Datavetenskap

Håkan Bergmark och Tony Bergh

Utvecklingsmiljö för Java med stöd för kontraktsprogrammering

Master's Thesis

2003:06

Utvecklingsmiljö för Java med stöd för kontraktsprogrammering

Håkan Bergmark och Tony Bergh

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en magisterexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Håkan Bergmark

Tony Bergh

Godkänd, 2003-06-23

Handledare: Martin Blom

Examinator: Donald F. Ross

Sammanfattning

Denna D-uppsats syftar till att beskriva bakgrunden till och tillkomsten av en utvecklingsmiljö för Java som har stöd för kontraktsprogrammering. Dokumentet specificerar vad syntax och semantik är, detta krävs för att man skall kunna sätta sig in i resten av dokumentet. Begreppet kontraktsprogrammering förklaras ingående då detta är ett nyckelbegrepp i utvecklingsmiljön och därmed denna uppsats. En genomgång av ett antal verktyg som används inom området kontraktsprogrammering görs och testas för att demonstrera deras funktionalitet. Dokumentet presenterar sedan ett idealsystem som vore det bästa möjliga resultatet av arbetet. Detta idealsystem används sedan för att jämföra det egentliga resultatet av arbetet med. Resultatet är en utvecklingsmiljö för Java som har stöd för att visa för- och eftervillkor vid metodanrop. Utvecklingsmiljön stöder även visning av klassinvarianter när nya instanser av klasser skall göras. Resultatet är alltså en utvecklingsmiljö som stöder kontraktsprogrammering. Denna utvecklingsmiljö ska förenkla för programmeraren att använda sig av kontraktsprogrammering vid utveckling av mjukvara. Författarnas förhoppning är att denna utvecklingsmiljö ska kunna bidra till bättre och mer pålitlig mjukvara.

Development Environment for Java with Support for Programming by Contract

Abstract

The purpose of this master's thesis is to describe the background to and creation of a development environment for Java that has support for programming by contract. The document specifies what syntax and semantics are since this is needed to understand the rest of the document. The conception of programming by contract is thoroughly explained since this is a key notion in the development environment and therewith also in this document. A review of some of the tools that is used within the area of programming by contract is made. These tools are also tested to demonstrate their functionality. The document then presents an ideal system that would be the best possible result of the project. This ideal system is later on used to compare the actual result of the project to. The result is a development environment for Java that has support for showing pre- and postconditions when invoking a method. The development environment also supports showing class invariants when new class instances are made. Consequently the result is a development environment that supports programming by contract. The intent of this development environment is to simplify for the programmer to use contracts in software development. It is the authors' hopes that this will lead to better and more reliable software.

Kommentar [TB1]: Var ska vi lägga in Tack-grejen? Efter Abstracten verkar det lite trångt…

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

Det finns ett ständigt krav på bättre och pålitligare mjukvara. Trots detta förekommer inte sällan fel, så kallade buggar, i mjukvaran som kan leda till ökade kostnader både för utvecklarna och användarna av mjukvaran. Det kan finnas många faktorer och orsaker till att buggar förekommer i mjukvara. Det kan till exempel vara slarv, logiska fel, missförstånd, för komplex kod och så vidare. Skulle kodkvaliteten kunna förbättras kan det finnas stora pengar för företagen att tjäna. Ett sätt att kunna förbättra kodkvaliteten är att se till att kodens semantik, eller innebörd, kan förstås enkelt. Om metoder beskrivs på ett sådant sätt att deras funktionalitet klart framgår utan att koden behöver studeras, skulle detta säkerligen minska antalet missförstånd. Kodkvaliteten skulle öka och det skulle finnas färre fel i mjukvaran. En utvecklingsmiljö som visar semantiken för metoder vid anrop skulle förenkla förståelsen för koden hos programmeraren. Om denna utvecklingsmiljö även skulle stödja kontraktsprogrammering, så att klassinvarianter och för- och eftervillkor visades, skulle det kunna öka pålitligheten hos mjukvaran och förhoppningsvis också öka kvaliteten på koden.

2 Bakgrund

I detta kapitel presenteras först kortfattat motiveringen till detta arbete följt av syftet och målet med detta arbete. Därefter följer ett stycke om syntax och semantik, något som är väsentligt för resten av arbetet. Även begreppet kontraktsprogrammering är viktigt för resten av arbetet, därför redogörs även detta i kapitlet. Därefter följer en liten presentation av ett tidigare arbete som har gjorts på Karlstads universitet inom samma område.

2.1 Motivering

Karlstads universitet har under ett flertal år bedrivit forskning inom området kontraktsprogrammering och dess inverkan på mjukvaran. Vi har fått i uppdrag, av forskningsgruppen för programvaruutveckling (SERG) vid Karlstads universitet, att ta fram en utvecklingsmiljö som stöder kontraktsprogrammering. Denna utvecklingsmiljö skall hämta och visa upp den semantiska informationen i form av kontrakt för metoder samtidigt som programmeraren anropar dessa.

Dagens utvecklingsmiljöer erbjuder oftast syntaktiskt stöd som till exempel *syntax highlighting* och *code completion*, men semantiskt stöd finns sällan att tillgå frånsett några få undantag. Däremot saknar vi kännedom om någon utvecklingsmiljö där kontraktsprogrammering stöds i form av uppvisning av kontrakt. Det finns dock ett antal separata verktyg för att underlätta kontraktsprogrammering, varav några presenteras i kapitel 3. Den semantiska informationen som skall presenteras kommer att vara i form av för- och eftervillkor för metoderna. För- och eftervillkoren skall visas när programmeraren anropar metoder. Meningen med detta är att programmeraren enklare skall kunna uppfylla kontrakten samt att detta skall uppmuntra till användning av kontraktsprogrammering vid utveckling av mjukvara. Om projektet faller ut väl så är det i förlängningen tänkt att utvecklingsmiljön ska användas i laborationssalarna på Karlstads universitet. Det öppnar då möjligheter för att undersöka om denna funktionalitet och stöd underlättar för programmeraren och gynnar mjukvarans kvalitet.

Kommentar [TB2]: Skall dessa förklaras här?

2.2 Syfte och mål

Syftet med detta arbete är att ta fram en utvecklingsmiljö. Denna utvecklingsmiljö skall tillhandahålla semantisk hjälp vid mjukvaruutveckling. Det är tänkt att detta verktyg skall användas tillsammans med programmeringsstilen kontraktsprogrammering. Fördelen med semantisk hjälp är att programmeraren vet vad metoder gör och behöver inte leta upp koden för metoden för att förstå vad den gör.

Kommentar [TB3]: Det känns lite som att detta repeterar lite av det som står i förra delkapitlet. Vi har tvistat lite om hur vi ska göra. Vad tycker du Martin?

Målet är att utveckla ett tilläggsprogram till en utvecklingsmiljö där man vid programmering kan få upp kontrakt för en viss metod i form av för- och eftervillkor. Målet är också att detta tilläggsprogram skall vara så oberoende av utvecklingsmiljön som möjligt. Detta oberoende öppnar möjligheten att kunna använda detta tilläggsprogram till andra utvecklingsmiljöer genom att man behöver göra ett så litet ingrepp som möjligt. För att man skall kunna uppnå ett oberoende kräver det att man använder sig av någon form av gränssnitt. Gränssnittet ska frigöra kopplingen mellan tilläggsprogrammet och utvecklingsmiljön. Istället finns det starka kopplingar mellan gränssnittet och utvecklingsmiljön. Detta innebär i sin tur att det endast är gränssnittet man behöver ändra om man vill anpassa tilläggsprogrammet till en annan utvecklingsmiljö.

2.3 Syntax och semantik

Språk kan definieras genom att beskriva hur meningar kan byggas upp och vad innebörden i meningarna är. Reglerna som specificerar hur meningarna i ett språk kan byggas upp kallas ett språks syntax. För att ett språk skall kunna användas bör dess meningar ha en betydelse, vilket kallas för språkets semantik. Tillämpar man ovanstående på programmeringsspråk kan det sägas att syntaxen beskriver hur programmen får konstrueras och semantiken beskriver programmens innebörd.

Att beskriva syntax är enklare än att beskriva semantik, troligtvis på grund av att det finns en koncis och universellt accepterad notation för att beskriva syntax. Någon motsvarande har ej utvecklats när det gäller semantik, åtminstone inte i samma omfattning [21]. Den notation som används för att beskriva ett programmeringsspråks syntax är *context-free grammar* eller *Backus-Naur form* (BNF). Figur 2.1 visar hur man kan definiera en heltalsoperation med de fyra räknesätten i BNF.

Kommentar [TB4]: Vad menar Hans med att ett språk är en context free grammar? Är det inte ett språks grammatik som är en CFG? Hur ska vi skriva?

```
<op> ::= <add> | <sub> | <mul> | <div>
<add> ::= <operand> ”+” <operand>
<sub> ::= <operand> ”-” <operand>
<mul> ::= <operand> ”*” <operand>
<div> ::= <operand> ”/” <operand>
<operand> ::= <integer> | <variable>
<variable> ::= ”a” | ”b” | ”c”
<integer> ::= ”0” | ”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” | ”8” | ”9”
```

Figur 2.1: Exempel på grammatik i BNF.

Grammatiken i Figur 2.1 består av åtta produktioner eller regler. Vänstersidan i produktionerna utgörs av *non-terminals* och högersidan utgörs av en sekvens av *non-terminals* och/eller *terminals*. *Terminals* är bassymbolerna som strängar utgörs av. När man pratar om programspråk är *token* ett synonym för *terminal*. *Non-terminals* är syntaktiska variabler som betecknar mängder av strängar. *Non-terminals* bestämmer också en hierarkisk struktur i språket som är användbart vid både syntaktisk analys och översättning. Grammatiken måste även bestå av en startsymbol som i det här fallet är <*op*>. [1]

Kommentar [t5]: Kan man översätta *strings* till strängar på svenska? Eller kan/bör vi skriva teckensekvens istället?

Om man kontrollerar Figur 2.1 ser man att även operationen division med noll är syntaktiskt korrekt. Eftersom division med noll inte är matematiskt definierat kan man anta att det inte skall vara tillåtet här heller. Detta har dock med semantiken i språket att göra. Man brukar

skilja på statisk och dynamisk semantik, där statisk semantik kan kontrolleras vid kompilering och dynamisk semantik inte kan kontrolleras förrän vid programkörning. Det är önskvärt att så mycket som möjligt av semantiken kontrolleras vid kompilering eftersom fel då blir mycket enklare att upptäcka än om de sker under programkörning. Den statiska semantiken för ett språk är bara indirekt kopplad till innebörden i program under körning, det har istället att göra med korrekt utformning av program (det vill säga syntax och inte semantik) [21]. I vissa fall kan man omforma den statiska semantiken så att den kan inkluderas i grammatiken. Figur 2.2 visar ett exempel på hur division med noll kan undvikas rent syntaktiskt genom att utöka grammatiken (gäller ej vid användning av variabler, detta förklaras lite senare). Figur 2.2 visar ett mycket enkelt exempel och det är lätt att förstå att det kan bli mycket svårt att utöka grammatiken om den är mycket omfattande och komplicerad. I andra fall är det omöjligt, som till exempel fallet när en variabel i ett programmeringsspråk måste ha blivit deklarerad innan den kan refereras till, [21].

```
<op> ::= <add> | <sub> | <mul> | <div>
<add> ::= <operand> ”+” <operand>
<sub> ::= <operand> ”-” <operand>
<mul> ::= <operand> ”*” <operand>
<div> ::= <operand> ”/” <denominator>
<operand> ::= <integer> | <variable>
<denominator> ::= <pos_int> | <variable>
<variable> ::= ”a” | ”b” | ”c”
<integer> ::= ”0” | ”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” | ”8” | ”9”
<pos_int> ::= ”1” | ”2” | ”3” | ”4” | ”5” | ”6” | ”7” | ”8” | ”9”
```

Figur 2.2: Exempel på omarbetad grammatik i BNF.

Dynamisk semantik är den typ av semantiska regler som inte kan kontrolleras under kompilering utan endast under programkörning. Statisk semantik har med programmets utformning att göra, vilket aldrig dynamisk semantik har. Dynamisk semantik har endast att göra med programmets innebörd. Ett exempel på dynamisk semantik är användningen av variabler som i Figur 2.2. Dessa kan tilldelas värden under programkörning och därmed kan värdet noll erhållas som nämnare vid division. Detta kan alltså inte kontrolleras vid kompilering, därmed måste man skydda sig mot detta under programkörning. Denna typ av semantiska regler kan upprätthållas genom antingen tester i programmet eller genom att använda sig av kontrakt vid programmeringen.

2.4 Kontraktsprogrammering

Kontraktsprogrammering innebär kortfattat att man sätter upp kontrakt mellan klient (användaren av metoden) och metod. Kontraktet beskriver vad som skall gälla före metodanropet, ett så kallat förvillkor, samt vad som gäller efter metodanropet om förvillkoret var uppfyllt, ett så kallat eftervillkor. Eftervillkoret kan endast garanteras om förvillkoret är uppfyllt. För att klienten skall uppfylla sin del av kontraktet måste klienten följa förvillkoret. Om förvillkoret är uppfyllt måste även eftervillkoret uppfyllas. Detta innebär att man alltid kommer att kunna garantera en metods beteende om kontrakten efterlevs, förutsatt att kontrakten är väldefinerade och korrekta. För att kontraktsprogrammeringen skall kunna fungera på ett tillfredsställande sätt så måste kontrakten specificeras på ett så tydligt och otvetydigt sätt som möjligt.

2.4.1 Olika nivåer av formalism

Enligt [4] finns det olika nivåer av formalism när det gäller specifikation av semantik. Dessa är: intuitiv semantik, strukturerad semantik, exekverbar semantik samt formell semantik. Författarna till [4] har även tagit med fallet där ingen information om semantik har angivits explicit och där endast rent syntaktisk information tillhandahålls. Anledningen till att den nivån har tagits med är för att göra beskrivningen av nivåerna komplett. Den nivån har dock utelämnats i denna rapport.

Intuitiv semantik är en beskrivning av en klass och dess metoder i form av ren text. Informationen är ostrukturerad till sin form och det finns ingen mall för vilken information beskrivningen skall innehålla. Därmed är det upp till den som författar beskrivningen att göra beskrivningen så komplett som möjligt. Detta kan ge upphov till tvetydighet och därmed flera olika tolkningar. Tvetydigheten kan ge upphov till att de egentliga förvillkoren av misstag utelämnas eller feltolkas. Denna nivå av semantikbeskrivning kan försvåra både för den som specificerar semantiken och även för de som skall tyda specifikationen. Detta illustreras i Figur 2.3.

Metoden `pop` avlägsnar det element som ligger överst på stacken. Stackens storlek minskas med ett.

Figur 2.3: Exempel på intuitiv semantik.

Beskrivningen i Figur 2.3 tar inte explicit upp förvillkoret om att stacken måste innehålla minst ett element före anrop till metoden *pop*. Genom att detta förvillkor utelämnats så kan metoden användas felaktigt vilket i sin tur kan leda till fel i systemet.

Genom att använda strukturerad semantik blir den semantiska informationen enklare att uttyda, men det är därmed inte sagt att den automatiskt blir enklare att förstå. Den senare utsagan baseras på att det fortfarande är upp till den som specificerar semantiken att uttrycka sig så tydligt som möjligt. Det är dock enklare för den som specificerar semantiken att veta vad han/hon skall skriva och skriva rätt saker tack vare att det finns rubriker, till exempel förvillkor och eftervillkor. Två fördelar med strukturerad semantik nämns i [4], nämligen att författaren av semantiken tänker igenom designen och att det underlättar förståelsen för semantiken hos klientprogrammerare och användare av metoden. Exempel på strukturerad semantik ges i Figur 2.4.

`pop` avlägsnar det element som ligger överst på stacken.

Förvillkor: stacken är inte tom.

Eftervillkor: det översta elementet har avlägsnats från stacken. Storleken på stacken har minskats med ett. Stacken är inte full.

Figur 2.4: Exempel på strukturerad semantik.

Exekverbar semantik är strukturerad semantik där villkoren kan översättas till exekverbar kod. Detta möjliggör att man kan kontrollera och verifiera kontrakten under programkörning. Det finns ett flertal verktyg som är anpassade för den här typen av semantiska beskrivningar, dessa beskrivs mer utförligt i kapitel 3. Figur 2.5 visar ett exempel på exekverbar semantik. Observara dock att eftervillkoret i kontraktet i Figur 2.5 inte uttryckligen anger att det är det översta elementet som har tagits bort, utan endast beskriver en konsekvens av att ett element har avlägsnats från stacken. Detta innebär i praktiken att eftervillkoret även uppfylls om till exempel det sista elementet på stacken avlägsnas.

Kommentar [TB6]: Kan vi trycka på ordet med understrykning? Eller är det förfulande?

`pop` avlägsnar det element som ligger överst på stacken.

Förvillkor: `!this.isEmpty()`

Eftervillkor: `this.size() == old.size()-1 && !isFull()`

Figur 2.5: Exempel på exekverbar semantik.

Formell semantik är när man beskriver metoder och klasser med hjälp av formella språk som matematiskt exakt beskriver vad som händer. Man kan i vissa fall använda sig av formella beskrivningar för att visa att ett program gör vad det ska. Dessa språk brukar dock ha en stor nackdel i att de endast kan uttydas av personer med matematisk bakgrund, eftersom deras beskrivningar grundar sig på logiska formler. Detta har då medfört att dessa är mycket svåra att uttyda för en person utan en matematisk bakgrund, vilket i sin tur har lett till att deras användning har begränsats. Ett exempel på ett formellt specifikationsspråk som är relativt svårt att uttyda är Z [5]. Ett formellt specifikationsspråk som har en betydligt enklare syntax än till exempel Z är språket Object Constraint Language (OCL) [6]. OCL använder sig av en syntax som påminner mycket om exekverbara uttryck blandat med nyckelord för logiska symboler. Figur 2.6 visar ett exempel på formell semantik, i figuren används OCL.

```
MyStack :: pop()
  pre : not isEmpty()
  post: size() = size()@pre – 1
  post: not isFull()
```

Figur 2.6: Exempel på formell semantik.

2.4.2 Kontrakt och arv

Bertrand Meyer, som anses av många vara den store pionjären inom området kontraktsprogrammering, anger två regler om vad som gäller vid kontrakt och arv i [17]. Dessa två regler är den så kallade *Parent's invariant rule* och *Assertion redefinition rule*.

Parent's invariant rule anger vad som skall gälla angående klassinvarianter vid arv. Regeln visas i Figur 2.7 och den säger att alla ärvda invarianter, antingen indirekt ärvda eller direkt ärvda, skall även gälla för klassen själv tillsammans med eventuella egna invarianter.

> **Parent's invariant rule:** The invariants of all the parents of a class apply to the class itself.

Figur 2.7: Parent's invariant rule [17].

Assertion redefinition rule anger vad som skall gälla vid arv av metoder med för- och eftervillkor. Regeln visas i Figur 2.8 och den säger att när en metod omdefinieras så måste den nya metodens förvillkor vara lika eller mindre restriktivt, samt dess eftervillkor måste vara lika eller mer restriktivt. Med andra ord så kan den nya metoden endast specificera ett lika starkt eller svagare förvillkor än den omdefinierade metoden, samt att den måste

garantera ett lika starkt eller starkare eftervillkor än den omdefinierade metoden. Anledningen till detta är den dynamiska bindningsmekanismen för objekt. Ett objekt ska kunna anropas genom dynamisk bindning genom dess superklass. Regeln garanterar då att den nya metodens kontrakt inte bryts vid dynamisk bindning, vilket kunde ha varit fallet om det omvända hade gällt.

> **Assertion redefinition rule:** Let *r* be a routine in class *A* and *s* a redefinition of *r* in a descendant of *A*, or an effective definition of *r* if *r* was deferred. Then pre_s must be weaker than or equal to pre_r, and $post_s$ must be stronger than or equal to $post_r$.

Figur 2.8: Assertion redefinition rule [17].

2.4.3 Kontrakt och undantag

Slarvigt använt degenererar undantagshantering till en form av *goto*-instruktion. När något har inträffat som inte anses som normalt hoppar man helt enkelt ut ur metoden och hamnar i en programdel som hanterar undantag. Undantagshantering är dessutom mycket krävande i form av processorkraft och en ökad komplexitet i programmen. Man bör således i möjligaste mån undvika användning av undantag vid programmering.

Det finns dock tillfällen när det kan vara lämpligt att använda sig av undantagshantering. Ett exempel är att i utvecklingsfasen använda sig av undantagshantering för att upptäcka fel. Annars kan man med fördel kombinera kontrakt och undantagshantering när man inte på förväg kan veta hur det kommer att gå. Exempel på detta är när det handlar om fel med kommunikationslänkar, operativsystemet eller hårdvaran, som *input*- eller *output*-fel. [17]

Kommentar [TB7]: Är inte detta motiverat i föregående mening?

2.4.4 Skillnader mellan kontraktsprogrammering och defensiv programmering

Kommentar [TB8]: Det heter väl defensiv programmering och inte defensivprogrammering?

Den stora skillnaden mellan kontraktsprogrammering och defensiv programmering är hur man ser på användaren av en metod. I kontraktsprogrammering så krävs det av användaren eller klienten att uppfylla vissa kriterier för att kunna få använda metoden. Om dessa kriterier uppfylls så garanteras ett visst resultat, vilket inte är fallet när dessa inte uppfylls. Således bygger kontraktsprogrammering på ett ömsesidigt förtroende mellan klient och metod.

Vid defensiv programmering saknas däremot helt förtroende mellan klient och metod. Varje värde måste testas och kontrolleras så att eventuella fel kan undvikas. Detta innebär att

mycket extra kod i form av att tester måste implementeras både i klienten och i metoden. Detta leder ofta till att koden ökar i omfattning och storlek samt att läsbarheten minskar. Det kan också vara mycket svårt att täcka in och förutse alla fel som kan tänkas förekomma.

En stor fördel med kontraktsprogrammering är att man inne i en metod inte behöver täcka in alla felaktiga fall, utan endast behöver koncentrera sig på de fall där metoden används korrekt. Detta medför att koden oftast blir mer kompakt eftersom den inte behöver innehålla tester på värden i samma omfattning, vilket i sin tur leder till ökad läsbarhet. Risken att metoder används felaktigt minskar eftersom användare respekterar förvillkoren och tar del av eftervillkoren. Nackdelen med kontraktsprogrammering är dock att detta inte kan användas i alla fall som i mjukvara som inte kan tolerera att systemen slutar att fungera bara för att ett förvillkor inte uppfylls. Exempel på sådan mjukvara kan vara mjukvara som ingår i kritiska system såsom kärnreaktorer och fordon. Sådan mjukvara måste innehålla tester för att kunna undvika fel vid onormala och oväntade situationer, så att katastrofer förhindras och undviks.

2.5 Tidigare arbeten

Våren 2000 gavs två studenter på dataingenjörsprogrammet vid Karlstads universitet, uppdraget att utveckla en Java-editor som skulle presentera semantisk information för användaren. Detta arbete ingick i dessa studenters C-uppsats [15] och utvecklingstiden låg därför på totalt 20 veckor. Den semantiska informationen skulle vara i form av för- och eftervillkor för metoder, vilka skulle presenteras i skrivande stund.

Man löste uppgiften genom att utveckla en ny editor, CoffeeMaker, som baserades på en redan befintlig editor, Jipe [13]. Även editorn Jipe baserade sin lösning på en redan befintlig editor nämligen jEdit [12]. Anledningen till att man valde att basera sin lösning på en redan befintlig editor var att själva slippa implementera *syntax highlightning*.

Man lyckades med uppgiften att utveckla en funktionalitet som presenterade kontrakt i skrivande stund. Dock hade denna lösning vissa begränsningar. Dessa begränsningar omfattade bland annat att endast kontrakt för klasser placerade i samma paketstruktur, kunde presenteras. Dessutom kunde inte ärvda metoders kontrakt presenteras och funktionaliteten kunde ej heller stängas av. Prestandan var också ett stort problem vilket berodde dels på att editorn implementerades i Java och dels på ineffektiva algoritmer.

På grund av editorns begränsade funktionalitet och ovan nämnda begränsningar, kunde CoffeeMaker aldrig tas i bruk i någon större utsträckning. Därför slutade CoffeeMaker vid att bara vara en prototyp utan egentligt användarvärde, dock med vissa utvecklingsmöjligheter.

Förutom CoffeMaker så saknar vi kännedom om någon annan editor eller utvecklingsmiljö som stöder kontraktsprogrammering i form av visning av kontrakt för programmeraren. Däremot finns en mängd separata verktyg och program, både kommersiella och icke kommersiella, för att underlätta programmering via kontrakt. Vi har valt att titta närmare på några av dessa verktyg nämligen iContract, Jass, jContractor och Jcontract, vilka samtliga är utvecklade för att användas vid kontraktsprogrammering i Java. Dessa fyra verktyg presenteras i kapitel 3.

3 Befintliga verktyg

I detta kapitel presenteras ett antal externa verktyg, vars syfte är att underlätta användningen av kontraktsprogrammering. De verktyg som beskrivs här är gjorda för att användas vid kontraktsprogrammering i programmeringsspråket Java. Dessa verktyg fungerar huvudsakligen på ett reaktivt sätt, det vill säga de underlättar inte direkt kontrakts-programmeringen vid editering av källkod, utan de försöker upptäcka kontrakt som ej följs under exekvering av program. Dessa verktyg används därför främst i testningssyfte för att kunna upptäcka kontraktsbrott i programmen, men de erbjuder ingen direkt hjälp för att förhindra att kontraktsbrotten uppstår under utveckling av programmen.

Kommentar [TB9]: Kan vi kalla dem så?

3.1 Inledning

De flesta programmeringsspråk saknar direkt stöd för kontraktsprogrammering och detta måste tillföras som en extra del. Ett programmeringsspråk som däremot har ett inbyggt stöd för kontraktsprogrammering är Eiffel. I Eiffel skrivs kontrakten direkt i koden med nyckel-orden *require* för förvillkor, *ensure* för eftervillkor och *invariant* för klassinvariant. Eiffel utvecklades av Bertrand Meyer.

Programmeringsspråket Java har i version 1.4 infört en form av semantikkontroll, så kallade *assertions*. Ett *assertion* är ett uttryck i Java som ger programmeraren möjligheter att

kontrollera de antaganden man har gjort för programmet. Om man till exempel skriver en metod som skall beräkna en partikels hastighet kan man skriva ett *assertion* om att den uträknade hastigheten är mindre än ljusets hastighet. Varje *assertion* innehåller ett booleskt uttryck som antas vara sant när detta *assertion* exekverar. Om det inte är sant kommer systemet att kasta ett undantag. Genom att verifiera att det booleska uttrycket verkligen är sant, bekräftar detta *assertion* programmets beteende vilket ökar tilltron till att programmet är felfritt. Erfarenheter har visat, enligt [20], att användandet av *assertions* när man programmerar är ett av de snabbaste och mest effektiva sätten att hitta och korrigera fel på. Som en extra bonus dokumenterar *assertions* programmets funktion och underlättar därmed underhållningsarbetet av programmet. [20]

Provkörningar av verktygen har genomförts där enkla kontrakt bryts för att se hur verktygen hanterar sådana situationer. Klassen som har använts i testerna presenteras i kapitel 3.2. Klassen i kapitel 3.2 har anpassats för varje verktyg där det specifika verktygets syntax har använts, de anpassade klasserna återfinns i bilaga A.

3.2 Testklass - MyStack

```
/**
 * Invariant: storleken på stacken ligger alltid mellan 0 och MAX (0<=size<=MAX).
 */
public class MyStack {
    private java.util.Vector myStack;
    private final int MAX = 10;
    private int size;

    /**
     * Förvillkor: true
     * Eftervillkor: stacken är initierad och har storleken 0.
     */
    public MyStack() {
        this.myStack = new java.util.Vector();
        this.size = 0;
    }

    /**
     * Förvillkor: stacken är inte tom.
     * Eftervillkor: det översta elementet är borttaget, storleken har minskat med 1 samt
     *               stacken är inte full.
     */
    public void pop() {
        this.myStack.remove(this.getSize() - 1);
        this.size--;
```

```
}

/**
 * Förvillkor: stacken är inte full och elementet e är ett giltigt objekt.
 * Eftervillkor: elementet e har lagts överst på stacken, storleken har ökat med 1 samt
 *               stacken är inte tom.
 */
public void push(Object e) {
    this.myStack.add(e);
    this.size++;
}

/**
 * Förvillkor: stacken är inte tom.
 * Eftervillkor: det översta elementet på stacken har returnerats.
 */
public Object top() {
    return this.myStack.get(this.getSize() - 1);
}

/**
 * Förvillkor: true
 * Eftervillkor: storleken på stacken har returnerats.
 */
public int getSize() {
    return this.size;
}

/**
 * Förvillkor: true
 * Eftervillkor: det returnerade värdet har samma sanningsvärde som påståendet att stacken
 *               är tom dvs size == 0.
 */
public boolean isEmpty() {
    return this.getSize() == 0;
}

/**
 * Förvillkor: true
 * Eftervillkor: det returnerade värdet har samma sanningsvärde som påståendet att stacken
 *               är full dvs size == MAX.
 */
public boolean isFull() {
    return this.getSize() == this.MAX;
}

/**
 * Förvillkor: true
 * Eftervillkor: stacken är tom.
 */
public void makeEmpty() {
```

```
        this.myStack.clear();
        this.size = 0;
    }

    /**
     * Förvillkor: true
     * Eftervillkor: true
     */
    public static void main(String[] args) {
        MyStack stack = new MyStack();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
        stack.makeEmpty();
    }
}
```

3.3 iContract

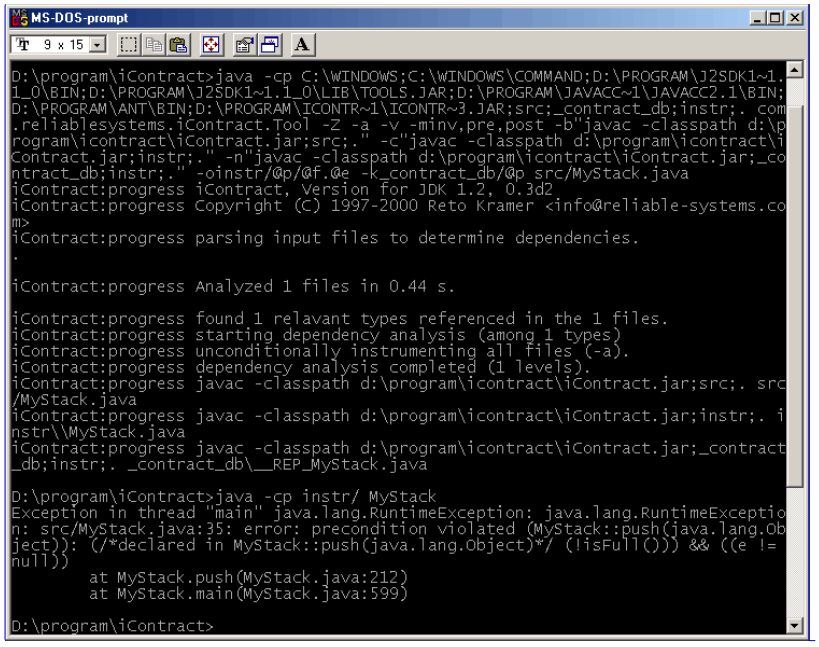
iContract är ett verktyg för Java som ger utvecklare stöd vid kontraktsprogrammering. Principen som används är att kontrakten specificeras som integrerade delar av källkoden. Likheten med den princip som används i programmeringsspråket Eiffel är tydlig, se kapitel 3.1. Utvecklarna av iContract nämner också denna likhet i sin beskrivning av verktyget. iContract är en gratis så kallad källkodspreprocessor. Att iContract är en källkodspreprocessor innebär att iContract går igenom koden före kompileringen och gör en egen källkodsfil där klassinvarianter och för- och eftervillkor för metoderna finns integrerade i koden. Kontrakt skrivs i Javadoc (se kapitel 5.3.1) med hjälp av specialtaggarna *@pre* för förvillkor, *@post* för eftervillkor och *@invariant* för klassinvariant. För mer ingående beskrivning hänvisas läsaren till [7] och [16].

3.3.1 Provkörning av iContract

Kontrakten för klassen *MyStack* i kapitel 3.2, har modifierats för att följa syntaxen för kontraktsbeskrivningar i iContract, se bilaga A.1. iContract följer till viss del OCL's [6] syntax och är därmed relativt enkelt att använda samt erbjuder en del kraftfulla operatorer. Symbolen *return* är motsvarigheten till *result* i OCL, det vill säga en metods returvärde,

vilken kan användas i eftervillkoret för en metod. På samma sätt som i OCL så kan man i iContract referera till tillståndet före ett metodanrop genom att lägga till *@pre* på typen som skall kontrolleras. När man använder iContract så skapas en ny version av .java-filen/-erna vilka inkluderar testerna av de specificerade kontrakten, se bilaga A.2. Även en fil som används av iContract för att möjliggöra arv av kontrakt genereras. För att sedan testa om applikationen uppfyller kontrakten, så kompileras de av iContract genererade .java-filerna. De resulterande .class-filerna kan sedan exekveras av iContract för att kontrollera om kontraktsbrott förekommer.

Figur 3.1 visar resultatet av att kompilera och exekvera den av iContract genererade *MyStack*-filen, innehållande kontraktstesterna, med iContract. Verktyget meddelar användaren att applikationen bryter mot ett kontrakt på rad 599 i main-metoden, genom att inte uppfylla förvillkoret (`!isFull() && e != null`) för metoden *push()*. Metoden vars förvillkor bryts återfinns på rad 212. Båda dessa radnummer refererar till den av iContract automatgenererade källkodsfilen MyStack.java (bilaga A.2). Metoden *push()* finns deklarerad på rad 35 i källkodsfilen MyStack.java (bilaga A.1), vilket också kan utläsas ur Figur 3.1.



Kommentar [TB10]: Är denna storlek på bilden stor nog?

Figur 3.1: Provkörning iContract.

3.3.2 Fördelar med iContract

- iContracts kontraktssyntax är enkel att använda samt innefattar även kraftfulla uttryck beroende på att syntaxen till viss del följer OCL-specifikationen.
- Eftersom kontrakten uttrycks i form av Javadoc, så hålls källkoden kompatibel med standard Java.
- Det är enkelt att generera dokumentation av kontrakten med hjälp av Javadoc-verktyget.
- Möjlighet att styra vilka typer av villkor (förvillkor, eftervillkor, invarianter) som skall testas och i vilka klasser.
- iContract stöder arv av kontrakt.
- iContract är gratis att använda.

3.3.3 Nackdelar med iContract

- Inställningarna för iContract är relativt krångliga och det är helt nödvändigt att skapa en script-fil för att slippa skriva de långa och krångliga uttrycken om och om igen.
- I iContract kan inte ett kontrakt innehålla en referens till det tidigare tillståndet för den givna metoden. Till exempel *@post return == top()@pre* för metoden *top()* resulterar i en rekursiv ”evighets” loop, vilket innebar att detta eftervillkor fick utelämnas i kontraktet för metoden *top()*.
- En annan nackdel är att en ny .java-fil måste skapas speciellt för kontraktstesterna. Detta medför att felmeddelandena om kontraktsbrott som visas upp i iContract refererar till denna fil, vilket innebär att det kan vara svårare att spåra kontraktsbrottet i originalkällkodsfilen.
- Ytterligare en nackdel är att om en metod har flera förvillkor och/eller eftervillkor så kommer dessa att slås samman till ett förvillkor respektive eftervillkor i koden. Detta resulterar i att det blir svårare att spåra vilket specifikt för- respektive eftervillkor som inte uppfylls i kontraktet. Detta ses i Figur 3.1 där det står *precondition violated … (!isFull() && (e != null))….* Användaren får här ingen information om det är förvillkoret *!isFull()* eller om det är förvillkoret *e != null* som inte uppfylls.
- Bara det första kontraktsbrottet presenteras. Detta beror på att undantag som kastas vid kontraktsbrott inte fångas, vilket leder till att applikationen terminerar.

3.4 Jass – Java with Assertions

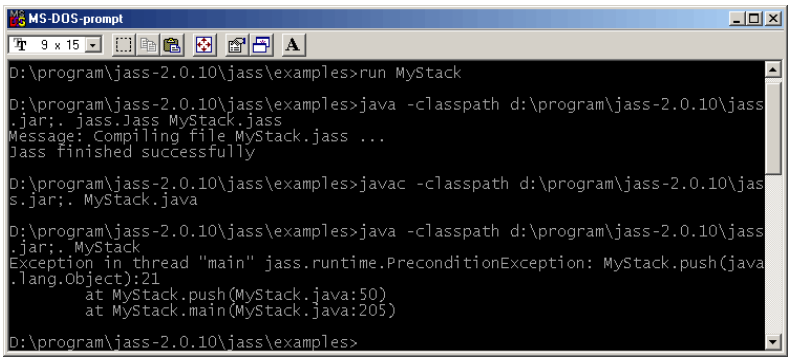
Jass är ett verktyg som erbjuder kontraktsprogrammering i Java enligt samma princip som programspråket Eiffel. Jass är en källkodspreprocessor som går igenom källkoden innan den kompileras och ändrar Jass-specifika nyckelord till integrerad kod. Den slutliga koden är en ren Javakod där kontrakt kontrolleras dynamiskt under programkörning.

Verktyget Jass använder sig av .jass-filer, vilka innehåller vanlig Java-kod samt kontraktsbeskrivningar som kan tolkas av verktyget. Dessa kontraktsbeskrivningar utgörs av de tre nyckelorden *require*, *ensure* samt *invariant*, vilka representerar förvillkor, eftervillkor respektive invariant (jämför med Eiffel i kapitel 3.1). Kontraktsbeskrivningarna skrivs mellan tecknen /** och **/ vilka placeras på olika ställen i källkoden. Nyckelordet *require* måste stå först i en metod, medan *ensure* skall stå sist i metoden. Klassinvarianten anges sist i klassen med nyckelordet *invariant*. I Jass syntax ingår även bland annat nyckelorden *Result* och *Old*, där *Result* representerar en metods returvärde och *Old* refererar till objektets tillstånd före metodanropet. *Old* är helt enkelt en kopia av objektet och därför måste klassen som använder sig av *Old* implementera interfacet *Cloneable* för att denna kopia skall kunna skapas.

När man kompilerar .jass-filen med hjälp av Jass så genereras en .java-fil med samma namn som .jass-filen. Denna fil innehåller förutom den ursprungliga koden i .jass-filen också testerna av kontrakten i form av Java-kod. Denna fil kan sedan kompileras och den resulterande .class-filen exekveras med hjälp av verktyget. Jass kommer att meddela användaren, om ett kontraktsbrott förekommer, genom att kasta ett undantag för respektive villkor. För ytterligare information om Jass hänvisas läsaren till [8] och [3].

3.4.1 Provkörning av Jass

För att kunna testa klassen *MyStack* i kapitel 3.2, så måste motsvarande .jass-fil skapas. Filen MyStack.jass återfinns i bilaga A.3 och som kan ses här implementerar klassen interfacet *Cloneable* för att möjliggöra användandet av *Old*. I Figur 3.2 visas resultatet av användningen av Jass på filen MyStack.jass och den av Jass genererade filen MyStack.java, den senare återfinns i bilaga A.4. Som kan ses i figuren så har ett undantag, jass.runtime.PreconditionException, kastas eftersom ett förvillkor för metoden *push()* i MyStack.java på rad 205 brutits. Även på vilken rad det brutna förvillkoret specificerats, i både MyStack.java (rad 50) samt MyStack.jass (rad 21), kan utläsas i figuren. Dock ges ingen information om vad detta förvillkor innefattar.



Figur 3.2: Provkörning Jass.

3.4.2 Fördelar med Jass

- Verktyget är enkelt att använda och kräver inga krångliga inställningar.
- Valmöjlighet finns om undantag skall kastas eller om endast en varning skall utfärdas vid kontraktsbrott. Detta möjliggör att exekvera hela applikationen färdigt och på så sätt kunna lokalisera flera kontraktsbrott utan att behöva exekvera applikationen om och om igen.
- Man har valmöjligheten att välja för vilka typer av villkor som skall testas eller alternativt att inga tester skall infogas i den genererade .java-filen.
- Stöd för arv av kontrakt.
- Det är möjligt att lägga in kommentarer i kontrakten utan att testerna påverkas.
- Jass är gratis att använda.

3.4.3 Nackdelar med Jass

- Verktyget kräver att man skapar en särskild sorts källkodsfiler, .jass-filer.
- Kontraktsbeskrivningen följer inte Javadoc-beskrivningen samt att istället för OCL används en mer Eiffel-orienterad kontraktsbeskrivning.
- Vilket specifikt villkor som bryts presenteras ej och felmeddelandet refererar till kontraktsbrottet i den av Jass genererade .java-filen, vilket försvårar spårningen av brottet i källkoden (.jass-filen).
- Alla filer som använder sig av *Old* i kontraktsbeskrivningen måste implementera interfacet *Cloneable*, vilket innebär att klasserna utökas.
- På grund av att kontrakten delas upp och specificeras i olika delar av källkoden, minskar överblicken samt läsbarheten av kontrakten.
- Jass kräver ett separat verktyg för att generera dokumentation över kontrakten.

3.5 jContractor

jContractor ger en programmerare stöd för kontraktsprogrammering i Java. Detta sker genom att kontrakt skrivs som metoder som följer en enkel benämningskonvention. På så sätt behövs ingen förbehandling av källkoden, den är redan skriven i ren Java.

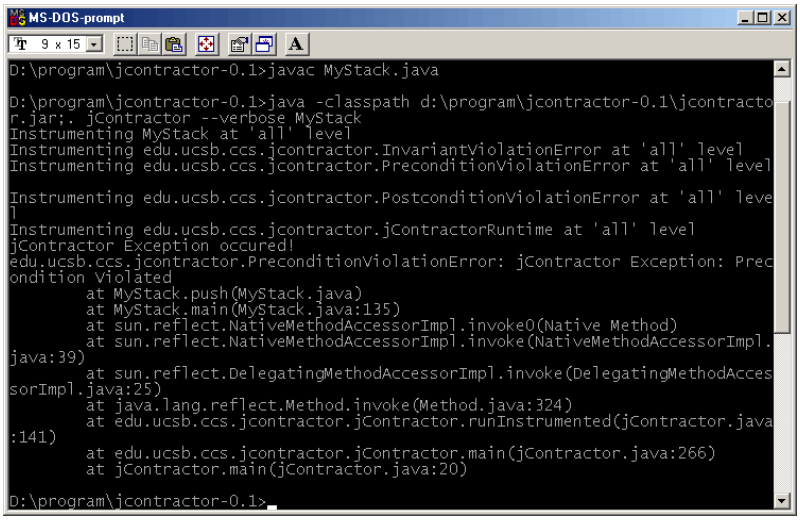
Verktyget jContractor arbetar med .java-filer i vilka kontrakt specificeras som booleska metoder. Dessa metoder följer benämningskonventionen enligt följande: förvillkor för en metod specificeras genom att definiera en metod med samma namn som den kontrakterade metoden + suffixet *_Precondition*. Förvillkorsmetoden tar in samma parametrar som metoden den gäller för och dess returvärde är ett booleskt uttryck som anger om det specificerade förvillkoret uppfylls eller ej. Eftervillkor specificeras på samma sätt som förvillkor med den skillnaden att suffixet hos dessa är *_Postcondition*. Dessa metoder tar in, förutom de parametrar som metoden den gäller för, även parametern *RESULT*. Denna parameter representerar returvärdet av den kontrakterade metoden. Om metoden saknar returvärde anges parametern att vara av typen *Void*. Klassinvarianter anges i metoden *_Invariant* vilken saknar inparametrar. Liksom Jass (se kapitel 3.4) använder sig jContractor av objektreferensen *OLD*, vilken måste deklareras explicit, för att kunna referera till tillståndet före ett metodanrop. Därmed måste även interfacet *Cloneable* implementeras av den kontrakterade klassen, för att möjliggöra kloning av objekt.

Den kontrakterade källkodsfilen kan köras antingen genom jContractor eller genom jInstrument. Skillnaden mellan de båda är att jContractor kontrollerar kontrakten utan att testerna behöver infogas i .class-filen vilket de måste om de körs genom jInstrument. jInstrument kan också användas för att avlägsna kontraktstester i .class-filen genom flaggan *–s*. För mer ingående information om jContractor eller jInstrument hänvisas läsaren till [11] och [14].

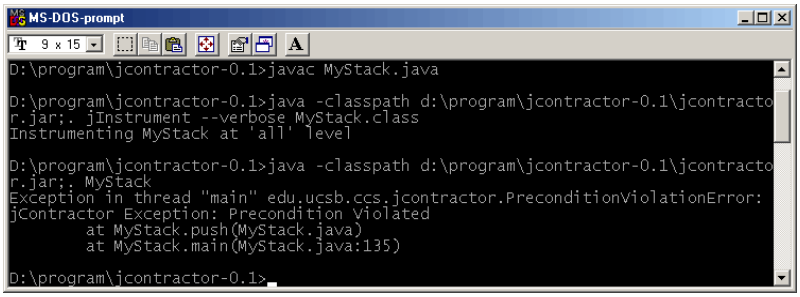
3.5.1 Provkörning av jContractor

I bilaga A.5 återfinns klassen *MyStack* vilken har anpassats till att följa jContractors kontraktsbeskrivning. I Figur 3.3 visas resultatet av att köra klassen *MyStack* genom jContractor. Som kan utläsas ur figuren så har ett undantag, *PreconditionViolationError*, kastats eftersom ett förvillkor har brutits på rad 135 i källkodsfilen. Förvillkoret gäller för metoden *push()* men ingen information om förvillkoret ges i felmeddelandet. I Figur 3.4 visas resultatet av att köra klassen *MyStack* genom jInstrument. Samma undantag kastas och samma

information ges till användaren om kontraktsbrottet som vid jContractor-exekveringen. Skillnaden ligger i att här finns alla tester inkluderade i MyStack.class vilket innebär att felmeddelandet blir betydligt kortare än det i Figur 3.3.



Figur 3.3: Provkörning jContractor.



Figur 3.4: Provkörning jInstrument.

3.5.2 Fördelar med jContractor

- Den största fördelen med jContractor är att man kan specificera kontrakt för klasser vars källkod inte är tillgänglig. Detta görs genom att skapa en klass med samma namn som den klass som kontraktet skall gälla för plus suffixet *_Contract*. I denna klass kan sedan kontrakten specificeras på samma sätt som beskrivits tidigare.
- Arv av kontrakt stöds av verktyget.
- Möjlighet att påverka vilka villkor i kontrakten som skall testas samt i vilka klasser dessa tester skall infogas.
- jContractor är gratis att använda.

3.5.3 Nackdelar med jContractor

- Kräver att man skriver kontraktsmetoderna själv i källkoden, vilket innebär mer arbete och mycket extra kod, vilket kan leda till mindre läsbara program.
- Kräver att alla klasser som använder sig av referensvariabeln *OLD* implementerar interfacet *Cloneable*, vilket innebär att klassen utökas.
- Felmeddelandet anger inte vilket specifikt villkor som inte uppfylls i kontraktet.
- Endast det första kontraktsbrottet visas eftersom applikationen avslutas när ett undantag kastas.
- Eftersom kontrakten inte specificeras i form av Javadoc, så krävs extern dokumentation av kontrakten.

3.6 Jcontract

Jcontract är ett verktyg som underlättar kontraktsprogrammering i Java. Kontrakt anges i Javadoc-kommentarerna med speciella taggar. Jcontract har en egen kompilator, *dbc_javac*, vilken tolkar kontraktsspecifikationen i Javadoc:en och omvandlar den till exekverbar kod, vilken infogas i .class-filerna. De resulterande .class-filerna innehåller extra byte-kod som kontrollerar kontrakten under programkörning. Eventuella kontraktsbrott presenteras i ett grafiskt användargränssnitt, i ett konsolfönster eller i en fil enligt konfiguration.

Kommentar [TB11]: Skall vi skriva så eller helt enkelt ”Javadocen” utan kolon?

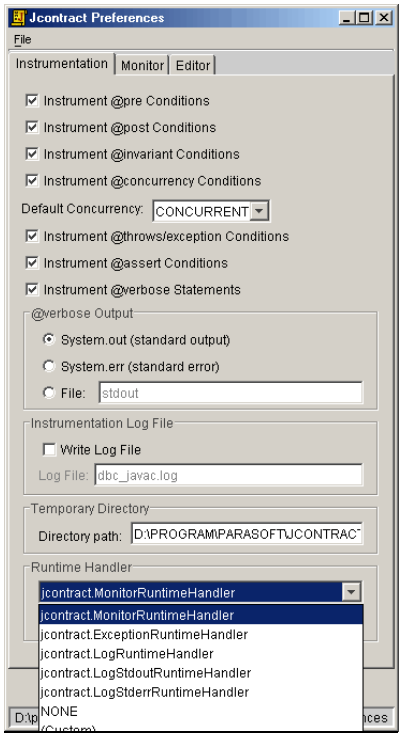
Jcontract använder sig av taggarna *@pre*, *@post* samt *@invariant* för att definiera förvillkor, eftervillkor respektive klassinvarianter i Javadoc:en. Syntaxen påminner om den syntax som beskrivs för OCL [6]. Symbolen *$result* används för att referera till metodens returvärde och symbolen *$pre* används till att referera till objektets tillstånd före anropet av metoden. Jcontract använder även symbolen *$none* för att specificera ”sanna” kontraktsvillkor, det vill säga där inga kontrakt finns eller villkor som alltid är uppfyllda. Genom att skriva *$none* efter kontraktstaggarna så kommer Jcontract att ignorera dessa och därmed genereras inga test för dessa villkor.

För ytterligare information om Jcontract hänvisas läsaren till [19].

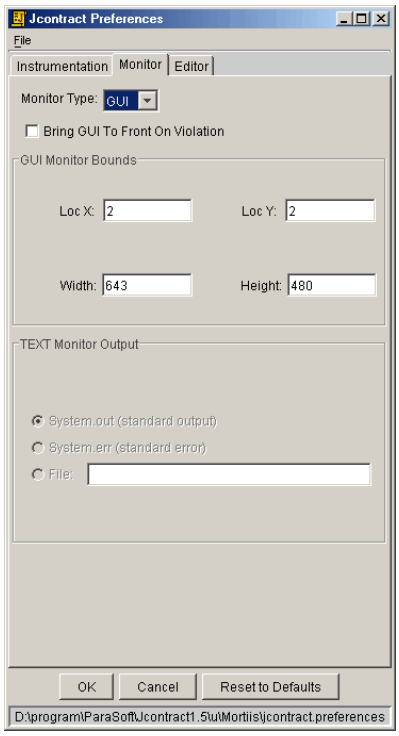
3.6.1 Provkörning av Jcontract

Alla inställningar sker genom ett grafiskt användargränssnitt *Jcontract Preferences*, som i sin tur utgörs av tre vyer. Dessa tre vyer är *Instrumentation*, *Monitor* och *Editor*. I vyn

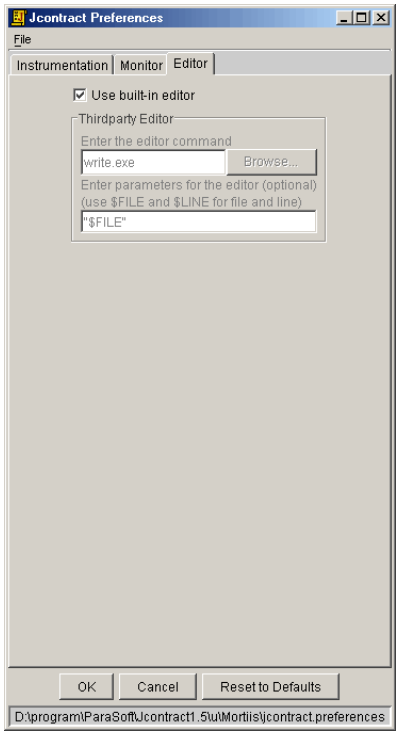
Instrumentation, se Figur 3.5, görs inställningar om vad som skall testas i kontrakten och hur dessa test skall behandlas under exekvering. I vyn *Monitor*, se Figur 3.6, anges om felmeddelanden skall presenteras i textform eller i ett grafiskt användargränssnitt. I vyn *Editor*, se Figur 3.7, så anges vilken editor som skall kopplas samman med Jcontract.



Figur 3.5: Jcontract inställningar - Instrumentation.

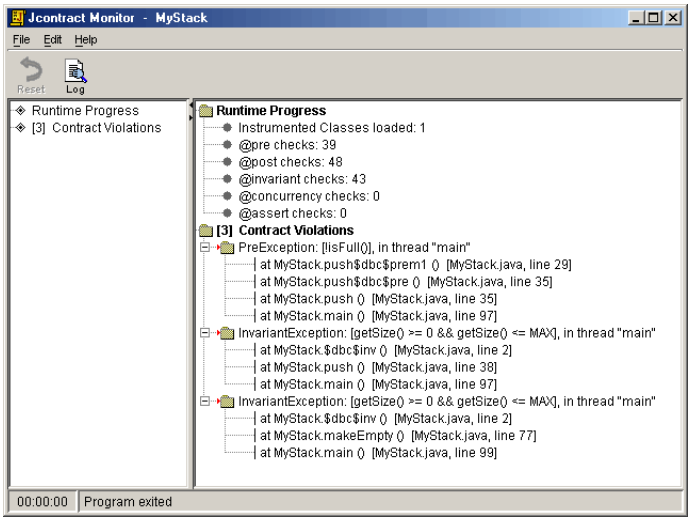


Figur 3.6: Jcontract inställningar - Monitor.

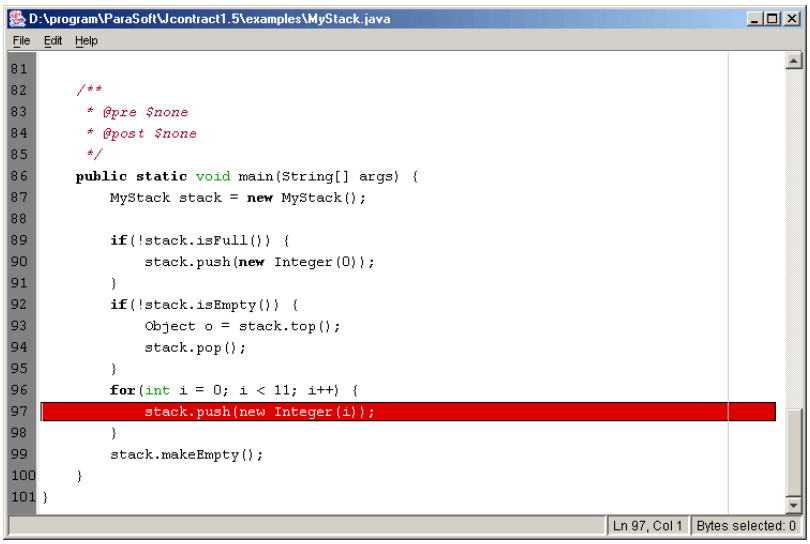


Figur 3.7: Jcontract inställningar - Editor.

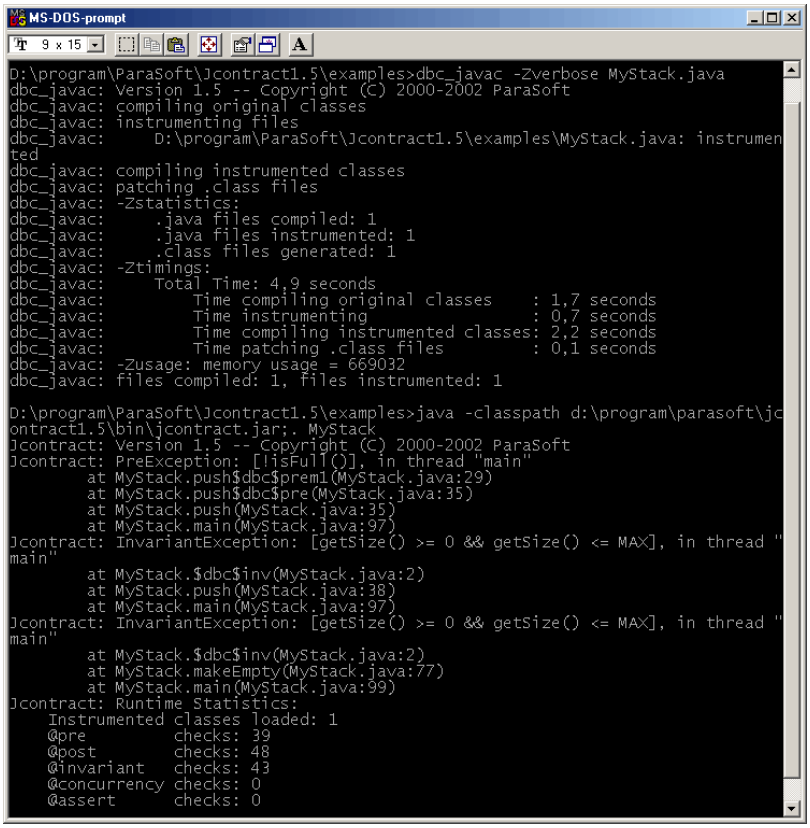
Figur 3.8 visar hur Jcontract beter sig när man exekverar klassen *MyStack*, vilken återfinns i bilaga A.6, i det grafiska monitorläget. I figuren kan utläsas hur många test av respektive villkor som har införts i .class-filen. Man kan också utläsa att applikationen bryter mot förvillkoret *(`!isFull()`)* för metoden *push()*, på rad 97 i main-metoden. Som en följd av detta kontraktsbrott bryts även klassinvarianten för klassen på två ställen i koden, vilket också framgår av figuren. Om man önskar kan man gå direkt till källkoden för att kunna rätta till kontraktsbrottet, vilket visas i Figur 3.9. Jcontract kan även köras så att informationen presenteras direkt i konsolen istället för i ett grafiskt användargränssnitt, vilket illustreras i Figur 3.10. Jcontract kan även köras i exception-läget, vilket innebär att ett undantag kastas för varje kontraktsbrott. Detta innebär att endast det första kontraktsbrottet presenteras, eftersom applikationen avslutas på grund av att detta undantag inte fångas. Detta läge presenteras i Figur 3.11.



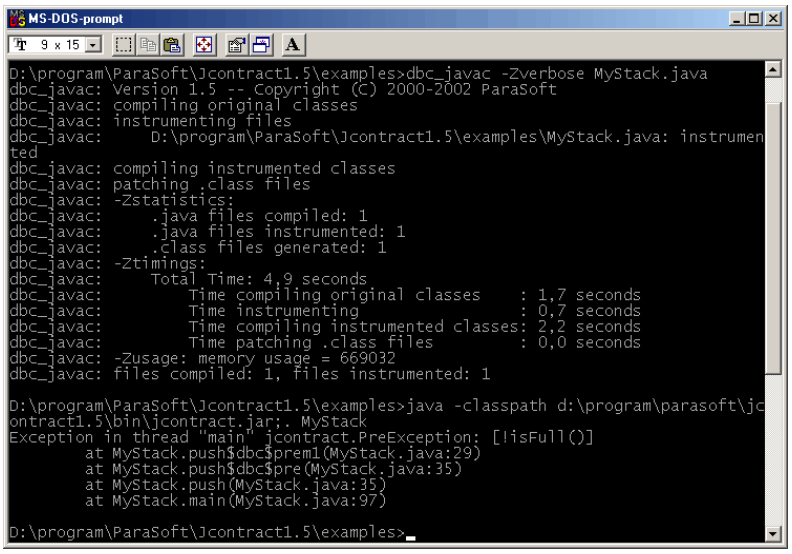
Figur 3.8: Jcontract monitor med GUI.



Figur 3.9: Jcontract editor.



Figur 3.10: Jcontract monitor i konsol.



Figur 3.11: Jcontract med undantag.

3.6.2 Fördelar med Jcontract

- Enkelt att skriva kontrakt eftersom dessa införs i övrig Javadoc för metoden.
- Inställningar är mycket enkla att genomföra genom det grafiska gränssnittet.
- Möjlighet att välja vilka villkor som skall testas i kontrakten.
- Mycket användbar funktion som möjliggör användande av den inbyggda editorn eller valfri editor för att kunna gå direkt till källkoden där villkoret bryts.
- Ett särskilt nyckelord för ”sanna” villkor, det vill säga sådana villkor som alltid är uppfyllda, erbjuds i form av *$none*. Denna deklaration vid en viss kontrakttagg innebär att inget test genereras för detta villkor.
- Mycket bra presentation av kontraktsbrottet genom att de olika villkoren i ett kontrakt särskiljs. Detta innebär att användaren får information om vilket specifikt villkor som inte uppfylls.

3.6.3 Nackdelar med Jcontract

- Den enda egentliga nackdelen med Jcontract är att detta verktyg kräver en licens för att brukas, vilket gör det till ett icke kostnadsfritt verktyg.

3.7 Sammanfattning

Jcontract är det verktyg som fungerade bäst och är det mest lättanvända av de testade verktygen, det är dock det enda av verktygen som kräver licens för att användas vilket kan ses

som en nackdel. Inget av de testade verktygen stöder dock en programmerare vid programmering. De ger endast stöd för kontraktskontroll efter det att programmen och kontrakten redan är skrivna. Detta innebär att testerna inte kan utföras förrän programmen är så pass färdigutvecklade att de kan kompileras och exekveras. Meningen med vårt tilläggs-program är att programmeraren skall få stöd under tiden han/hon programmerar.

4 Idealsystem

Ett idealsystem vore ett system som ger användaren både semantisk och syntaktisk hjälp vid programmering. Detta system skall gå att koppla till olika utvecklingsmiljöer utan att man behöver ändra i systemet alltför mycket. Detta möjliggörs med olika gränssnitt för olika utvecklingsmiljöer.

Här följer kortfattade beskrivningar på idealsystemets funktionalitet. När användaren skall anropa till exempel metoden *getValue()* i klassen *Sifr* (se Figur 4.1) så skall den syntaktiska hjälpen dyka upp när man (i main-metoden i detta fallet) skriver ”sifrInstance.”, det vill säga när man skriver punkten som avgränsar objektet från metoden. Det skall komma fram en lista med alla tänkbara fortsättningar på koden. Det vill säga alla metodnamn som klassen känner till och som den får anropa. Fortsätter man att skriva till exempel ”sifrInstans.ge” så skall listan maska bort de alternativ som inte längre stämmer överens med den inmatade koden. Denna funktionalitet kallas *code completion*.

```
public class Sifr {
    private int value;

    public Sifr(int startVal) {
        value = startVal;
    }

    public int getValue() {
        return value;
    }

    public static void main(String[] args) {
        Sifr sifrInstance = new Sifr(3);
    }
}
```

Figur 4.1: Exempelklassen Sifr.

När väl den syntaktiska hjälpen har kommit upp så skall det även komma upp semantisk hjälp i form av för- och eftervillkor för den metod man anropar. Med hjälp av detta kan programmeraren se direkt om han bryter mot kontrakt som är uppsatta för den anropade metoden. En stor fördel med detta är att en programmerare inte behöver sitta och kontrollera mot ett API vid sidan om utan endast behöver koncentrera sig på programmeringen eftersom all relevant information kommer upp på skärmen. Systemet skall även kunna stödja kompilering och naturligtvis *syntax highlighting*. En programmerare skall kunna infoga förvillkorstest i sin kod automatiskt, för att på så sätt försäkra sig om att uppfylla sin del av kontraktet. Det skulle kunna vara till exempel en *if-sats* på ett exekverbart förvillkor. Även verifiering av att kontraktsbeskrivningarna stämmer överens med implementationen, skall vara möjlig i systemet. Detta leder till att kvalitén på kontraktsbeskrivningarna ökar och att tvetydigheter i specifikationen elimineras samt att implementationsfel upptäcks på ett tidigt stadium. Det skall även finnas möjlighet att testa om programmen uppfyller kontrakten under exekvering, liknande de som beskrivs i kapitel 3. På så sätt minskar risken för att klasser och metoder används på ett felaktigt sätt.

5 Implementation

I detta kapitel behandlas lösningen och implementationen av uppgiften. Motiveringar till de olika valen på lösningen presenteras också. Kapitlet behandlar även hur man möjliggör visning av kontrakt med hjälp av Javadoc-verktyget.

5.1 Inledning

Det finns ett stort antal utvecklingsmiljöer som fungerar klanderfritt. Det känns därför onödigt att ta fram en helt ny utvecklingsmiljö eftersom det redan finns ett flertal sådana att tillgå vilka dessutom har öppen källkod vilket möjliggör förändringar i den befintliga koden. Eftersom detta arbete är tidsbegränsat till endast 20 veckor, så är risken att en egenutvecklad utvecklingsmiljö blir betydligt sämre än redan befintliga utvecklingsmiljöer. Det som detta arbete därför borde koncentrera sig på är själva huvuduppgiften, det vill säga att se till så att resultatet blir en utvecklingsmiljö som ger en programmerare semantisk information i form av för- och eftervillkor, lämpligen baserad på en befintlig utvecklingsmiljö.

5.2 Avgränsning

Arbetet är avgränsat till att endast tillföra ny funktionalitet till en redan befintlig utvecklinsmiljö. Anledningen till detta är att de utvecklingsmiljöer som redan finns tillgängliga är tillräckligt bra och att en helt egenutvecklad utvecklingsmiljö knappast skulle bli lika bra på grund av tidsbrist. Det detta arbete då har koncentrera sig på är att tillföra stöd för semantisk information i form av för- och eftervillkor till en redan befintlig utvecklingsmiljö.

Det finns dock minimikrav som ställs på utvecklingsmiljön. Först och främst så skall den ha öppen källkod och vara tillgänglig för allmänheten. Det skall inte finnas några licensproblem som inskränker en utvecklare att använda sig av källkoden. Den skall ha stöd för *syntax highlighting* och helst *code completion*. Detta för att den skall vara användbar och effektiv att arbeta med.

Andra avgränsningar är också tvungna att göras för att arbetet skall kunna färdigställas inom de uppsatta tidsramarna. Om man utgår ifrån det beskrivna idealsystemet i kapitel 4, så är lösningenavgränsad till att endast innefatta visning av kontrakt samt möjlighet att infoga tester

på förvillkor i koden. Något av det som har avgränsats bort är verifiering av att kontrakt följs under exekvering. Denna avgränsning har skett dels på grund av att det krävs mycket ändringar för att åstadkomma detta, vilket skulle kunna vara en fullgod uppgift för sig, och dels för att det redan finns ett flertal sådana verktyg att tillgå vilket visades i kapitel 3. Huvuduppgiften är just att presentera semantisk information i form av kontrakt, för programmeraren i skrivande stund, inte verifiering av kontrakt.

5.3 Val av implementationsmiljö

Lösningen implementeras i och för programmeringsspråket Java. Det finns en rad anledningar till varför Java har valts:

- Koden är plattformsoberoende och kan därför användas på olika plattformar utan att först modifieras.
- Språket är objektorienterat. Används objektorienteringen korrekt medför det att metoder och klasser har klara avgränsningar. Detta förenklar kontraktsbeskrivningen.
- I Java har man definierat ett sätt att skriva kommentarer i koden på ett strukturerat sätt i form av dokumentationskommentarer, eller mer allmänt kallat Javadoc.
- Man erbjuder ett verktyg att presentera denna dokumentation, Javadoc, i form av till exempel HTML-sidor.
- Dessutom finns även ett flertal kostnadsfria utvecklingsmiljöer för språket, som har öppen källkod, vilket underlättar utvecklingen betydligt.
- I övrigt så finns mycket färdig kod i form av standardbibliotek, vilket gör språket mycket enkelt att använda.

Utvecklingsmiljön, som har valts ut att implementera lösningen i och till, är NetBeans IDE. Anledningen till valet av NetBeans är dels för den öppna källkoden och dels för att redan kraftfulla funktioner som *code completion* och *Javadoc completion* redan finns att tillgå. Detta innebär att allt som krävs runt omkring uppgiften redan finns implementerat och lösningen är därmed mer renodlad till den ursprungliga uppgiften, nämligen att erbjuda kontraktsvisning i en utvecklingsmiljö. Dessutom är NetBeans' källkod skriven i Java, vilket innebär att IDE:n är plattformsoberoende.

Kommentar [TB12]: Behöver kanske förklaras eftersom det är första gången vi nämner det och det nog inte är alltför välkänt utanför NetBeans-världen vad detta är..

5.3.1 Javadoc

Java stöder kommentarer i koden enligt samma konventioner som i C/C++, nämligen enkelradskommentarer genom teckenkombinationen // och flerradskommentarer genom /* */.

Det som står efter enkelradskommentartecknen på samma rad kommer att ignoreras av Javakompilatorn och ej översättas till byte-kod. Det samma gäller för allt som står mellan flerradskommentartecknen också, med den skillnaden att den kan sträcka sig över flera rader.

Java har även infört så kallade dokumentationskommentarer, /** */, vilka har samma funktion som flerradskommentarerna med den skillnad att dessa kommentarer kan tolkas av Javas verktyg *Javadoc*. Genom att dokumentera koden i form av dokumentationskommentarer, eller mer allmänt kallat Javadoc, kan man på ett enkelt sätt skilja ut dokumentationen från koden med hjälp av Javadoc-verktyget, vilket bidrar till att förbättra läsbarheten samt medför att källkoden inte behöver lämnas ut i dokumentationssyfte. Javadoc-verktyget genererar dokumentation i form av till exempel HTML-sidor som beskriver klassen/-erna och dess metoder. En dokumentationskommentar består av en ”huvud”-beskrivning följd av en så kallad taggsektion. ”Huvud”-beskrivningen börjar direkt efter starttecknen för dokumentationskommentaren, /**, och pågår tills den första taggen påträffas. En tagg specificeras med tecknet @ före namnet på taggen. Alla blanktecken och asterisker i början på en rad kommer att filtreras bort liksom nyradstecken. En dokumentationskommentar behöver inte innehålla någon huvudbeskrivning, liksom den ej heller måste innehålla en taggsektion. Alla dokumentationskommentarer placeras direkt före antingen en klass-, interface-, konstruktor-, metod- eller fältdeklaration om de skall kunna tolkas av Javadoc-verktyget. Varje deklaration kan endast ha en dokumentationskommentar, om den har flera så kommer dessa att ignoreras liksom de som inte är placerade enligt ovanstående.
Java har ett flertal fördefinierade taggar vilka finns angivna på [10]. När Javadoc-verktyget stöter på ett taggtecken, så kommer verktyget först att kontrollera om det är en standardtagg som stötts på. Om så är fallet så kommer taggen att bytas ut mot en fördefinierad rubrik i HTML-koden. Om inte taggen är en standardtagg, så kommer den att ignoreras eller alternativt används taggens namn som rubrik i HTML-koden, beroende på vilka inställningar man använder för Javadoc-verktyget. Något som är nytt för Java version 1.4, är att man kan definiera egna taggar, genom valen *–tag* och *–taglet* till Javadoc-verktyget, se [10].

Som sammanfattning kan sägas att Javadoc-verktyget erbjuder en enkel och användbar funktionalitet genom att lyfta ut dokumentationen av koden i separata dokument, som kan vara i form av HTML-sidor. Den stora fördelen med detta verktyg är att man kan erbjuda implementationsdokumentation eller API-dokumentation redan innan koden är färdigutvecklad, eftersom inte Javadoc-verktyget tar någon hänsyn till implementationen, utan

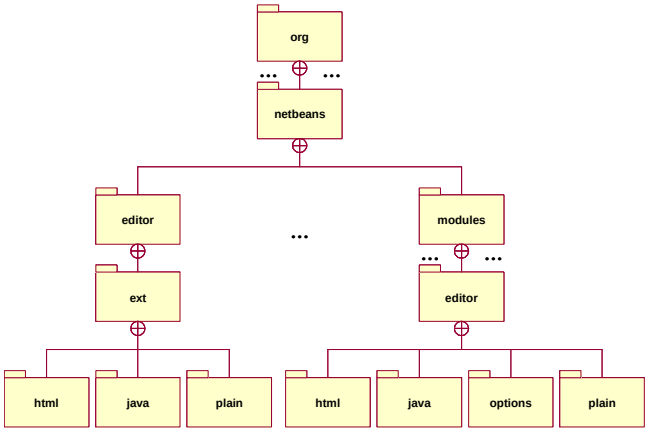
endast använder sig av dokumentationskommentarerna tillsammans med respektive deklaration.

5.4 Editor-modulen i NetBeans

Koden i NetBeans är organiserad i olika moduler för att strukturera upp koden på ett bra sätt. En av dessa moduler är editor-modulen, vilken är den modul där ändringarna införts för att tillhandahålla lösningen. Källkoden för denna modul återfinns i katalogen *editor*, i huvudkatalogen för källkoden, *netbeans-src*. Källkoden till NetBeans IDE:n kan laddas ned från [18].

5.4.1 Filstruktur

Katalogen *editor* innehåller ett flertal filer och underkataloger varav två är *src* och *libsrc*. Dessa två underkataloger innehåller den egentliga källkoden för editorn i NetBeans. *Libsrc*-katalogen innehåller ett bibliotek av klasser som är oberoende av IDE:n och dess implementation. *Src*-katalogen däremot innehåller IDE:ns implementation av editor-modulen, och dessa klasser använder sig bland annat av klasserna i *libsrc*. Filerna i *libsrc* ingår i paketet *org.netbeans.editor* och filerna i *src* ingår i paketet *org.netbeans.modules.editor*. Figur 5.1 visar paketstrukturen.



Figur 5.1: Paketstruktur.

5.5 Design

För att kontraktsvisningen skall kunna möjliggöras så måste man definiera en syntax för hur ett kontrakt skall se ut. Det här arbetet definierar tre nya Javadoc-taggar som skall representera förvillkor, eftervillkor samt klassinvariant.

5.5.1 Kontraktstaggar

Vi har skapat tre nya Javadoc-taggar som representerar förvillkor, eftervillkor samt klassinvariant.

@pre: vid denna tagg så kan förvillkoret för metoden uttryckas antingen i textform som exekverbart uttryck eller på ett formellt sätt. I Figur 5.2 visas ett exempel på användningen av denna tagg.

```
/**
  * …
  * @pre stacken är inte full.
  * …
  */
public void push(Object element)
```

Figur 5.2: Exempel @pre.

@post: vid denna tagg så kan eftervillkoret för metoden uttryckas i antingen textform som exekverbart uttryck eller på ett formellt sätt. I Figur 5.3 visas ett exempel på användningen av *@post*-taggen.

```
/**
  * …
  * @post Objektet element har lagts överst på stacken och storleken
  * på stacken har ökats med ett.
  * …
 */
public void push(Object element)
```

Figur 5.3: Exempel @post.

@invariant: vid denna tagg så kan invarianten för en klass uttryckas i antingen textform som exekverbart uttryck eller på ett formellt sätt. I Figur 5.4 visas ett exempel på användningen av *@invariant*-taggen.

```
/**
   * …
   * @invariant size >= 0 && size <= MAX
   * …
  */
public class MyStack {
```

Figur 5.4: Exempel @invariant.

Anledningen till att just *@pre*, *@post* och *@invariant* har valts som kontraktstaggar, är dels att dessa finns specificerade i OCL och dels för att det redan finns befintliga verktyg som stöder dessa taggar, se kapitel 3. Detta innebär att dessa verktyg då kan användas tillsammans med den tänkta lösningen, vilket ytterligare ökar programmerarens möjligheter att utveckla robusta och pålitliga program.

5.5.2 Syntaxbeskrivning för kontrakt

Det är tillåtet att skriva kontrakten i form av ren text, exekverbar kod eller formella uttryck, eller en kombination av dessa former. Anledningen till denna valfrihet är att kontraktsskrivaren själv skall kunna välja och avgöra vilken form som passar bäst för den givna situationen. I vissa fall kan det vara bra att uttrycka kontrakten i textform för att ge tydliga och ingående förklaringar av dessa. I andra fall kanske ett kontrakt uttryckt i exekverbar kod, eller i ett formellt språk, passar bättre. Det viktiga är inte att kontrakten skrivs efter en viss syntax, utan det viktiga är att de är korrekta och otvetydiga samt överensstämmer med implementationen. En rekommendation är dock att varje villkor skrivs vid en separat tagg. Detta för att det skall vara enklare att urskilja de olika villkoren i kontrakten och på så sätt öka läsbarheten av dessa. Eftersom inga krav på syntaktisk utformning av villkoren i kontrakten ställs, så är det upp till kontraktsskrivaren att själv välja syntax för innehållet. På så sätt kan kontrakten anpassas till att eventuellt användas tillsammans med ett verktyg, till exempel något av de verktyg som nämns i kapitel 3. I Figur 5.5 visas ett exempel på ett kontrakt som blandar exekverbara och icke exekverbara uttryck.

```
/**
  * @pre !isEmpty()
  * @pre Stacken måste innehålla minst ett element.
  * @post Det översta elementet har tagits bort och storleken på stacken
  * har minskats med ett.
  */
    public Object pop()
```

Figur 5.5: Exempel syntaxbeskrivning.

Det exekverbara uttrycket *!isEmpty()* kan på ett enkelt sätt särskiljas från övrig beskrivning i kontraktet, genom att det uttrycks vid en separat tagg. Detta gör det enklare för programmeraren att uppfylla förvillkoret, genom att han/hon nu vet hur förvillkoret skall kunna testas och uppfyllas.

5.6 Realisering

I detta delkapitel beskrivs de olika steg som måste utföras för att kunna realisera implementationen.

5.6.1 Visning av kontrakt i NetBeans

NetBeans IDE:n har en egen uppsättning med standardtaggar och använder sig alltså inte av Javadoc-verktygets standardtaggar när det gäller presentationen av dokumentationen i *Javadoc completion*-fönstret. Därför är man tvungen att lägga till de tidigare presenterade kontraktstaggarna i NetBeans om man vill ha en korrekt visning av kontrakt i IDE:n. Detta möjliggörs genom att modifiera filen *Bundle.properties*, i paketet *org.netbeans.editor*. Genom att här infoga följande rader, så kan kontraktstaggarna tolkas och ersättas med lämpliga rubriker:

javadoc-tag-@pre=Precondition:
javadoc-tag-@post=Postcondition:
javadoc-tag-@invariant=Invariant:

Detta tillägg i filen *Bundle.properties*, innebär att NetBeans kommer att ersätta taggarna *@pre*, *@post*, och *@invariant* mot rubrikerna *Precondition:*, *Postcondition:*, samt *Invariant:*

respektive, när de presenteras i *Javadoc-completion*-fönstret. Denna modifiering innebär att kontraktstaggarna nu kan tolkas av NetBeans.

Eftersom NetBeans sorterar alla taggarna i bokstavsordning så betyder detta att *@post*-taggen kommer presenteras före *@pre*-taggen. Detta blir ett logiskt fel eftersom det som skall vara uppfyllt efter metoden har använts presenteras före skyldigheterna som måste uppfyllas innan man använder metoden. Därför är det nödvändigt att samla ihop och sortera om taggarna, så att de kan presenteras i en logisk ordning. Denna sortering implementeras på ett sådant sätt att kontraktstaggarna placeras överst i taggsektionen, så att dessa är de första som presenteras för en metod. Denna sortering utförs i metoden *sortContractTags()* i klassen *CompletionJavaDoc.java* i paketet *org.netbeans.editor.java*. Dessa modifieringar medför att kontrakt nu skall kunna presenteras i NetBeans IDE:n, på ett tillfredställande sätt.

5.6.2 Inställning av kontraktsvisning i NetBeans

För att kunna styra innehållet i *Javadoc-completion*-fönstret gällande om enbart kontrakt eller all dokumentation skall presenteras, så måste en sådan inställningsmöjlighet införas i IDE:n. Denna inställningsmöjlighet kan erbjudas genom att modifiera ett flertal filer i editor-modulen, nämligen *ExtSettingsDefaults.java*, *ExtSettingsNames.java* och *ExtSettingsInitilizer.java* i paketet *org.netbeans.editor.ext*. Dessa klasser definierar inställningar för editorn, och genom att lägga till parametrar för kontraktsvisningen i dessa, så möjliggörs inställningsmöjligheter för innehållet i *Javadoc-completion*-fönstret. Dessa filer är som tidigare nämnts oberoende av IDE:n och därför måste även de filer som har med konfigurering av inställningar inifrån IDE:n modifieras. För att kunna ändra inställningarna inne i IDE:n så måste även filerna *Bundle.properties*, *JavaOptions.java* och *JavaOptionsBeanInfo.java*, i paketet *org.netbeans.modules.editor.options*, modifieras.

5.6.3 Infoga förvillkorstest i NetBeans

Nästa steg efter att ha infört kontraktsvisning i NetBeans, är att möjliggöra infogande av tester på exekverbara förvillkor i koden. Detta skall fungera på ett sådant sätt att när en metod har ett exekverbart så skall det vara möjligt att infoga ett test på detta förvillkor, för att försäkra sig om att förvillkoret uppfylls före anrop till metoden. Detta innebär att när en *@pre*-tagg finns deklarerad i kontraktet för den anropade metoden, så skall dess uttryck kunna kontrolleras genom en *if-sats* för att försäkra sig om att förvillkoret uppfylls innan anrop sker.

Figur 5.6 visar ett scenario när detta används. Infogandet av det exekverbara förvillkoret resulterar i testet *if(!isFull())*.

```
/**
   * ...
   * @pre !isFull()
   * ...
   */
  public void push(Object element) {
  ...
  }
  ...
  if(!isFull()) {
      push(object);
  }
```

Figur 5.6: Exempel infogning av förvillkorstest.

Detta infogande av förvillkorstest skall antingen kunna väljas från *Javadoc completion*-fönstret, genom en enkel knapptryckning, eller genom ett snabbkommando.

Ett flertal filer måste modifieras för att denna funktionalitet skall kunna erbjudas i NetBeans IDE:n. Själva bearbetningen av kontrakten sker som nämnts tidigare i *CompletionJavaDoc.java* och *NbCompletionJavaDoc.java*, i paketen *org.netbeans.editor.ext* respektive org.*netbeans.modules.editor.java*. Därför måste dessa filer modifieras för att denna funktionalitet skall kunna införas. Även *JDCPopupPanel.java*, *ScrollJavaDocPane.java*, *JavaDocPane.java*, i paketet *org.netbeans.editor.ext*, samt filerna Bundle.*properties* och *NbScrollJavaDocPane.java*, i paketet *org.netbeans.modules.editor.java*, måste modifieras för att kunna infoga test av förvillkor från *Javadoc completion*-fönstret. Detta test införs antingen genom en knapptryckning eller genom ett snabbkommando i *Javadoc completion*-fönstret.

5.6.4 Kompilering

För att nu skapa en IDE-version där tidigare nämnda ändringar brukas, så måste en ny IDE-*build* skapas. Detta görs genom att använda verktyget Ant vilket kan laddas ned gratis från [2]. Ant är Javas motsvarighet till Make i Linux och använder sig av XML-filer på samma sätt som Make använder sig av Make-filer. När man laddar ner källkoden för NetBeans IDE:n så finns dessa XML-filer inkluderade och de har namnet *build.xml*. Det enda man behöver göra är att köra kommandot *ant* så sköter verktyget resten. För att bygga en IDE, gå till mappen

Kommentar [TB13]: Kan man använda nå't svenskt ord för build?

nbbuild, i huvudmappen för källkoden *netbeans-src*, och kör kommandot *ant.*, Resultatet blir en *zip*-fil med namnet *NetBeans-release**versionsnummer-datum**.zip*, vilken är den nya IDE-*build:en*. Sedan är det bara att packa upp denna *zip*-fil och starta IDE:n.

Kommentar [H14]: Hur ska vi skriva detta på ett bra sätt? I vårt fall heter filen NetBeans-release3.5-20030018.zip.

Kommentar [TB15]: NetBeans-release[versionsnummer]-[datum].zip kanske?

5.6.5 Stöd för kontraktstaggarna med Javadoc-verktyget

Om man vill att kontraktstaggarna, som definierades i kapitel 5.5.1, skall stödjas även av Javadoc-verktyget, så finns här flera möjligheter beroende på vilken *JRE*-version (Java Runtime Environment) man använder sig av. *JRE 1.4* har nämligen stöd för egendefinierade taggar genom *–tag* respektive *–taglet* som parametrar till verktyget. Genom att använda *–tag* parametern så kan man här lista upp sina egna taggar samt vilka rubriker som skall ersätta dessa i de genererade HTML-sidorna. Detta sätt kräver att man specificerar de egna taggarna varje gång Javadoc-verktyget används för att få stöd för dessa taggar. I Figur 5.7 visas vilka parametrar som måste ges till Javadoc-verktyget för att detta skall kunna tolka och generera HTML-kod för kontraktstaggarna. Lite förklaring till figuren är på sin plats: *-d* parametern används för att specificera vart de genererade HTML-sidorna skall läggas, *-tag* parametern består av ett uttryck i flera delar. Den första delen anger namnet på taggen det vill säga det namn som följer direkt efter @-symbolen. Nästa del vilken kommer efter det första kolonet, *:*, anger vart i Javadoc:en taggen får förekomma. Som kan ses i figuren så har pre-taggen och post-taggen bokstäverna *c* och *m* satta, vilket innebär att dessa taggar endast får förekomma i konstruktor- och metodbeskrivningar. Invariant-taggen får däremot endast förekomma i typ-beskrivningar såsom klass- och interface-beskrivningar, vilket anges med bokstaven *t*. Sista delen i tagguttrycket är vilket text som skall ersätta taggen i HTML-koden. Något som är viktigt att observera är att taggarna kommer att placeras i den ordning som de anges till Javadoc-verktyget. Det är därför viktigt att pre-taggen anges före post-taggen om man vill att kontraktsbeskrivningen skall ha en logisk ordning.

```
$javadoc –d html-directory –tag pre:cm:”Precondition:” –tag
post:cm:”Postcondition:” –tag:invariant:t:”Invariant:”
klassnamn.java
```

Figur 5.7: Javadoc-tag.

Om man vill slippa att ange egenskaperna för egendefinierade taggar, såsom rubrik och giltig placering i dokumentationen, varje gång man använder sig av Javadoc-verktyget, kan man skapa så kallade *Taglet* för dessa taggar. Dessa filer fungerar på så sätt att de anger vad en tagg har för egenskaper, med andra ord så är det själva implementationen av taggen. I bilaga

B finns tre Taglet-filer, en för varje kontraktstagg vilket kan underlätta när Javadoc:en skall genereras. Figur 5.8 visar ett exempel på hur man kan använda dessa tillsammans med Javadoc-verktyget. Även här så spelar ordningen, i vilken Taglet-filerna anges, roll för hur taggarna placeras i de genererade HTML-sidorna. På samma sätt som tidigare anges vart de genererade HTML-filerna skall placeras med *–d* parametern, *-tagletpath* parametern anger vart de olika Taglet-filerna finns. Denna sökväg kan innehålla flera olika vägar, vilka skiljs åt med kolon, *:*. Vid *–taglet* parametern anges namnet på Taglet-filen.

```
$javadoc –d html-directory –tagletpath path –taglet PreTaglet –
taglet PostTaglet –taglet InvariantTaglet klassnamn.java
```

Figur 5.8: Javadoc-taglet.

Om man använder sig av en äldre *JRE* än 1.4, så finns varken *–tag* eller *–taglet* parametrarna att tillgå. För att här få stöd för egendefinierade taggar krävs att man antingen modifierar de befintliga filerna i *tools.jar* eller skapar en ny så kallad *Doclet*, en mall som används för att bestämma och formatera innehållet i utdatan från Javadoc-verktyget. Denna information finns att tillgå på Suns hemsida [10] liksom övrig information om Javadoc-verktyget. I bilaga B bifogas *Taglet*-filerna för kontraktstaggarna.

5.7 Filer som har modifierats

I detta kapitel följer en kortfattad beskrivning av de filer som har modifieras för att kunna implementera lösningen. Dock bör nämnas att beskrivningarna av de olika filerna är våra egna tolkningar av dessa. Detta på grund av att egentlig dokumentation av dessa filer saknades. Vi hoppas dock att våra beskrivningar till större delen stämmer överens med dessa filers syfte. Även de klassdiagram som presenteras är mycket förenklade genom att kardinaliteter har utelämnats och att arvsrelationerna är ofullständiga. Anledningen till dessa förenklingar är att syftet med dessa klassdiagram inte är att ge en fullständig bild av systemet utan endast en överblick. Kompletta klassdiagram skulle bli allt för stora och oöverskådliga och därmed inte uppfylla vårt syfte, nämligen att ge en översikt över de klasser som berörs av modifieringarna.

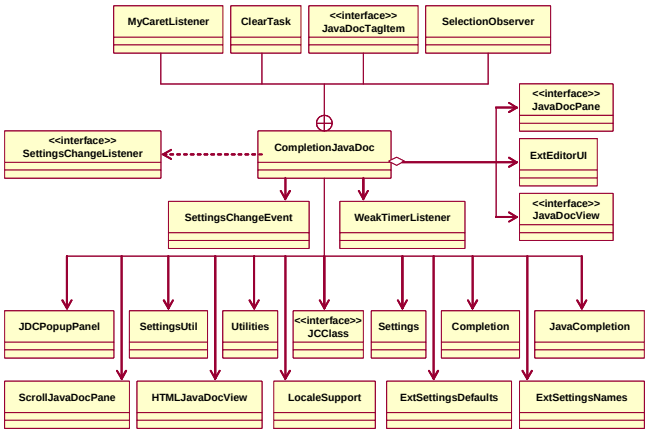
Filerna som har paketprefixet *org.netbeans.editor* återfinns i underkataloger till katalogen *libsrc* och filerna med paketprefixet *org.netbeans.modules.editor* återfinns i underkataloger till katalogen *src*, vilka båda återfinns i *editor*-katalogen i *netbeans-src*. De modifierade filerna har inkluderats i bilaga E.

5.7.1 org.netbeans.editor.Bundle.properties

I denna fil specificeras och definieras bland annat de Javadoc-taggar som NetBeans använder sig av. Denna fil har modifierats så att de tre kontraktstaggarna *@pre*, *@post* samt *@invariant* kan tolkas av NetBeans.

5.7.2 org.netbeans.editor.ext.CompletionJavaDoc.java

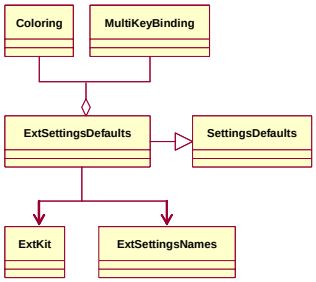
Denna klass är den klass som implementerar *Javadoc completion* funktionaliteten. Det är här som Javadoc:en bearbetas och presenteras i editorn. Ändringarna som har införts här innebär att kontrakten specificerade med kontraktstaggarna kan presenteras på ett tillfredställande sätt. Figur 5.9 visar klassdiagrammet för klassen.



Figur 5.9: CompletionJavaDoc.

5.7.3 org.netbeans.editor.ext.ExtSettingsDefaults.java

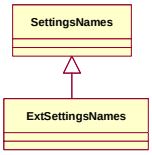
I denna klass anges *default*-inställningarna för editorn, det vill säga de inställningar som ska gälla första gången man använder sig av editorn i NetBeans. Den modifiering vi har infört i denna klass består i att valet för *default*-värdet för visning av endast kontrakt i *Javadoc completion*-fönstret är satt till falskt, det vill säga all dokumentation presenteras i *Javadoc completion*-fönstret. Figur 5.10 visar klassdiagrammet för klassen.



Figur 5.10: ExtSettingsDefaults.

5.7.4 org.netbeans.editor.ext.ExtSettingsNames.java

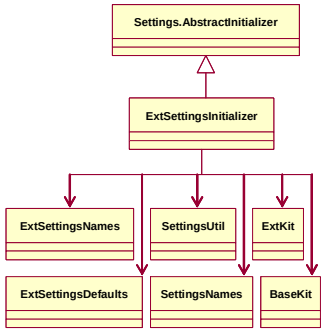
Denna klass innehåller namnen på de olika inställningarna i editorn och när dessa ska visas. Namnet på inställningen av kontraktsvisning har införts i denna klass. Figur 5.11 visar klassdiagrammet för klassen.



Figur 5.11: ExtSettingsNames.

5.7.5 org.netbeans.editor.ext.ExtSettingsInitializer.java

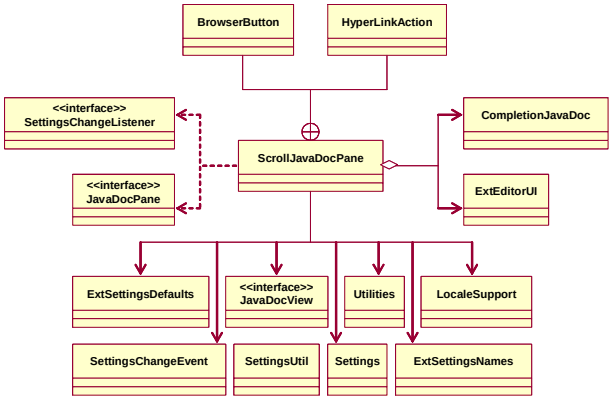
Denna klass används för att initiera inställningarna i editorn. Här har inställningen för visning av kontrakt lagts till. Figur 5.12 visar klassdiagrammet för klassen.



Figur 5.12: ExtSettingsInitializer.

5.7.6 org.netbeans.editor.ext.ScrollJavaDocPane.java

Denna klass är implementationen av ytan på vilken Javadoc:en presenteras i editorn. En metod som möjliggör aktivering/deaktivering av infogande av förvillkorstest har definierats i klassen. Figur 5.13 visar klassdiagrammet för klassen.



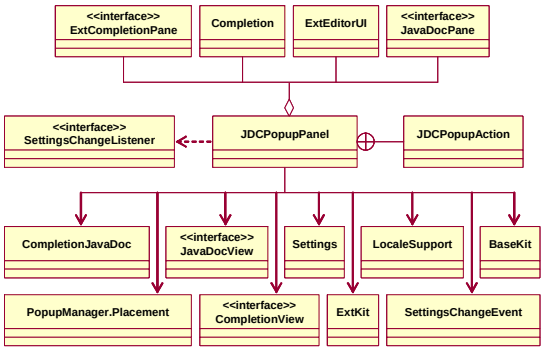
Figur 5.13: ScrollJavaDocPane.

5.7.7 org.netbeans.editor.ext.JavaDocPane.java

Detta är ett interface vilket används av *ScrollJavaDocPane* det vill säga ett interface för ytan på vilken Javadoc:en presenteras. En metod som möjliggör aktivering/deaktivering av infogande av förvillkorstest har deklarerats i interfacet.

5.7.8 org.netbeans.editor.ext.JDCPopupPanel.java

Denna klass representerar ett osynligt fönster vilket innehåller information om *code completion*-fönstret och *Javadoc completion*-fönstret. Modifieringarna som har införts i klassen syftar till att möjliggöra infogande av förvillkorstest genom snabbkommandot *alt+p*. Figur 5.14 visar klassdiagrammet för klassen.



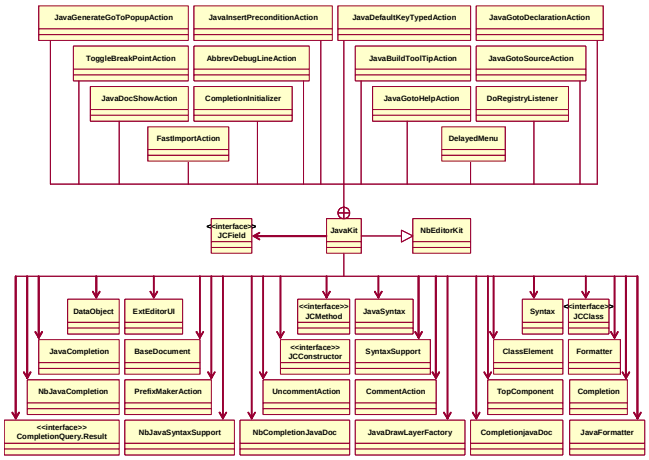
Figur 5.14: JDCPopupPanel.

5.7.9 org.netbeans.modules.editor.java.Bundle.properties

Denna fil innehåller bland annat texter som kan presenteras i IDE:n, som till exempel så kallade *tool-tips* eller *hints*. Filen har modifierats så att ett tips ges om vad knappen för infogning av förvillkorstest i *Javadoc completion*-fönstret gör.

5.7.10 org.netbeans.modules.editor.java.JavaKit.java

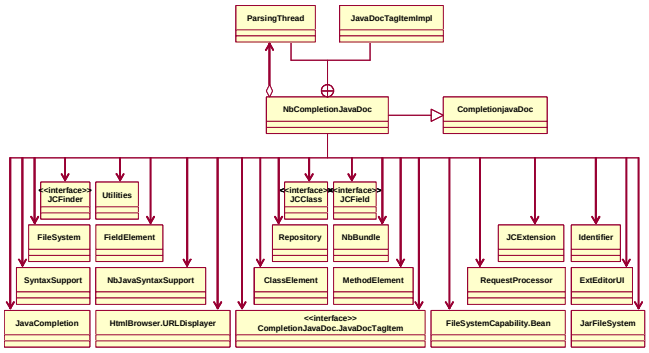
Denna klass används för att tillhandahålla egenskaper som krävs för att en textkomponent skall kunna fungera som en Java-editor. Här definieras en mängd så kallade *actions*, det vill säga operationer, för Java-editorn i NetBeans. Denna klass har ej modifieras utan anledningen till varför den beskrivs i detta kapitel är att den är den centrala klassen i editor-modulen och är därför viktig för översikten av lösningen. Figur 5.15 visar klassdiagrammet för klassen.



Figur 5.15: JavaKit.

5.7.11 org.netbeans.modules.editor.java.NbCompletionJavaDoc.java

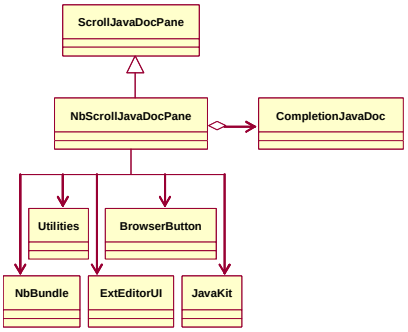
Klassen är IDE:n implementation av *Javadoc completion* funktionaliteten. Denna klass ärver från klassen *CompletionJavaDoc*. I denna klass så har funktionalitet som krävs för att infoga förvillkorstest i koden införts. Figur 5.16 visar klassdiagrammet för klassen.



Figur 5.16: NbCompletionJavaDoc.

5.7.12 org.netbeans.modules.editor.java.NbScrollJavaDocPane.java

Klassen är IDE:ns implementation av *ScrollJavaDocPane* vilken också är dess superklass. I denna klass så har knappen för infogande av förvillkorstest lagts till så att den presenteras i verktygsfältet i *Javadoc completion*-fönstret. Figur 5.17 visar klassdiagrammet för klassen.



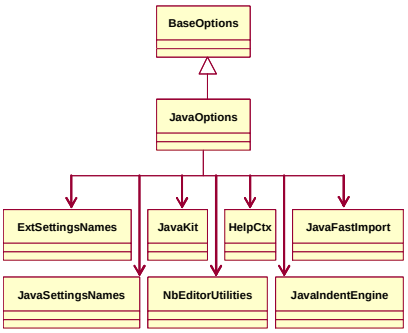
Figur 5.17: NbScrollJavaDocPane.

5.7.13 org.netbeans.modules.editor.options.Bundle.properties

Här finns inställningar och egenskaper för editorerna i IDE:n. Modifieringen av denna fil innebär att man kan styra innehållet som presenteras i *Javadoc completion*-fönstret. Antingen kan man nu välja att endast presentera kontraktstaggarna och deras specifikationer, eller så kan man presentera övrig Javadoc tillsammans med kontraktstaggarna.

5.7.14 org.netbeans.modules.editor.options.JavaOptions.java

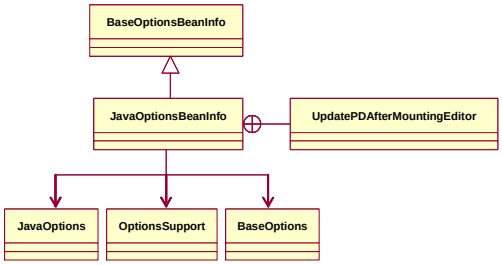
I denna klass definieras en mängd val som gäller för Java-editorn i IDE:n. Modifieringen av denna klass innebär att man får möjlighet att styra innehållet i *Javadoc completion*-fönstret, se kapitel 5.7.13, genom att detta val nu finns att tillgå som en inställning i editorn. Figur 5.18 visar klassdiagrammet för klassen.



Figur 5.18: JavaOptions.

5.7.15 org.netbeans.modules.editor.options.JavaOptionsBeanInfo.java

Denna klass används för att sätta valen i NetBeans IDE:n för Java-editorn. Modifieringen av denna klass möjliggör att värdet för vad som ska visas i *Javadoc completion*-fönstret kan ändras. Figur 5.19 visar klassdiagrammet för klassen.



Figur 5.19: JavaOptionsBeanInfo.

5.8 Sammanfattning

Detta kapitel har visat hur man inför ökat stöd till kontraktsprogrammering i NetBeans IDE genom att tillhandahålla visning av kontrakt och infogande av test på exekverbara förvillkor. I kapitlet har tre nya taggar definierats och deras användningsområde har specificerats för att kunna skriva kontrakt på ett strukturerat sätt i form av Javadoc. Dessa tre taggar är *@pre*, *@post*, samt *@invariant*. Dessa taggar representerar förvillkor, eftervillkor och klassinvarianter respektive. Dessa taggar har sedan införts i NetBeans IDE:n. Även några lösningar på hur man får Javadoc-verktyget att känna igen de egendefinierade kontraktstaggarna, och därmed kunna generera kontraktsbeskrivningar i HTML-format, har presenterats. Till sist har även en kortfattad beskrivning av syfte och ändringar av de filer som modifierats presenterats.

6 Arbetets gång

Arbetet startade med en analys av det tidigare arbete som hade gjorts på universitetet, se kapitel 2.5. Det arbetet hade använt sig av editorn Jipe [13]. Jipe studerades översiktligt och det framgick att Jipe var baserat på en annan editor, jEdit [12].

Koncentrationen lades nu på jEdit. Detta på grund av att det verkade som att utvecklingen av Jipe hade avstannat plus att jEdit verkade ha ett mycket bra stöd för programmering av

tilläggsprogram. jEdit hade inga inskränkande licensavtal och källkoden var öppen. jEdit hade stöd för *syntax highlighting*, det var positivt. Användandet av jEdit skulle dock kräva utveckling av *code completion*. En sådan lösning krävde att man kunde parsa källkoden för att se vilka syntaktiska möjligheter som fanns. Detta var tänkt att lösas genom att använda verktyget JavaCC [9] för att generera en parser. Tilläggsprogrammet skulle tillhandahålla den funktionalitet som projektet krävde. Tilläggsprogrammet skulle kunna parsa koden som skrevs i jEdit och kunna hantera *code completion* och till slut även visning av för- och efter-villkor som skrevs i Javadoc-kommentarerna.

jEdit befanns vara svårt att implementera all funktionalitet som beskrivs i föregående stycke. Något tilläggsprogram blev aldrig realiserat på grund av att den hjälp som jEdit gav för programmering av tilläggsprogram inte var tillräcklig. Beslutet att avbryta arbetet med jEdit fattades och sökandet efter en annan editor med öppen källkod påbörjades. Tyvärr hade redan flera veckors arbete lagts ner på att sätta sig in i jEdit.

En avgränsning måste nu göras för att det skulle vara möjligt att slutföra projektet i tid. Sökandet efter en editor som erbjöd ännu mer funktionalitet som skulle vara till nytta för projektet påbörjades.

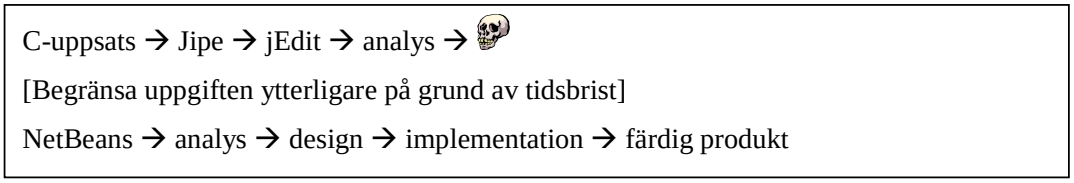
En sökning på Internet gav resultat. NetBeans IDE [18] var en utvecklingsmiljö som hade öppen källkod. Denna IDE erbjöd redan från början kraftfulla funktioner som var tätt sammankopplade med uppgiften. Detta omfattade förutom *syntax highlighting* bland annat *code completion*, vilken erbjuder en syntaktisk hjälp, och *Javadoc completion* vilken erbjuder en semantisk hjälp i form av visning av dokumentationskommentarerna. Dessutom var det en stor fördel att Sun, skaparna av programmeringsspråket Java, var projektsponsorer till IDE eftersom den då alltid skulle anpassas efter eventuella förbättringar i språket. Detta innebar att NetBeans borde vara en av de bästa Java IDE:rna som fanns att tillgå. En annan fördel var att koden var under Sun Public License (se bilaga F) vilket innebar att man kunde få tillgång till koden, modifiera den samt använda den i kommersiella syften om man så önskade. IDE:n var även kostnadsfri. Den nackdel som kunde ses med NetBeans var att denna IDE var väldigt resurskrävande, vilket hade att göra med att den var helt skriven i Java. Detta medförde att IDE:n inte var lika effektiv prestandamässigt, som den skulle kunna vara om dess kod vore anpassad för en specifik plattform. Detta var dock ett mindre problem som kunde lösas rent hårdvarumässigt genom snabbare datorer. Fördelen med att NetBeans var skriven i Java, var

att IDE:n blev plattformsoberoende och därmed kunde köras på ett flertal plattformar utan att koden måste modifieras.

Eftersom mycket användbara hjälpmedel som underlättar programutvecklingen redan fanns implementerade i NetBeans, så föll valet på denna IDE.

En analys av NetBeans' källkod genomfördes. Detta ansågs vara ganska avancerat och tog mycket tid i anspråk. Ett designarbete påbörjades efter analysen av NetBeans' källkod där det bestämdes i stort hur själva implementationen skulle gå till väga. Implementationsfasen påbörjades då den utökade funktionaliteten skulle läggas in i NetBeans IDE.

Figur 6.1 visar arbetets gång mer kortfattat och abstrakt.



Figur 6.1: Arbetets gång - översikt.

7 Resultat

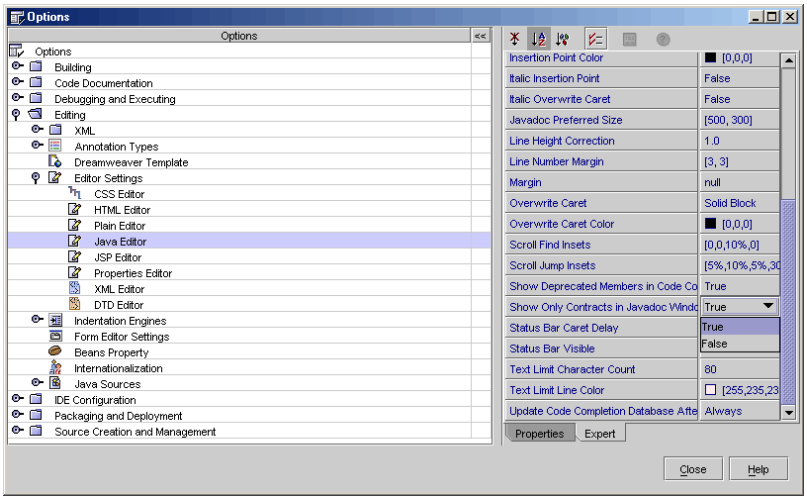
I detta kapitel demonstreras den implementerade lösningen i NetBeans IDE:n. För att demonstrera kontraktsvisningen används de tre klasserna *StackInterface.java*, *StackImpl.java* samt *StackApplication.java* vilka återfinns i bilaga C. Demonstrationen inkluderar bara inställningar och verktyg som har med själva lösningen att göra. Övrigt användande av IDE:n lämnas åt läsaren att själv utforska.

Sist i kapitlet beskrivs även hur IDE:n ytterligare kan anpassas för att bättre stödja användaren vid kontraktsprogrammering.

Se bilaga D för de systemkrav som ställs för att NetBeans IDE skall fungera.

7.1 Demonstration

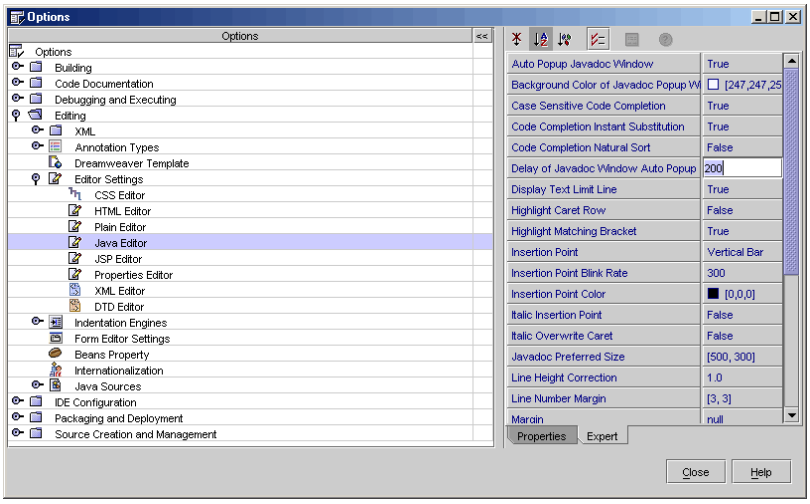
Den första inställning som måste göras är vad som skall presenteras i *Javadoc completion*-fönstret. Vill man att endast kontrakt skall visas eller skall även resterande dokumentation presenteras. Default-inställningen är att all dokumentation skall visas (med undantag för vissa taggar). Denna inställning görs genom att gå in i menyn *Tools->Options*, under *Editing* välj *Java Editor* under *Editor Settings*, gå in på fliken *Expert* och ändra till önskat värde för fältet *Show Only Contracts in Javadoc Window*, se Figur 7.1. Värdet *true* innebär att endast kontraktstaggarna visas, medan värdet *false* innebär att all Javadoc presenteras.



Figur 7.1: Inställning om endast kontrakt skall visas.

Sedan skall man ställa in vilken fördröjning man vill ha för *code completion*-fönstret och *Javadoc completion*-fönstret innan dessa aktiveras. Dessa inställningar görs som tidigare genom att gå in i menyn *Tools->Options*, välj *Java Editor* under *Editor Settings* i huvudkatalogen *Editing*. Sätt önskat fördröjning för *code completion*-fönstret genom att klicka på fliken Properties och ändra värdet för *Delay of Completion Window Auto Popup*. *Javadoc completion*-fönstrets fördröjning sätts under *Expert*-fliken vid värdet *Delay of Javadoc Window Auto Popup*, se Figur 7.2. Anledningen till dessa värden är att man skall kunna kontrollera om de olika fönstren skall komma upp eller ej beroende på hur snabbt man skriver. Om man skriver snabbare än fördröjningstiderna kommer fönstren ej att aktiveras och om man skriver långsammare än dessa tiden så aktiveras fönstren. Detta är användbart när man inte vill ha någon hjälp från fönstren utan vet vad som gäller. Då är det bara att skriva på och man slipper därför se fönstret. Om man däremot är osäker på vad som gäller så kan man

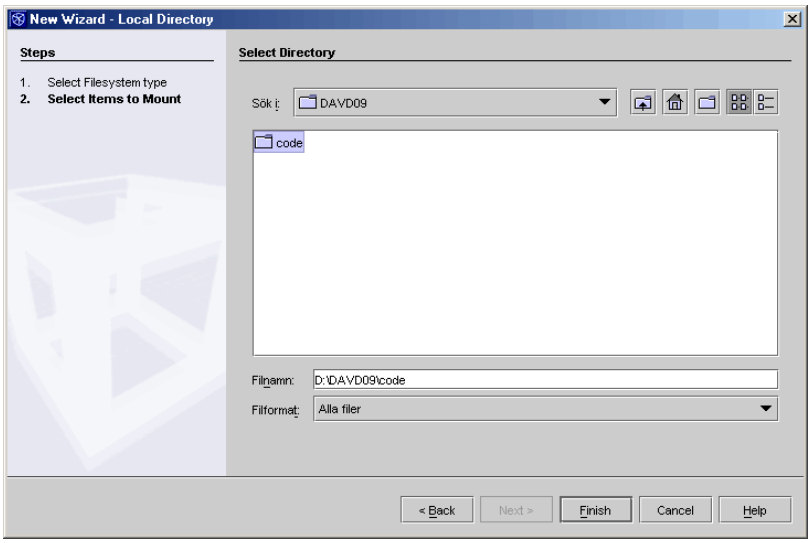
vänta tills fönstren aktiveras för att få den nödvändiga hjälp man behöver. En fördröjning satt till 0 kommer alltså att innebära att fönstret aktiveras omedelbart.



Figur 7.2: Javadoc popup-fönstrets fördröjning.

För att få upp dokumentationen för en typ så måste denna klass vara *mountad*, det vill säga man måste inkludera filsystemet innan den känns igen av IDE:n. Detta görs genom att välja *File->Mount Filesystem* och mappen som skall *mountas*, observera att man väljer mappen i vilken paketstrukturen ligger, se Figur 7.3.

Kommentar [TB16]: Används monterad, monteras osv. på svenska?



Figur 7.3: Mountning av filsystem.

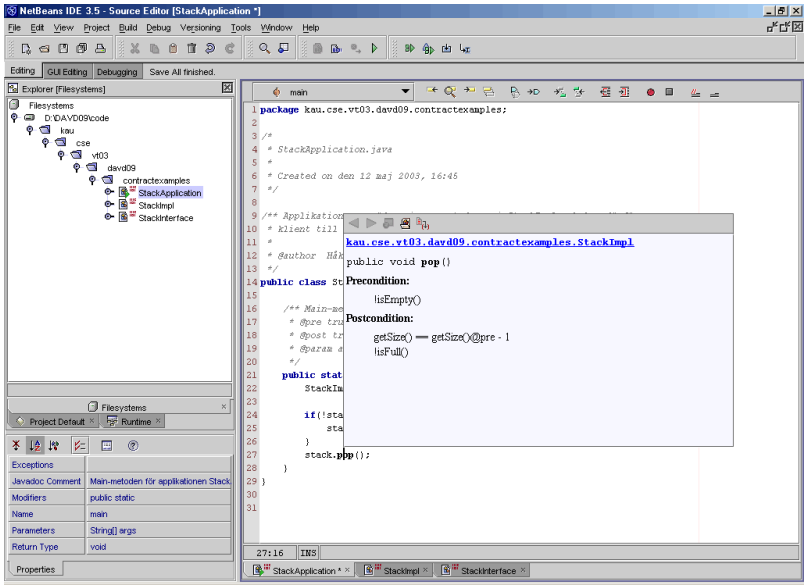
Man kan få upp *Javadoc completion*-fönstret på flera olika sätt:

- Genom att skriva nyckelordet *new* och vänta ut fördröjningen. Detta alternativ leder till att *code completion*-fönstret aktiveras (förutsatt att det inte är deaktiverat) och

efter stund aktiveras även *Javadoc completion*-fönstret. *Javadoc completion*-fönstret uppvisar dokumentationen för den markerade datatypen i *code completion*-fönstret.

- Genom att skriva en punkt efter objektsreferensen och vänta ut fördröjningen. Detta kommer att leda till samma resultat som i den ovanstående punkten.
- Markera eller placera markören över den datatyp man vill se Javadoc:en för och sedan trycka *ctrl+space*. Detta leder till att *code completion*-fönstret omedelbart aktiveras varav sedan även *Javadoc completion*-fönstret aktiveras.
- Markera eller placera markören över den datatyp man vill se Javadoc:en för och sedan trycka *ctrl+shift+space*. Detta leder till att *Javadoc completion*-fönstret omedelbart aktiveras utan att *code completion*-fönstret först aktiveras.

I Figur 7.4 visas ett aktiverat *Javadoc completion*-fönster vilket har aktiverats genom snabbkommandot *ctrl+shift+space*. Fönstret uppvisar kontraktet för metoden *pop()* i klassen *StackImpl*.

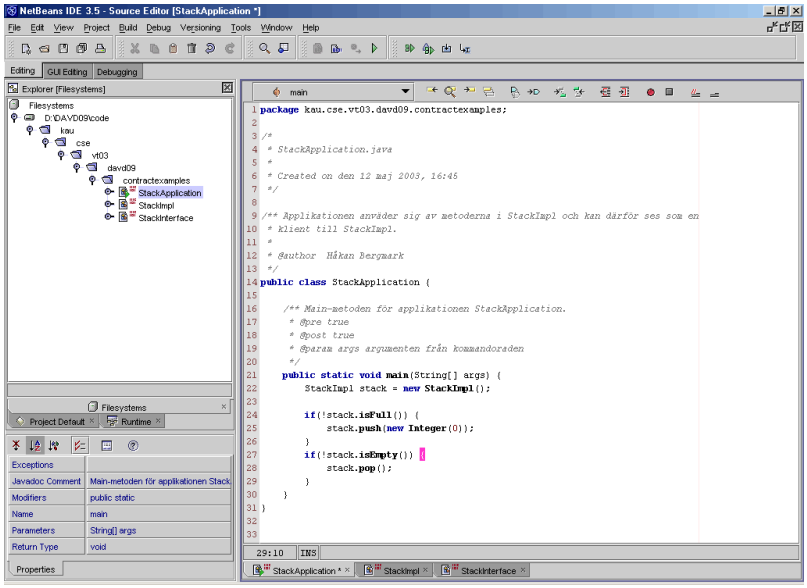


Figur 7.4: Javadoc completion-fönster.

När ett *Javadoc completion*-fönster har aktiverats, så kan förvillkorstest infogas, förutsatt att sådana existerar. Detta kan göras antingen genom att klicka på knappen längst ut till höger i *Javadoc completion*-fönstrets knappmeny (se Figur 7.5) eller genom snabbkommandot *alt+p*. Båda resulterar i att en *if-sats*, innehållande förvillkoret, infogas i koden.

Figur 7.5: Knapp för infogande av förvillkorstest.

I Figur 7.6 ses att i kontraktet för *pop()* anges det exekverbara uttrycket *!isEmpty()* som förvillkor till metoden. I Figur visas att detta förvillkor placerats i en *if-sats* genom att snabbkommandot *alt+p* har använts. Den aktuella objektreferensen, *stack*, har infogats automatiskt framför metodanropet.



Figur 7.6: Förvillkorstest.

7.2 Utvärdering

Resultatet är en utvecklingsmiljö som erbjuder visning av kontrakt vid programmering. Den erbjuder även möjligheten för en programmerare att genom en knapptryckning eller ett snabbkommando infoga test på förvillkoret till en anropad metod. Om man jämför resultatet med det uppsatta idealsystemet (se kapitel 4) så stämmer resultatet överens med idealsystemet på ett flertal punkter. Idealsystemet skulle ha stöd för *syntax highlighting*, *code completion* och kompileringsmöjlighet. Detta hade den utvecklingsmiljö resultatet bygger på stöd för redan från början. Idealsystemet skulle ha stöd för visning av kontrakt vid programmering. Detta stöd har införts i utvecklingsmiljön. Tillsammans med detta så har även möjligheten till infogande av förvillkorstest införts. Detta var också specificerat att ingå i ett idealsystem. Ett idealsystem skall ha möjlighet till att verifiera att uppsatta kontrakt stämmer överens med

implementationen. Detta stöd finns inte i utvecklingsmiljön. Det finns inte heller stöd för att kunna kontrollera att programmen följer uppsatta kontrakt under exekvering, vilket ett idealsystem skulle ha. Dock finns möjligheten att utforma kontrakten på ett sådant sätt att de följer den syntax som något av de testade verktygen i kapitel 3 använder. Då kan man använda sig av ett sådant verktyg i samarbete med utvecklingsmiljön.

Enligt de mål som var uppsatta skulle ett tilläggsprogram till en utvecklingsmiljö göras. Detta tilläggsprogram skulle vara så oberoende av utvecklingsmiljön som möjligt. För att öppna möjligheten till att använda tilläggsprogrammet till andra utvecklingsmiljöer utan att behöva ändra allt för mycket i koden (se kapitel 2.2). Detta tilläggsprogram skulle ha stöd för visning av kontrakt vid programmering. Resultatet blev inte ett tilläggsprogram till en utvecklingsmiljö även om detta var det som författarna strävade mot, se kapitel 6. Resultatet blev dock en fungerande utvecklingsmiljö som erbjuder visning av kontrakt vid programmering. Utvecklingsmiljön är plattformsoberoende, detta är möjligtvis en liten kompensation för att det inte blev ett tilläggsprogram med så svaga band till utvecklingsmiljön som möjligt.

7.3 Framtida arbete

Visningen av kontrakt anses fungera väl och det finns därför ingen anledning till att förändra denna något ytterligare. Däremot så finns vissa brister när det gäller infogandet av förvillkorstest. Ingen skillnad görs här på exekverbara och icke exekverbara uttryck, vilket innebär att allt som står angivet som förvillkor kommer att infogas i testet. Detta betyder att användaren själv måste ta bort de icke exekverbara uttrycken ur testet, för att testet skall vara giltigt. En annan begränsning är att markören måste befinna sig till vänster om öppningsparantesen i metodanropet när *Javadoc completion*-fönstret aktiveras. Om markören befinner sig till höger om denna parantes så kommer inte den korrekta objektreferensen att kunna urskiljas och infogas i testet. Ytterligare en begränsning är att objektreferensen för konstruktor-anrop inte urskiljs och därmed måste användaren själv ange denna i testet.. Ett förslag till hur detta automatiska infogande av förvillkorstest, skulle kunna förbättras är att möjliggöra automatisk identifiering av exekverbara uttryck. Om en sådan funktionalitet finns tillgänglig innebär det att man kan förbättra det automatiska infogandet av förvillkorstestet, till att endast vara möjlig för exekverbara uttryck. Även tillhandahållandet av en funktionalitet som verifierar kontraktsbeskrivningarna mot implementationen kan ses som ett framtida arbete liksom möjligheten att kontrollera om program bryter mot kontrakt under exekvering.

8 Sammanfattning

I detta arbete har en lösning gällande visning av kontrakt vid programmering tagits fram och presenterats till IDE:n NetBeans. Även andra verktyg som förenklar kontraktsprogrammering har presenterats och en beskrivning av kontrakt och deras betydelse för programvaru-utvecklingen har lagts fram. Ett idealsystem har presenterats, detta skulle fungera som något att sträva mot och jämföra med i slutet. Även möjligheten till test av förvillkor har implementerats. En programmerare har möjlighet att infoga test av förvillkoret för den metod han/hon anropar. Denna funktionalitet med infogande av förvillkorstest är inte fullt utvecklad. Den skall därför ses mer som en hjälp än som ett komplett stöd vid programmeringen. Det är tänkt att användaren skall kunna infoga testet på ett enkelt sätt och endast behöva göra mindre modifieringar av testet. Visningen av kontrakt fungerar tillfredställande. Hela nyttan med visning av kontrakt bygger på att kontrakten är korrekta. Detta arbete tar ingen hänsyn till om kontrakten är korrekta eller ej, utan det är upp till kontraktsskrivaren att tillhandahålla bra och korrekta kontrakt.

Referenser

[1] A. V. Aho, R. Sethi, J. D. Ullman. *Compilers: Principles, Techniques and Tools.* Addison Wesley, 1986.

[2] Apache Ant
http://ant.apache.org/
(2003-05-15)

[3] D. Bartetzko, C. Fischer, M. Möller, H. Wehrheim. *Jass – Java with Assertions*. Från *Workshop on Runtime Verification, 2001 held in conjunction with the 13th Conference on Computer Aided Verification, CAV'01*, Paris, Frankrike, juli 2001.

[4] M. Blom. *Semantic Aspects in Software Development*. Licentiatavhandling Karlstads universitet, Karlstad 2002.

[5] A. Diller. *Z: An Introduction to Formal Methods, Second edition.* Wiley & Sons Ltd., 1995.

[6] IBM – OCL
http://www-3.ibm.com/software/awdtools/library/standards/ocl.html
(2003-05-20)

[7] iContract
http://www.reliable-systems.com/tools/iContract/iContract.htm
(2003-05-20)

[8] Jass
http://csd.informatik.uni-oldenburg.de/~jass/index.html
(2003-05-20)

[9] JavaCC
http://www.experimentalstuff.com/Technologies/JavaCC/
(2003-05-21)

[10] Javadoc Tool
http://java.sun.com/j2se/1.4.1/docs/tooldocs/javadoc/index.html
(2003-05-15)

[11] jContractor
http://jcontractor.sourceforge.net/
(2003-05-20)

[12] jEdit
http://www.jedit.org/
(2003-05-20)

[13] Jipe
http://jipe.sourceforge.net/
(2003-05-20)

[14] M. Karaorman, U. Hölzle, J. Bruno. *jContractor: A Reflective Java Library to Support Design By Contract*. Från *In Preceedings of Meta-Level Architectures and Reflection, 2nd International Conference, Reflection '99*, Saint-Malo, Frankrike, juli 1999.

[15] L. Nguyen, E. Tångring. *Javaeditor som stöder för- och eftervillkor*. C-uppsats Karlstads universitet, Karlstad 2000.

[16] R. Kramer. i*Contract the Java Design by Contract™ Tool*. Från *Proceedings of the TOOLS'98 Conference*, Santa Barbara, USA, 1998.

[17] B. Meyer. *Object-oriented Software Construction*. Precentice Hall, 1988.

[18] NetBeans
http://www.netbeans.org/
(2003-05-15)

[19] Parasoft – Jcontract
http://www.parasoft.com/
(2003-05-20)

[20] Programming With Assertions
http://java.sun.com/j2se/1.4.1/docs/guide/lang/assert.html
(2003-05-20)

[21] R. W. Sebasta. *Concepts of Programming Languages*. Addison Wesley, 1999.

A Klasser vid provkörning av verktyg

A.1 MyStack – iContract

```
/**
 * @invariant getSize() >= 0 && getSize() <= MAX
 */
public class MyStack {
    private java.util.Vector myStack;
    private final int MAX = 10;
    private int size;

    /**
     * @pre true
     * @post isEmpty()
     */
    public MyStack() {
        this.myStack = new java.util.Vector();
        this.size = 0;
    }

    /**
     * @pre !isEmpty()
     * @post getSize() == getSize()@pre - 1
     * @post !isFull()
     */
    public void pop() {
        this.myStack.remove(this.getSize() - 1);
        this.size--;
    }

    /**
     * @pre !isFull()
     * @pre e != null
     * @post getSize () == getSize()@pre + 1
     * @post top() == e
     * @post !isEmpty()
     */
    public void push(Object e) {
        this.myStack.add(e);
        this.size++;
    }

    /**
     * @pre !isEmpty()
     * @post return != null
     */
    public Object top() {
```

```
    return this.myStack.get(this.getSize() - 1);
}

/**
 * @pre true
 * @post return == size
 */
public int getSize() {
    return this.size;
}

/**
 * @pre true
 * @post return == (getSize() == 0)
 */
public boolean isEmpty() {
    return this.getSize() == 0;
}

/**
 * @pre true
 * @post return == (getSize() == MAX)
 */
public boolean isFull() {
    return this.getSize() == this.MAX;
}

/**
 * @pre true
 * @post isEmpty()
 */
public void makeEmpty() {
    this.myStack.clear();
    this.size = 0;
}

/**
 * @pre true
 * @post true
 */
public static void main(String[] args) {
    MyStack stack = new MyStack();

    if(!stack.isFull()) {
        stack.push(new Integer(0));
    }
    if(!stack.isEmpty()) {
        Object o = stack.top();
        stack.pop();
    }
    for(int i = 0; i < 11; i++) {
```

```
            stack.push(new Integer(i));
        }
        stack.makeEmpty();
    }
}
```

A.2 MyStack – Automatgenererad av iContract

```
/**
 * @invariant getSize() >= 0 && getSize() <= MAX
 */
public class MyStack {
/*|*/ //#*#---------------------------------------------------------------------------
/*|*/ // Keeps track of calling chain to avoid recursive invariant checks.
/*|*/ // Avoids inv checks in public methods that are called from private ones.
/*|*/ // Stores bookkeeping information -- key: thread, value: call level
/*|*/ protected transient java.util.Hashtable __icl_ = new java.util.Hashtable(1);
/*|*/ // Update bookkeeping of method nesting and check the invariant if appropriate
/*|*/ private synchronized void __inv_check_at_entry__MyStack(Thread thread, String loc)
throws RuntimeException {
/*|*/   // perform lazy initialization after de-serialization (transient __icl_)
/*|*/   if (__icl_ == null) __icl_ = new java.util.Hashtable(1);
/*|*/   if ( !__icl_.containsKey(thread) ) {
/*|*/     __icl_.put(thread, new Integer(1));
/*|*/     __check_invariant____MyStack(loc);
/*|*/   }
/*|*/   else
/*|*/     __icl_.put(thread, new Integer(((Integer)__icl_.get(thread)).intValue()+1));
/*|*/ }
/*|*/ // Update bookkeeping of method nesting and check the invariant if appropriate
/*|*/ private synchronized void __inv_check_at_exit__MyStack(Thread thread, String loc) throws
RuntimeException {
/*|*/   // perform lazy initialization after de-serialization (transient __icl_)
/*|*/   if (__icl_ == null) __icl_ = new java.util.Hashtable(1);
/*|*/   if (((Integer)__icl_.get(thread)).intValue() == 1 ) {
/*|*/     try {
/*|*/       __check_invariant____MyStack(loc);
/*|*/     } finally {
/*|*/       __icl_.remove(thread); // remove from bookkeeping, before checking (resoliant wrt
exceptions)
/*|*/   }}
/*|*/   else
/*|*/     __icl_.put(thread, new Integer(((Integer)__icl_.get(thread)).intValue()-1));
/*|*/ }
/*|*/ // Update bookkeeping of method nesting DO NOT check the invariant (i.e. for non-default
constr.)
/*|*/ private synchronized void __inc_icl_at_entry__MyStack(Thread thread)  {
/*|*/   // perform lazy initialization after de-serialization (transient __icl_)
/*|*/   if (__icl_ == null) __icl_ = new java.util.Hashtable(1);
/*|*/   if ( !__icl_.containsKey(thread) ) {
/*|*/     __icl_.put(thread, new Integer(1));
```

```
/*|*/   }
/*|*/   else
/*|*/     __icl_.put(thread, new Integer(((Integer)__icl_.get(thread)).intValue()+1));
/*|*/ }
/*|*/ // Tests the invariants of the class and its superclasses.
/*|*/ // This method is public (see note below) to give subclasses (potentially in different
packages),
/*|*/ // acccess to the inv of superclasses (and to let the reflection API find the.
/*|*/ // method)
/*|*/ //
/*|*/ public synchronized void __check_invariant____MyStack( String location ) throws
RuntimeException {
/*|*/ try {
/*|*/ if (!(getSize() >= 0 && getSize() <= MAX))
/*|*/   throw new RuntimeException ( location + "error: invariant violated (MyStack): "+
/*|*/   "getSize() >= 0 && getSize() <= MAX");}
/*|*/ catch ( RuntimeException ex ) {
/*|*/   String msg = "";
/*|*/   if (ex.getClass()==RuntimeException.class) { msg = ex.toString(); }
/*|*/   else msg = location + " exception <<"+ex+">> occured while evaluating the class
INVARIANT.";
/*|*/   throw new RuntimeException(msg);}
/*|*/
/*|*/ }
/*|*/ //-------------------------------------------------------------------------------#*#
/*|*/
    private java.util.Vector myStack;
    private final int MAX = 10;
    private int size;

    /**
     * @pre true
     * @post isEmpty()
     */
    public MyStack() {
        this.myStack = new java.util.Vector();

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   __inc_icl_at_entry__MyStack(Thread.currentThread());
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::MyStack()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::MyStack()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::MyStack()
```

```
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:13: error: precondition violated
(MyStack::MyStack()): (/*declared in MyStack::MyStack()*/ (true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:13:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:13:  MyStack::MyStack()): (/*declared in MyStack::MyStack()*/
(true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   this.size = 0;

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(isEmpty()))
/*|*/     throw new RuntimeException ("src/MyStack.java:13: error: postcondition violated
(MyStack::MyStack()): "+
/*|*/     "isEmpty()");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:13:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:13:  MyStack::MyStack()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/   __inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:13:  just before
exit MyStack::MyStack() ");}
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
}

    /**
     * @pre !isEmpty()
     * @post getSize() == getSize()@pre - 1
     * @post !isFull()
     */
    public void pop() {
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:23:  just
after entry MyStack::pop() ");
/*|*/   //-------------------------------------------------------------------------------#*#
```

```
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   int getSize___pre = getSize(); // save getSize()@pre (in MyStack::pop())
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::pop()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (!isEmpty() /* do not check prepassed */ ))   __pre_passed = true; // succeeded
in: MyStack::pop()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::pop()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:23: error: precondition violated
(MyStack::pop()): (/*declared in MyStack::pop()*/ (!isEmpty())) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:23:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:23:  MyStack::pop()): (/*declared in MyStack::pop()*/
(!isEmpty())) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
        this.myStack.remove(this.getSize() - 1);
        this.size--;

/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(getSize() == getSize___pre - 1))
/*|*/     throw new RuntimeException ("src/MyStack.java:23: error: postcondition violated
(MyStack::pop()): "+
/*|*/     "getSize() == getSize()@pre - 1");
/*|*/   if (!(!isFull()))
/*|*/     throw new RuntimeException ("src/MyStack.java:23: error: postcondition violated
(MyStack::pop()): "+
/*|*/     "!isFull()");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
```

```
/*|*/     else txt = "src/MyStack.java:23:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:23:  MyStack::pop()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:23:  just
before exit MyStack::pop() ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
}

    /**
     * @pre !isFull()
     * @pre e != null
     * @post getSize () == getSize()@pre + 1
     * @post top() == e
     * @post !isEmpty()
     */
    public void push(Object e) {
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:35:  just
after entry MyStack::push(java.lang.Object) ");
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   int getSize___pre = getSize(); // save getSize()@pre (in
MyStack::push(java.lang.Object))
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::push(java.lang.Object)
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (!isFull() /* do not check prepassed */ ))   __pre_passed = true; // succeeded
in: MyStack::push(java.lang.Object)
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::push(java.lang.Object)
/*|*/
/*|*/   if ( (__pre_passed && (e != null /*...*/)))   __pre_passed = __pre_passed && true; //
conditionally succeeded in: MyStack::push(java.lang.Object)
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::push(java.lang.Object)
/*|*/   }
/*|*/   if (!__pre_passed) {
```

```
/*|*/     throw new RuntimeException ("src/MyStack.java:35: error: precondition violated
(MyStack::push(java.lang.Object)): (/*declared in MyStack::push(java.lang.Object)*/
(!isFull())) && ((e != null)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:35:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:35:  MyStack::push(java.lang.Object)): (/*declared in
MyStack::push(java.lang.Object)*/ (!isFull())) && ((e != null)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
        this.myStack.add(e);
        this.size++;

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(getSize () == getSize___pre + 1))
/*|*/     throw new RuntimeException ("src/MyStack.java:35: error: postcondition violated
(MyStack::push(java.lang.Object)): "+
/*|*/     "getSize () == getSize()@pre + 1");
/*|*/   if (!(top() == e))
/*|*/     throw new RuntimeException ("src/MyStack.java:35: error: postcondition violated
(MyStack::push(java.lang.Object)): "+
/*|*/     "top() == e");
/*|*/   if (!(!isEmpty()))
/*|*/     throw new RuntimeException ("src/MyStack.java:35: error: postcondition violated
(MyStack::push(java.lang.Object)): "+
/*|*/     "!isEmpty()");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:35:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:35:  MyStack::push(java.lang.Object)";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:35:  just
before exit MyStack::push(java.lang.Object) ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------------#*#
```

```
/*|*/
}

    /**
     * @pre !isEmpty()
     * @post return != null
     */
    public Object top() {
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:44:  just
after entry MyStack::top() ");
/*|*/   //---------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   java.lang.Object __return_value_holder_;
/*|*/   //---------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::top()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (!isEmpty() /* do not check prepassed */ ))   __pre_passed = true; // succeeded
in: MyStack::top()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::top()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:44: error: precondition violated
(MyStack::top()): (/*declared in MyStack::top()*/ (!isEmpty())) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:44:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:44:  MyStack::top()): (/*declared in MyStack::top()*/
(!isEmpty())) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //---------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#---------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //---------------------------------------------------------------------------#*#
/*|*/

/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   /*
return this.myStack.get(this.getSize() - 1);
```

```
/*|*/
/*|*/   __return_value_holder_ =  this.myStack.get(this.getSize() - 1);
/*|*/   //-------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(__return_value_holder_ != null))
/*|*/     throw new RuntimeException ("src/MyStack.java:44: error: postcondition violated
(MyStack::top()): "+
/*|*/     "(/*return*/  this.myStack.get(this.getSize() - 1)) != null");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:44:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:44:  MyStack::top()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:44:  just
before exit MyStack::top() ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   return __return_value_holder_;
/*|*/   //-------------------------------------------------------------------------#*#

}

    /**
     * @pre true
     * @post return == size
     */
    public int getSize() {
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:52:  just
after entry MyStack::getSize() ");
/*|*/   //-------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------
------
/*|*/   int __return_value_holder_;
/*|*/   //-------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::getSize()
/*|*/   if (! __pre_passed ) {
```

```
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::getSize()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::getSize()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:52: error: precondition violated
(MyStack::getSize()): (/*declared in MyStack::getSize()*/ (true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:52:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:52:  MyStack::getSize()): (/*declared in MyStack::getSize()*/
(true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   /*
return this.size;

/*|*/
/*|*/   __return_value_holder_ =  this.size;
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(__return_value_holder_ == size))
/*|*/     throw new RuntimeException ("src/MyStack.java:52: error: postcondition violated
(MyStack::getSize()): "+
/*|*/     "(/*return*/  this.size) == size");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:52:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:52:  MyStack::getSize()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
```

```
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:52:  just
before exit MyStack::getSize() ");
/*|*/   }
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#-----------------------------------------------------------------------------
/*|*/   return __return_value_holder_;
/*|*/   //-----------------------------------------------------------------------------#*#

}

    /**
     * @pre true
     * @post return == (getSize() == 0)
     */
    public boolean isEmpty() {
/*|*/   //#*#-----------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:60:  just
after entry MyStack::isEmpty() ");
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------------
------
/*|*/   boolean __return_value_holder_;
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::isEmpty()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::isEmpty()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::isEmpty()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:60: error: precondition violated
(MyStack::isEmpty()): (/*declared in MyStack::isEmpty()*/ (true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:60:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:60:  MyStack::isEmpty()): (/*declared in MyStack::isEmpty()*/
(true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-----------------------------------------------------------------------------
------
```

```
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   /*
return this.getSize() == 0;

/*|*/
/*|*/   __return_value_holder_ =  this.getSize() == 0;
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(__return_value_holder_ == (getSize() == 0)))
/*|*/     throw new RuntimeException ("src/MyStack.java:60: error: postcondition violated
(MyStack::isEmpty()): "+
/*|*/     "(/*return*/  this.getSize() == 0) == (getSize() == 0)");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:60:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:60:  MyStack::isEmpty()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:60:  just
before exit MyStack::isEmpty() ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   return __return_value_holder_;
/*|*/   //-------------------------------------------------------------------------------#*#

}

    /**
     * @pre true
     * @post return == (getSize() == MAX)
     */
    public boolean isFull() {
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:68:  just
after entry MyStack::isFull() ");
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
```

```
/*|*/   boolean __return_value_holder_;
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::isFull()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::isFull()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::isFull()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:68: error: precondition violated
(MyStack::isFull()): (/*declared in MyStack::isFull()*/ (true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:68:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:68:  MyStack::isFull()): (/*declared in MyStack::isFull()*/
(true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/

/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   /*
return this.getSize() == this.MAX;

/*|*/
/*|*/   __return_value_holder_ =  this.getSize() == this.MAX;
/*|*/   //-----------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(__return_value_holder_ == (getSize() == MAX)))
/*|*/     throw new RuntimeException ("src/MyStack.java:68: error: postcondition violated
(MyStack::isFull()): "+
/*|*/     "(/*return*/  this.getSize() == this.MAX) == (getSize() == MAX)");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
```

```
/*|*/     else txt = "src/MyStack.java:68:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:68:  MyStack::isFull()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:68:  just
before exit MyStack::isFull() ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   return __return_value_holder_;
/*|*/   //-------------------------------------------------------------------------------#*#

}

    /**
     * @pre true
     * @post isEmpty()
     */
    public void makeEmpty() {
/*|*/   //#*#---------------------------------------------------------------------------
/*|*/   this.__inv_check_at_entry__MyStack(Thread.currentThread(),"src/MyStack.java:76:  just
after entry MyStack::makeEmpty() ");
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::makeEmpty()
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::makeEmpty()
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::makeEmpty()
/*|*/   }
/*|*/   if (!__pre_passed) {
/*|*/     throw new RuntimeException ("src/MyStack.java:76: error: precondition violated
(MyStack::makeEmpty()): (/*declared in MyStack::makeEmpty()*/ (true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:76:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:76:  MyStack::makeEmpty()): (/*declared in
MyStack::makeEmpty()*/ (true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
```

```
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   try {
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
        this.myStack.clear();
        this.size = 0;

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(isEmpty()))
/*|*/     throw new RuntimeException ("src/MyStack.java:76: error: postcondition violated
(MyStack::makeEmpty()): "+
/*|*/     "isEmpty()");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:76:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:76:  MyStack::makeEmpty()";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/   /*|*/   //#*#-------------------------------------------------------------------------
------
/*|*/   } finally {
/*|*/     this.__inv_check_at_exit__MyStack(Thread.currentThread(),"src/MyStack.java:76:  just
before exit MyStack::makeEmpty() ");
/*|*/   }
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
}

    /**
     * @pre true
     * @post true
     */
    public static void main(String[] args) {
/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   boolean __pre_passed = false; // true if at least one pre-cond conj. passed.
/*|*/   // checking MyStack::main(java.lang.String[])
/*|*/   if (! __pre_passed ) {
/*|*/   if ( (true /* do not check prepassed */ ))   __pre_passed = true; // succeeded in:
MyStack::main(java.lang.String[])
/*|*/   else
/*|*/     __pre_passed = false; // failed in: MyStack::main(java.lang.String[])
/*|*/   }
/*|*/   if (!__pre_passed) {
```

```
/*|*/     throw new RuntimeException ("src/MyStack.java:85: error: precondition violated
(MyStack::main(java.lang.String[])): (/*declared in MyStack::main(java.lang.String[])*/
(true)) "
/*|*/   ); }}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:85:  exception <<"+ex+">> occured while evaluating PRE-
condition in src/MyStack.java:85:  MyStack::main(java.lang.String[])): (/*declared in
MyStack::main(java.lang.String[])*/ (true)) "
/*|*/   ;
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
        MyStack stack = new MyStack();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
        stack.makeEmpty();

/*|*/   //#*#-------------------------------------------------------------------------------
/*|*/   try {
/*|*/   if (!(true))
/*|*/     throw new RuntimeException ("src/MyStack.java:85: error: postcondition violated
(MyStack::main(java.lang.String[])): "+
/*|*/     "true");}
/*|*/   catch ( RuntimeException ex ) {
/*|*/     String txt = "";  if (ex.getClass()==RuntimeException.class) { txt = ex.toString();;
}
/*|*/     else txt = "src/MyStack.java:85:  exception <<"+ex+">> occured while evaluating
POST-condition in src/MyStack.java:85:  MyStack::main(java.lang.String[])";
/*|*/     throw new RuntimeException(txt);}
/*|*/
/*|*/   //-------------------------------------------------------------------------------#*#
/*|*/
}
}
```

A.3 MyStack – Jass

```
public class MyStack implements Cloneable {
    private java.util.Vector myStack;
```

```
private final int MAX = 10;
private int size;

public MyStack() {
    /** require true; **/
    this.myStack = new java.util.Vector();
    this.size = 0;
    /** ensure isEmpty(); **/
}

public void pop() {
    /** require !isEmpty(); **/
    this.myStack.remove(this.getSize() - 1);
    this.size--;
    /** ensure getSize() == Old.getSize() - 1; !isFull(); **/
}

public void push(Object e) {
    /** require !isFull(); e != null; **/
    this.myStack.add(e);
    this.size++;
    /** ensure getSize() == Old.getSize() + 1; top() == e; !isEmpty(); **/
}

public Object top() {
    /** require !isEmpty(); **/
    return this.myStack.get(this.getSize() - 1);
    /** ensure Result == Old.top(); Result != null; changeonly{}; **/
}

public int getSize() {
    /** require true; **/
    return this.size;
    /** ensure Result == size; changeonly{}; **/
}

public boolean isEmpty() {
    /** require true; **/
    return this.getSize() == 0;
    /** ensure Result == (getSize() == 0); changeonly{}; **/
}

public boolean isFull() {
    /** require true; **/
    return this.getSize() == this.MAX;
    /** ensure Result == (getSize() == MAX); changeonly{}; **/
}

public void makeEmpty() {
    /** require true; **/
    this.myStack.clear();
    this.size = 0;
    /** ensure isEmpty(); changeonly{myStack, size}; **/
}

protected Object clone() {
    Object b = null;
    try {
        b = super.clone();
    }
    catch (CloneNotSupportedException e){;}
    return b;
}

public static void main(String[] args) {
    /** require true; **/
    MyStack stack = new MyStack();

    if(!stack.isFull()) {
        stack.push(new Integer(0));
    }
    if(!stack.isEmpty()) {
        Object o = stack.top();
        stack.pop();
    }
    for(int i = 0; i < 11; i++) {
        stack.push(new Integer(i));
```

```
        }
        stack.makeEmpty();
        /** ensure true; **/
    }
    /** invariant getSize() >= 0; getSize() <= MAX; **/
}
```

A.4 MyStack – Automatgenererad av Jass

```
public class MyStack implements Cloneable {
    private java.util.Vector myStack;
    private final int MAX = 10;
    private int size;

    public MyStack() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "MyStack()", true),
jassParameters);


             /* precondition */
             if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","MyStack()",7,null);
        this.myStack = new java.util.Vector();
        this.size = 0;
             /* postcondition */
             if (!(jassInternal_isEmpty())) throw new
jass.runtime.PostconditionException("MyStack","MyStack()",10,null);
             /* invariant */
             jassCheckInvariant("at end of method MyStack().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "MyStack()", false),
jassParameters);

    }

    public void pop() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "pop()", true),
jassParameters);
```

```
            /* invariant */
            jassCheckInvariant("at begin of method pop().");
            MyStack jassOld = (MyStack)this.clone();
            /* precondition */
            if (!(!jassInternal_isEmpty())) throw new
jass.runtime.PreconditionException("MyStack","pop()",14,null);
        this.myStack.remove(this.getSize() - 1);
        this.size--;
            /* postcondition */
            if (!(jassInternal_getSize()==jassOld.jassInternal_getSize()-1)) throw new
jass.runtime.PostconditionException("MyStack","pop()",17,null);
            if (!(!jassInternal_isFull())) throw new
jass.runtime.PostconditionException("MyStack","pop()",17,null);
            /* invariant */
            jassCheckInvariant("at end of method pop().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "pop()", false),
jassParameters);

    }

    public void push(Object e) {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {new jass.runtime.traceAssertion.Parameter(e)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "push(java.lang.Object)",
true), jassParameters);


            /* invariant */
            jassCheckInvariant("at begin of method push(java.lang.Object).");
            MyStack jassOld = (MyStack)this.clone();
            /* precondition */
            if (!(!jassInternal_isFull())) throw new
jass.runtime.PreconditionException("MyStack","push(java.lang.Object)",21,null);
            if (!(e!=null)) throw new
jass.runtime.PreconditionException("MyStack","push(java.lang.Object)",21,null);
        this.myStack.add(e);
        this.size++;
            /* postcondition */
            if (!(jassInternal_getSize()==jassOld.jassInternal_getSize()+1)) throw new
jass.runtime.PostconditionException("MyStack","push(java.lang.Object)",24,null);
            if (!(jassInternal_top()==e)) throw new
jass.runtime.PostconditionException("MyStack","push(java.lang.Object)",24,null);
            if (!(!jassInternal_isEmpty())) throw new
jass.runtime.PostconditionException("MyStack","push(java.lang.Object)",24,null);
```

```
            /* invariant */
            jassCheckInvariant("at end of method push(java.lang.Object).");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "push(java.lang.Object)",
false), jassParameters);

    }

    public Object top() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "top()", true),
jassParameters);
            java.lang.Object jassResult;

            /* invariant */
            jassCheckInvariant("at begin of method top().");
            MyStack jassOld = (MyStack)this.clone();
            /* precondition */
            if (!(!jassInternal_isEmpty())) throw new
jass.runtime.PreconditionException("MyStack","top()",28,null);
            jassResult = ( this.myStack.get(this.getSize() - 1));
            /* postcondition */
            if (!(jassResult==jassOld.jassInternal_top())) throw new
jass.runtime.PostconditionException("MyStack","top()",30,null);
            if (!(jassResult!=null)) throw new
jass.runtime.PostconditionException("MyStack","top()",30,null);
            if (!(jass.runtime.Tool.referenceEquals(myStack,jassOld.myStack) && MAX ==
jassOld.MAX && size == jassOld.size)) throw new
jass.runtime.PostconditionException("MyStack","top()",-1,"Method has changed old value.");
            /* invariant */
            jassCheckInvariant("before return in method top().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(jassResult)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "top()", false),
jassParameters);


            return jassResult;
    }

    public int getSize() {
```

```
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "getSize()", true),
jassParameters);
              int jassResult;

              /* invariant */
              jassCheckInvariant("at begin of method getSize().");
              MyStack jassOld = (MyStack)this.clone();
              /* precondition */
              if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","getSize()",34,null);
              jassResult = ( this.size);
              /* postcondition */
              if (!(jassResult==size)) throw new
jass.runtime.PostconditionException("MyStack","getSize()",36,null);
              if (!(jass.runtime.Tool.referenceEquals(myStack,jassOld.myStack) && MAX ==
jassOld.MAX && size == jassOld.size)) throw new
jass.runtime.PostconditionException("MyStack","getSize()",-1,"Method has changed old value.");
              /* invariant */
              jassCheckInvariant("before return in method getSize().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(jassResult)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "getSize()", false),
jassParameters);


              return jassResult;
    }

    public boolean isEmpty() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "isEmpty()", true),
jassParameters);
              boolean jassResult;

              /* invariant */
              jassCheckInvariant("at begin of method isEmpty().");
              MyStack jassOld = (MyStack)this.clone();
              /* precondition */
```

```
            if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","isEmpty()",40,null);
            jassResult = ( this.getSize() == 0);
            /* postcondition */
            if (!(jassResult==(jassInternal_getSize()==0))) throw new
jass.runtime.PostconditionException("MyStack","isEmpty()",42,null);
            if (!(jass.runtime.Tool.referenceEquals(myStack,jassOld.myStack) && MAX ==
jassOld.MAX && size == jassOld.size)) throw new
jass.runtime.PostconditionException("MyStack","isEmpty()",-1,"Method has changed old value.");
            /* invariant */
            jassCheckInvariant("before return in method isEmpty().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(jassResult)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "isEmpty()", false),
jassParameters);


            return jassResult;
    }

    public boolean isFull() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "isFull()", true),
jassParameters);
            boolean jassResult;

            /* invariant */
            jassCheckInvariant("at begin of method isFull().");
            MyStack jassOld = (MyStack)this.clone();
            /* precondition */
            if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","isFull()",46,null);
            jassResult = ( this.getSize() == this.MAX);
            /* postcondition */
            if (!(jassResult==(jassInternal_getSize()==MAX))) throw new
jass.runtime.PostconditionException("MyStack","isFull()",48,null);
            if (!(jass.runtime.Tool.referenceEquals(myStack,jassOld.myStack) && MAX ==
jassOld.MAX && size == jassOld.size)) throw new
jass.runtime.PostconditionException("MyStack","isFull()",-1,"Method has changed old value.");
            /* invariant */
            jassCheckInvariant("before return in method isFull().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(jassResult)};
```

```
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "isFull()", false),
jassParameters);


            return jassResult;
    }

    public void makeEmpty() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "makeEmpty()", true),
jassParameters);


            /* invariant */
            jassCheckInvariant("at begin of method makeEmpty().");
            MyStack jassOld = (MyStack)this.clone();
            /* precondition */
            if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","makeEmpty()",52,null);
        this.myStack.clear();
        this.size = 0;
            /* postcondition */
            if (!(jassInternal_isEmpty())) throw new
jass.runtime.PostconditionException("MyStack","makeEmpty()",55,null);
            if (!(MAX == jassOld.MAX)) throw new
jass.runtime.PostconditionException("MyStack","makeEmpty()",-1,"Method has changed old
value.");
            /* invariant */
            jassCheckInvariant("at end of method makeEmpty().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "makeEmpty()", false),
jassParameters);

    }

    protected Object clone() {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
```

```
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "clone()", true),
jassParameters);
              java.lang.Object jassResult;

              /* invariant */
              jassCheckInvariant("at begin of method clone().");
        Object b = null;
        try {
            b = super.clone();
        }
        catch (CloneNotSupportedException e){;}
              jassResult = ( b);
              /* invariant */
              jassCheckInvariant("before return in method clone().");
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(jassResult)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "clone()", false),
jassParameters);


              return jassResult;
    }

    public static void main(String[] args) {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {new jass.runtime.traceAssertion.Parameter(args)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(null, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "main(java.lang.String[])",
true), jassParameters);


              /* precondition */
              if (!(true)) throw new
jass.runtime.PreconditionException("MyStack","main(java.lang.String[])",68,null);
        MyStack stack = new MyStack();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
```

```
        stack.makeEmpty();
             /* postcondition */
             if (!(true)) throw new
jass.runtime.PostconditionException("MyStack","main(java.lang.String[])",82,null);
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(null, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "main(java.lang.String[])",
false), jassParameters);

    }

             /* --- The following methods of class MyStack are generated by JASS --- */


    public int jassInternal_getSize() {
        return this.size;
    }


    public boolean jassInternal_isEmpty() {
        return this.getSize() == 0;
    }


    public boolean jassInternal_isFull() {
        return this.getSize() == this.MAX;
    }


    public Object jassInternal_top() {
        return this.myStack.get(this.getSize() - 1);
    }

             private void jassCheckInvariant(String msg) {
                          if (!(jassInternal_getSize()>=0)) throw new
jass.runtime.InvariantException("MyStack",null,84,"Exception occured "+msg+" (null)");
                          if (!(jassInternal_getSize()<=MAX)) throw new
jass.runtime.InvariantException("MyStack",null,84,"Exception occured "+msg+" (null)");
             }


             protected void finalize () throws java.lang.Throwable {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "finalize()", true),
jassParameters);
```

```
                        super.finalize();
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "finalize()", false),
jassParameters);
            }

            public boolean equals (java.lang.Object par0) {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {new jass.runtime.traceAssertion.Parameter(par0)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "equals(java.lang.Object)",
true), jassParameters);
                        boolean returnValue = super.equals(par0);
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(returnValue)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "equals(java.lang.Object)",
false), jassParameters);
                        return returnValue;
            }

            public java.lang.String toString () {
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true;
jass.runtime.traceAssertion.Parameter[] jassParameters; jassParameters = new
jass.runtime.traceAssertion.Parameter[] {};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "toString()", true),
jassParameters);
                        java.lang.String returnValue = super.toString();
      jass.runtime.traceAssertion.CommunicationManager.internalAction = true; jassParameters =
new jass.runtime.traceAssertion.Parameter[] {new
jass.runtime.traceAssertion.Parameter(returnValue)};
jass.runtime.traceAssertion.CommunicationManager.internalAction = false;
jass.runtime.traceAssertion.CommunicationManager.communicate(this, new
jass.runtime.traceAssertion.MethodReference("null", "MyStack", "toString()", false),
jassParameters);
                        return returnValue;
            }
}
```

A.5 MyStack – jContractor

```
public class MyStack implements Cloneable {
```

```
private java.util.Vector myStack;
private final int MAX = 10;
private int size;
private MyStack OLD;

public MyStack() {
    this.myStack = new java.util.Vector();
    this.size = 0;
}

public void pop() {
    this.myStack.remove(this.getSize() - 1);
    this.size--;
}

public void push(Object e) {
    this.myStack.add(e);
    this.size++;
}

public Object top() {
    return this.myStack.get(this.getSize() - 1);
}

public int getSize() {
    return this.size;
}

public boolean isEmpty() {
    return this.getSize() == 0;
}

public boolean isFull() {
    return this.getSize() == this.MAX;
}

public void makeEmpty() {
    this.myStack.clear();
    this.size = 0;
}

protected boolean MyStack_Precondition() {
    return true;
}

protected boolean MyStack_Postcondition(Void RESULT) {
    return isEmpty();
}

// contract-methods
```

```
protected boolean pop_Precondition() {
    return !isEmpty();
}

protected boolean pop_Postcondition(Void RESULT) {
    return getSize() == OLD.getSize() - 1 && !isFull();
}

protected boolean push_Precondition(Object e) {
    return !isFull() && e != null;
}

protected boolean push_Postcondition(Object e, Void RESULT) {
    return getSize() == OLD.getSize() + 1 && top() == e && !isEmpty();
}

protected boolean top_Precondition() {
    return !isEmpty();
}

protected boolean top_Postcondition(Object RESULT) {
    return RESULT == OLD.top() && RESULT != null;
}

protected boolean getSize_Precondition() {
    return true;
}

protected boolean getSize_Postcondition(int RESULT) {
    return RESULT == size;
}

protected boolean isEmpty_Precondition() {
    return true;
}

protected boolean isEmpty_Postcondition(boolean RESULT) {
    return RESULT == (getSize() == 0);
}

protected boolean isFull_Precondition() {
    return true;
}

protected boolean isFull_Postcondition(boolean RESULT) {
    return RESULT == (getSize() == MAX);
}

protected boolean makeEmpty_Precondition() {
    return true;
}
```

```
    protected boolean makeEmpty_Postcondition(Void RESULT) {
        return isEmpty();
    }

    protected boolean _Invariant() {
        return getSize() >= 0 && getSize() <= MAX;
    }

    //end contract-methods

    protected Object clone() {
        Object b = null;
        try {
            b = super.clone();
        }
        catch (CloneNotSupportedException e){;}
        return b;
    }

    public static void main(String[] args) {
        MyStack stack = new MyStack();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
        stack.makeEmpty();
    }
}
```

A.6 MyStack – Jcontract

```
/**
 * @invariant getSize() >= 0 && getSize() <= MAX
 */
public class MyStack {
    private java.util.Vector myStack;
    private final int MAX = 10;
    private int size;

    /**
     * @pre true
     * @post isEmpty()
     */
```

```
public MyStack() {
    this.myStack = new java.util.Vector();
    this.size = 0;
}

/**
 * @pre !isEmpty()
 * @post getSize() == $pre(int, getSize()) - 1
 * @post !isFull()
 */
public void pop() {
    this.myStack.remove(this.getSize() - 1);
    this.size--;
}

/**
 * @pre !isFull()
 * @pre e != null
 * @post getSize () == $pre(int, getSize()) + 1
 * @post top() == e
 * @post !isEmpty()
 */
public void push(Object e) {
    this.myStack.add(e);
    this.size++;
}

/**
 * @pre !isEmpty()
 * @post $result == $pre(Object, top())
 * @post $result != null
 */
public Object top() {
    return this.myStack.get(this.getSize() - 1);
}

/**
 * @pre true
 * @post $result == size
 */
public int getSize() {
    return this.size;
}

/**
 * @pre true
 * @post $result == (getSize() == 0)
 */
public boolean isEmpty() {
    return this.getSize() == 0;
}
```

```
    /**
     * @pre true
     * @post $result == (getSize() == MAX)
     */
    public boolean isFull() {
        return this.getSize() == this.MAX;
    }

    /**
     * @pre true
     * @post isEmpty()
     */
    public void makeEmpty() {
        this.myStack.clear();
        this.size = 0;
    }

    /**
     * @pre true
     * @post true
     */
    public static void main(String[] args) {
        MyStack stack = new MyStack();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
        stack.makeEmpty();
    }
}
```

B Taglet-filer

B.1 PreTaglet.java

```
package com.sun.tools.doclets.standard.tags;
/*
 * Copyright 2002 Sun Microsystems, Inc. All  Rights Reserved.
 *
```

```
 * Redistribution and use in source and binary forms, with or
 * without modification, are permitted provided that the following
 * conditions are met:
 *
 * -Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *
 * -Redistribution in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in
 *  the documentation and/or other materials provided with the
 *  distribution.
 *
 * Neither the name of Sun Microsystems, Inc. or the names of
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * This software is provided "AS IS," without a warranty of any
 * kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND
 * WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY
 * EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY
 * DAMAGES OR LIABILITIES  SUFFERED BY LICENSEE AS A RESULT OF OR
 * RELATING TO USE, MODIFICATION OR DISTRIBUTION OF THE SOFTWARE OR
 * ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE
 * FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT,
 * SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF
 * THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 *
 * You acknowledge that Software is not designed, licensed or
 * intended for use in the design, construction, operation or
 * maintenance of any nuclear facility.
 */

import com.sun.tools.doclets.Taglet;
import com.sun.javadoc.*;
import java.util.Map;

/**
 * A Taglet representing @pre. It is not an inline tag. The purpose
 * of the @pre tag is to specify preconditions for a data type in a clear
 * and structural manner.
 *
 * @author Håkan Bergmark
 * @author Tony Bergh
 */

public class PreTaglet implements Taglet {

    private static final String NAME = "pre";
```

```
private static final String HEADER = "Precondition:";

/**
 * Return the name of this custom tag.
 */
public String getName() {
    return NAME;
}

/**
 * Will return false since <code>@pre</code>
 * cannot be used in field documentation.
 * @return false since <code>@pre</code>
 * cannot be used in field documentation and true
 * otherwise.
 */
public boolean inField() {
    return false;
}

/**
 * Will return true since <code>@pre</code>
 * can be used in constructor documentation.
 * @return true since <code>@pre</code>
 * can be used in constructor documentation and false
 * otherwise.
 */
public boolean inConstructor() {
    return true;
}

/**
 * Will return true since <code>@pre</code>
 * can be used in method documentation.
 * @return true since <code>@pre</code>
 * can be used in method documentation and false
 * otherwise.
 */
public boolean inMethod() {
    return true;
}

/**
 * Will return false since <code>@pre</code>
 * cannot be used in method documentation.
 * @return false since <code>@pre</code>
 * cannot be used in overview documentation and true
 * otherwise.
 */
public boolean inOverview() {
    return false;
```

```
}

/**
 * Will return false since <code>@pre</code>
 * cannot be used in package documentation.
 * @return false since <code>@pre</code>
 * cannot be used in package documentation and true
 * otherwise.
 */
public boolean inPackage() {
    return false;
}

/**
 * Will return false since <code>@pre</code>
 * cannot be used in type documentation (classes or interfaces).
 * @return false since <code>@pre</code>
 * cannot be used in type documentation and true
 * otherwise.
 */
public boolean inType() {
    return false;
}

/**
 * Will return false since <code>@post</code>
 * is not an inline tag.
 * @return false since <code>@post</code>
 * is not an inline tag.
 */

public boolean isInlineTag() {
    return false;
}

/**
 * Register this Taglet.
 * @param tagletMap  the map to register this tag to.
 */
public static void register(Map tagletMap) {
   PreTaglet tag = new PreTaglet();
   Taglet t = (Taglet) tagletMap.get(tag.getName());
   if (t != null) {
       tagletMap.remove(tag.getName());
   }
   tagletMap.put(tag.getName(), tag);
}

/**
 * Given the <code>Tag</code> representation of this custom
 * tag, return its string representation.
```

```
     * @param tag   the <code>Tag</code> representation of this custom tag.
     */
    public String toString(Tag tag) {
        return "<DT><B>" + HEADER + "</B><DD>"
               + "<table cellpadding=2 cellspacing=0>"
               + tag.text()
               + "</table></DD>\n";
    }

    /**
     * Given an array of <code>Tag</code>s representing this custom
     * tag, return its string representation.
     * @param tags  the array of <code>Tag</code>s representing of this custom tag.
     */
    public String toString(Tag[] tags) {
        if (tags.length == 0) {
            return null;
        }
        String result = "\n<DT><B>" + HEADER + "</B><DD>";
        result += "<table cellpadding=2 cellspacing=0>";
        for (int i = 0; i < tags.length; i++) {
            if (i > 0) {
                result += "<BR>";
            }
            result += tags[i].text();
        }
        return result + "</table></DD>\n";
    }
}
```

B.2 PostTaglet.java

```
package com.sun.tools.doclets.standard.tags;
/*
 * Copyright 2002 Sun Microsystems, Inc. All  Rights Reserved.
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, are permitted provided that the following
 * conditions are met:
 *
 * -Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *
 * -Redistribution in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in
 *  the documentation and/or other materials provided with the
 *  distribution.
 *
 * Neither the name of Sun Microsystems, Inc. or the names of
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
```

```
 *
 * This software is provided "AS IS," without a warranty of any
 * kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND
 * WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY
 * EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY
 * DAMAGES OR LIABILITIES  SUFFERED BY LICENSEE AS A RESULT OF OR
 * RELATING TO USE, MODIFICATION OR DISTRIBUTION OF THE SOFTWARE OR
 * ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE
 * FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT,
 * SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF
 * THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 *
 * You acknowledge that Software is not designed, licensed or
 * intended for use in the design, construction, operation or
 * maintenance of any nuclear facility.
 */

import com.sun.tools.doclets.Taglet;
import com.sun.javadoc.*;
import java.util.Map;

/**
 * A Taglet representing @post. It is not an inline tag. The purpose
 * of the @post tag is to specify postconditions for a data type in a clear
 * and structural manner.
 *
 * @author Håkan Bergmark
 * @author Tony Bergh
 */

public class PostTaglet implements Taglet {

    private static final String NAME = "post";
    private static final String HEADER = "Postcondition:";

    /**
     * Return the name of this custom tag.
     */
    public String getName() {
        return NAME;
    }

    /**
     * Will return false since <code>@post</code>
     * cannot be used in field documentation.
     * @return false since <code>@post</code>
     * cannot be used in field documentation and true
     * otherwise.
```

```
 */
public boolean inField() {
    return false;
}

/**
 * Will return true since <code>@post</code>
 * can be used in constructor documentation.
 * @return true since <code>@post</code>
 * can be used in constructor documentation and false
 * otherwise.
 */
public boolean inConstructor() {
    return true;
}

/**
 * Will return true since <code>@post</code>
 * can be used in method documentation.
 * @return true since <code>@post</code>
 * can be used in method documentation and false
 * otherwise.
 */
public boolean inMethod() {
    return true;
}

/**
 * Will return false since <code>@post</code>
 * cannot be used in method documentation.
 * @return false since <code>@post</code>
 * cannot be used in overview documentation and true
 * otherwise.
 */
public boolean inOverview() {
    return false;
}

/**
 * Will return false since <code>@post</code>
 * cannot be used in package documentation.
 * @return false since <code>@post</code>
 * cannot be used in package documentation and true
 * otherwise.
 */
public boolean inPackage() {
    return false;
}

/**
 * Will return false since <code>@post</code>
```

```
 * cannot be used in type documentation (classes or interfaces).
 * @return false since <code>@post</code>
 * cannot be used in type documentation and true
 * otherwise.
 */
public boolean inType() {
    return false;
}

/**
 * Will return false since <code>@post</code>
 * is not an inline tag.
 * @return false since <code>@post</code>
 * is not an inline tag.
 */

public boolean isInlineTag() {
    return false;
}

/**
 * Register this Taglet.
 * @param tagletMap  the map to register this tag to.
 */
public static void register(Map tagletMap) {
   PostTaglet tag = new PostTaglet();
   Taglet t = (Taglet) tagletMap.get(tag.getName());
   if (t != null) {
       tagletMap.remove(tag.getName());
   }
   tagletMap.put(tag.getName(), tag);
}

/**
 * Given the <code>Tag</code> representation of this custom
 * tag, return its string representation.
 * @param tag   the <code>Tag</code> representation of this custom tag.
 */
public String toString(Tag tag) {
    return "<DT><B>" + HEADER + "</B><DD>"
           + "<table cellpadding=2 cellspacing=0>"
           + tag.text()
           + "</table></DD>\n";
}

/**
 * Given an array of <code>Tag</code>s representing this custom
 * tag, return its string representation.
 * @param tags  the array of <code>Tag</code>s representing of this custom tag.
 */
public String toString(Tag[] tags) {
```

```
        if (tags.length == 0) {
            return null;
        }
        String result = "\n<DT><B>" + HEADER + "</B><DD>";
        result += "<table cellpadding=2 cellspacing=0>";
        for (int i = 0; i < tags.length; i++) {
            if (i > 0) {
                result += "<BR>";
            }
            result += tags[i].text();
        }
        return result + "</table></DD>\n";
    }
}
```

B.3 InvariantTaglet.java

```
package com.sun.tools.doclets.standard.tags;
/*
 * Copyright 2002 Sun Microsystems, Inc. All  Rights Reserved.
 *
 * Redistribution and use in source and binary forms, with or
 * without modification, are permitted provided that the following
 * conditions are met:
 *
 * -Redistributions of source code must retain the above copyright
 *  notice, this list of conditions and the following disclaimer.
 *
 * -Redistribution in binary form must reproduce the above copyright
 *  notice, this list of conditions and the following disclaimer in
 *  the documentation and/or other materials provided with the
 *  distribution.
 *
 * Neither the name of Sun Microsystems, Inc. or the names of
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 *
 * This software is provided "AS IS," without a warranty of any
 * kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND
 * WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY
 * EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY
 * DAMAGES OR LIABILITIES  SUFFERED BY LICENSEE AS A RESULT OF OR
 * RELATING TO USE, MODIFICATION OR DISTRIBUTION OF THE SOFTWARE OR
 * ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE
 * FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT,
 * SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER
 * CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF
 * THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN
 * ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
 *
```

```
 * You acknowledge that Software is not designed, licensed or
 * intended for use in the design, construction, operation or
 * maintenance of any nuclear facility.
 */

import com.sun.tools.doclets.Taglet;
import com.sun.javadoc.*;
import java.util.Map;

/**
 * A Taglet representing @invariant. It is not an inline tag. The purpose
 * of the @invariant tag is to specify invariants for a data type in a
 * clear and structural manner.
 *
 * @author Håkan Bergmark
 * @author Tony Bergh
 */

public class InvariantTaglet implements Taglet {

    private static final String NAME = "invariant";
    private static final String HEADER = "Invariant:";

    /**
     * Return the name of this custom tag.
     */
    public String getName() {
        return NAME;
    }

    /**
     * Will return false since <code>@invariant</code>
     * cannot be used in field documentation.
     * @return false since <code>@invariant</code>
     * cannot be used in field documentation and true
     * otherwise.
     */
    public boolean inField() {
        return false;
    }

    /**
     * Will return false since <code>@invariant</code>
     * cannot be used in constructor documentation.
     * @return false since <code>@invariant</code>
     * cannot be used in constructor documentation and true
     * otherwise.
     */
    public boolean inConstructor() {
        return false;
    }
```

```
/**
 * Will return false since <code>@invariant</code>
 * cannot be used in method documentation.
 * @return false since <code>@invariant</code>
 * cannot be used in method documentation and true
 * otherwise.
 */
public boolean inMethod() {
    return false;
}

/**
 * Will return false since <code>@invariant</code>
 * cannot be used in overview documentation.
 * @return false since <code>@invariant</code>
 * cannot be used in overview documentation and true
 * otherwise.
 */
public boolean inOverview() {
    return false;
}

/**
 * Will return false since <code>@invariant</code>
 * cannot be used in package documentation.
 * @return false since <code>@invariant</code>
 * cannot be used in package documentation and true
 * otherwise.
 */
public boolean inPackage() {
    return false;
}

/**
 * Will return true since <code>@invariant</code>
 * can be used in type documentation (classes or interfaces).
 * @return true since <code>@invariant</code>
 * can be used in type documentation and false
 * otherwise.
 */
public boolean inType() {
    return true;
}

/**
 * Will return false since <code>@post</code>
 * is not an inline tag.
 * @return false since <code>@post</code>
 * is not an inline tag.
 */
```

```
    public boolean isInlineTag() {
        return false;
    }

    /**
     * Register this Taglet.
     * @param tagletMap  the map to register this tag to.
     */
    public static void register(Map tagletMap) {
       InvariantTaglet tag = new InvariantTaglet();
       Taglet t = (Taglet) tagletMap.get(tag.getName());
       if (t != null) {
           tagletMap.remove(tag.getName());
       }
       tagletMap.put(tag.getName(), tag);
    }

    /**
     * Given the <code>Tag</code> representation of this custom
     * tag, return its string representation.
     * @param tag   the <code>Tag</code> representation of this custom tag.
     */
    public String toString(Tag tag) {
        return "<DT><B>" + HEADER + "</B><DD>"
               + "<table cellpadding=2 cellspacing=0>"
               + tag.text()
               + "</table></DD>\n";
    }

    /**
     * Given an array of <code>Tag</code>s representing this custom
     * tag, return its string representation.
     * @param tags  the array of <code>Tag</code>s representing of this custom tag.
     */
    public String toString(Tag[] tags) {
        if (tags.length == 0) {
            return null;
        }
        String result = "\n<DT><B>" + HEADER + "</B><DD>";
        result += "<table cellpadding=2 cellspacing=0>";
        for (int i = 0; i < tags.length; i++) {
            if (i > 0) {
                result += "<BR>";
            }
            result += tags[i].text();
        }
        return result + "</table></DD>\n";
    }
}
```

C Klasser för provkörning av NetBeans IDE

Följande klasser används för testning av den modifierade versionen av NetBeans IDE.

C.1 StackInterface.java

```
package kau.cse.vt03.davd09.contractexamples;

/** Interfacet deklarerar alla nödvändiga metoder för en stack.
 */
public interface StackInterface {

    /** Tar bort det översta elementet på stacken.
     * @pre !isEmpty()
     * @post getSize() == getSize()@pre - 1
     * @post !isFull()
     */
    public void pop();

    /** Lägger till ett element på stacken.
     * @pre !isFull()
     * @pre e != null
     * @post top() == e
     * @post getSize() == getSize()@pre + 1
     * @post !isEmpty()
     * @param e elementet som ska läggas överst på stacken.
     */
    public void push(Object e);

    /** Returnerar det översta elementet på stacken.
     * @pre !isEmpty()
     * @post result == top()@pre
     * @return det översta elementet på stacken.
     */
    public Object top();

    /** Returnerar storleken på stacken, det vill säga antal element på stacken.
     * @pre true
     * @post storleken på stacken har returnerats
     * @return storleken på stacken.
     */
    public int getSize();

    /** Berättar om stacken är tom eller ej.
     * @pre true
     * @post true har returnerats om stacken var tom, annars har false returnerats
     * @return true returneras om stacken är tom, annars returneras false.
     */
    public boolean isEmpty();
```

```
    /** Berättar om stacken är full eller ej.
     * @pre true
     * @post true har returnerats om stacken var full, annars har false returnerats
     * @return true returneras om stacken är full, annars returneras false.
     */
    public boolean isFull();

    /** Tömmer stacken på alla element.
     * @pre true
     * @post isEmpty()
     */
    public void makeEmpty();
}
```

C.2 StackImpl.java

```
package kau.cse.vt03.davd09.contractexamples;

/** Denna klass implementerar interfacet StackInterface.
 * @invariant getSize() >= 0 && getSize() <= MAX. Storleken
 * på stacken kan endast anta ett värde mellan 0 och och MAX.
 */
public class StackImpl implements StackInterface {

    private java.util.Vector stack;
    private final int MAX = 10;
    private int size;

    /** Konstruktor för StackImpl-klassen.
     * @pre true
     * @post getSize() == 0
     */
    public StackImpl() {
        this.stack = new java.util.Vector();
        this.size = 0;
    }

    /** Tar bort det översta elementet på stacken.
     * @pre !isEmpty()
     * @post getSize() == getSize()@pre - 1
     * @post !isFull()
     */
    public void pop() {
        this.stack.remove(this.getSize() - 1);
        this.size--;
    }

    /** Lägger till ett element på stacken.
     * @pre !isFull()
     * @pre e != null
```

```
 * @post top() == e
 * @post getSize() == getSize()@pre + 1
 * @post !isEmpty()
 * @param e elementet som ska läggas överst på stacken.
 */
public void push(Object e) {
    this.stack.add(e);
    this.size++;
}

/** Returnerar det översta elementet på stacken.
 * @pre !isEmpty()
 * @post result == top()@pre
 * @post result != null
 * @return returnerar det översta elementet på stacken.
 */
public Object top() {
    return this.stack.get(this.getSize() - 1);
}

/** Returnerar storleken på stacken, det vill säga antal element på stacken.
 * @pre true
 * @post result == size
 * @return storleken på stacken.
 */
public int getSize() {
    return this.size;
}

/** Berättar om stacken är tom eller ej.
 * @pre true
 * @post result == (getSize() == 0)
 * @return true returneras om getSize() == 0, annars returneras false.
 */
public boolean isEmpty() {
    return this.getSize() == 0;
}

/** Berättar om stacken är full eller ej.
 * @pre true
 * @post result == (getSize() == MAX)
 * @return true returneras om getSize() == MAX, annars returneras false.
 */
public boolean isFull() {
    return this.getSize() == this.MAX;
}

/** Tömmer stacken på alla element.
 * @pre true
 * @post isEmpty()
 */
```

```
    public void makeEmpty() {
        this.stack.clear();
        this.size = 0;
    }
}
```

C.3 StackApplication.java

```
package kau.cse.vt03.davd09.contractexamples;

/** Applikationen använder sig av metoderna i StackImpl och kan därför ses som en
 * klient till StackImpl.
 */
public class StackApplication {

    /** Main-metoden för applikationen StackApplication.
     * @pre true
     * @post true
     * @param args argumenten från kommandoraden
     */
    public static void main(String[] args) {
        StackImpl stack = new StackImpl();

        if(!stack.isFull()) {
            stack.push(new Integer(0));
        }
        if(!stack.isEmpty()) {
            Object o = stack.top();
            stack.pop();
        }
        for(int i = 0; i < 11; i++) {
            stack.push(new Integer(i));
        }
        stack.makeEmpty();
    }
}
```

D Systemkrav

NetBeans	Version 3.5	
Java	Version 1.4	
Hårdvara	Sekundärminne	125 Mb
	Primärminne	256 Mb
	Processor	PIII 500 MHz (Windows och Linux)

		Ultra 10 (Solaris)
		500 MHz Alpha (Open VMS)
Operativsystem	Windows 95, 98, ME, NT 4.0, 2000 och XP	
	Solaris 8 och Solaris 9	
	Linux – alla distributioner	
	OS/2	
	Open VMS 7.2-1 eller nyare	
	Mac OS X 10.1.1 eller nyare	
	HP-UX	

Tabell D.1: Systemkrav.

E Källkod

Denna bilaga innehåller källkoden till de filer i NetBeans IDE:n som har modifierats för att realisera lösningen. Modifikationer och tillägg i källkoden är markerade med fet text. Filerna lyder under Sun Public License, se bilaga F.

E.1 org.netbeans.editor.Bundle.properies

```
#                 Sun Public License Notice
#
# The contents of this file are subject to the Sun Public License
# Version 1.0 (the "License"). You may not use this file except in
# compliance with the License. A copy of the License is available at
# http://www.sun.com/
#
# The Original Code is NetBeans. The Initial Developer of the Original
# Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
# Microsystems, Inc. All Rights Reserved.
#
# Contributor(s): Håkan Bergmark, Tony Bergh

# Editor ResourceBundle properties file

# Editor Actions
annotations-cycling=Annotations Cycling
abbrev-debug-line=Debug Filename and Line Number
abbrev-expand=Expand Abbreviation
abbrev-reset=Reset Abbreviation
toggle-breakpoint=Toggle Breakpoint
```

```
add-watch=New Watch...
adjust-caret-bottom=Move Insertion Point to Bottom
adjust-caret-center=Move Insertion Point to Center
adjust-caret-top=Move Insertion Point to Top
adjust-window-bottom=Scroll Insertion Point to Bottom
adjust-window-center=Scroll Insertion Point to Center
adjust-window-top=Scroll Insertion Point to Top
beep=Beep
bookmark-next=Next Bookmark
bookmark-toggle=Toggle Bookmark
brace-code-select=Select Code Between Braces
bracket-match=Match Bracket
build-popup-menu=Build Popup Menu
pd-file-not-found-title=Code Completion DB file missing.
pd-file-not-found=Code Completion DB file {0} was not found.
build-tool-tip=Build Tool Tip
caret-backward=Insertion Point Backward
caret-begin-line=Insertion Point to Beginning of Text on Line
caret-line-first-column=Insertion Point to Beginning of Line
caret-begin-word=Insertion Point to Beginning of Word
caret-begin=Insertion Point to Beginning of Document
caret-down=Insertion Point Down
caret-end-line=Insertion Point to End of Line
caret-end-word=Insertion Point to End of Word
caret-end=Insertion Point to End of Document
caret-forward=Insertion Point Forward
caret-next-word=Insertion Point to Next Word
caret-previous-word=Insertion Point to Previous Word
caret-up=Insertion Point Up
completion-hide=Hide Code Completion
completion-show=Show Code Completion
comment=Comment
copy-to-clipboard=Copy
cut-to-clipboard=Cut
default-typed=Default Typed
delete-next=Delete Next Character
delete-previous=Delete Previous Character
escape=Simulate Escape Key
fast-import=Fast Import
find-next=Find Next Occurrence
find-previous=Find Previous Occurrence
find-selection=Find Selection
find=Find
first-non-white=Go to First Non-whitespace Char
format=Reformat Code
goto-declaration=Declaration
goto-help=Go to Help
goto-source=Go to Source
toggle-toolbar=Toggle Toolbar
goto=Go to Line
insert-break=Insert Newline
```

```
insert-content=Insert Content
insert-date-time=Insert Current Date and Time
insert-tab=Insert Tab
javadoc-show-action=Show the JavaDoc Window
jdc-popup-show=Show Completion and JavaDoc Window
jdc-popup-hide=Hide Completion and JavaDoc Window
jcompletion-hide=Hide the JCompletion Window
jcompletion-show-help=Show the JCompletion Window
jump-list-next=Jump to Next Edit
jump-list-next-component=Jump to Next Edit Cross Window
jump-list-prev=Jump to Previous Edit
jump-list-prev-component=Jump to Previous Edit Cross Window
last-non-white=Go to Last Non-whitespace Char
make-getter=Replace Variable With its Getter
make-is=Replace Variable With its is* Method
make-setter=Replace Variable With its Setter
match-brace=Match Brace
page-down=Page Down
page-up=Page Up
paste-from-clipboard=Paste
popup-remove-selection=Delete
redo=Redo
remove-line=Delete Line
remove-line-begin=Delete Preceding Characters in Line
remove-selection=Delete Selection
remove-tab=Delete Tab
remove-word=Delete Word
replace=Replace
scroll-bottom=Scroll to Bottom
scroll-center=Scroll to Center
scroll-down=Scroll Down
scroll-top=Scroll to Top
scroll-up=Scroll Up
select-all=Select All
select-identifier=Select Identifier
select-line=Select Line
select-next-parameter=Select Next Parameter
select-word=Select Word
selection-backward=Extend Selection Backward
selection-begin-line=Extend Selection to Beginning of Text on Line
selection-line-first-column=Extend Selection to Beginning of Line
selection-begin-word=Extend Selection to Beginning of Word
selection-begin=Extend Selection to Beginning of Document
selection-bracket-match=Extend Selection to Matching Bracket
selection-down=Extend Selection Down
selection-end-line=Extend Selection to End of Line
selection-end-word=Extend Selection to End of Word
selection-end=Extend Selection to End of Document
selection-first-non-white=Extend Selection to First Non-whitespace Char
selection-forward=Extend Selection Forward
selection-last-non-white=Extend Selection to Last Non-whitespace Char
```

```
selection-match-brace=Extend Selection to Matching Brace
selection-next-word=Extend Selection to Next Word
selection-page-down=Extend Selection to Next Page
selection-page-up=Extend Selection to Previous Page
selection-previous-word=Extend Selection Back
selection-up=Extend Selection Up
set-dot=Set Dot
set-read-only=Set Read Only
set-writable=Set Writable
shift-insert-break=Substitute Code Completion Text
shift-line-left=Shift Line Left
shift-line-right=Shift Line Right
show-popup-menu=Show Popup Menu
start-macro-recording=Start Macro Recording
stop-macro-recording=Stop Macro Recording
switch-case=Switch Case
to-lower-case=To Lowercase
to-upper-case=To Uppercase
toggle-case-identifier-begin=Switch Capitalization of Identifier
toggle-case-word-begin=Switch Capitalization of Word
toggle-highlight-search=Toggle Highlight Search
toggle-line-numbers=Toggle Line Numbers
toggle-typing-mode=Toggle Typing Mode
uncomment=Uncomment
undo=Undo
unselect=Unselect
word-match-next=Next Matching Word
word-match-prev=Previous Matching Word

# Macro related stuff
macro-recording=Recording

#MacroDialogSupport texts
MDS_title=Recorded Macro
MDS_ok=OK
MDS_cancel=Cancel
ACSD_MDS_ok=
ACSD_MDS_cancel=

#MacroSavePanel [PENDING: Clean up the class to not contain defaults]
MSP_Name=Name:
MSP_Name_Mnemonic=N
ACSD_MSP_Name=
MSP_Macro=Macro:
MSP_Macro_Mnemonic=M
ACSD_MSP_Macro=
MSP_Keys=Keybindings:
MSP_Keys_Mnemonic=K
ACSD_MSP_Keys=
MSP_Add=Add...
MSP_Add_Mnemonic=A
```

```
MSP_AddToolTip=Add a keybinding for this macro
MSP_Remove=Remove
MSP_Remove_Mnemonic=R
MSP_RemoveToolTip=Remove a keybinding from this macro
MSP_AddTitle=Add Keybinding
ACSD_MSP=

#MacroSavePanel.KeySequenceRequester
MSP_ok=OK
MSP_cancel=Cancel
MSP_clear=Clear
MSP_clear_Mnemonic=C
ACSD_MSP_ok=
ACSD_MSP_cancel=
ACSD_MSP_clear=
MSP_FMT_Collision=This shortcut sequence conflicts with sequence <{0}> bound to "{1}".
MSP_Collision=This shortcut sequence is already bound to this Macro.

#KeySequenceInputPanel
LBL_KSIP_Sequence=Shortcut Sequence:
LBL_KSIP_Sequence_Mnemonic=S
ACSD_LBL_KSIP_Sequence=
ACSD_KSIP=

#FindSupport related stuff
find-found=found at
find-not-found=not found
find-wrap-start=End of document reached. Continuing search from beginning.
find-wrap-end=Beginning of document reached. Continuing search from end.
find-items-replaced={0} of {1} items replaced

#FindDialogPanel
find-what=Find What:
find-what-mnemonic=F
ACSD_find-what=
find-replace-with=Replace With:
find-replace-with-mnemonic=l
ACSD_find-replace-with=
find-highlight-search=\ Highlight Search
find-highlight-search-mnemonic=t
find-highlight-search-tooltip=Highlights occurrences in the text.
find-inc-search=\ Incremental Search
find-inc-search-mnemonic=I
find-inc-search-tooltip=Tries to find text as you type.
find-match-case=\ Match Case
find-match-case-mnemonic=M
find-match-case-tooltip=Matches search text only to text with the same capitalization.
find-smart-case=\ Smart Case
find-smart-case-mnemonic=S
find-smart-case-tooltip=Matches case if at least one character in the searched text is
uppercase.
```

```
find-whole-words=\ Match Whole Words Only
find-whole-words-mnemonic=W
find-whole-words-tooltip=Matches search text only to whole words in the document.
find-backward-search=\ Backward Search
find-backward-search-mnemonic=B
find-backward-search-tooltip=Searches backward from the current position of the insertion
point.
find-wrap-search=\ Wrap Search
find-wrap-search-mnemonic=p
find-wrap-search-tooltip=Continues search from the beginning or end of the document.
find-reg-exp=\ Regular Expressions
find-reg-exp-mnemonic=E
find-reg-exp-tooltip=Finds occurrences of a regular expression in the text.
ACSD_find=

#FindDialogSupport
find-title=Find
replace-title=Replace
find-button-find=Find
find-button-find-mnemonic=D
find-button-replace=Replace
find-button-replace-mnemonic=R
find-button-replace-all=Replace All
find-button-replace-all-mnemonic=A
find-button-cancel=Close
ACSD_find-button-find=
ACSD_find-button-replace=
ACSD_find-button-replace-all=
ACSD_find-button-cancel=

# GotoDialogPanel
goto-line=Go to Line:
goto-line-mnemonic=L
ACSD_goto-line=
ACSD_goto=

# GotoDialogSupport
goto-title=Go to Line
goto-button-goto=Goto
goto-button-goto-mnemonic=G
goto-button-cancel=Close
ACSD_goto-button-goto=
ACSD_goto-button-cancel=

#Status bar INS/OVR labels
status-bar-insert=INS
status-bar-overwrite=OVR

#Utilities.debugPosition
wrong_position=Wrong position
```

```
#Guarded document exception formats
FMT_guarded_insert=Attempt to insert into guarded block at position {0}.
FMT_guarded_remove=Attempt to remove from guarded block at position {0}.

#GlyphGutter
cycling-glyph_tooltip=Multiple annotations here [{0}] - click to cycle
ACSN_Glyph_Gutter=Glyph Gutter
ACSD_Glyph_Gutter=Component for displaying line numbers and annotation glyphs

line-numbers-menuitem=Show Line Numbers
generate-gutter-popup=Margin

#Utilities
key-prefix-numpad=NumPad-

ACSN_CompletionView=Code Completion
ACSD_CompletionView=Code Completion Window

javadoc-loading=Searching...

#JavaDoc Tags
javadoc-tag-@param=Parameters:
javadoc-tag-@see=See Also:
javadoc-tag-@exception=Throws:
javadoc-tag-@return=Returns:
javadoc-tag-@since=Since:
javadoc-tag-@author=Author:
javadoc-tag-@deprecated=Deprecated:
javadoc-tag-@throws=Throws:
javadoc-tag-@version=Version:
javadoc-tag-@pre=Precondition:
javadoc-tag-@post=Postcondition:
javadoc-tag-@invariant=Invariant:

#Javadoc accessibility
ACSD_JAVADOC_javaDocPane=


## ext.Completion.java
ext.Completion.wait=Please wait...

#---
```

E.2 org.netbeans.editor.ext.ExtSettingsNames.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
```

```
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.editor.ext;

import org.netbeans.editor.SettingsNames;

/**
* Names of the extended editor settings.
*
* @author Miloslav Metelka
* @version 1.00
*/

public class ExtSettingsNames extends SettingsNames {

    /** List of the action names that should be shown in the popup menu.
    * Null name means separator.
    * Values: java.util.List containing java.lang.String instances
    */
    public static final String POPUP_MENU_ACTION_NAME_LIST = "popup-menu-action-name-list"; //
NOI18N

    /** List of the action names that should be shown in the popup menu
     * when JEditorPane is shown in the dialogs. It corresponds
     * Null name means separator.
     * Values: java.util.List containing java.lang.String instances
     */
    public static final String DIALOG_POPUP_MENU_ACTION_NAME_LIST
    = "dialog-popup-menu-action-name-list"; // NOI18N

    /** Whether popup menu will be displayed on mouse right-click or not.
     * It's set to true by default.
     * Values: java.lang.Boolean
     */
    public static final String POPUP_MENU_ENABLED = "popup-menu-enabled"; // NOI18N

    /** Highlight the row where the caret currently is. The ExtCaret must be used.
    * Values: java.lang.Boolean
    */
    public static final String HIGHLIGHT_CARET_ROW = "highlight-caret-row"; // NOI18N

    /** Highlight the matching brace (if the caret currently stands after the brace).
    * The ExtCaret must be used.
    * Values: java.lang.Boolean
```

```
    */
    public static final String HIGHLIGHT_MATCH_BRACE = "highlight-match-brace"; // NOI18N

    /** Coloring used to highlight the row where the caret resides */
    public static final String HIGHLIGHT_CARET_ROW_COLORING = "highlight-caret-row"; // NOI18N

    /** Coloring used to highlight the matching brace */
    public static final String HIGHLIGHT_MATCH_BRACE_COLORING = "highlight-match-brace"; //
NOI18N

    /** Delay (milliseconds) after which the matching brace
    * will be updated. This is intended to eliminate flicker
    * if the user holds the arrow key pressed.
    */
    public static final String HIGHLIGHT_MATCH_BRACE_DELAY = "highlight-match-brace-delay"; //
NOI18N

    /** Whether the fast and simple matching should be used for higlighting
    * the matching brace. Its disadvantage is that it doesn't ignore the comments
    * and string and character constants in the search.
    */
    public static final String CARET_SIMPLE_MATCH_BRACE = "caret-simple-match-brace"; //
NOI18N

    /** Whether the code completion window should popup automatically.
    * Values: java.lang.Boolean
    */
    public static final String COMPLETION_AUTO_POPUP = "completion-auto-popup"; // NOI18N

    /** Whether the code completion query search will be case  sensitive
    * Values: java.lang.Boolean
    */
    public static final String COMPLETION_CASE_SENSITIVE = "completion-case-sensitive"; //
NOI18N

    /** Whether the code completion sorting will be natural
    * Values: java.lang.Boolean
    */
    public static final String COMPLETION_NATURAL_SORT = "completion-natural-sort"; // NOI18N

    /** Whether perform instant substitution, if the search result contains only one item
    * Values: java.lang.Boolean
    */
    public static final String COMPLETION_INSTANT_SUBSTITUTION = "completion-instant-
substitution"; // NOI18N

    /** The delay after which the completion window is shown automatically.
    * Values: java.lang.Integer
    */
    public static final String COMPLETION_AUTO_POPUP_DELAY = "completion-auto-popup-delay"; //
NOI18N
```

```
    /** The delay after which the completion window is refreshed.
    * Values: java.lang.Integer
    */
    public static final String COMPLETION_REFRESH_DELAY = "completion-refresh-delay"; //
NOI18N

    /** The minimum size of the completion pane component.
    * Values: java.awt.Dimension
    */
    public static final String COMPLETION_PANE_MIN_SIZE = "completion-pane-min-size"; //
NOI18N

    /** The maximum size of the completion pane component.
    * Values: java.awt.Dimension
    */
    public static final String COMPLETION_PANE_MAX_SIZE = "completion-pane-max-size"; //
NOI18N

    /** Acceptor sensitive to characters that cause that
     * that the current line will be reformatted immediately.
     */
    public static final String INDENT_HOT_CHARS_ACCEPTOR = "indent-hot-chars-acceptor"; //
NOI18N

    /** Whether lines should be indented on an indent hot key if there is non whitespace
before
     * the typed hot key. See editor issue #10771.
     * Values: java.lang.Boolean
     */
    public static final String REINDENT_WITH_TEXT_BEFORE = "reindent-with-text-before"; //
NOI18N

    /** Whether the fast import should offer packages instead of classes
     * Values: java.lang.Integer
     */
    public static final String FAST_IMPORT_SELECTION = "fast-import-selection"; // NOI18N

    /** Whether the fast import should offer packages instead of classes
     * Values: java.lang.Boolean
     * @deprecated replaced by FAST_IMPORT_SELECTION
     */
    public static final String FAST_IMPORT_PACKAGE = "fast-import-package"; // NOI18N


    /** Display info before Go To Class feature
     *  Values: java.lang.Boolean
     */
    public static final String DISPLAY_GO_TO_CLASS_INFO = "display-go-to-class-info"; //
NOI18N
```

```
    /** Background color of javaDoc popup window
     *  Values: java.awt.Color
     */
    public static final String JAVADOC_BG_COLOR = "javadoc-bg-color"; //NOI18N

    /** The delay after which the javaDoc window is shown automatically.
    *   Values: java.lang.Integer
    */
    public static final String JAVADOC_AUTO_POPUP_DELAY = "javadoc-auto-popup-delay"; //NOI18N

    /** The preferred size of javaDoc popup window
    *   Values: java.awt.Dimension
    */
    public static final String JAVADOC_PREFERRED_SIZE = "javadoc-preferred-size"; //NOI18N

    /** Whether the javaDoc window should popup automatically.
    * Values: java.lang.Boolean
    */
    public static final String JAVADOC_AUTO_POPUP = "javadoc-auto-popup"; // NOI18N

    /** Whether to only show contracts in javaDoc window
    * Values: java.lang.Boolean
    */
    public static final String SHOW_CONTRACT = "show-contract"; // NOI18N

    /** Whether show deprecated members in code completion popup window
    * Values: java.lang.Boolean
    */
    public static final String SHOW_DEPRECATED_MEMBERS = "show-deprecated-members"; // NOI18N

    /** Whether update Code Completion DB after mounting new FS
    * Values: java.lang.String
    */
    public static final String UPDATE_PD_AFTER_MOUNTING = "update_pd_after_mounting"; //
NOI18N

    /** Two levels of performing auto update of Code Completion DB
     *  ALWAYS - PD is updated automatically after mounting a new filesystem
     *  NEVER - PD is never updated after mounting a new filesystem
     */
    public static final String ALWAYS = "pd_always";//NOI18N
    public static final String NEVER = "pd_never";//NOI18N

}
```

E.3 org.netbeans.editor.ext.ExtSettingsDefaults.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
```

```
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.editor.ext;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.util.Arrays;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import javax.swing.KeyStroke;
import javax.swing.text.JTextComponent;
import org.netbeans.editor.Coloring;
import org.netbeans.editor.Settings;
import org.netbeans.editor.SettingsUtil;
import org.netbeans.editor.SettingsNames;
import org.netbeans.editor.SettingsDefaults;
import org.netbeans.editor.BaseKit;
import org.netbeans.editor.MultiKeyBinding;

/**
* Initializer for the extended editor settings.
*
* @author Miloslav Metelka
* @version 1.00
*/

public class ExtSettingsDefaults extends SettingsDefaults {

    // Highlight row with caret coloring
    public static final Color defaultHighlightCaretRowBackColor = new Color(255, 255, 220);
    public static final Coloring defaultHighlightCaretRowColoring
    = new Coloring(null, null, defaultHighlightCaretRowBackColor);
    // Highlight matching brace coloring
    public static final Color defaultHighlightMatchBraceForeColor = Color.white;
    public static final Color defaultHighlightMatchBraceBackColor = new Color(255, 50, 210);
    public static final Coloring defaultHighlightMatchBraceColoring
    = new Coloring(null, defaultHighlightMatchBraceForeColor,
defaultHighlightMatchBraceBackColor);
```

```
public static final Boolean defaultHighlightCaretRow = Boolean.FALSE;
public static final Boolean defaultHighlightMatchBrace = Boolean.TRUE;
public static final Integer defaultHighlightMatchBraceDelay = new Integer(100);
public static final Boolean defaultCaretSimpleMatchBrace = Boolean.TRUE;

public static final Boolean defaultCompletionAutoPopup = Boolean.TRUE;
public static final Boolean defaultCompletionCaseSensitive = Boolean.TRUE;
public static final Boolean defaultCompletionNaturalSort = Boolean.FALSE;
public static final Integer defaultCompletionAutoPopupDelay = new Integer(500);
public static final Integer defaultCompletionRefreshDelay = new Integer(200);
public static final Dimension defaultCompletionPaneMaxSize = new Dimension(400, 300);
public static final Dimension defaultCompletionPaneMinSize = new Dimension(60, 34);
public static final Boolean defaultDisplayGoToClassInfo = Boolean.TRUE;
public static final Boolean defaultFastImportPackage = Boolean.FALSE;
public static final Integer defaultFastImportSelection = new Integer(0);
public static final Boolean defaultShowDeprecatedMembers = Boolean.TRUE;
public static final Boolean defaultCompletionInstantSubstitution = Boolean.TRUE;

public static final Color defaultJavaDocBGColor = new Color(247, 247, 255);
public static final Integer defaultJavaDocAutoPopupDelay = new Integer(200);
public static final Dimension defaultJavaDocPreferredSize = new Dimension(500, 300);
public static final Boolean defaultJavaDocAutoPopup = Boolean.TRUE;
public static final String defaultUpdatePDAfterMounting = ExtSettingsNames.ALWAYS;
public static final Boolean defaultShowContract = Boolean.FALSE;

public static final MultiKeyBinding[] defaultExtKeyBindings
= new MultiKeyBinding[] {
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_G, InputEvent.ALT_MASK),
          ExtKit.gotoDeclarationAction
      ),
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_F, InputEvent.CTRL_MASK),
          ExtKit.findAction
      ),
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_FIND, 0),
          ExtKit.findAction
      ),
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_H, InputEvent.CTRL_MASK),
          ExtKit.replaceAction
      ),
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_G, InputEvent.CTRL_MASK),
          ExtKit.gotoAction
      ),
      new MultiKeyBinding(
          KeyStroke.getKeyStroke(KeyEvent.VK_SPACE, InputEvent.CTRL_MASK),
          ExtKit.completionShowAction
```

```
        ),
        new MultiKeyBinding( // Japanese Solaris uses CTRL+SPACE for IM
            KeyStroke.getKeyStroke(KeyEvent.VK_BACK_SLASH, InputEvent.CTRL_MASK),
            ExtKit.completionShowAction
        ),
        new MultiKeyBinding(
            KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
            ExtKit.escapeAction
        ),
        new MultiKeyBinding(
            KeyStroke.getKeyStroke(KeyEvent.VK_OPEN_BRACKET, InputEvent.CTRL_MASK),
            ExtKit.matchBraceAction
        ),
        new MultiKeyBinding(
            KeyStroke.getKeyStroke(KeyEvent.VK_OPEN_BRACKET, InputEvent.CTRL_MASK |
InputEvent.SHIFT_MASK),
            ExtKit.selectionMatchBraceAction
        ),
        new MultiKeyBinding(
            KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, InputEvent.SHIFT_MASK),
            ExtKit.shiftInsertBreakAction
        ),
        new MultiKeyBinding(
            KeyStroke.getKeyStroke(KeyEvent.VK_F10, InputEvent.SHIFT_MASK),
            ExtKit.showPopupMenuAction
        ),
        /*      new MultiKeyBinding(
                  KeyStroke.getKeyStroke(KeyEvent.VK_U, InputEvent.CTRL_MASK),
        //          KeyStroke.getKeyStroke(KeyEvent.VK_BRACELEFT, InputEvent.CTRL_MASK),
                  ExtKit.braceCodeSelectAction
                ),
        */

    };

}
```

E.4 org.netbeans.editor.ext.ExtSettingsInitializer.java

```
/*
 *                Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
```

```
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.editor.ext;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.util.Arrays;
import java.util.List;
import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import javax.swing.KeyStroke;
import javax.swing.text.JTextComponent;
import org.netbeans.editor.Coloring;
import org.netbeans.editor.Settings;
import org.netbeans.editor.SettingsUtil;
import org.netbeans.editor.SettingsNames;
import org.netbeans.editor.BaseKit;
import org.netbeans.editor.MultiKeyBinding;

/**
* Initializer for the extended editor settings.
*
* @author Miloslav Metelka
* @version 1.00
*/

public class ExtSettingsInitializer extends Settings.AbstractInitializer {

    public static final String NAME = "ext-settings-initializer";

    public ExtSettingsInitializer() {
        super(NAME);
    }

    /** Update map filled with the settings.
    * @param kitClass kit class for which the settings are being updated.
    *   It is always non-null value.
    * @param settingsMap map holding [setting-name, setting-value] pairs.
    *   The map can be empty if this is the first initializer
    *   that updates it or if no previous initializers updated it.
    */
    public void updateSettingsMap(Class kitClass, Map settingsMap) {

        // ----------------------- BaseKit Settings -------------------------------------
        if (kitClass == BaseKit.class) {
            // Add key-bindings
            SettingsUtil.updateListSetting(settingsMap, SettingsNames.KEY_BINDING_LIST,
```

```
                                        ExtSettingsDefaults.defaultExtKeyBindings);
        }

        // ------------------------ ExtKit Settings -----------------------------------
        if (kitClass == ExtKit.class) {

            // List of the additional colorings
            SettingsUtil.updateListSetting(settingsMap, SettingsNames.COLORING_NAME_LIST,
                                        new String[] {
                                            ExtSettingsNames.HIGHLIGHT_CARET_ROW_COLORING,

ExtSettingsNames.HIGHLIGHT_MATCH_BRACE_COLORING,
                                        }
                                       );

            // ExtCaret highlighting options
            settingsMap.put(ExtSettingsNames.HIGHLIGHT_CARET_ROW,
                            ExtSettingsDefaults.defaultHighlightCaretRow);
            settingsMap.put(ExtSettingsNames.HIGHLIGHT_MATCH_BRACE,
                            ExtSettingsDefaults.defaultHighlightMatchBrace);

            // ExtCaret highlighting colorings
            SettingsUtil.setColoring(settingsMap,
ExtSettingsNames.HIGHLIGHT_CARET_ROW_COLORING,
                                     ExtSettingsDefaults.defaultHighlightCaretRowColoring);
            SettingsUtil.setColoring(settingsMap,
ExtSettingsNames.HIGHLIGHT_MATCH_BRACE_COLORING,
                                     ExtSettingsDefaults.defaultHighlightMatchBraceColoring);

            // Popup menu default action names
            String[] popupMenuActionNames
                = new String[] {
                    BaseKit.cutAction,
                    BaseKit.copyAction,
                    BaseKit.pasteAction,
                    null,
                    BaseKit.removeSelectionAction
                };


            List pml = (List)settingsMap.get(ExtSettingsNames.POPUP_MENU_ACTION_NAME_LIST);
            if (pml == null || pml.indexOf(BaseKit.cutAction) == -1) {
                SettingsUtil.updateListSetting(settingsMap,
                    ExtSettingsNames.POPUP_MENU_ACTION_NAME_LIST, popupMenuActionNames);

                SettingsUtil.updateListSetting(settingsMap,
                    ExtSettingsNames.DIALOG_POPUP_MENU_ACTION_NAME_LIST,
popupMenuActionNames);
            }

            settingsMap.put(ExtSettingsNames.POPUP_MENU_ENABLED, Boolean.TRUE);
```

```
        settingsMap.put(ExtSettingsNames.FAST_IMPORT_PACKAGE,
                        ExtSettingsDefaults.defaultFastImportPackage);

        // Completion settings
        settingsMap.put(ExtSettingsNames.COMPLETION_AUTO_POPUP,
                        ExtSettingsDefaults.defaultCompletionAutoPopup);

        settingsMap.put(ExtSettingsNames.COMPLETION_CASE_SENSITIVE,
                        ExtSettingsDefaults.defaultCompletionCaseSensitive);

        settingsMap.put(ExtSettingsNames.COMPLETION_NATURAL_SORT,
                        ExtSettingsDefaults.defaultCompletionNaturalSort);

        settingsMap.put(ExtSettingsNames.COMPLETION_INSTANT_SUBSTITUTION,
                        ExtSettingsDefaults.defaultCompletionInstantSubstitution);

        settingsMap.put(ExtSettingsNames.COMPLETION_AUTO_POPUP_DELAY,
                        ExtSettingsDefaults.defaultCompletionAutoPopupDelay);

        settingsMap.put(ExtSettingsNames.COMPLETION_REFRESH_DELAY,
                        ExtSettingsDefaults.defaultCompletionRefreshDelay);

        settingsMap.put(ExtSettingsNames.COMPLETION_PANE_MIN_SIZE,
                        ExtSettingsDefaults.defaultCompletionPaneMinSize);

        settingsMap.put(ExtSettingsNames.COMPLETION_PANE_MAX_SIZE,
                        ExtSettingsDefaults.defaultCompletionPaneMaxSize);

        // re-indentation settings
        settingsMap.put(ExtSettingsNames.REINDENT_WITH_TEXT_BEFORE,
                        Boolean.TRUE);

        settingsMap.put(ExtSettingsNames.JAVADOC_BG_COLOR,
                        ExtSettingsDefaults.defaultJavaDocBGColor);

        settingsMap.put(ExtSettingsNames.JAVADOC_AUTO_POPUP_DELAY,
                        ExtSettingsDefaults.defaultJavaDocAutoPopupDelay);

        settingsMap.put(ExtSettingsNames.JAVADOC_PREFERRED_SIZE,
                        ExtSettingsDefaults.defaultJavaDocPreferredSize);

        settingsMap.put(ExtSettingsNames.JAVADOC_AUTO_POPUP,
                        ExtSettingsDefaults.defaultJavaDocAutoPopup);

        settingsMap.put(ExtSettingsNames.SHOW_CONTRACT,
                        ExtSettingsDefaults.defaultShowContract);

    }

}
```

```
}
```

E.5 org.netbeans.editor.ext.JavaDocPane.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */


package org.netbeans.editor.ext;

import javax.swing.JComponent;

/**
 *  Pane displaying the javadoc view and accompanying components
 *  like toolbar for navigation etc.
 *
 *  @author  Martin Roskanin
 *  @since   03/2002
 */
public interface JavaDocPane {

    /** Returns component of JavaDocPane implementation */
    public JComponent getComponent();

    /** enables/disables forward button */
    public void setForwardEnabled(boolean enable);

    /** enables/disables back button */
    public void setBackEnabled(boolean enable);

    /** enables/disables 'show in external browser' button */
    public void setShowWebEnabled(boolean enable);

    /** enables/disables 'insert precondition test in code' button */
    public void setInsertPreEnabled(boolean enable);

    public JComponent getJavadocDisplayComponent();
}
```

E.6 org.netbeans.editor.ext.ScrollJavaDocPane.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */


package org.netbeans.editor.ext;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.Dimension;
import java.awt.Rectangle;

import javax.swing.JScrollPane;
import javax.swing.JComponent;
import javax.swing.JRootPane;
import javax.swing.text.JTextComponent;
import javax.swing.JLayeredPane;
import javax.swing.JList;
import javax.swing.SwingUtilities;
import javax.swing.JPanel;

import org.netbeans.editor.SettingsChangeListener;
import org.netbeans.editor.Utilities;
import org.netbeans.editor.SettingsUtil;
import org.netbeans.editor.ext.ExtSettingsNames;
import org.netbeans.editor.ext.ExtSettingsDefaults;
import org.netbeans.editor.SettingsChangeEvent;
import org.netbeans.editor.Settings;
import javax.swing.JButton;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EtchedBorder;
import javax.swing.border.EmptyBorder;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.ImageIcon;
import java.net.URL;
import javax.swing.Icon;
import java.awt.FlowLayout;
```

```
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.GridLayout;
import java.awt.ComponentOrientation;
import java.awt.GridBagLayout;
import javax.swing.BoxLayout;
import javax.swing.AbstractAction;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.event.HyperlinkListener;
import javax.swing.event.HyperlinkEvent;
import javax.swing.JEditorPane;
import javax.swing.UIManager;
import javax.swing.border.Border;
import javax.swing.border.LineBorder;
import org.netbeans.editor.LocaleSupport;

/**
 *  JScrollPane implementation of JavaDocPane.
 *
 *  @author  Martin Roskanin
 *  @since   03/2002
 */
public class ScrollJavaDocPane extends JPanel implements JavaDocPane, SettingsChangeListener{

    protected  ExtEditorUI extEditorUI;
    private JComponent view;
    private  CompletionJavaDoc cjd;
    protected JScrollPane scrollPane = new JScrollPane();
    Border lineBorder;

    /** Creates a new instance of ScrollJavaDocPane */
    public ScrollJavaDocPane(ExtEditorUI extEditorUI) {

//        new RuntimeException("ScrollJavaDocPane.<init>").printStackTrace();

        setLayout(null);

        this.extEditorUI = extEditorUI;

        // Add the completionJavaDoc view
        cjd = extEditorUI.getCompletionJavaDoc();
        if (cjd!=null){
            JavaDocView javaDocView = cjd.getJavaDocView();
            if (javaDocView instanceof JComponent) {
                if (javaDocView instanceof JEditorPane){
                    ((JEditorPane)javaDocView).addHyperlinkListener(new HyperlinkAction());
                }
                view = (JComponent)javaDocView;
                scrollPane.setViewportView(view);
```

```
            }

            Settings.addSettingsChangeListener(this);
            setMinimumSize(new Dimension(100,100)); //[PENDING] put it into the options
            setMaximumSize(getMaxPopupSize());
        }else{
            setMinimumSize(new Dimension(0,0));
            setMaximumSize(new Dimension(0,0));
        }
        super.setVisible(false);
        add(scrollPane);

getAccessibleContext().setAccessibleDescription(LocaleSupport.getString("ACSD_JAVADOC_javaDocP
ane")); //NOI18N

        // !!! virtual method called from contructor!!
        installTitleComponent();
        setBorder(new LineBorder(javax.swing.UIManager.getColor("controlDkShadow"))); //NOI18N
    }


    public void setBounds(Rectangle r){
        super.setBounds(r);
        scrollPane.setBounds(r.x, 0, r.width+1, r.height );
    }

    public void setVisible(boolean visible){
        super.setVisible(visible);
        if (cjd!=null && !visible){
            cjd.clearHistory();
        }
    }

    protected ImageIcon resolveIcon(String res){
        ClassLoader loader = this.getClass().getClassLoader();
        URL resource = loader.getResource( res );
        if( resource == null ) resource = ClassLoader.getSystemResource( res );
        return  ( resource != null ) ? new ImageIcon( resource ) : null;
    }

    protected void installTitleComponent() {
    }

    private Dimension getMaxPopupSize(){
        Class kitClass = Utilities.getKitClass(extEditorUI.getComponent());
        if (kitClass != null) {
            return (Dimension)SettingsUtil.getValue(kitClass,
                      ExtSettingsNames.JAVADOC_PREFERRED_SIZE,
                      ExtSettingsDefaults.defaultJavaDocAutoPopupDelay);

        }
```

```
    return ExtSettingsDefaults.defaultJavaDocPreferredSize;
}

public void settingsChange(SettingsChangeEvent evt) {
    if (ExtSettingsNames.JAVADOC_PREFERRED_SIZE.equals(evt.getSettingName())){
        setMaximumSize(getMaxPopupSize());
    }
}

public JComponent getComponent() {
    return this;
}

public void setForwardEnabled(boolean enable) {
}

public void setBackEnabled(boolean enable) {
}

public void setShowWebEnabled(boolean enable) {
}

public void setInsertPreEnabled(boolean enable) {
}

public JComponent getJavadocDisplayComponent() {
    return scrollPane;
}

public class BrowserButton extends JButton {
    public BrowserButton() {
        setBorderPainted(false);
        setFocusPainted(false);
    }

    public BrowserButton(String text){
        super(text);
        setBorderPainted(false);
        setFocusPainted(false);
    }

    public BrowserButton(Icon icon){
        super(icon);
        setBorderPainted(false);
        setFocusPainted(false);
    }
}


protected  class HyperlinkAction implements HyperlinkListener{
    public void hyperlinkUpdate(HyperlinkEvent e) {
```

```
            if (e!=null && HyperlinkEvent.EventType.ACTIVATED.equals(e.getEventType())){
                if (e.getDescription() != null){
                    Object obj = cjd.parseLink(e.getDescription(), null);
                    if (obj!=null){
                        cjd.setContent(obj);
                        cjd.addToHistory(obj);
                    }
                }
            }
        }
    }
    /*
    private class BackAction implements ActionListener{
        public void actionPerformed(ActionEvent evt) {
            if (cjd!=null){
                System.out.println("back");
                cjd.backHistory();
            }
        }
    }

    private class ForwardAction implements ActionListener {
        public void actionPerformed(ActionEvent evt) {
            if (cjd!=null){
                System.out.println("fwd");
                cjd.forwardHistory();
            }
        }
    }
    */
}
```

E.7 org.netbeans.editor.ext.JDCPopupPanel.java

```
/*
 *                Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.editor.ext;
```

```
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Rectangle;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.awt.event.KeyEvent;
import java.awt.event.FocusListener;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import javax.swing.JComponent;
import javax.swing.JPanel;
import javax.swing.text.JTextComponent;
import javax.swing.KeyStroke;
import javax.swing.SwingUtilities;
import javax.swing.Action;
import javax.swing.AbstractAction;

import org.netbeans.editor.PopupManager;
import org.netbeans.editor.ext.ExtEditorUI;
import org.netbeans.editor.Utilities;
import org.netbeans.editor.BaseKit;
import org.netbeans.editor.ext.CompletionJavaDoc;
import org.netbeans.editor.ext.ExtKit;
import javax.swing.plaf.TextUI;
import javax.swing.text.Keymap;
import javax.swing.text.EditorKit;
import java.util.Iterator;
import java.util.ArrayList;
import java.util.List;
import org.netbeans.editor.SettingsChangeEvent;
import org.netbeans.editor.SettingsChangeListener;
import org.netbeans.editor.Settings;


/**
 *  Invisible panel that contains code completion panel and javadoc panel,
 *  computes preferred size and provides access to these both components.
 *
 *  @author  Martin Roskanin
 *  @since   03/2002
 */
public class JDCPopupPanel extends JPanel implements PropertyChangeListener,
SettingsChangeListener {

    private ExtCompletionPane completion;

    private ExtEditorUI extEditorUI;
    private FocusListener focusL;
```

```
    private List keyActionPairsList;

    // Completion can create documentation pane
    private Completion documentationProvider;
    private JavaDocPane javadoc;

    // gap between javadoc and code completion window
    private static final int WINDOW_GAP = 1;

    private static final String POPUP_HIDE = "jdc-popup-hide"; //NOI18N

    private static final String COMPLETION_UP = "completion-up"; //NOI18N
    private static final String COMPLETION_DOWN = "completion-down"; //NOI18N
    private static final String COMPLETION_PGUP = "completion-pgup"; //NOI18N
    private static final String COMPLETION_PGDN = "completion-pgdn"; //NOI18N
    private static final String COMPLETION_BEGIN = "completion-begin"; //NOI18N
    private static final String COMPLETION_END = "completion-end"; //NOI18N

    private static final String JAVADOC_UP = "javadoc-up"; //NOI18N
    private static final String JAVADOC_DOWN = "javadoc-down"; //NOI18N
    private static final String JAVADOC_PGUP = "javadoc-pgup"; //NOI18N
    private static final String JAVADOC_PGDN = "javadoc-pgdn"; //NOI18N
    private static final String JAVADOC_BEGIN = "javadoc-begin"; //NOI18N
    private static final String JAVADOC_END = "javadoc-end"; //NOI18N
    private static final String JAVADOC_LEFT = "javadoc-left"; //NOI18N
    private static final String JAVADOC_RIGHT = "javadoc-right"; //NOI18N
    private static final String JAVADOC_BACK = "javadoc-back"; //NOI18N
    private static final String JAVADOC_FORWARD = "javadoc-forward"; //NOI18N
    private static final String JAVADOC_OPEN_IN_BROWSER = "javadoc-open-in-browser"; //NOI18N
    private static final String JAVADOC_OPEN_SOURCE = "javadoc-open-source"; //NOI18N
    private static final String JAVADOC_INSERT_PRECONDITION = "javadoc-insert-precondition";
//NOI18N

    private static final int ACTION_POPUP_HIDE = 1;

    private static final int ACTION_COMPLETION_UP = 2;
    private static final int ACTION_COMPLETION_DOWN = 3;
    private static final int ACTION_COMPLETION_PGUP = 4;
    private static final int ACTION_COMPLETION_PGDN = 5;
    private static final int ACTION_COMPLETION_BEGIN = 6;
    private static final int ACTION_COMPLETION_END = 7;

    private static final int ACTION_JAVADOC_UP = 8;
    private static final int ACTION_JAVADOC_DOWN = 9;
    private static final int ACTION_JAVADOC_PGUP = 10;
    private static final int ACTION_JAVADOC_PGDN = 11;
    private static final int ACTION_JAVADOC_BEGIN = 12;
    private static final int ACTION_JAVADOC_END = 13;
    private static final int ACTION_JAVADOC_LEFT = 14;
    private static final int ACTION_JAVADOC_RIGHT = 15;
    private static final int ACTION_JAVADOC_BACK = 16;
```

```
    private static final int ACTION_JAVADOC_FORWARD = 17;
    private static final int ACTION_JAVADOC_OPEN_IN_BROWSER = 18;
    private static final int ACTION_JAVADOC_OPEN_SOURCE = 19;
    private static final int ACTION_JAVADOC_INSERT_PRECONDITION = 20;


    /** Creates a new instance of JDCPopupPanel */
    public JDCPopupPanel(ExtEditorUI extEditorUI, ExtCompletionPane completion, Completion
documentationProvider) {
        super();
        this.completion = completion;
        this.documentationProvider = documentationProvider;
        this.extEditorUI = extEditorUI;
        this.keyActionPairsList = new ArrayList();

        setLayout(null);
        setOpaque(false); // make panel background invisible

        focusL = new FocusAdapter() {
            public void focusLost(FocusEvent evt) {
                SwingUtilities.invokeLater( new Runnable() {
                    public void run() {
                        if (isVisible()) {
                            JTextComponent component =
JDCPopupPanel.this.extEditorUI.getComponent();
                            if (component != null) {
                                java.awt.Window w = SwingUtilities.windowForComponent(
                                component);
                                Component focusOwner = (w == null) ?
                                null : w.getFocusOwner();

                                if (focusOwner == null) {
                                    setVisible(false);
                                }

                                boolean docO = javadoc != null && focusOwner ==
JDCPopupPanel.this.getJavaDocView();
                                boolean docA = javadoc != null &&
JDCPopupPanel.this.getJavaDocPane().getComponent().isAncestorOf(focusOwner);
                                boolean comO = focusOwner ==
JDCPopupPanel.this.getCompletionView();
                                if (docO || comO || docA) {
                                    component.requestFocus();
                                }else if ( focusOwner != component ) {
                                    setVisible(false); // completion, javadoc and component
don't own the focus
                                }
                            }
                        }
                    }
                } );
```

```
        }
    };


    synchronized (extEditorUI.getComponentLock()) {
        // if component already installed in ExtEditorUI simulate installation
        JTextComponent component = extEditorUI.getComponent();
        if (component != null) {
            propertyChange(new PropertyChangeEvent(extEditorUI,
            ExtEditorUI.COMPONENT_PROPERTY, null, component));
        }
        extEditorUI.addPropertyChangeListener(this);
    }

    super.setVisible(false);

    Settings.addSettingsChangeListener(this);
}

/** Returns completion pane */
public ExtCompletionPane getCompletionPane(){
    return completion;
}

private void cancelJavaDocTask(){
    CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
    if (cjd!=null){
        cjd.cancelPerformingThread();
    }
}

private JavaDocView getJavaDocView(){
    CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
    if (cjd!=null){
        return cjd.getJavaDocView();
    }
    return null;
}

private CompletionView getCompletionView(){
    Completion c = extEditorUI.getCompletion();
    if (c!=null){
        return c.getView();
    }
    return null;
}

/** Returns javadoc pane */
public JavaDocPane getJavaDocPane(){
    if (javadoc == null) {
        javadoc = documentationProvider.getJavaDocPane();
```

```
        }
        return javadoc;
    }

    /** Returns minimum size that this component can occupy. Because javadoc can be omitted,
     *  the minimum size is the minimum size of code completion */
    public Dimension getMinimumSize(){
        return completion.getComponent().getMinimumSize();
    }

    /** Returns the maximum size of this component. Maximum size is computed from
     *  javadoc's and completion's maximum size */
    public Dimension getMaximumSize(){
        Dimension compDim = completion.getComponent().getMaximumSize();
        Dimension javadocDim = new Dimension();
        if (javadoc != null) {
            javadocDim = javadoc.getComponent().getMaximumSize();
        }
        return new Dimension(Math.max(compDim.width,javadocDim.width),
compDim.height+javadocDim.height);
    }

    /** Returns the preferred size of this component. Preferred size is computed from
     *  javadoc's and completion's preferred size trimmed to fit maximum size */
    public Dimension getPreferredSize(){
        Dimension ret = new Dimension();
        Dimension compPref = completion.getComponent().getPreferredSize();
        Dimension compMax = completion.getComponent().getMaximumSize();

        Dimension javadocPref = new Dimension();
        Dimension javadocMax = new Dimension();
        if (javadoc != null) {
            javadocPref = javadoc.getComponent().getPreferredSize();
            javadocMax = javadoc.getComponent().getMaximumSize();
        }

        ret.width = Math.min(compPref.width, compMax.width) + Math.min(javadocPref.width,
javadocMax.width);
        ret.height = Math.min(compPref.height, compMax.height) + Math.min(javadocPref.height,
javadocMax.height);

        return ret;
    }

    /** Sets completion pane visiblility */
    public void setCompletionVisible(boolean visible){
        completion.setVisible(visible);
        setVisible(visible);
    }

    /** Sets javadoc pane visibility */
```

```
    public void setJavaDocVisible(boolean visible){
        getJavaDocPane().getComponent().setVisible(visible);
        if (visible) {
            extEditorUI.getPopupManager().install(this);
            setVisible(visible);
        }
        if (!getCompletionPane().isVisible()){
            setVisible(visible);
        }
    }

    /** Sets this popup comonent visibility */
    public void setVisible(boolean visible){
        if (visible == true){
            extEditorUI.getPopupManager().install(this);
        }
        boolean docv = javadoc != null && javadoc.getComponent().isVisible();
        if (visible && (docv || completion.isVisible())){
            super.setVisible(visible);
        }
        if (!visible){
            cancelJavaDocTask();
            Completion c = extEditorUI.getCompletion();
            if (c!=null){
                c.completionCancel();
            }
            if (javadoc != null) {
                javadoc.getComponent().setVisible(visible);
            }
            completion.getComponent().setVisible(visible);
            super.setVisible(visible);
        }
    }

    /** Setting size of popup panel. The height and size is computed to fit the best place on
the screen */
    public void setSize(int width, int height){
        PopupManager.Placement placement = (PopupManager.Placement)getClientProperty(
            PopupManager.Placement.class);

        Dimension completionMinSize = completion.getComponent().getMinimumSize();
        if (completionMinSize.height > height) { // cannot fit
            putClientProperty(PopupManager.Placement.class, null);
        }

        // first we will set size to completion and then if there will be space for javadoc we
will display it.
        completion.getComponent().setSize(width, height);

        Dimension javaDocMinSize = new Dimension();
        if (javadoc != null) {
```

```
            javaDocMinSize = javadoc.getComponent().getMinimumSize();
        }
        Dimension completionMaxSize = completion.getComponent().getMaximumSize();

        Rectangle completionBounds = new Rectangle(completion.getComponent().getSize());
        Rectangle javadocBounds = new Rectangle();
        if (javadoc != null) {
            javadocBounds = new Rectangle(javadoc.getComponent().getMaximumSize());
        }

        boolean showJavaDoc = true;
        if (javadoc == null || javadoc.getComponent().isVisible() == false) {
            showJavaDoc = false;
        }

        CompletionJavaDoc completionJavaDoc = extEditorUI.getCompletionJavaDoc();
        if (completionJavaDoc!=null){
            if(completionJavaDoc.autoPopup()){
                showJavaDoc = javadoc != null;
            }
        }

        boolean showCompletion = getCompletionPane().isVisible();

        if (!showCompletion){
            completionMinSize.height = 0;
            completionMinSize.width = 0;
            completionBounds = new Rectangle();
        }else{
            completionBounds.width = Math.min(completionMaxSize.width,completionBounds.width);
            completionBounds.height =
Math.min(completionMaxSize.height,completionBounds.height);
        }

        // do not show javaDoc. There is no space available
        if ((javaDocMinSize.height+completionMinSize.height)>height) showJavaDoc = false;

        if (showJavaDoc) {

            if ((completionBounds.height + javadocBounds.height) > height ){
                // javadocBounds.height should be resized
                completionBounds.height =
Math.max(Math.min((int)(height/2),completionBounds.height), completionMinSize.height);
                completionBounds.height = Math.min(130,completionBounds.height); // [PENDING]
- maybe this should be in options
                javadocBounds.height = Math.min((height - completionBounds.height - WINDOW_GAP
), javadocBounds.height);
            }

            if (placement == PopupManager.Below) {
                completionBounds.y = 0;
```

```
                javadocBounds.y = completionBounds.height + WINDOW_GAP;
            }else{
                completionBounds.y = javadocBounds.height + WINDOW_GAP;
                javadocBounds.y = 0;
            }

        }

        //width algorhitm
        if (width < javaDocMinSize.width){
            showJavaDoc = false;
        }else{
            // going to compute the width of both windows
            javadocBounds.width = Math.min(width,javadocBounds.width);

            JTextComponent component = extEditorUI.getComponent();
            Rectangle extBounds = extEditorUI.getExtentBounds();
            Rectangle caretRect = (Rectangle)component.getCaret();

            completionBounds.x = ((width - javadocBounds.width/2) < (caretRect.x -
extBounds.x)) ?
                javadocBounds.width - completionBounds.width : 0;
        }


        completion.setVisible(false);

        remove(completion.getComponent());
        if (javadoc != null) {
            remove(javadoc.getComponent());
        }

        if (showJavaDoc) {
            if (showCompletion){
                completion.getComponent().setBounds(completionBounds);
                getJavaDocPane().getComponent().setBounds(javadocBounds);

super.setBounds(0,0,Math.max(completionBounds.width,javadocBounds.width),completionBounds.heig
ht+javadocBounds.height+WINDOW_GAP);
                add(completion.getComponent());
                add(getJavaDocPane().getComponent());
                completion.setVisible(true);
            }else{
                javadocBounds.x = 0; javadocBounds.y = 0;
                getJavaDocPane().getComponent().setBounds(javadocBounds);
                super.setBounds(0,0,javadocBounds.width,javadocBounds.height);
                add(getJavaDocPane().getComponent());
            }
        }else{
            completionBounds.x = 0; completionBounds.y = 0;
            completion.getComponent().setBounds(completionBounds);
```

```
            super.setBounds(0,0,completionBounds.width,completionBounds.height+WINDOW_GAP);
            add(completion.getComponent());
            completion.setVisible(true);
        }
    }

    public void setSize(Dimension d){
        setSize(d.width, d.height);
    }

    /** Refreshes the code completion content */
    public void refresh(){
        setVisible(true);
    }

    private void performJavaDocAction(KeyStroke ks){
        ActionListener act =
getJavaDocPane().getJavadocDisplayComponent().getActionForKeyStroke(ks);
        if (act!=null){
            act.actionPerformed(new ActionEvent(getJavaDocPane().getJavadocDisplayComponent(),
ActionEvent.ACTION_PERFORMED, "")); //NOI18N
            getJavaDocPane().getJavadocDisplayComponent().repaint();
        }
    }

    /** Attempt to find the editor keystroke for the given editor action. */
    private KeyStroke[] findEditorKeys(String editorActionName, KeyStroke defaultKey) {
        // This method is implemented due to the issue
        // #25715 - Attempt to search keymap for the keybinding that logically corresponds to
the action
        KeyStroke[] ret = new KeyStroke[] { defaultKey };
        if (editorActionName != null && extEditorUI != null) {
            JTextComponent component = extEditorUI.getComponent();
            if (component != null) {
                TextUI ui = component.getUI();
                Keymap km = component.getKeymap();
                if (ui != null && km != null) {
                    EditorKit kit = ui.getEditorKit(component);
                    if (kit instanceof BaseKit) {
                        Action a = ((BaseKit)kit).getActionByName(editorActionName);
                        if (a != null) {
                            KeyStroke[] keys = km.getKeyStrokesForAction(a);
                            if (keys != null && keys.length > 0) {
                                ret = keys;
                            }
                        }
                    }
                }
            }
        }
```

```
        return ret;
    }

    private void registerKeybinding(int action, String actionName, KeyStroke stroke, String
editorActionName){
        KeyStroke[] keys = findEditorKeys(editorActionName, stroke);
        for (int i = 0; i < keys.length; i++) {
            getInputMap().put(keys[i], actionName);
            keyActionPairsList.add(actionName); // add action-name
            keyActionPairsList.add(keys[i]); // add keystroke
        }

        getActionMap().put(actionName, new JDCPopupAction(action));
    }

    private void unregisterKeybinding(String actionName, KeyStroke stroke) {
        getInputMap().remove(stroke);
        getActionMap().remove(actionName);
    }

    private void installKeybindings() {
        // Register escape key
        registerKeybinding(ACTION_POPUP_HIDE, POPUP_HIDE,
        KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0),
        ExtKit.escapeAction
        );

        // Register up key
        registerKeybinding(ACTION_COMPLETION_UP, COMPLETION_UP,
        KeyStroke.getKeyStroke(KeyEvent.VK_UP, 0),
        BaseKit.upAction
        );

        // Register down key
        registerKeybinding(ACTION_COMPLETION_DOWN, COMPLETION_DOWN,
        KeyStroke.getKeyStroke(KeyEvent.VK_DOWN, 0),
        BaseKit.downAction
        );

        // Register PgDn key
        registerKeybinding(ACTION_COMPLETION_PGDN, COMPLETION_PGDN,
        KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_DOWN, 0),
        BaseKit.pageDownAction
        );

        // Register PgUp key
        registerKeybinding(ACTION_COMPLETION_PGUP, COMPLETION_PGUP,
        KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP, 0),
        BaseKit.pageUpAction
        );
```

```
// Register home key
registerKeybinding(ACTION_COMPLETION_BEGIN, COMPLETION_BEGIN,
KeyStroke.getKeyStroke(KeyEvent.VK_HOME, 0),
BaseKit.beginLineAction
);

// Register end key
registerKeybinding(ACTION_COMPLETION_END, COMPLETION_END,
KeyStroke.getKeyStroke(KeyEvent.VK_END, 0),
BaseKit.endLineAction
);

// Register javadoc up key
registerKeybinding(ACTION_JAVADOC_UP, JAVADOC_UP,
KeyStroke.getKeyStroke(KeyEvent.VK_UP, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc down key
registerKeybinding(ACTION_JAVADOC_DOWN, JAVADOC_DOWN,
KeyStroke.getKeyStroke(KeyEvent.VK_DOWN, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc PgDn key
registerKeybinding(ACTION_JAVADOC_PGDN, JAVADOC_PGDN,
KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_DOWN, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc PgUp key
registerKeybinding(ACTION_JAVADOC_PGUP, JAVADOC_PGUP,
KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc home key
registerKeybinding(ACTION_JAVADOC_BEGIN, JAVADOC_BEGIN,
KeyStroke.getKeyStroke(KeyEvent.VK_HOME, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc end key
registerKeybinding(ACTION_JAVADOC_END, JAVADOC_END,
KeyStroke.getKeyStroke(KeyEvent.VK_END, KeyEvent.SHIFT_MASK),
null
);

// Register javadoc left key
registerKeybinding(ACTION_JAVADOC_LEFT, JAVADOC_LEFT,
KeyStroke.getKeyStroke(KeyEvent.VK_LEFT, KeyEvent.SHIFT_MASK),
```

```
    null
    );

    // Register javadoc right key
    registerKeybinding(ACTION_JAVADOC_RIGHT, JAVADOC_RIGHT,
    KeyStroke.getKeyStroke(KeyEvent.VK_RIGHT, KeyEvent.SHIFT_MASK),
    null
    );

    // Register javadoc back key
    registerKeybinding(ACTION_JAVADOC_BACK, JAVADOC_BACK,
    KeyStroke.getKeyStroke(KeyEvent.VK_LEFT, KeyEvent.ALT_MASK),
    null
    );

    // Register javadoc forward key
    registerKeybinding(ACTION_JAVADOC_FORWARD, JAVADOC_FORWARD,
    KeyStroke.getKeyStroke(KeyEvent.VK_RIGHT, KeyEvent.ALT_MASK),
    null
    );

    // Register open in external browser key
    registerKeybinding(ACTION_JAVADOC_OPEN_IN_BROWSER, JAVADOC_OPEN_IN_BROWSER,
    KeyStroke.getKeyStroke(KeyEvent.VK_F1, KeyEvent.ALT_MASK | KeyEvent.SHIFT_MASK),
    null
    );

    // Register open the source in editor key
    registerKeybinding(ACTION_JAVADOC_OPEN_SOURCE, JAVADOC_OPEN_SOURCE,
    KeyStroke.getKeyStroke(KeyEvent.VK_O, KeyEvent.ALT_MASK | KeyEvent.CTRL_MASK),
    null
    );

    // Register insert the precondition in source key
    registerKeybinding(ACTION_JAVADOC_INSERT_PRECONDITION, JAVADOC_INSERT_PRECONDITION,
    KeyStroke.getKeyStroke(KeyEvent.VK_P, KeyEvent.ALT_MASK),
    null
    );

}

private void uninstallKeybindings() {
    for (Iterator it = keyActionPairsList.iterator(); it.hasNext();) {
        unregisterKeybinding(
            (String)it.next(), // action-name
            (KeyStroke)it.next() // keystroke
        );
    }
    keyActionPairsList.clear();
}
```

```
public void propertyChange(PropertyChangeEvent evt) {
    String propName = evt.getPropertyName();

    if (ExtEditorUI.COMPONENT_PROPERTY.equals(propName)) {
        if (evt.getNewValue() != null) { // just installed
            JTextComponent component = extEditorUI.getComponent();
            installKeybindings();
            component.addFocusListener(focusL);

        } else { // just deinstalled
            JTextComponent component = (JTextComponent)evt.getOldValue();
            uninstallKeybindings();
            component.removeFocusListener(focusL);
        }

    }
}

public void settingsChange(SettingsChangeEvent evt) {
    // Refresh keybindings
    uninstallKeybindings();
    installKeybindings();
}

private class JDCPopupAction extends AbstractAction {
    private int action;

    private JDCPopupAction(int action) {
        this.action = action;
    }

    public void actionPerformed(java.awt.event.ActionEvent actionEvent) {
        switch (action) {
            case ACTION_POPUP_HIDE:
                setVisible(false);
                break;
            case ACTION_COMPLETION_UP:
                if (completion.isVisible()){
                    getCompletionView().up();
                }else{
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_UP, 0));
                }
                break;
            case ACTION_COMPLETION_DOWN:
                if (completion.isVisible()){
                    getCompletionView().down();
                }else {
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_DOWN, 0));
                }
                break;
```

```
                case ACTION_COMPLETION_PGUP:
                    if (completion.isVisible()){
                        getCompletionView().pageUp();
                    } else{
                        performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP, 0));
                    }
                    break;
                case ACTION_COMPLETION_PGDN:
                    if (completion.isVisible()){
                        getCompletionView().pageDown();
                    }else{
                        performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_DOWN,
0));
                    }
                    break;
                case ACTION_COMPLETION_BEGIN:
                    if (completion.isVisible()){
                        getCompletionView().begin();
                    }else{
                        performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_HOME, 0));
                    }
                    break;
                case ACTION_COMPLETION_END:
                    if (completion.isVisible()){
                        getCompletionView().end();
                    }else{
                        performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_END, 0));
                    }
                    break;
                case ACTION_JAVADOC_UP:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_UP, 0));
                    break;
                case ACTION_JAVADOC_DOWN:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_DOWN, 0));
                    break;
                case ACTION_JAVADOC_PGUP:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_UP, 0));
                    break;
                case ACTION_JAVADOC_PGDN:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_PAGE_DOWN, 0));
                    break;
                case ACTION_JAVADOC_BEGIN:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_HOME, 0));
                    break;
                case ACTION_JAVADOC_END:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_END, 0));
                    break;
                case ACTION_JAVADOC_LEFT:
                    performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_LEFT, 0));
                    break;
                case ACTION_JAVADOC_RIGHT:
```

```
                performJavaDocAction(KeyStroke.getKeyStroke(KeyEvent.VK_RIGHT, 0));
                break;
            case ACTION_JAVADOC_BACK: {
                CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
                if (cjd!=null){
                    cjd.backHistory();
                }
            }
            break;
            case ACTION_JAVADOC_FORWARD: {
                CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
                if (cjd!=null){
                    cjd.forwardHistory();
                }
            }
            break;
            case ACTION_JAVADOC_OPEN_IN_BROWSER: {
                CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
                if (cjd!=null && cjd.isExternalJavaDocMounted()){
                    cjd.openInExternalBrowser();
                }
            }
            break;
            case ACTION_JAVADOC_OPEN_SOURCE: {
                CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
                if (cjd!=null){
                    cjd.goToSource();
                }
            }
            break;
            case ACTION_JAVADOC_INSERT_PRECONDITION: {
                CompletionJavaDoc cjd = extEditorUI.getCompletionJavaDoc();
                if (!cjd.getPrecondition().isEmpty()) {
                    cjd.insertPrecondition();
                }
            }
            break;
        }
    }
}
```

E.8 org.netbeans.editor.ext.CompletionJavaDoc.java

```
/*
 *                Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
```

```
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */


package org.netbeans.editor.ext;

import java.awt.Color;
import java.awt.Rectangle;
import java.lang.Comparable;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.lang.StringBuffer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.StringTokenizer;
import java.util.NoSuchElementException;

import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.JList;
import javax.swing.SwingUtilities;
import javax.swing.event.CaretEvent;
import javax.swing.event.CaretListener;
import javax.swing.text.JTextComponent;
import javax.swing.Timer;

import org.netbeans.editor.ext.java.JCClass;
import org.netbeans.editor.ext.java.JCField;
import org.netbeans.editor.ext.java.JCMethod;
import org.netbeans.editor.WeakTimerListener;
import org.netbeans.editor.SettingsChangeListener;
import org.netbeans.editor.Settings;
import org.netbeans.editor.Utilities;
import org.netbeans.editor.ext.ExtSettingsNames;
import org.netbeans.editor.ext.ExtSettingsDefaults;
import org.netbeans.editor.SettingsChangeEvent;
import org.netbeans.editor.SettingsUtil;
import org.netbeans.editor.ext.java.JCClass;
import org.netbeans.editor.LocaleSupport;
import java.util.List;
import org.netbeans.editor.ext.java.JCFinder;
import org.netbeans.editor.ext.java.JavaCompletion;
```

```
/**
 *  Support for javadoc in code completion.
 *  Contains also static utilities methods for preparing javadoc HTML content
 *
 *  @author  Martin Roskanin
 *  @since   03/2002
 */
public class CompletionJavaDoc implements ActionListener, SettingsChangeListener,
PropertyChangeListener  {

    /** Editor UI supporting this completion */
    protected ExtEditorUI extEditorUI;

    // javadoc browser history
    private List history = new ArrayList(5);

    private int curHistoryItem = -1;

    private JavaDocPane pane;
    private JavaDocView view;
    private int javaDocDelay;
    private Timer timer;
    protected Object currentContent;
    private ListSelectionListener completionListener;
    private boolean javaDocAutoPopup;
    private CaretListener caretL;
    private boolean showContract;
    private ArrayList preCondition;

    public static final String BUNDLE_PREFIX = "javadoc-tag-"; //NOI18N
    public static final String LOADING = "javadoc-loading"; //NOI18N

    private static ArrayList omittedTags = new ArrayList();
    static{
        omittedTags.add("@serial"); //NOI18N
        omittedTags.add("@serialField"); //NOI18N
        omittedTags.add("@serialData"); //NOI18N
        omittedTags.add("@author"); //NOI18N
        omittedTags.add("@version"); //NOI18N
        omittedTags.add("@beaninfo"); //NOI18N
    }

    private static ArrayList contractTags = new ArrayList();
    static{
        contractTags.add("@invariant"); //NOI18N
        contractTags.add("@pre"); //NOI18N
        contractTags.add("@post"); //NOI18N
    }

    /** Creates a new instance of CompletionJavaDoc */
    public CompletionJavaDoc(ExtEditorUI extEditorUI) {
```

```
this.extEditorUI = extEditorUI;

// Initialize timer
timer = new Timer(0, new WeakTimerListener(this)); // delay will be set later
timer.setRepeats(false);
Settings.addSettingsChangeListener(this);

javaDocDelay = getJavaDocDelay();
javaDocAutoPopup = getJavaDocAutoPopup();
preCondition = new ArrayList();
showContract = getShowContract();

// add completion listener
final ExtEditorUI extUI = extEditorUI;
class ClearTask implements Runnable {
    public void run(){
        Completion com = extUI.getCompletion();
        if (com != null && com.isPaneVisible()){
            CompletionJavaDoc completionJavaDoc = extUI.getCompletionJavaDoc();
            if (completionJavaDoc!=null){
                if(completionJavaDoc.autoPopup()){
                    completionJavaDoc.setContent(LocaleSupport.getString(LOADING));
                    clearHistory();
                    completionJavaDoc.setContent(com.getSelectedValue());
                    addToHistory(com.getSelectedValue());
                }else{
                    completionJavaDoc.setContent(null);
                }
            }
        }
    }
}

class SelectionObserver implements ListSelectionListener {
    public void valueChanged(ListSelectionEvent e){
        SwingUtilities.invokeLater(new ClearTask());
    }
};

completionListener = new SelectionObserver();

Completion completion = extEditorUI.getCompletion();
if (completion != null){
    if (completion.getView() instanceof JList){
        JList completionList = (JList)completion.getView();
        completionList.addListSelectionListener(completionListener);
    }
}

/**
 * Hides JavaDoc if completion is hidden.
```

```
         */
        class MyCaretListener implements CaretListener {
            public void caretUpdate( CaretEvent e ) {
                Completion com = extUI.getCompletion();
                if (com == null) return;
                JDCPopupPanel panel = com.getJDCPopupPanelIfExists();
                if (panel == null) return;
                if (panel.isVisible() && !com.isPaneVisible()){
                    setJavaDocVisible(false);
                }
            }
        }
        caretL = new MyCaretListener();

        synchronized (extEditorUI.getComponentLock()) {
            // if component already installed in ExtEditorUI simulate installation
            JTextComponent component = extEditorUI.getComponent();
            if (component != null) {
                propertyChange(new PropertyChangeEvent(extEditorUI,
                                                       ExtEditorUI.COMPONENT_PROPERTY, null,
component));
            }

            extEditorUI.addPropertyChangeListener(this);
        }

    }

    public void propertyChange(PropertyChangeEvent evt) {
        String propName = evt.getPropertyName();

        if (ExtEditorUI.COMPONENT_PROPERTY.equals(propName)) {
            JTextComponent component = (JTextComponent)evt.getNewValue();
            if (component != null) { // just installed
                component.addCaretListener( caretL );
            } else { // just deinstalled
                cancelPerformingThread();
                component = (JTextComponent)evt.getOldValue();

                if( component != null ) {
                    component.removeCaretListener( caretL );
                }
            }

        }
    }

    private JDCPopupPanel getJDCPopupPanel(){
        Completion completion = extEditorUI.getCompletion();
        if (completion != null){
            return completion.getJDCPopupPanelIfExists();
```

```
    }
    return null;
}

/** Returns JavaDoc popup pane */
public JavaDocPane getJavaDocPane(){
    Completion completion = extEditorUI.getCompletion();
    if (completion != null){
        return completion.getJDCPopupPanel().getJavaDocPane();
    }

    if (pane == null){
        pane = new ScrollJavaDocPane(extEditorUI);
    }
    return pane;
 }

/** Returns JavaDoc View */
public JavaDocView getJavaDocView(){
    if (view == null) {
        view = new HTMLJavaDocView(getJavaDocBGColor());
    }
    return view;
}

/** Sets javadoc popup window visibility */
public void setJavaDocVisible(final boolean visible){
    final JDCPopupPanel jdc = getJDCPopupPanel();
    if (jdc!=null){
        getJavaDocPane().setShowWebEnabled(isExternalJavaDocMounted());
        if (!SwingUtilities.isEventDispatchThread()){
            SwingUtilities.invokeLater(
            new Runnable() {
                public void run() {
                    jdc.setJavaDocVisible(visible);
                }
            });
        }else{
            jdc.setJavaDocVisible(visible);
        }
    }
}

public synchronized void addToHistory(Object url){
    int histSize = history.size();
    for (int i=curHistoryItem+1; i<histSize; i++){
        history.remove(history.size()-1);
    }
    history.add(url);
    curHistoryItem = history.size()-1;
    if (curHistoryItem > 0) getJavaDocPane().setBackEnabled(true);
```

```
        getJavaDocPane().setForwardEnabled(false);
    }

    public synchronized void backHistory(){
        if (curHistoryItem > 0) {
            curHistoryItem--;
            setContent(history.get(curHistoryItem), false);
            if (curHistoryItem == 0) getJavaDocPane().setBackEnabled(false);
            getJavaDocPane().setForwardEnabled(true);
        }
    }

    public synchronized void forwardHistory(){
        if (curHistoryItem <history.size()-1){
            curHistoryItem++;
            setContent(history.get(curHistoryItem), false);
            if (curHistoryItem == history.size()-1) getJavaDocPane().setForwardEnabled(false);
            getJavaDocPane().setBackEnabled(true);
        }
    }

    public synchronized void clearHistory(){
        curHistoryItem = -1;
        history.clear();
        getJavaDocPane().setBackEnabled(false);
        getJavaDocPane().setForwardEnabled(false);
    }

    public boolean isVisible(){
        return getJavaDocPane().getComponent().isVisible();
    }

    /** Interrupts timer that is responsible for delayed popup of javadoc window */
    public void cancelPerformingThread(){
        timer.stop();
    }

    /** Sets content of javadoc
     *  @param content it is Object of the java member such as JCClass, JCMethod, JCField or
JCConstructor
     *  @param postRequest if false, javadoc window is popuped without delay
     */
    public synchronized void setContent(Object content, boolean postRequest){
        timer.stop();
        if (content == null) {
            setJavaDocVisible(false);
            return;
        }
        currentContent = content;
        if (postRequest){
            timer.setInitialDelay(javaDocDelay);
```

```
        timer.setDelay(javaDocDelay);
        timer.start();
    }else{
        actionPerformed(null);
    }

}

/** Sets content of javadoc with postRequest setted to true
 *  @see #setContent(java.lang.Object, boolean)
 */
public void setContent(Object content){
    setContent(content, true);
}

/** Immediately sets Content of javadoc withou popup delay
 *  @param content String representation of the displayed text.
 *      In the case of current implementation it is an HTML document
 *      Can be <code>null</code> in this case javaDoc popup will be hidden
 */
public void setContent(String content){
    if (content == null){
        setJavaDocVisible(false);
        return;
    }
    getJavaDocView().setContent(content);
}

/**
 * Invoked when an action occurs.
 */
public synchronized void actionPerformed(ActionEvent e) {
    //[PENDING] - javaDoc for standalone editor
}

/** Retrieve a background color of javadoc from options */
private Color getJavaDocBGColor(){
    Class kitClass = Utilities.getKitClass(extEditorUI.getComponent());
    if (kitClass != null) {
        return (Color)SettingsUtil.getValue(kitClass,
                  ExtSettingsNames.JAVADOC_BG_COLOR,
                  ExtSettingsDefaults.defaultJavaDocBGColor);
    }
    return ExtSettingsDefaults.defaultJavaDocBGColor;
}

/** Retrieve a javadoc popup delay from options */
private int getJavaDocDelay(){
    Class kitClass = Utilities.getKitClass(extEditorUI.getComponent());
    if (kitClass != null) {
        return ((Integer)SettingsUtil.getValue(kitClass,
```

```
                ExtSettingsNames.JAVADOC_AUTO_POPUP_DELAY,
                ExtSettingsDefaults.defaultJavaDocAutoPopupDelay)).intValue();
    }
    return ExtSettingsDefaults.defaultJavaDocAutoPopupDelay.intValue();
}

/** Retrieve a auto popup of javadoc property from options */
private boolean  getJavaDocAutoPopup(){
    Class kitClass = Utilities.getKitClass(extEditorUI.getComponent());
    if (kitClass != null) {
        return ((Boolean)SettingsUtil.getValue(kitClass,
                ExtSettingsNames.JAVADOC_AUTO_POPUP,
                ExtSettingsDefaults.defaultJavaDocAutoPopup)).booleanValue();
    }
    return ExtSettingsDefaults.defaultJavaDocAutoPopup.booleanValue();
}

/** Returns whether javadoc window should be invoked automatically */
public boolean autoPopup(){
    return javaDocAutoPopup;
}

public void settingsChange(SettingsChangeEvent evt) {
    if (ExtSettingsNames.JAVADOC_BG_COLOR.equals(evt.getSettingName())){
        getJavaDocView().setBGColor(getJavaDocBGColor());
    }

    if (ExtSettingsNames.JAVADOC_AUTO_POPUP_DELAY.equals(evt.getSettingName())){
        javaDocDelay = getJavaDocDelay();
    }

    if (ExtSettingsNames.JAVADOC_AUTO_POPUP.equals(evt.getSettingName())){
        javaDocAutoPopup = getJavaDocAutoPopup();
    }

    if (ExtSettingsNames.SHOW_CONTRACT.equals(evt.getSettingName())) {
        showContract = getShowContract();
    }

}

/** Prepares raw javadoc content
 *  @param cls root class
 *  @param member String representation of field or method
 *  @param content raw javadoc content text
 */
public String prepareJavaDocContent(JCClass cls, String member, String content){
    return prepareJavaDocContent(cls, member, content, null);
}

/** Checks whether a tag should be omitted in javadoc */
```

```
    public boolean isOmittedJavaDocTag(String tag){
        return omittedTags.contains(tag);
    }

    /** Parses given link such as <code>java.awt.Component#addHierarchyListener</code>
     *  and returns parsed Object
     *  @return Object of JCClass, JCMethod, JCConstructor or JCField
     */
    public Object parseLink(String link, JCClass cls){
        return null;
    }

    protected boolean isNotFullyQualifiedInnerClass (String inner, JCClass cls){
        if (inner.indexOf(".") != inner.lastIndexOf(".")){ //NOI18N
            // fully qualified
            return false;
        }
        String pkgName = cls.getPackageName();
        return (JavaCompletion.getFinder().getExactClass( (pkgName.length()>0) ?
(pkgName+"."+inner) : inner ) != null); //NOI18N
    }

    protected String findProperClass(String name, JCClass cls){
        if (cls == null) return null;
        String ret = null;

        JCClass retClass;
        if (cls.getPackageName()!=null && cls.getPackageName().length()>0){
            retClass =
JavaCompletion.getFinder().getExactClass(cls.getPackageName()+"."+name); //NOI18N
        }else{
            retClass = JavaCompletion.getFinder().getExactClass(name);
        }

        if (retClass != null){
            return retClass.getFullName();
        }

        List classes = JavaCompletion.getFinder().findClasses(null, name, true);
        if (classes.size()>0){
            ret = ((JCClass)classes.get(0)).getFullName();
        }

        return ret;
    }

    /** Creates a link from given javadoc @see or @link parameter and root class
     *  @param prm link parameter such as <code>java.awt.Component#addHierarchyListener</code>
     *  @param cls root class to which link a link parameter without class specification like
<code>#addHierarchyListener</code>
     */
```

```
    public String createAnchor(String prm, JCClass cls){
        StringBuffer ret= new StringBuffer(prm.length() + 50); //NOI18N
        String label = ""; //NOI18N
        int lastParenthesisOccurence = prm.lastIndexOf(")"); //NOI18N

        boolean linkedObjectExists = (parseLink(prm, cls) != null );

        int indexOfSpaceAfterLastParenthesis = prm.indexOf(" ",lastParenthesisOccurence);
        if ( indexOfSpaceAfterLastParenthesis > 0) label =
prm.substring(indexOfSpaceAfterLastParenthesis+1);

        if (prm.startsWith("#")){ //NOI18N
            if (linkedObjectExists){
                ret.append("<a href='").append(cls.getFullName()).append(((label.length()>0) ?
prm.substring(0,indexOfSpaceAfterLastParenthesis-1):prm)).append( //NOI18N
                "'><code>").append(((label.length() >
0)?label:prm.substring(1))).append("</code></a>"); //NOI18N
            }else{
                ret.append("<code>").append(((label.length() >
0)?label:prm.substring(1))).append("</code>"); //NOI18N
            }

        }else{
            if (linkedObjectExists){
                String href =
((label.length()>0)?prm.substring(0,indexOfSpaceAfterLastParenthesis):prm);
                if (href.indexOf(".")<0 || isNotFullyQualifiedInnerClass(href, cls)){ //NOI18N
                    // stick a package before
                    String refCls = (href.indexOf("#") > 0) ? href.substring(0,
href.indexOf("#")) : href; //NOI18N
                    String refMember = (href.indexOf("#") > 0) ?
href.substring(href.indexOf("#")+1) : ""; //NOI18N
                    String fullClassName = findProperClass(refCls, cls);
                    if (fullClassName != null){
                        href = (refMember.length()>0) ? fullClassName+"#"+refMember :
fullClassName; //NOI18N
                    }else{
                        return "<a href='"+href+"'><code>"+ //NOI18N
                        ((label.length() > 0)?label:prm.replace('#','.'))+"</code></a>";
//NOI18N
                    }
                }
                ret.append("<a href='").append(href).append("'><code>").append( //NOI18N
                ((label.length() > 0)?label:prm.replace('#','.'))).append("</code></a>");
//NOI18N
            }else{
                ret.append("<code>").append(((label.length() >
0)?label:prm.replace('#','.'))).append("</code>"); //NOI18N
            }
        }
        return ret.toString();
```

```
    }

    /** Processes given javadoc content and replaces @link parameters to anchors
     *  @param content raw javadoc content
     *  @param cls root class
     */
    public String createLinks(String content, JCClass cls){
        StringBuffer ret = new StringBuffer(content.length());
        StringTokenizer strTok = new StringTokenizer(content ,"{"); //NOI18N
        String next=""; //NOI18N
        boolean linkAppended = false;
        boolean useOriginalContent = true;

        while (strTok.hasMoreTokens()){
            next = strTok.nextToken("{"); //NOI18N
            if (strTok.hasMoreTokens()) {
                String link = strTok.nextToken("}"); //NOI18N
                linkAppended = false;
                if (link.indexOf("@link ") > 0) { //NOI18N
                    useOriginalContent = false;
                    int linkPos = link.indexOf("@link ")+6; //NOI18N
                    if (next.startsWith("}")) next = next.substring(1); //NOI18N
                    link = createAnchor(link.substring(linkPos), cls);
                    linkAppended = true;
                }

                ret.append(next);
                ret.append(link);
            } else {
                if (linkAppended && next.startsWith("}")) next = next.substring(1); //NOI18N
                ret.append(next);
            }
        }
        return useOriginalContent ? content : ret.toString();
    }

     /** Prepares raw javadoc content with given javadoc parameters
     *  @param cls root class
     *  @param member String representation of field or method
     *  @param content raw javadoc content text
     *  @param tags javadoc tag items array
     */
    public String prepareJavaDocContent(JCClass cls, String member, String content,
CompletionJavaDoc.JavaDocTagItem tags[]){
        if (content != null){
            content = createLinks(content, cls);
        }else{
            content = ""; //NOI18N
        }
        StringBuffer sb = new StringBuffer(512);
        clearPrecondition();
```

```
        //sb.insert(0,"<font size='4'><code><b>"+cls.toString()+"</b></code></font>");
//NOI18N
        sb.append("<font
size='+0'><b>").append(createAnchor(cls.getFullName(),cls)).append("</b></font>"); //NOI18N
        if (member.length()>0) sb.append("<pre>").append(member).append("</pre>"); //NOI18N
        if(!showContract()) {
            sb.append("<blockquote>").append(content).append("</blockquote>"); //NOI18N
        }
        if(sb.indexOf("</pre>") ==  -1) {
            sb.append("<br>");
        }
        if (tags!=null){
            Arrays.sort(tags);
            tags = sortContractTags(tags);
            String curParam = ""; //NOI18N
            int curItem = 0;
            for (int i=0; i<tags.length; i++){
                if (!curParam.equals(tags[i].getName())){
                    if ((isOmittedJavaDocTag(tags[i].getName()) || showContract()) &&
!isContractTag(tags[i].getName())) {
                        continue;
                    }

                    if(!curParam.equals("")) sb.append("</blockquote>"); //NOI18N

sb.append("<b>").append(LocaleSupport.getString(BUNDLE_PREFIX+tags[i].getName(),tags[i].getNam
e())).append("</b><blockquote>"); //NOI18N
                    curParam = tags[i].getName();
                    curItem = 0;
                }

                if (curItem>0)
                   sb.append("<br>"); //NOI18N


                if (curParam.equals("@param")){ //NOI18N
                    String prm = tags[i].getText();
                    int spacePos = prm.indexOf(" "); //NOI18N
                    if (spacePos>0){
                        String anchoredText = createLinks(prm.substring(spacePos), cls);
                        sb.append("<code>").append(prm.substring(0,spacePos)).append("</code>
- ").append(anchoredText); //NOI18N
                    }else{
                        sb.append(prm);
                    }
                }else if (curParam.equals("@see")){ //NOI18N
                    String txt = tags[i].getText();
                    if (txt.startsWith("\"")){ //NOI18N
                        txt = txt.substring(1);
```

```
                        if (txt.endsWith("\"")) txt = txt.substring(0,txt.length()-1);
//NOI18N
                        sb.append(txt);
                    }else if(txt.startsWith("<")){ //NOI18N
                        sb.append(txt);
                    }else{
                        sb.append(createAnchor(tags[i].getText(),cls));
                    }
                }else{
                    sb.append(createLinks(tags[i].getText(), cls));
                }
                if(curParam.equals("@pre")) { //NOI18N
                    preCondition.add(tags[i].getText());
                    getJavaDocPane().setInsertPreEnabled(true);
                }

                curItem++;
            }
            if(!curParam.equals("")) sb.append("</blockquote>"); //NOI18N
        }

        return sb.toString();
    }

    public void goToSource(){
    }

    public void openInExternalBrowser(){
    }

    public boolean isExternalJavaDocMounted(){
        return false;
    }


    public interface JavaDocTagItem extends Comparable{
        public String getName();
        public String getText();
    }

    /** Removes previously stored precondition. */
    public synchronized void clearPrecondition() {
        preCondition.clear();
        getJavaDocPane().setInsertPreEnabled(false);
    }

    /** Returnes the stored raw precondition.
     * @return the raw precondition.
     */
    public ArrayList getPrecondition() {
        return preCondition;
```

```
    }

    /** Retrieves the show contract property from options.
     * @return true if only show contract tags is set; false otherwise.
     */
    private boolean  getShowContract() {
        Class kitClass = Utilities.getKitClass(extEditorUI.getComponent());
        if (kitClass != null) {
            return ((Boolean)SettingsUtil.getValue(kitClass,
                      ExtSettingsNames.SHOW_CONTRACT,
                      ExtSettingsDefaults.defaultShowContract)).booleanValue();
        }
        return ExtSettingsDefaults.defaultShowContract.booleanValue();
    }

    /** Checks whether a tag is an contract tag i.e. @pre, @post, or @invariant.
     * @param tag the tag to be checked.
     * @return true if the tag is a contract tag; false otherwise.
     */
    public boolean isContractTag(String tag) {
        return contractTags.contains(tag);
    }

    public void insertPrecondition() {
    }

    /** Checks whether only contract should be shown in javadoc window.
     * @return true if only contract should be shown; false otherwise.
     */
    public boolean showContract() {
        return showContract;
    }

    /** Sorts the tags, making the @pre tags appear before the @post tags.
     * @param tags[] the array of tags to be sorted.
     * @return the sorted array of tags.
     */
    private CompletionJavaDoc.JavaDocTagItem[]
sortContractTags(CompletionJavaDoc.JavaDocTagItem tags[]) {
        int precur = 0;
        int postcur = 0;
        ArrayList list = new ArrayList(Arrays.asList(tags));
        CompletionJavaDoc.JavaDocTagItem ret[] = tags;

        for(int i = 0; i < tags.length; i++) {
            if(tags[i].getName().equals("@pre")) {
                CompletionJavaDoc.JavaDocTagItem temp =
(CompletionJavaDoc.JavaDocTagItem)list.get(i);
                list.remove(i);
                list.add(precur, temp);
                precur++;
```

```
            }
            else if(tags[i].getName().equals("@post")) {
                CompletionJavaDoc.JavaDocTagItem temp =
(CompletionJavaDoc.JavaDocTagItem)list.get(i);
                list.remove(i);
                list.add(precur + postcur, temp);
                postcur++;
            }
        }

        for(int i = 0; i < list.size(); i++) {
            ret[i] = (CompletionJavaDoc.JavaDocTagItem)list.get(i);
        }

        return ret;
    }
}
```

E.9 org.netbeans.modules.editor.java.Bundle.properties

```
#                 Sun Public License Notice
#
# The contents of this file are subject to the Sun Public License
# Version 1.0 (the "License"). You may not use this file except in
# compliance with the License. A copy of the License is available at
# http://www.sun.com/
#
# The Original Code is NetBeans. The Initial Developer of the Original
# Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
# Microsystems, Inc. All Rights Reserved.
#
# Contributor(s): Håkan Bergmark, Tony Bergh

JC_title=Update Code Completion Database
JC_progress_title=Updating Code Completion Database
JC_db_prefix=Code Completion Database File Name
JC_db_prefix.mnemonic=P
JC_title_tooltip=Two files will appear (with .jcs and .jcb suffixes) in the system/ParserDB
directory.
JC_class_level=Classes
JC_field_level=Fields
JC_method_level=Methods
JC_storage_levels=Storage Levels
JC_auto_update=\ Automatic Update
JC_auto_update_mnemonic=A
JC_mount_filesystem=\ Mount as Filesystem
JC_mount_filesystem_mnemonic=M
JC_inspecting=Inspecting folders
CTL_JC_level_package=\ All Except Private
CTL_JC_level_private=\ All
CTL_JC_level_protected=\ Protected and Public
```

```
CTL_JC_level_public=\ Public
ask_overwrite=Code Completion DB file with name {0} already exists. Overwrite?
JC_building_class=Building Class List
JC_init_parser=Invoking parser
JC_updating_db=Updating Database
JC_entries=entries
JC_updating_memory=Updating Memory Database
JC_classes_done=folders processed
JC_classes_found=folders found
JC_classes_done_of=of
JC_creating_parserDB_status_displayer=Creating code completion DB for the filesystem {0}...
JC_parsing_finished_status_displayer=Code completion DB has been created for the filesystem
{0}.
JC_parsing_cancelled_status_displayer=Creation of the code completion DB for the filesystem
{0} has been cancelled.
ERR_Wrong_mounting_point=Class \"{0}\" claims to exist in package \"{1}\", yet it is mounted
under package \"{2}\". The package \"{2}\" will not be updated."
ACSD_CTL_JC_level_package=
ACSD_CTL_JC_level_private=
ACSD_CTL_JC_level_protected=
ACSD_CTL_JC_level_public=
ACSD_JC=
ACSD_JC_progress=
ACSD_JC_AUTOUPDATE=
ACSD_JC_MOUNT=

update_action=U&pdate Code Completions...
ccdbm_action=Code Completion &Database Manager

CTL_Watch_Title=New Watch
CTL_Watch_Name=Watch Expression:
CTL_Watch_Name_Mnemonic=W
ACSD_CTL_Watch_Name=
ACSD_WatchPanel=

NAME_JavaFastOpenAction=Go &To Class...
MSG_JavaFastOpen=The Go To Class feature can find and open a file only if its filesystem is
mounted in the Explorer and the Code Completion Database has been updated for that
filesystem.\n\nTo update the Code Completion Database, right-click in the Explorer on the
package or filesystem containing the classes you want to add to the database, then choose
Tools > Update Code Completions... from the contextual menu.
CTL_DNSTDNT=\ Do Not Show This Dialog Next Time
CTL_DNSTDNT_Mnemonic=D
ACSD_CTL_DNSTDNT=

show_javadoc=Show Javadoc
cannot_find_javadoc=Cannot find javadoc. Check the Javadoc Manager to make sure the
documentation is mounted.
generate-goto-popup=Go To
goto_source_of=Go to Source of
goto_source_explore_package=Explore package {0}
```

```
goto_source_open_source=Source ({0})
goto_source_open_source_not_formatted=Source
goto_source_package_not_found=Package {0} not found in Filesystems.
goto_source_source_not_found=Source {0} not found in Filesystems.
LAB_JavaIndentEngine=Java Indentation Engine
ask_update=Do you want to create a Code Completion Database for filesystem: \n {0}?
HINT_JavaIndentEngine=Indentation engine for Java files.
PROP_indentEngine_javaFormatNewlineBeforeBrace=Add Newline Before Brace
HINT_indentEngine_javaFormatNewlineBeforeBrace=If True, put curly braces on the following
line.
PROP_indentEngine_javaFormatSpaceBeforeParenthesis=Add Space Before Parenthesis
HINT_indentEngine_javaFormatSpaceBeforeParenthesis=If True, add a space before opening
parenthesis.
PROP_indentEngine_javaFormatLeadingStarInComment=Add Leading Star in Comment
HINT_indentEngine_javaFormatLeadingStarInComment=If True, add asterisk to lines beginning
inside block comments.

HINT_javadoc_browser_back_button=Go to previous page
HINT_javadoc_browser_forward_button=Go to next page
HINT_javadoc_browser_show_web_button=Show javadoc in external web browser
HINT_javadoc_browser_goto_source_button=Open source in editor
HINT_javadoc_browser_insert_pre_button=Insert precondition test

javadoc_content_not_found=<hr size=1 color=black noshade><font color=\"#7c0000\">Javadoc not
found.</font> Either Javadoc documentation for this item does not exist or you have not
mounted a filesystem that contains the source for this class. The Javadoc documentation that
is displayed in this box is generated from source files, not from compiled Javadoc
documentation.


#Code Completion Database Customizer
PDCD_PREFIX=Name
PDCD_FILESYSTEM=Filesystem
PDCD_AUTOUPDATE=Autoupdate
PDCD_STATUS=Status
PDCD_STATUS_PARSING=Creating...
PDCD_STATUS_QUEUED=Queued
PDCD_STATUS_CANCELLED=Cancelled
PDCD_FILES=Code Completion Database Files:
PDCD_FILES_Mnemonic=F
PDCD_UPDATE=Update...
PDCD_UPDATE_Mnemonic=U
PDCD_DELETE=Delete
PDCD_DELETE_Mnemonic=D
PDCD_ADD=Add...
PDCD_ADD_Mnemonic=A
PDCD_STOP=Stop
PDCD_STOP_Mnemonic=S
PDCD_PDCUSTOMIZER=Code Completion Database Manager
PDCD_CLOSE=Close
PDCD_CLOSE_Mnemonic=C
```

```
PDC_JarArchivesMask=JAR and ZIP Archives
PDC_ChooserTitle=Add
PDC_SelectButton=Select
PDC_SelectButton_Mnemonic=S
PDC_DeleteQuestion=Are you sure you want to delete {0} Code Completion Database file?
PDC_DeleteDialogTitle=Confirm Deletion
ACSD_PDCD_ADD=
ACSD_PDCD_UPDATE=
ACSD_PDCD_DELETE=
ACSD_PDCD_TABLE=
ACSD_PDCD_CLOSE=
ACSD_PDCD_PANEL=

#AutoUpdateQuestionPanel
AUQP_DO_NOT_SHOW_NEXT_TIME=\ Do Not Show This Dialog Next Time
AUQP_DO_NOT_SHOW_NEXT_TIME_Mnemonic=D
AUQP_YES=\ Yes
AUQP_YES_Mnemonic=Y
AUQP_NO=\ No
AUQP_NO_Mnemonic=N
AUQP_UPDATE_QUESTION=Do you want to include newly mounted filesystem \"{0}\" in the Code
Completion Database?
AUQP_DELETE_QUESTION=Do you also want to delete Code Completion Database file for unmounted
filesystem \"{0}\"?


java-desc-goto-help=Goto Javadoc
```

E.10 org.netbeans.modules.editor.java.NbScrollJavaDocPane.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.modules.editor.java;

import javax.swing.JToolBar;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.Rectangle;
import org.netbeans.editor.ext.ScrollJavaDocPane;
import java.awt.event.MouseAdapter;
import javax.swing.JButton;
import java.awt.event.MouseEvent;
import javax.swing.ImageIcon;
import org.netbeans.editor.ext.CompletionJavaDoc;
import org.netbeans.editor.ext.ExtEditorUI;
import org.openide.util.NbBundle;

/**
```

```
 *
 *  @author  Martin Roskanin
 *  @since   03/2002
 */
public class NbScrollJavaDocPane extends ScrollJavaDocPane {

    private JToolBar toolbar;
    private ImageIcon iBack, iForward, iGoToSource, iShowWeb, iInsertPre;
    private JButton bBack, bForward, bGoToSource, bShowWeb, bInsertPre;
    private static final String BACK = "org/netbeans/modules/editor/resources/back.gif";
//NOI18N
    private static final String FORWARD = "org/netbeans/modules/editor/resources/forward.gif";
//NOI18N
    private static final String GOTO_SOURCE =
"org/netbeans/modules/editor/resources/gotosource.gif"; //NOI18N
    private static final String SHOW_WEB =
"org/netbeans/modules/editor/resources/htmlView.gif"; //NOI18N
    private static final String INSERT_PRE =
"org/netbeans/modules/editor/resources/insert_pre.gif"; //NOI18N
    private  CompletionJavaDoc cjd;

    /** Creates a new instance of NbScrollCompletionPane */
    public NbScrollJavaDocPane(ExtEditorUI extEditorUI) {
        super(extEditorUI);
        cjd = extEditorUI.getCompletionJavaDoc();
    }


    protected ImageIcon resolveIcon(String res){
        return new ImageIcon(org.openide.util.Utilities.loadImage (res));
    }


    protected void installTitleComponent() {
        toolbar = new JToolBar();
        toolbar.setFloatable(false);

        toolbar.setBorder(new javax.swing.border.EmptyBorder(new java.awt.Insets(1, 2, 1,
2)));
        toolbar.setLayout(new GridBagLayout());
        GridBagConstraints gdc = new GridBagConstraints();
        gdc.gridx = 0;
        gdc.gridy = 0;
        gdc.anchor = GridBagConstraints.WEST;

        iBack = resolveIcon(BACK);
        if (iBack !=null){

            bBack = new BrowserButton(iBack);
            bBack.addMouseListener(new MouseEventListener(bBack));
            bBack.setEnabled(false);
            bBack.setMargin(new Insets(0, 0, 0, 0));

bBack.setToolTipText(NbBundle.getBundle(JavaKit.class).getString("HINT_javadoc_browser_back_bu
tton")); //NOI18N
            toolbar.add(bBack, gdc);
        }

        gdc.gridx = 1;
        gdc.gridy = 0;
        gdc.anchor = GridBagConstraints.WEST;

        iForward = resolveIcon(FORWARD);
        if (iForward !=null){
            bForward = new BrowserButton(iForward);
            bForward.addMouseListener(new MouseEventListener(bForward));
            bForward.setEnabled(false);

bForward.setToolTipText(NbBundle.getBundle(JavaKit.class).getString("HINT_javadoc_browser_forw
ard_button")); //NOI18N
            bForward.setMargin(new Insets(0, 0, 0, 0));
            toolbar.add(bForward, gdc);
        }

        gdc.gridx = 2;
        gdc.gridy = 0;
        gdc.anchor = GridBagConstraints.WEST;
```

```
        iShowWeb = resolveIcon(SHOW_WEB);
        if (iShowWeb !=null){

            bShowWeb = new BrowserButton(iShowWeb);

bShowWeb.setToolTipText(NbBundle.getBundle(JavaKit.class).getString("HINT_javadoc_browser_show
_web_button")); //NOI18N
            bShowWeb.addMouseListener(new MouseEventListener(bShowWeb));
            bShowWeb.setMargin(new Insets(0, 0, 0, 0));
            toolbar.add(bShowWeb, gdc);
        }

        gdc.gridx = 3;
        gdc.gridy = 0;
        gdc.anchor = GridBagConstraints.WEST;

        iGoToSource = resolveIcon(GOTO_SOURCE);
        if (iGoToSource !=null){
            bGoToSource = new BrowserButton(iGoToSource);

bGoToSource.setToolTipText(NbBundle.getBundle(JavaKit.class).getString("HINT_javadoc_browser_g
oto_source_button")); //NOI18N
            bGoToSource.addMouseListener(new MouseEventListener(bGoToSource));
            bGoToSource.setMargin(new Insets(0, 0, 0, 0));
            toolbar.add(bGoToSource, gdc);
        }

        gdc.gridx = 4;
        gdc.gridy = 0;
        gdc.weightx = 1.0;
        gdc.anchor = GridBagConstraints.WEST;

        iInsertPre = resolveIcon(INSERT_PRE);
        if(iInsertPre != null) {
            bInsertPre = new BrowserButton(iInsertPre);
            bInsertPre.setEnabled(false);

bInsertPre.setToolTipText(NbBundle.getBundle(JavaKit.class).getString("HINT_javadoc_browser_in
sert_pre_button")); //NOI18N
            bInsertPre.addMouseListener(new MouseEventListener(bInsertPre));
            bInsertPre.setMargin(new Insets(0, 0, 0, 0));
            toolbar.add(bInsertPre, gdc);
        }

        add(toolbar);
    }

    public void setBounds(Rectangle r){
        super.setBounds(r);
        scrollPane.setBounds(r.x, 25, r.width+1, r.height - 25);
        toolbar.setBounds(r.x+1, 1, r.width-2, 24);
    }


    public void setForwardEnabled(boolean enable) {
        bForward.setEnabled(enable);
    }

    public void setBackEnabled(boolean enable) {
        bBack.setEnabled(enable);
    }

    public void setShowWebEnabled(boolean enable) {
        bShowWeb.setEnabled(enable);
    }

    public void setInsertPreEnabled(boolean enable) {
        bInsertPre.setEnabled(enable);
    }

    class MouseEventListener extends MouseAdapter {
        JButton button;
        MouseEventListener(JButton button) {
            this.button = button;
        }
```

```
        public void mouseEntered(MouseEvent ev) {
            if (button.isEnabled()){
                button.setBorderPainted(true);
            }
        }
        public void mouseExited(MouseEvent ev) {
            button.setBorderPainted(false);
        }

        public void mouseClicked(MouseEvent evt) {
            if (button.equals(bBack)){
                cjd.backHistory();
            }else if(button.equals(bForward)){
                cjd.forwardHistory();
            }else if(button.equals(bGoToSource)){
                cjd.goToSource();
            }else if (button.equals(bShowWeb)){
                cjd.openInExternalBrowser();
            }else if(button.equals(bInsertPre)) {
                cjd.insertPrecondition();
            }
        }
    }
}
```

E.11 org.netbeans.modules.editor.java.NbCompletionJavaDoc.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */


package org.netbeans.modules.editor.java;

import java.awt.event.ActionEvent;
import java.beans.PropertyVetoException;
import java.io.File;
import java.io.IOException;
import java.lang.IllegalArgumentException;

import org.openide.src.JavaDoc;
import org.openide.src.ClassElement;
import org.openide.util.RequestProcessor;
import org.openide.src.Identifier;
import org.openide.src.FieldElement;
import org.openide.src.Type;
import org.openide.src.MethodElement;
import org.openide.src.ConstructorElement;
import org.openide.src.JavaDocTag;

import org.netbeans.editor.ext.CompletionJavaDoc;
import org.netbeans.editor.ext.ExtEditorUI;
import org.netbeans.editor.ext.java.JCField;
import org.netbeans.editor.ext.java.JCMethod;
import org.netbeans.editor.ext.java.JCClass;
import org.netbeans.editor.ext.java.JCParameter;
import org.netbeans.editor.ext.java.JCConstructor;
import java.util.List;
import java.util.ArrayList;
import java.util.StringTokenizer;
import java.util.Map;
import java.util.HashMap;
import org.netbeans.editor.ext.java.JCFinder;
```

```
import org.netbeans.editor.ext.java.JavaCompletion;
import java.net.URL;
import java.util.Enumeration;
import org.netbeans.modules.editor.java.NbJavaSyntaxSupport;
import org.netbeans.modules.editor.NbEditorUtilities;
import org.netbeans.editor.Utilities;
import org.netbeans.editor.SyntaxSupport;
import org.openide.filesystems.FileObject;
import org.openide.filesystems.FileSystem;
import org.openide.filesystems.FileSystemCapability;
import org.openide.filesystems.JarFileSystem;
import org.openide.filesystems.Repository;
import org.openide.util.NbBundle;

/**
 *
 *  @author  Martin Roskanin
 *  @since 03/2002
 */
public class NbCompletionJavaDoc extends CompletionJavaDoc{

    ParsingThread task = null;
    Map lookupCache;
    private static final String CONTENT_NOT_FOUND =
NbBundle.getMessage(NbCompletionJavaDoc.class, "javadoc_content_not_found"); //NOI18N
    private static boolean javaSourcesMounted;
    private static boolean inited = false;

    /** Creates a new instance of NbCompletionJavaDoc */
    public NbCompletionJavaDoc(ExtEditorUI extEditorUI) {
        super(extEditorUI);
        lookupCache = new HashMap(89);
        RequestProcessor.postRequest(new Runnable(){
            public void run(){
                Enumeration e = Repository.getDefault().getFileSystems();
                while (e.hasMoreElements()) {
                    FileSystem fs = (FileSystem)e.nextElement();
                    if (fs.findResource("java/lang/String.java") !=null){ //NOI18N
                        javaSourcesMounted = true;
                        break;
                    }
                }
                inited = true;
            }
        });
    }

    private CompletionJavaDoc.JavaDocTagItem[] getJavaDocTags(JavaDocTag jdt[]){
        CompletionJavaDoc.JavaDocTagItem ret [] = new
CompletionJavaDoc.JavaDocTagItem[jdt.length];
        for (int i=0; i<jdt.length; i++){
            ret[i] = new JavaDocTagItemImpl(jdt[i].name(),jdt[i].text());
        }
        return ret;
    }

    private void mountSources(){
        if (inited == false) return;
        if (javaSourcesMounted) return;
        javaSourcesMounted = true;
        try{
            String javaVersion = System.getProperty("java.version"); //NOI18N
            if (javaVersion.indexOf("1.4") != -1){ //NOI18N
                String fileSeparator = System.getProperty("file.separator"); //NOI18N
                String javaRuntimeRoot = System.getProperty("java.home") + fileSeparator;
//NOI18N
                String javaRoot = javaRuntimeRoot + ".." + fileSeparator; //NOI18N

                File srcFile = new File(javaRoot+"src.zip"); //NOI18N
                if (!srcFile.exists()){
                    srcFile = new File(javaRoot+"src.jar"); //NOI18N
                    if (!srcFile.exists()) return;
                }
                FileSystemCapability.Bean cap = new FileSystemCapability.Bean ();
                cap.setCompile (false);
                cap.setDebug (false);
                cap.setExecute (false);
```

```
                cap.setDoc (false);

                JarFileSystem  mountedFS = new JarFileSystem(cap);
                mountedFS.setJarFile(srcFile);
                mountedFS.setHidden(true);
                if (Repository.getDefault().findFileSystem(mountedFS.getSystemName()) == null)
{
                    Repository.getDefault().addFileSystem(mountedFS);
                }
            }
        }catch(IOException ioe){
        }catch(PropertyVetoException pve){
        }
    }

    private ClassElement getClassElement(String clsFullName){
            ClassElement ce=null;
            if (clsFullName == null) return null;
            try {
                ce = ClassElement.forName(clsFullName);
                if (ce == null && !javaSourcesMounted){
                    //mount java sources
                    mountSources();
                    ce = ClassElement.forName(clsFullName);
                }
            } catch ( ThreadDeath td ) {
                throw td;
            } catch (Throwable t) { // Parser sometimes sensitive to forName call !!!
                System.err.println("Error occurred during name resolving"); // NOI18N
                t.printStackTrace();
            }
            return ce;
    }


    private ParsingThread setInRequestProcessorThread(final Object content){
        ParsingThread pt = new ParsingThread(content);
        RequestProcessor.postRequest(pt);
        return pt;
    }

    public void cancelPerformingThread(){
        super.cancelPerformingThread();
        if (task!=null){
            task.stopTask();
        }
    }

    public synchronized void actionPerformed(ActionEvent e) {
        if (task!=null){
            task.stopTask();
        }
        task = setInRequestProcessorThread(currentContent);
    }

    /** Opens source of current javadoc in editor */
    public void goToSource(){
        SyntaxSupport sup = Utilities.getSyntaxSupport(extEditorUI.getComponent());
        NbJavaSyntaxSupport nbJavaSup =
(NbJavaSyntaxSupport)sup.get(NbJavaSyntaxSupport.class);
        nbJavaSup.openSource(currentContent, false, false);
    }

    /** Opens javadoc in external browser */
    public void openInExternalBrowser(){
        SyntaxSupport sup = Utilities.getSyntaxSupport(extEditorUI.getComponent());
        NbJavaSyntaxSupport nbJavaSup =
(NbJavaSyntaxSupport)sup.get(NbJavaSyntaxSupport.class);
        URL[] urls = nbJavaSup.getJavaDocURLs(currentContent);

        if (urls != null && urls.length > 0) {
            org.openide.awt.HtmlBrowser.URLDisplayer.getDefault().showURL(urls[0]); // show
first URL
        }
    }

    /** Gets the object reference for a method invocation. Either the object
```

```
     * reference is explicit or implicit i.e. 'this'.
     * @param pos the carets' position in the document.
     * @return the object reference for the method invocation.
     */
    public String getObjectReference(int pos) throws Exception {
        int start = Utilities.getWordStart(extEditorUI.getDocument(), pos);
        String id = Utilities.getIdentifier(extEditorUI.getDocument(), start - 1);

        if((id == null) || (id.equals("="))) { //NOI18N
            return "this"; //NOI18N
        }else if(id.equals("new")){ //NOI18N
            return ""; //NOI18N
        }else {
            return id; //NOI18N
        }
    }

    /** Inserts the given object reference before method invocations in the
     * expression, if any.
     * @param objref the object reference to be inserted.
     * @param expr the expression where to insert the object reference.
     * @return the expression given to the method including object references.
     */
    public String insertObjectReference(String objref, String expr) {
        StringBuffer sb = new StringBuffer(512);
        String token;
        StringTokenizer st = new StringTokenizer(expr);

        while(st.hasMoreTokens()) {
            token = st.nextToken();
            if(isMethodInvocation(token)) {
                if(token.startsWith("!")) { //NOI18N
                    sb.append(token.charAt(0) + objref + "." + token.substring(1));
                }else {
                    sb.append(objref + "." + token);
                }
            }else {
                sb.append(token);
            }
            if(st.countTokens() != 0) {
                sb.append(" "); //NOI18N
            }
        }
        return sb.toString();
    }

    /** Complements the raw precondition by inserting object references before
     * method invocations.
     * @param raw the raw precondition.
     * @param objref the object reference to be inserted.
     * @return the modified precondition containing object references before method
     * invocations.
     */
    public String complementPrecondition(ArrayList raw, String objref) {
        StringBuffer sb = new StringBuffer(512);

        sb.append(insertObjectReference(objref, (String)raw.get(0)));
        for(int i = 1; i < raw.size(); i++) {
            sb.append(" && " + insertObjectReference(objref, (String)raw.get(i))); //NOI18N
        }
        return sb.toString();
    }

    /** Checks whether the expression is a method invocation. The regular expression
     * representing a method invocation is as follows:
[!]?[\\p{Alpha}|_|$][\\w|$]*[\\(][\\p{ASCII}]*[\\)].
     * @param expr the expression to checked.
     * @return true if the expression is a method invocation; false otherwise.
     */
    public boolean isMethodInvocation(String expr) {
        return expr.matches("[!]?[\\p{Alpha}|_|$][\\w|$]*[\\(][\\p{ASCII}]*[\\)]"); //NOI18N
    }

    /** Inserts a test on the the precondition into the code. */
    public void insertPrecondition() {
        ArrayList al = super.getPrecondition();
         if(!al.isEmpty()) {
```

```
            try{
                int cpos = extEditorUI.getComponent().getCaret().getDot();
                int start = Utilities.getRowStart(extEditorUI.getDocument(), cpos);

                String objref = getObjectReference(cpos);
                String expr = complementPrecondition(al, objref);
                extEditorUI.getDocument().insertString(start, "if(" + expr + ") {\n", null);
//NOI18N

                int end = Utilities.getRowEnd(extEditorUI.getDocument(),
                                              extEditorUI.getComponent().getCaret().getDot());
                extEditorUI.getDocument().insertString(end, "\n}", null); //NOI18N
                Utilities.reformat(extEditorUI.getDocument(), start,
                                   Utilities.getRowStart(extEditorUI.getDocument(), end, 2));

            }catch(Exception e) {
                e.printStackTrace();
            }
         }
    }

    protected String findProperClass(String name, JCClass cls){
        if (cls == null || name==null) return null;
        String ret = null;

        if (cls.getPackageName()!=null && cls.getPackageName().length()>0){
            if (getClassElement(cls.getPackageName()+"."+name) != null){ //NOI18N
                return cls.getPackageName()+"."+name; //NOI18N
            }
        }else{
            if (getClassElement(name) != null){
                return name;
            }
        }

        List classes = JavaCompletion.getFinder().findClasses(null, name, true);
        if (classes.size()>0){
            ret = ((JCClass)classes.get(0)).getFullName();
            // found in Code Completion DB, but it is not mounted
            if (getClassElement(ret) == null) ret = null;
        }

        return ret;
    }

    protected boolean isNotFullyQualifiedInnerClass (String inner, JCClass cls){
        if (inner.indexOf(".") != inner.lastIndexOf(".")){ //NOI18N
            // fully qualified
            return false;
        }
        String pkgName = cls.getPackageName();
        return (getClassElement((pkgName.length()>0) ? (pkgName+"."+inner) : inner) != null);
//NOI18N
    }


    private List parseMethodTypes(String parameters){
        ArrayList ret = new ArrayList();
        if (parameters == null) return ret;
        StringTokenizer st = new StringTokenizer(parameters,","); //NOI18N
        while (st.hasMoreTokens()) {
            String param = st.nextToken();
            param.trim();
            if (param.startsWith(" ")) { //NOI18N
                param = param.substring(1);
            }
            if (param.endsWith(" ")){ //NOI18N
                param = param.substring(0,param.length()-1);
            }
            if (param.indexOf(".") < 0){
                //it could be i.e String without java.lang package
                String testParam = "java.lang."+param; //NOI18N
                if (testParam.indexOf("[")>0){ //NOI18N
                    // if param is an array
                    testParam = testParam.substring(0,testParam.indexOf("["));
                }
```

```
                ClassElement ce = getClassElement(testParam); //NOI18N
                if (ce!=null){
                    param = "java.lang."+param; //NOI18N
                }

            }
            ret.add(param);
        }

        return ret;
    }

    /** Returns true if javadoc is mounted in FS */
    public boolean  isExternalJavaDocMounted(){
        SyntaxSupport sup = Utilities.getSyntaxSupport(extEditorUI.getComponent());
        NbJavaSyntaxSupport nbJavaSup =
(NbJavaSyntaxSupport)sup.get(NbJavaSyntaxSupport.class);
        URL urls[] = nbJavaSup.getJavaDocURLs(currentContent);
        return (urls == null || urls.length < 1) ? false : true;
    }

    /** Parses given link such as <code>java.awt.Component#addHierarchyListener</code>
     *  and returns parsed Object
     *  @return Object of JCClass, JCMethod, JCConstructor or JCField
     */
    public Object parseLink(String link, JCClass cls){
        link = link.trim();
        ClassElement  linkClsElem;
        Object  linkMember;
        JCClass linkClass;
        JCFinder finder = JavaCompletion.getFinder();

        if (link.indexOf("#") > -1 ){ //NOI18N
            if (link.startsWith("#")){ //NOI18N
                /* Referencing a member of the current class i.e:
                 * @see  #field
                 * @see  #method(Type, Type,...)
                 * @see  #method(Type argname, Type argname,...)
                 */

                if (cls == null) return null;
                linkClsElem = getClassElement(cls.getFullName());
                if (linkClsElem == null) return null;
                linkClass = finder.getExactClass(cls.getFullName());
                if(linkClass == null){
                    linkClass = JCExtension.parseClassElement(linkClsElem, 0, 0, 0,
lookupCache, false);
                }
                if (linkClass == null) return null;

            }else{
                /* Referencing another class in the current or imported packages
                 * @see  Class#field
                 * @see  Class#method(Type, Type,...)
                 * @see  Class#method(Type argname, Type argname,...)
                 * @see  package.Class#field
                 * @see  package.Class#method(Type, Type,...)
                 * @see  package.Class#method(Type argname, Type argname,...)
                 */

                String refCls = link.substring(0, link.indexOf("#")); //NOI18N

                if (refCls.indexOf(".") < 0){
                    // it can be class in current package
                    if (cls != null){
                        String curPkgCls = (cls.getPackageName().length()>0) ?
cls.getPackageName()+"."+refCls : refCls; //NOI18N
                        if (getClassElement(curPkgCls) == null){
                            // try to find a class via finder
                            List outCls = finder.findClasses(null, refCls, true);
                            if (outCls.size() > 0){
                                refCls = ((JCClass) outCls.get(0)).getFullName();
                            }
                        }else{
                            refCls = curPkgCls;
                        }
                    }
```

```
                }
                linkClsElem = getClassElement(refCls);
                if (linkClsElem == null) return null;

                linkClass = finder.getExactClass(linkClsElem.getName().getFullName());
                if(linkClass == null){
                    linkClass = JCExtension.parseClassElement(linkClsElem, 0, 0, 0,
lookupCache, false);
                }
                if (linkClass == null) return null;
            }

            if (link.indexOf("(")>0){ //NOI18N
            //method or constructor
                String memberLink = link.substring(link.indexOf("#")+1) ;  //NOI18N
                String memberFullName =  (memberLink.indexOf(" ")>0) ?
memberLink.substring(0,memberLink.indexOf(" ")) : memberLink; //NOI18N
                String memberName = memberLink.substring(0,memberLink.indexOf("(")); //NOI18N
                String memberParams = null;
                if (link.indexOf(")")>0){ //NOI18N
                    memberParams = link.substring(link.indexOf("(")+1,link.indexOf(")"));
//NOI18N
                }
                List params = parseMethodTypes(memberParams);
                Type types[] = new Type[params.size()];
                try{
                    for (int i=0; i<types.length; i++){
                        types[i] = Type.parse((String)params.get(i));
                    }
                }catch(IllegalArgumentException iae){
                    types = new Type[] {};
                }

                MethodElement me = linkClsElem.getMethod(Identifier.create(memberName),
types);

                if (me!=null){
                    JCMethod mtds[] = linkClass.getMethods();
                    for (int i = 0; i<mtds.length; i++){
                        if (JCExtension.equals(mtds[i], me)){
                            return mtds[i];
                        }
                    }
                }
                return null;
            }else{
            //field or method or constructor
                String memberLink = link.substring(link.indexOf("#")+1) ; //NOI18N
                String memberName =  (memberLink.indexOf(" ")>0) ?
memberLink.substring(0,memberLink.indexOf(" ")) : memberLink; //NOI18N

                // look for fields first. If it will be the method with the same name it
should be named with ()
                FieldElement fieldElement =
linkClsElem.getField(Identifier.create(memberName));
                if (fieldElement !=null){
                    JCField flds[] = linkClass.getFields();
                    for (int i = 0; i<flds.length; i++){
                        if (JCExtension.equals(flds[i], fieldElement)) return flds[i];
                    }
                    return null;
                }else{
                    // Now try to find a method or constructor

                    //process all super classes
                    MethodElement me;
                    ClassElement processedCE = linkClsElem;
                    do {
                        me = processedCE.getMethod(Identifier.create(memberName), new Type[]
{});
                        Identifier superCls = processedCE.getSuperclass();
                        processedCE = (superCls == null) ? null :
ClassElement.forName(superCls.getFullName());
                    }while(processedCE != null && me == null);

                    // it could be method with some parameter
                    if (me == null){
```

```
                        // go through all super classes
                        processedCE = linkClsElem;
                        do{
                            MethodElement mes[] = processedCE.getMethods();
                            for (int i=0; i<mes.length; i++){

                                if (Identifier.create(memberName).equals(mes[i].getName())){
                                    // found, now just pick up the JCMethod
                                    JCClass linkClassLoop =
finder.getExactClass(processedCE.getName().getFullName());

                                    if (linkClassLoop != null){
                                        JCMethod mtds[] = linkClassLoop.getMethods();
                                        for (int j = 0; j<mtds.length; j++){
                                            if (JCExtension.equals(mtds[j], mes[i])) {
                                                return mtds[j];
                                            }
                                        }
                                    }
                                }

                            }

                            Identifier superCls = processedCE.getSuperclass();
                            processedCE = (superCls == null) ? null :
ClassElement.forName(superCls.getFullName());
                        }while (processedCE != null);

                    }else{
                        do {
                            JCMethod mtds[] = linkClass.getMethods();
                            for (int i = 0; i<mtds.length; i++){
                                if (JCExtension.equals(mtds[i], me)) return mtds[i];
                            }
                            linkClass = linkClass.getSuperclass();
                        }while(linkClass!=null);
                    }
                    return null;
                }
            }

        }else{
            /* no member available, it can be package or class i.e:
             * @see  Class
             * @see  package.Class
             * @see  package
             */
            String refCls = (link.indexOf(" ")>0) ? link.substring(0, link.indexOf(" ")) :
link; //NOI18N
            if (refCls.indexOf(".") > 0){ //NOI18N
                // fully qualified name or inner class
                linkClsElem = getClassElement(refCls);
                if (linkClsElem == null) {
                    if (cls != null){
                        refCls = findProperClass(refCls, cls);
                    }
                }
            }else{
                // class in current package
                if (cls != null){
                    refCls = findProperClass(refCls, cls);
                }
            }

            linkClsElem = getClassElement(refCls);
            if (linkClsElem == null) return null;
            linkClass = finder.getExactClass(linkClsElem.getName().getFullName());
            if(linkClass == null){
                linkClass = JCExtension.parseClassElement(linkClsElem, 0, 0, 0, lookupCache,
false);
            }
            return linkClass;
        }

    }
```

```
    private class ParsingThread implements Runnable{

        Object content;
        boolean running = true;

        public ParsingThread(Object content){
            this.content = content;
        }

        void stopTask(){
            running = false;
        }

        private String getBoldName(String description, String name){
            if (description.indexOf(name) > -1){
                StringBuffer sb = new StringBuffer();
                sb.append(description.substring(0, description.indexOf(name)));
                sb.append("<b>"+name+"</b>"); //NOI18N
                sb.append(description.substring(description.indexOf(name)+name.length()));
                return sb.toString().replace('$', '.'); //NOI18N
            }
            return description;
        }

        private void showJavaDoc(String preparedText){
            if (running){
                getJavaDocView().setContent(preparedText);
                if (running){
                    setJavaDocVisible(true);
                }
            }
        }

        private void setMethod(MethodElement me, JCMethod jcMethod){
            if (me == null || jcMethod == null){
                if (jcMethod == null) {
                    showJavaDoc(CONTENT_NOT_FOUND);
                }else{
                    String preparedText = prepareJavaDocContent(jcMethod.getClazz(),
                    getBoldName(jcMethod.toString(), jcMethod.getName()),
                    CONTENT_NOT_FOUND, null);
                    showJavaDoc(preparedText);
                }
                return;
            }
            JavaDoc.Method jdMethod = me.getJavaDoc();
            String preparedText = prepareJavaDocContent(jcMethod.getClazz(),
            getBoldName(jcMethod.toString(), jcMethod.getName()),
            (jdMethod == null || (jdMethod.getText().length() == 0 &&
jdMethod.getTags().length == 0)) ? CONTENT_NOT_FOUND : jdMethod.getText(),
            (jdMethod == null) ? null : getJavaDocTags(jdMethod.getTags()));
            showJavaDoc(preparedText);
        }

        private void setClass(ClassElement ce, JCClass cls){
            if (ce == null) {
                String preparedText;
                if (cls != null){
                    preparedText = prepareJavaDocContent(cls,"",  //NOI18N
                    CONTENT_NOT_FOUND,
                    null);
                }else{
                    preparedText = CONTENT_NOT_FOUND;
                }
                showJavaDoc(preparedText);
                return;
            }
            JavaDoc.Class jdClass = ce.getJavaDoc();

            String preparedText = prepareJavaDocContent(cls,"",  //NOI18N
            (jdClass == null || (jdClass.getText().length() == 0 && jdClass.getTags().length
== 0)) ? CONTENT_NOT_FOUND : jdClass.getText(),
            (jdClass == null) ? null : getJavaDocTags(jdClass.getTags()));
            showJavaDoc(preparedText);
        }
```

```
        private void setField(FieldElement fe, JCField jcField){
            if (fe == null || jcField == null) {
                if (jcField == null){
                    showJavaDoc(CONTENT_NOT_FOUND);
                }else{
                    String preparedText = prepareJavaDocContent(jcField.getClazz(),
                    getBoldName(jcField.toString(), jcField.getName()),
                    CONTENT_NOT_FOUND, null);
                    showJavaDoc(preparedText);
                }
                return;
            }
            JavaDoc.Field jdField = fe.getJavaDoc();
            String preparedText = prepareJavaDocContent(jcField.getClazz(),
            getBoldName(jcField.toString(), jcField.getName()),
            (jdField == null || (jdField.getText().length() == 0 && jdField.getTags().length
== 0) ) ? CONTENT_NOT_FOUND : jdField.getText(),
            (jdField == null) ? null : getJavaDocTags(jdField.getTags()));
            showJavaDoc(preparedText);
        }

        private void setConstructor(ConstructorElement conElem, JCConstructor jcConstuctor){
            if (conElem == null || jcConstuctor == null) {
                if (jcConstuctor == null){
                    showJavaDoc(CONTENT_NOT_FOUND);
                }else{
                    String preparedText = prepareJavaDocContent(jcConstuctor.getClazz(),
                    getBoldName(jcConstuctor.toString(), jcConstuctor.getClazz().getName()),
                    CONTENT_NOT_FOUND, null);
                    showJavaDoc(preparedText);
                }
                return;
            }
            JavaDoc.Method jdMethod = conElem.getJavaDoc();

            String preparedText = prepareJavaDocContent(jcConstuctor.getClazz(),
            getBoldName(jcConstuctor.toString(), jcConstuctor.getClazz().getName()),
            (jdMethod == null || ( jdMethod.getText().length() == 0 &&
jdMethod.getTags().length == 0 )) ? CONTENT_NOT_FOUND : jdMethod.getText(),
            (jdMethod == null) ? null : getJavaDocTags(jdMethod.getTags()));
            showJavaDoc(preparedText);
        }

        public void run(){
            if (!javaSourcesMounted) mountSources();

            if (content instanceof JCClass){
                setClass(getClassElement(((JCClass)content).getFullName()),
((JCClass)content));
            }else if(content instanceof JCField){
                JCField fld = (JCField)content;
                ClassElement ce = getClassElement(fld.getClazz().getFullName());

                if (ce == null){
                    showJavaDoc(CONTENT_NOT_FOUND);
                    return;
                }
                setField(ce.getField(Identifier.create(((JCField)content).getName())), fld);

            }else if(content instanceof JCMethod){
                JCMethod mtd = (JCMethod)content;
                ClassElement ce = getClassElement(mtd.getClazz().getFullName());

                if (ce == null) {
                    showJavaDoc(CONTENT_NOT_FOUND);
                    return;
                }
                JCParameter jcp [] = mtd.getParameters();
                try{
                    Type types[] = new Type[jcp.length];
                    for (int i=0; i<jcp.length; i++){
                        String array = ""; //NOI18N
                        for (int j = 0; j<jcp[i].getType().getArrayDepth(); j++){
                            array += "[]"; //NOI18N
                        }
                        types[i] =
Type.parse(jcp[i].getType().getClazz().getFullName()+array);
```

```
                    }
                    setMethod(ce.getMethod(Identifier.create(((JCMethod)content).getName()),
types), mtd);

                }catch(IllegalArgumentException iae){
                    iae.printStackTrace();
                }
            }else if (content instanceof JCConstructor){
                JCConstructor con = (JCConstructor)content;
                ClassElement ce = getClassElement(con.getClazz().getFullName());

                if (ce == null) {
                    showJavaDoc(CONTENT_NOT_FOUND);
                    return;
                }
                JCParameter jcp [] = con.getParameters();
                try {
                    Type types[] = new Type[jcp.length];
                    for (int i=0; i<jcp.length; i++){
                        String array = ""; //NOI18N
                        for (int j = 0; j<jcp[i].getType().getArrayDepth(); j++){
                            array += "[]"; //NOI18N
                        }
                        types[i] =
Type.parse(jcp[i].getType().getClazz().getFullName()+array);
                    }

                    setConstructor(ce.getConstructor(types), con);

                }catch (IllegalArgumentException iae){
                    iae.printStackTrace();
                }
            }else{
                showJavaDoc(CONTENT_NOT_FOUND);
            }
        }
    }

    class JavaDocTagItemImpl implements CompletionJavaDoc.JavaDocTagItem{
        String name;
        String text;

        public JavaDocTagItemImpl(String name, String text){
            this.name = name;
            this.text = text;
        }

        public String getName(){
            return name;
        }

        public String getText(){
            return text;
        }

        public int compareTo(Object o) {
            if (this == o) {
                return 0;
            }
            JavaDocTagItemImpl p = (JavaDocTagItemImpl)o;
            return name.compareTo(p.getName());
        }

        public int hashCode() {
            return name.hashCode();
        }

        public boolean equals(Object o) {
            if (this == o) {
                return true;
            }
            if (o instanceof JavaDocTagItemImpl) {
                return name.equals(((JavaDocTagItemImpl)o).getName());
            }
            if (o instanceof String) {
                return name.equals((String)o);
            }
```

```
            return false;
        }

        public String toString() {
            return name;
        }
    }

}
```

E.12 org.netbeans.modules.editor.options.Bundle.properties

```
#                 Sun Public License Notice
#
# The contents of this file are subject to the Sun Public License
# Version 1.0 (the "License"). You may not use this file except in
# compliance with the License. A copy of the License is available at
# http://www.sun.com/
#
# The Original Code is NetBeans. The Initial Developer of the Original
# Code is Sun Microsystems, Inc. Portions Copyright 1997-2000 Sun
# Microsystems, Inc. All Rights Reserved.
#
# Contributor(s): Håkan Bergmark, Tony Bergh

OPTIONS_all=Editor Settings
OPTIONS_plain=Plain Editor
OPTIONS_java=Java Editor
OPTIONS_print=Plain Editor
OPTIONS_html=HTML Editor

# Consider this garbage, as this property is hidden
PROP_base_editorState=Editor State
HINT_base_editorState=Editor state.

# Following localizations MUST be changed
# These are the labels and tooltips for all the editors properties
PROP_abbrevMap=Abbreviations
HINT_abbrevMap=Abbreviations for commonly typed key sequences.

PROP_caretBlinkRate=Insertion Point Blink Rate
HINT_caretBlinkRate=How fast the insertion point or overwrite caret blinks.

PROP_caretColorInsertMode=Insertion Point Color
HINT_caretColorInsertMode=Color of the insertion point.

PROP_caretColorOverwriteMode=Overwrite Caret Color
HINT_caretColorOverwriteMode=Color of the caret when in overwrite mode.

PROP_caretItalicInsertMode=Italic Insertion Point
HINT_caretItalicInsertMode=If True, the insertion point is italicized.

PROP_caretItalicOverwriteMode=Italic Overwrite Caret
HINT_caretItalicOverwriteMode=If True, the caret is italicized when in overwrite mode.

PROP_caretTypeInsertMode=Insertion Point
HINT_caretTypeInsertMode=Insertion point style.

PROP_caretTypeOverwriteMode=Overwrite Caret
HINT_caretTypeOverwriteMode=Style of caret when in overwrite mode.

PROP_coloringMap=Fonts and Colors
HINT_coloringMap=Fonts and colors for various elements displayed in the Source Editor.

PROP_completionAutoPopup=Auto Popup Completion Window
HINT_completionAutoPopup=If True, the completion window automatically appears when
appropriate.

PROP_completionCaseSensitive=Case Sensitive Code Completion
HINT_completionCaseSensitive=If True, the completion query search will be case sensitive

PROP_completionNaturalSort=Code Completion Natural Sort
HINT_completionNaturalSort=If True, the completion search results will be sorted naturally
```

```
PROP_fastImportPackage=Fast Import Package
HINT_fastImportPackage=If True, fast import dialog will offer packages

PROP_completionInstantSubstitution=Code Completion Instant Substitution
HINT_completionInstantSubstitution=If True, invocation of code completion perform instant
substitution, if the search result contains only one item

PROP_completionLowerCase=Code Completion Lowercase Substitution
HINT_completionLowerCase=If True, HTML Code Completion will be available in lower case.

PROP_completionAutoPopupDelay=Delay of Completion Window Auto Popup
HINT_completionAutoPopupDelay=How long to wait (in milliseconds) before automatically popping
up the completion window.

PROP_displayGoToClassInfo=Display Go To Class Info Window
HINT_displayGoToClassInfo=If true, Info Window will be displayed before Go To Class feature.

PROP_javaDocBGColor=Background Color of Javadoc Popup Window
HINT_javaDocBGColor=Background color of Javadoc popup window

PROP_javaDocAutoPopupDelay=Delay of Javadoc Window Auto Popup
HINT_javaDocAutoPopupDelay=How long to wait (in milliseconds) before automatically popping up
the Javadoc window.

PROP_javaDocPreferredSize=Javadoc Preferred Size
HINT_javaDocPreferredSize=Preferred size of Javadoc popup window.

PROP_javaDocAutoPopup=Auto Popup Javadoc Window
HINT_javaDocAutoPopup=If True, the Javadoc window automatically appears when appropriate.

PROP_updatePDAfterMounting=Update Code Completion Database After Mounting
HINT_updatePDAfterMounting=Mounted filesystem will be added into the Code Completion Database.

PROP_showContract=Show Only Contracts in Javadoc Window
HINT_showContract=If True, only contracts are shown in the Javadoc window.

PROP_showDeprecatedMembers=Show Deprecated Members in Code Completion
HINT_showDeprecatedMembers=If True, deprecated members are shown in code completion.

PROP_expandTabs=Expand Tabs to Spaces
HINT_expandTabs=If True, automatically converts tabs in the document to spaces padded to the
same column.

PROP_fontSize=Font Size
HINT_fontSize=Default font size for tokens not assigned value in Fonts and Colors property.

PROP_formatCompoundBracketAddNL=Curly Brace on Next Line
HINT_formatCompoundBracketAddNL=Curly brace on next line.

PROP_formatSpaceBeforeParenthesis=Add Space Before Parentheses
HINT_formatSpaceBeforeParenthesis=If True, add space before opening parenthesis.

PROP_highlightCaretRow=Highlight Caret Row
HINT_highlightCaretRow=If True, highlight row where the insertion point is.

PROP_highlightMatchingBracket=Highlight Matching Bracket
HINT_highlightMatchingBracket=Highlight bracket that matches the bracket before the insertion
point.

PROP_indentEngine=Indentation Engine
HINT_indentEngine=Indentation engine used for this type of editor.

PROP_keyBindingList=Key Bindings
HINT_keyBindingList=Actions bound to shortcut keys.

PROP_lineHeightCorrection=Line Height Correction
HINT_lineHeightCorrection=Multiplier by which to adjust line height.

PROP_lineNumberMargin=Line Number Margin
HINT_lineNumberMargin=Determines placement of line number margin.

PROP_lineNumberMargin2=Line Number Margin
HINT_lineNumberMargin2=Determines placement of line number margin.


PROP_lineNumberVisible=Line Numbers
```

```
HINT_lineNumberVisible=If True, line numbers in the document are displayed along the left side
of the window.

PROP_macroMap=Macros
HINT_macroMap=Click ellipsis button to display, edit, add, and remove macros.

PROP_margin=Margin
HINT_margin=Determines placement of margin.

PROP_printColoringMap=Print Fonts and Colors
HINT_printColoringMap=Fonts and colors to use when printing document.

PROP_printLineNumberVisible=Print Line Numbers
HINT_printLineNumberVisible=If True, print line numbers.

PROP_scrollFindInsets=Scroll Find Insets
HINT_scrollFindInsets=Specifies how much space to reserve on each side of text located with
the Find command.

PROP_scrollJumpInsets=Scroll Jump Insets
HINT_scrollJumpInsets=How much the view should jump in each direction when scrolling goes off
screen.

PROP_spacesPerTab=Number of Spaces per Tab
HINT_spacesPerTab=Number of spaces to be inserted when pressing tab.

PROP_statusBarCaretDelay=Status Bar Caret Delay
HINT_statusBarCaretDelay=Delay (in milliseconds) between time the caret stops moving and the
update of its position in the status bar.

PROP_statusBarVisible=Status Bar Visible
HINT_statusBarVisible=If True, the status bar (showing insert mode, line number, and so on) is
displayed at the bottom of the window.

PROP_tabSize=Tab Size
HINT_tabSize=Number of spaces per tab stop, for documents that contain real tabs.

PROP_textLimitLineColor=Text Limit Line Color
HINT_textLimitLineColor=Color of the line showing the text limit.

PROP_textLimitLineVisible=Display Text Limit Line
HINT_textLimitLineVisible=If True, display the text limit line.

PROP_textLimitWidth=Text Limit Character Count
HINT_textLimitWidth=Count of characters after which the text limit line is displayed.

PROP_base_keyBindingList=Global Key Bindings
HINT_base_keyBindingList=Actions bound to shortcut keys for all editor types.

PROP_base_toolbarVisible=Show Toolbar
HINT_base_toolbarVisible=If True, display the editor toolbar.

PROP_base_textAntialiasing=Text Antialiasing
HINT_base_textAntialiasing=If true, text antialiasing will be used for font rendering.


# Component colorings related stuff
NAME_coloring_base-default=Default
HINT_coloring_base-default=Regular text in document.
EXAMPLE_coloring_base-default=regular text

NAME_coloring_base-line-number=Glyph and Line Number Margin
HINT_coloring_base-line-number=Line numbers and glyph icon maring along left side of document.
EXAMPLE_coloring_base-line-number=123

NAME_coloring_base-selection=Selected Text
HINT_coloring_base-selection=Selected text in document.
EXAMPLE_coloring_base-selection=selected text

NAME_coloring_base-inc-search=Incremental Search
HINT_coloring_base-inc-search=Text currently found during an incremental search.
EXAMPLE_coloring_base-inc-search=incremental search

NAME_coloring_base-highlight-search=Highlighted by Search
HINT_coloring_base-highlight-search=Text highlighted after an incremental search.
EXAMPLE_coloring_base-highlight-search=highlight search
```

```
NAME_coloring_base-bookmark=Bookmark
HINT_coloring_base-bookmark=Bookmark in document.
EXAMPLE_coloring_base-bookmark=bookmark

NAME_coloring_base-guarded=Guarded Block
HINT_coloring_base-guarded=Guarded block (cannot be edited).
EXAMPLE_coloring_base-guarded=guarded block

NAME_coloring_base-status-bar=Status Bar
HINT_coloring_base-status-bar=Regular status bar text.
EXAMPLE_coloring_base-status-bar=30:54

NAME_coloring_base-status-bar-bold=Search Wrapped
HINT_coloring_base-status-bar-bold=Bold text in the status bar.
EXAMPLE_coloring_base-status-bar-bold=Search wrapped

NAME_coloring_base-highlight-caret-row=Highlight Insertion Point Row
HINT_coloring_base_highlight-caret-row=Highlight the insertion point row.
EXAMPLE_coloring_base_highlight-caret-row=row text

NAME_coloring_base-highlight-match-brace=Matching Bracket
HINT_coloring_base_highlight-match-brace=Highlight bracket that matches the bracket before the
insertion point.
EXAMPLE_coloring_base-highlight-match-brace=(


# the Base Colorings
NAME_coloring_base_text=Plain Text
HINT_coloring_base_text=Plain text.
EXAMPLE_coloring_base_text=text

NAME_coloring_base_error=Erroneous Text
HINT_coloring_base_error=Erroneous text.
EXAMPLE_coloring_base_error=error

# The Plain Colorings (those of PlainKit)
NAME_coloring_plain_text=Plain Text
HINT_coloring_plain_text=Plain text.
EXAMPLE_coloring_plain_text=text

NAME_coloring_plain_error=Erroneous Text
HINT_coloring_plain_error=Erroneous text.
EXAMPLE_coloring_plain_error=error



# The Java colorings. Some of them are generalized for a group
# of the same type, others are independent


# java-keywords
NAME_coloring_java-keywords=Java Keyword
HINT_coloring_java-keywords=Java keyword.
EXAMPLE_coloring_java-keywords=keyword
    # java-boolean
    NAME_coloring_java-boolean=boolean
    HINT_coloring_java-boolean=
    EXAMPLE_coloring_java-boolean=boolean
    # java-byte
    NAME_coloring_java-byte=byte
    HINT_coloring_java-byte=
    EXAMPLE_coloring_java-byte=byte
    # java-char
    NAME_coloring_java-char=char
    HINT_coloring_java-char=
    EXAMPLE_coloring_java-char=char
    # java-double
    NAME_coloring_java-double=double
    HINT_coloring_java-double=
    EXAMPLE_coloring_java-double=double
    # java-float
    NAME_coloring_java-float=float
    HINT_coloring_java-float=
    EXAMPLE_coloring_java-float=float
    # java-int
    NAME_coloring_java-int=int
    HINT_coloring_java-int=
```

```
    EXAMPLE_coloring_java-int=int
    # java-long
    NAME_coloring_java-long=long
    HINT_coloring_java-long=
    EXAMPLE_coloring_java-long=long
    # java-short
    NAME_coloring_java-short=short
    HINT_coloring_java-short=
    EXAMPLE_coloring_java-short=short
    # java-void
    NAME_coloring_java-void=void
    HINT_coloring_java-void=
    EXAMPLE_coloring_java-void=void
    # java-abstract
    NAME_coloring_java-abstract=abstract
    HINT_coloring_java-abstract=
    EXAMPLE_coloring_java-abstract=abstract
    # java-break
    NAME_coloring_java-break=break
    HINT_coloring_java-break=
    EXAMPLE_coloring_java-break=break
    # java-case
    NAME_coloring_java-case=case
HINT_coloring_java-case=
HINT_base_optionsVersion=Version of the options.
PROP_base_optionsVersion=Version of the Options
PROP_optionsVersion=Version of the Options
HINT_optionsVersion=Version of the options.
    EXAMPLE_coloring_java-case=case
    # java-catch
    NAME_coloring_java-catch=catch
    HINT_coloring_java-catch=
    EXAMPLE_coloring_java-catch=catch
    # java-class
    NAME_coloring_java-class=class
    HINT_coloring_java-class=
    EXAMPLE_coloring_java-class=class
    # java-const
    NAME_coloring_java-const=const
    HINT_coloring_java-const=
    EXAMPLE_coloring_java-const=const
    # java-continue
    NAME_coloring_java-continue=continue
    HINT_coloring_java-continue=
    EXAMPLE_coloring_java-continue=continue
    # java-default
    NAME_coloring_java-default=default
    HINT_coloring_java-default=
    EXAMPLE_coloring_java-default=default
    # java-do
    NAME_coloring_java-do=do
    HINT_coloring_java-do=
    EXAMPLE_coloring_java-do=do
    # java-else
    NAME_coloring_java-else=else
    HINT_coloring_java-else=
    EXAMPLE_coloring_java-else=else
    # java-extends
    NAME_coloring_java-extends=extends
    HINT_coloring_java-extends=
    EXAMPLE_coloring_java-extends=extends
    # java-false
    NAME_coloring_java-false=false
    HINT_coloring_java-false=
    EXAMPLE_coloring_java-false=false
    # java-final
    NAME_coloring_java-final=final
    HINT_coloring_java-final=
    EXAMPLE_coloring_java-final=final
    # java-finally
    NAME_coloring_java-finally=finally
    HINT_coloring_java-finally=
    EXAMPLE_coloring_java-finally=finally
    # java-for
    NAME_coloring_java-for=for
    HINT_coloring_java-for=
    EXAMPLE_coloring_java-for=for
```

```
# java-goto
NAME_coloring_java-goto=goto
HINT_coloring_java-goto=
EXAMPLE_coloring_java-goto=goto
# java-if
NAME_coloring_java-if=if
HINT_coloring_java-if=
EXAMPLE_coloring_java-if=if
# java-implements
NAME_coloring_java-implements=implements
HINT_coloring_java-implements=
EXAMPLE_coloring_java-implements=implements
# java-import
NAME_coloring_java-import=import
HINT_coloring_java-import=
EXAMPLE_coloring_java-import=import
# java-instanceof
NAME_coloring_java-instanceof=instanceof
HINT_coloring_java-instanceof=
EXAMPLE_coloring_java-instanceof=instanceof
# java-interface
NAME_coloring_java-interface=interface
HINT_coloring_java-interface=
EXAMPLE_coloring_java-interface=interface
# java-native
NAME_coloring_java-native=native
HINT_coloring_java-native=
EXAMPLE_coloring_java-native=native
# java-new
NAME_coloring_java-new=new
HINT_coloring_java-new=
EXAMPLE_coloring_java-new=new
# java-null
NAME_coloring_java-null=null
HINT_coloring_java-null=
EXAMPLE_coloring_java-null=null
# java-package
NAME_coloring_java-package=package
HINT_coloring_java-package=
EXAMPLE_coloring_java-package=package
# java-private
NAME_coloring_java-private=private
HINT_coloring_java-private=
EXAMPLE_coloring_java-private=private
# java-protected
NAME_coloring_java-protected=protected
HINT_coloring_java-protected=
EXAMPLE_coloring_java-protected=protected
# java-public
NAME_coloring_java-public=public
HINT_coloring_java-public=
EXAMPLE_coloring_java-public=public
# java-return
NAME_coloring_java-return=return
HINT_coloring_java-return=
EXAMPLE_coloring_java-return=return
# java-static
NAME_coloring_java-static=static
HINT_coloring_java-static=
EXAMPLE_coloring_java-static=static
# java-super
NAME_coloring_java-super=super
HINT_coloring_java-super=
EXAMPLE_coloring_java-super=super
# java-switch
NAME_coloring_java-switch=switch
HINT_coloring_java-switch=
EXAMPLE_coloring_java-switch=switch
# java-synchronized
NAME_coloring_java-synchronized=synchronized
HINT_coloring_java-synchronized=
EXAMPLE_coloring_java-synchronized=synchronized
# java-this
NAME_coloring_java-this=this
HINT_coloring_java-this=
EXAMPLE_coloring_java-this=this
# java-throw
```

```
    NAME_coloring_java-throw=throw
    HINT_coloring_java-throw=
    EXAMPLE_coloring_java-throw=throw
    # java-throws
    NAME_coloring_java-throws=throws
    HINT_coloring_java-throws=
    EXAMPLE_coloring_java-throws=throws
    # java-transient
    NAME_coloring_java-transient=transient
    HINT_coloring_java-transient=
    EXAMPLE_coloring_java-transient=transient
    # java-true
    NAME_coloring_java-true=true
    HINT_coloring_java-true=
    EXAMPLE_coloring_java-true=true
    # java-try
    NAME_coloring_java-try=try
    HINT_coloring_java-try=
    EXAMPLE_coloring_java-try=try
    # java-volatile
    NAME_coloring_java-volatile=volatile
    HINT_coloring_java-volatile=
    EXAMPLE_coloring_java-volatile=volatile
    # java-while
    NAME_coloring_java-while=while
    HINT_coloring_java-while=
    EXAMPLE_coloring_java-while=while



# java-operators
NAME_coloring_java-operators=Java Operator
HINT_coloring_java-operators=Java operator.
EXAMPLE_coloring_java-operators=+-*/&|
    # java-eq
    NAME_coloring_java-eq=Assignment
    HINT_coloring_java-eq=Assignment operator.
    EXAMPLE_coloring_java-eq==
    # java-lt
    NAME_coloring_java-lt=Less Than
    HINT_coloring_java-lt="Less than" operator.
    EXAMPLE_coloring_java-lt=<
    # java-gt
    NAME_coloring_java-gt=Greater Than
    HINT_coloring_java-gt="Greater than" operator.
    EXAMPLE_coloring_java-gt=>
    # java-lshift
    NAME_coloring_java-lshift=Left Shift
    HINT_coloring_java-lshift=Left shift operator.
    EXAMPLE_coloring_java-lshift=<<
    # java-rsshift
    NAME_coloring_java-rsshift=Right shift
    HINT_coloring_java-rsshift=Right Shift operator.
    EXAMPLE_coloring_java-rsshift=>>
    # java-rushift
    NAME_coloring_java-rushift=Right Unsigned Shift
    HINT_coloring_java-rushift=Right unsigned shift operator.
    EXAMPLE_coloring_java-rushift=>>>
    # java-plus
    NAME_coloring_java-plus=Plus
    HINT_coloring_java-plus=Plus operator.
    EXAMPLE_coloring_java-plus=+
    # java-minus
    NAME_coloring_java-minus=Minus
    HINT_coloring_java-minus=Minus operator.
    EXAMPLE_coloring_java-minus=-
    # java-mul
    NAME_coloring_java-mul=Multiply
    HINT_coloring_java-mul=Multiply operator.
    EXAMPLE_coloring_java-mul=*
    # java-div
    NAME_coloring_java-div=Division
    HINT_coloring_java-div=Division operator.
    EXAMPLE_coloring_java-div=/
    # java-and
    NAME_coloring_java-and=And
```

```
HINT_coloring_java-and=And operator.
EXAMPLE_coloring_java-and=&
# java-or
NAME_coloring_java-or=Or
HINT_coloring_java-or=Or operator.
EXAMPLE_coloring_java-or=|
# java-xor
NAME_coloring_java-xor=Xor
HINT_coloring_java-xor=Xor operator.
EXAMPLE_coloring_java-xor=^
# java-mod
NAME_coloring_java-mod=Modulo
HINT_coloring_java-mod=Modulo (reminder) operator.
EXAMPLE_coloring_java-mod=%
# java-not
NAME_coloring_java-not=Not
HINT_coloring_java-not=Logical not operator.
EXAMPLE_coloring_java-not=!
# java-neg
NAME_coloring_java-neg=Negation
HINT_coloring_java-neg=Negation operator.
EXAMPLE_coloring_java-neg=~
# java-eq-eq
NAME_coloring_java-eq-eq=Equivalence
HINT_coloring_java-eq-eq=Equivalence operator.
EXAMPLE_coloring_java-eq-eq===
# java-le
NAME_coloring_java-le=Less Or Equal
HINT_coloring_java-le="Less or equal" operator.
EXAMPLE_coloring_java-le=<=
# java-ge
NAME_coloring_java-ge=Greater Or Equal
HINT_coloring_java-ge="Greater or equal" operator.
EXAMPLE_coloring_java-ge=>=
# java-lshift-eq
NAME_coloring_java-lshift-eq=
HINT_coloring_java-lshift-eq=
EXAMPLE_coloring_java-lshift-eq=<<=
# java-rsshift-eq
NAME_coloring_java-rsshift-eq=
HINT_coloring_java-rsshift-eq=
EXAMPLE_coloring_java-rsshift-eq=>>=
# java-rushift-eq
NAME_coloring_java-rushift-eq=
HINT_coloring_java-rushift-eq=
EXAMPLE_coloring_java-rushift-eq=>>>=
# java-plus-eq
NAME_coloring_java-plus-eq=
HINT_coloring_java-plus-eq=
EXAMPLE_coloring_java-plus-eq=+=
# java-minus-eq
NAME_coloring_java-minus-eq=
HINT_coloring_java-minus-eq=
EXAMPLE_coloring_java-minus-eq=-=
# java-mul-eq
NAME_coloring_java-mul-eq=
HINT_coloring_java-mul-eq=
EXAMPLE_coloring_java-mul-eq=*=
# java-div-eq
NAME_coloring_java-div-eq=
HINT_coloring_java-div-eq=
EXAMPLE_coloring_java-div-eq=/=
# java-and-eq
NAME_coloring_java-and-eq=
HINT_coloring_java-and-eq=
EXAMPLE_coloring_java-and-eq=&=
# java-or-eq
NAME_coloring_java-or-eq=
HINT_coloring_java-or-eq=
EXAMPLE_coloring_java-or-eq=!=
# java-xor-eq
NAME_coloring_java-xor-eq=
HINT_coloring_java-xor-eq=
EXAMPLE_coloring_java-xor-eq=^=
# java-mod-eq
NAME_coloring_java-mod-eq=
HINT_coloring_java-mod-eq=
```

```
    EXAMPLE_coloring_java-mod-eq=%=
    # java-not-eq
    NAME_coloring_java-not-eq=
    HINT_coloring_java-not-eq=
    EXAMPLE_coloring_java-not-eq=!=
    # java-dot
    NAME_coloring_java-dot=Dot
    HINT_coloring_java-dot=Dot operator.
    EXAMPLE_coloring_java-dot=.
    # java-comma
    NAME_coloring_java-comma=Comma
    HINT_coloring_java-comma=Comma operator.
    EXAMPLE_coloring_java-comma=,
    # java-colon
    NAME_coloring_java-colon=
    HINT_coloring_java-colon=
    EXAMPLE_coloring_java-colon=:
    # java-semicolon [PENDING: Not an operator]
    # NAME_coloring_java-semicolon=
    # HINT_coloring_java-semicolon=
    # EXAMPLE_coloring_java-semicolon=
    # java-question
    NAME_coloring_java-question=
    HINT_coloring_java-question=
    EXAMPLE_coloring_java-question=?
    # java-lparen
    NAME_coloring_java-lparen=
    HINT_coloring_java-lparen=
    EXAMPLE_coloring_java-lparen=(
    # java-rparen
    NAME_coloring_java-rparen=
    HINT_coloring_java-rparen=
    EXAMPLE_coloring_java-rparen=)
    # java-lbracket
    NAME_coloring_java-lbracket=
    HINT_coloring_java-lbracket=
    EXAMPLE_coloring_java-lbracket=[
    # java-rbracket
    NAME_coloring_java-rbracket=
    HINT_coloring_java-rbracket=
    EXAMPLE_coloring_java-rbracket=]
    # java-lbrace [PENDING: Not an operator]
    # NAME_coloring_java-lbrace=
    # HINT_coloring_java-lbrace=
    # EXAMPLE_coloring_java-lbrace={
    # java-rbrace [PENDING: Not an operator]
    # NAME_coloring_java-rbrace=
    # HINT_coloring_java-rbrace=
    # EXAMPLE_coloring_java-rbrace=}
    # java-plus-plus
    NAME_coloring_java-plus-plus=Increment
    HINT_coloring_java-plus-plus=Increment operator.
    EXAMPLE_coloring_java-plus-plus=++
    # java-minus-minus
    NAME_coloring_java-minus-minus=Decrement
    HINT_coloring_java-minus-minus=Decrement operator.
    EXAMPLE_coloring_java-minus-minus=--
    # java-and-and
    NAME_coloring_java-and-and=
    HINT_coloring_java-and-and=
    EXAMPLE_coloring_java-and-and=&&
    # java-or-or
    NAME_coloring_java-or-or=
    HINT_coloring_java-or-or=
    EXAMPLE_coloring_java-or-or=||


# java-numeric-literals, the grouping and the elements
NAME_coloring_java-numeric-literals=Numeric Literal
HINT_coloring_java-numeric-literals=Numeric constant.
EXAMPLE_coloring_java-numeric-literals=5
    # java-int-literal
    NAME_coloring_java-int-literal=Integer Literal
    HINT_coloring_java-int-literal=Integer constant.
    EXAMPLE_coloring_java-int-literal=1234567
    # java-long-literal
```

```
    NAME_coloring_java-long-literal=Long Integer Literal
    HINT_coloring_java-long-literal=Long integer constant.
    EXAMPLE_coloring_java-long-literal=12345L
    # java-hex-literal
    NAME_coloring_java-hex-literal=Hexadecimal Literal
    HINT_coloring_java-hex-literal=Hexadecimal constant.
    EXAMPLE_coloring_java-hex-literal=0x9A
    # java-octal-literal
    NAME_coloring_java-octal-literal=Octal Literal
    HINT_coloring_java-octal-literal=Octal constant.
    EXAMPLE_coloring_java-octal-literal=057
    # java-float-literal
    NAME_coloring_java-float-literal=Float Literal
    HINT_coloring_java-float-literal=Floating-point constant.
    EXAMPLE_coloring_java-float-literal=1.02f
    # java-double-literal
    NAME_coloring_java-double-literal=Double Literal
    HINT_coloring_java-double-literal=Double floating-point constant.
    EXAMPLE_coloring_java-double-literal=3.55d



# java-errors
NAME_coloring_java-errors=Error In Sources
HINT_coloring_java-errors=Erroneous text (marked by the compiler).
EXAMPLE_coloring_java-errors=error
    # java-incomplete-string-literal
    NAME_coloring_java-incomplete-string-literal=
    HINT_coloring_java-incomplete-string-literal=
    EXAMPLE_coloring_java-incomplete-string-literal="tex
    # java-incomplete-char-literal
    NAME_coloring_java-incomplete-char-literal=
    HINT_coloring_java-incomplete-char-literal=
    EXAMPLE_coloring_java-incomplete-char-literal='a
    # java-incomplete-hex-literal
    NAME_coloring_java-incomplete-hex-literal=
    HINT_coloring_java-incomplete-hex-literal=
    EXAMPLE_coloring_java-incomplete-hex-literal=0x
    # java-invalid-char
    NAME_coloring_java-invalid-char=
    HINT_coloring_java-invalid-char=
    EXAMPLE_coloring_java-invalid-char='aa'
    # java-invalid-operator
    NAME_coloring_java-invalid-operator=
    HINT_coloring_java-invalid-operator=
    EXAMPLE_coloring_java-invalid-operator=
    # java-invalid-octal-literal
    NAME_coloring_java-invalid-octal-literal=
    HINT_coloring_java-invalid-octal-literal=
    EXAMPLE_coloring_java-invalid-octal-literal=
    # java-invalid-comment-end
    NAME_coloring_java-invalid-comment-end=
    HINT_coloring_java-invalid-comment-end=
    EXAMPLE_coloring_java-invalid-comment-end=


# java-whitespace
NAME_coloring_java-whitespace=Whitespace
HINT_coloring_java-whitespace=Whitespace.
EXAMPLE_coloring_java-whitespace=

# java-identifier
NAME_coloring_java-identifier=Java Identifier
HINT_coloring_java-identifier=Java identifier.
EXAMPLE_coloring_java-identifier=id

# java-line-comment
NAME_coloring_java-line-comment=Single-line Comment
HINT_coloring_java-line-comment=Single-line comment.
EXAMPLE_coloring_java-line-comment=// line-comment

# java-block-comment
NAME_coloring_java-block-comment=Block Comment
HINT_coloring_java-block-comment=Block (multi-line) comment.
EXAMPLE_coloring_java-block-comment=/* block comment */
```

```
# java-char-literal
NAME_coloring_java-char-literal=Character Literal
HINT_coloring_java-char-literal=Character constant.
EXAMPLE_coloring_java-char-literal='\\n'

# java-string-literal
NAME_coloring_java-string-literal=String Literal
HINT_coloring_java-string-literal=String constant.
EXAMPLE_coloring_java-string-literal="string"

#-----------------------------------------------------------------------------
NAME_coloring_java-layer-method=Java Method Call
HINT_coloring_java-layer-method=Java method (or constructor) call.
EXAMPLE_coloring_java-layer-method=method()
#-----------------------------------------------------------------------------
NAME_coloring_html-text=Plain Text
HINT_coloring_html-text=Plain text.
EXAMPLE_coloring_html-text=text

NAME_coloring_html-ws=White Space
HINT_coloring_html-ws=White space.
EXAMPLE_coloring_html-ws=

NAME_coloring_html-error=Erroneous Text
HINT_coloring_html-error=Erroneous text.
EXAMPLE_coloring_html-error=<A\

NAME_coloring_html-tag=HTML Tag
HINT_coloring_html-tag=HTML tag.
EXAMPLE_coloring_html-tag=<TABLE>

NAME_coloring_html-argument=Attribute of a Tag
HINT_coloring_html-argument=Attribute of a tag.
EXAMPLE_coloring_html-argument=href

NAME_coloring_html-operator=HTML Special Syntax
HINT_coloring_html-operator=HTML special syntax.
EXAMPLE_coloring_html-operator==

NAME_coloring_html-value=Attribute Value
HINT_coloring_html-value=Value of an attribute.
EXAMPLE_coloring_html-value="value"

NAME_coloring_html-block-comment=HTML Comment
HINT_coloring_html-block-comment=HTML comment.
EXAMPLE_coloring_html-block-comment=<!-- comment -->

NAME_coloring_html-sgml-comment=SGML-type Comment
HINT_coloring_html-sgml-comment=Comment in SGML declaration.
EXAMPLE_coloring_html-sgml-comment=-- comment --

NAME_coloring_html-sgml-declaration=SGML Declaration
HINT_coloring_html-sgml-declaration=SGML declaration.
EXAMPLE_coloring_html-sgml-declaration=<!DOCTYPE ... >


NAME_coloring_html-character=Character Reference
HINT_coloring_html-character=Character reference.
EXAMPLE_coloring_html-character=&amp;

#-----------------------------------------------------------------------------

# Indentation Engine customizer
PROP_indentEngine_expandTabs=Expand Tabs to Spaces
HINT_indentEngine_expandTabs=If True, automatically converts tabs in the document to spaces
padded to the same column.
PROP_indentEngine_spacesPerTab=Number of Spaces per Tab
HINT_indentEngine_spacesPerTab=Number of spaces to be inserted when pressing tab.
LAB_SimpleIndentEngine=Simple Indentation Engine
HINT_SimpleIndentEngine=Simple indentation engine.

# Update PD after FS mounting or delete PD after unmounting
ALWAYS=Always
NEVER=Never
ASK=Ask
```

```
# CaretTypeEditor
LINE_CARET=Vertical Bar
THIN_LINE_CARET=Thin Vertical Bar
BLOCK_CARET=Solid Block

#Settings for font and color property editor
#ColoringEditorPanel
CEP_FontTitle=\ Font
ACSD_CEP_Font=
CEP_FontTrans=\ Inherit
CEP_FontTrans_Mnemonic=I
ACSD_CEP_FontTrans=
CEP_FgTitle=\ Foreground Color
ACSD_CEP_Fg=
CEP_FgTrans=\ Inherit
CEP_FgTrans_Mnemonic=N
ACSD_CEP_FgTrans=
CEP_BgTitle=\ Background Color
ACSD_CEP_Bg=
CEP_BgTrans=\ Inherit
CEP_BgTrans_Mnemonic=E
ACSD_CEP_BgTrans=
CEP_PreviewTitle=\ Preview
ACSN_CEP_Preview=Color Preview
ACSD_CEP_Preview=

#ColoringArrayEditorPanel
CAEP_SyntaxLabel=Syntax:
CAEP_SyntaxLabel_Mnemonic=S
ACSD_CAEP_Syntax=
CAEP_ColoringLabel=Coloring
CAEP_DefaultColoringName=DEFAULT
ACSD_CAEP_Panel=

#KeyBindingsEditorPanel
KBEP_Actions=Actions:
ACSN_KBEP_Actions=Actions
ACSD_KBEP_Actions=
KBEP_Sequences=Keybindings:
KBEP_Sequences_Mnemonic=K
ACSD_KBEP_Sequences=
KBEP_Add=Add...
KBEP_Add_Mnemonic=A
ACSD_KBEP_Add=
KBEP_Remove=Remove
KBEP_Remove_Mnemonic=R
ACSD_KBEP_Remove=
KBEP_action_sort_button=\ Sort by Action
KBEP_action_sort_button_mnemonic=S
ACSD_KBEP_action_sort_button=
KBEP_name_sort_button=\ Sort by Name
KBEP_name_sort_button_mnemonic=N
ACSD_KBEP_name_sort_button=
ACSD_KBEP_Panel=

#KeyBindingsEditorPanel.KeySequenceRequester
KBEP_AddSequence=Add Keybinding
KBEP_OK_LABEL=OK
KBEP_CLEAR_LABEL=Clear
KBEP_CLEAR_Mnemonic=C
ACSD_KBEP_OK=
ACSD_KBEP_CLEAR=
KBEP_FMT_Collision=This shortcut sequence conflicts with sequence <{0}> bound to "{1}".
LBL_KSIP_Sequence=Shortcut Sequence:

PROP_KeyBindings=Key Bindings
PROP_Abbreviations=Abbreviations
PROP_Macros=Macros

#AbbrevsEditorPanel
AEP_EnterAbbrev=Enter Abbreviation
AEP_AbbrevTitle=Abbreviation
AEP_ExpandTitle=Expanded String
AEP_Add=Add...
AEP_Add_Mnemonic=A
ACSD_AEP_Add=
AEP_Edit=Edit...
```

```
AEP_Edit_Mnemonic=E
ACSD_AEP_Edit=
AEP_Remove=Remove
AEP_Remove_Mnemonic=R
ACSD_AEP_Remove=
AIP_Abbrev=Abbreviation:
AIP_Abbrev_Mnemonic=A
ACSD_AIP_Abbrev=
AIP_Expand=Expansion:
AIP_Expand_Mnemonic=E
ACSD_AIP_Expand
ACSN_AEP_Table=Abbreviations
ACSD_AEP_Table=
ACSD_AEP=
ACSD_AIP=
AEP_Overwrite=Abbreviation <{0}> already exists for "{1}". \nReplace with "{2}"?

#MacrosEditorPanel
MEP_EnterMacro=Enter Macro
MEP_MacroTitle=Macro
MEP_ExpandTitle=Expanded String
MEP_Add=Add...
MEP_Add_Mnemonic=A
ACSD_MEP_Add=
MEP_Edit=Edit...
MEP_Edit_Mnemonic=E
ACSD_MEP_Edit=
MEP_Remove=Remove
MEP_Remove_Mnemonic=R
ACSD_MEP_Remove=
ACSN_MEP_Table=Macros
ACSD_MEP_Table=
ACSD_MEP=

MIP_Macro=Macro:
MIP_Macro_Mnemonic=M
ACSD_MIP_Macro=
MIP_Expand=Expansion:
MIP_Expand_Mnemonic=E
ACSD_MIP_Expand=
ACSD_MIP=

# ScrollInsetsCustomEditor
SIE_EXC_BadFormatValue=Value should be in format [top[%],left[%],bottom[%],right[%]]
SIC_InvalidValue=Invalid value entered
SICE_Insets=\ Insets
SICE_Top=Top:
SICE_Top_Mnemonic=T
ACSD_SICE_Top=
SICE_Bottom=Bottom:
SICE_Bottom_Mnemonic=B
ACSD_SICE_Bottom=
SICE_Left=Left:
SICE_Left_Mnemonic=L
ACSD_SICE_Left=
SICE_Right=Right:
SICE_Right_Mnemonic=R
ACSD_SICE_Right=
ACSD_SICE_Top=
ACSD_SICE_Bottom=
ACSD_SICE_Left=
ACSD_SICE_Right=
ACSD_SICE=

# Coloring desc
PROP_Coloring=Fonts & Colors
HINT_Coloring=Font, background color, foreground color.

# Tool tip for "green" fields
HINT_HitEnter=Hit the action key to make the change.

# Add/Rem buttons
Add=Add
Remove=Remove

# title
TITLE_EditPanel=Edit
```

```
HINT_BOOKMARK=Bookmark

ATN_AnnotationTypesNode_Name=Annotation Types
ATN_AnnotationTypesNode_Description=All registered annotation types

PROP_AT_HIGHLIGHT=Highlight Color
HINT_AT_HIGHLIGHT=Color of the background highlight.
PROP_AT_USE_HIGHLIGHT=Use Highlight Color
HINT_AT_USE_HIGHLIGHT=If True, the highlight color is used.
PROP_AT_FOREGROUND=Foreground Color
HINT_AT_FOREGROUND=Color of the text.
PROP_AT_WAVEUNDERLINE=Wave Underline Color
HINT_AT_WAVEUNDERLINE=Color of the wave underline.
PROP_AT_USE_WAVEUNDERLINE=Use Wave Underline
HINT_AT_USE_WAVEUNDERLINE=If True, the wave underline is used.
PROP_AT_INHERIT_FOREGROUND=Inherit Foreground Colors
HINT_AT_INHERIT_FOREGROUND=If True, the foreground color is used.
PROP_AT_WHOLELINE=Whole Line
HINT_AT_WHOLELINE=If True, this annotation type is whole line annotation type.
ERR_NoContentTypeDefined =KitClass {0} has not defined method getContentType(). It should not
be displayed as node in Editor Settings.

PROP_backgroundDrawing=Draw Glyphs on Background
HINT_backgroundDrawing=If True, annotation glyphs that are covered by another annotation glyph
are drawn on the background under the Source Editor text.
PROP_backgroundGlyphAlpha=Glyph Icon Alpha
HINT_backgroundGlyphAlpha=Percentage of alpha value for the glyph icons that are drawn on the
background.
PROP_combineGlyphs=Combine Annotations
HINT_combineGlyphs=If True, annotation glyphs can be combined.
PROP_glyphsOverLineNumbers=Draw Glyph Over Line Numbers
HINT_glyphsOverLineNumbers=If True, glyph icons can be drawn over the line numbers.
PROP_showGlyphGutter=Always Show Glyph Margin
HINT_showGlyphGutter=Show glyph margin even when there are no annotations.
```

E.13 org.netbeans.modules.editor.options.JavaOptions.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.modules.editor.options;

import java.util.Enumeration;
import java.util.List;
import org.netbeans.editor.ext.ExtSettingsNames;
import org.netbeans.editor.ext.java.JavaSettingsNames;
import org.netbeans.modules.editor.java.JavaKit;
import org.netbeans.modules.editor.java.JavaIndentEngine;
import org.openide.ServiceType;
import org.openide.util.HelpCtx;
import org.netbeans.modules.editor.NbEditorUtilities;
import java.awt.Color;
import java.awt.Dimension;
import org.netbeans.editor.ext.java.JavaFastImport;

/**
* Options for the java editor kit
*
* @author Miloslav Metelka
* @version 1.00
*/
```

```
public class JavaOptions extends BaseOptions {

    public static final String JAVA = "java"; // NOI18N

    public static final String COMPLETION_AUTO_POPUP_PROP = "completionAutoPopup"; // NOI18N

    public static final String COMPLETION_CASE_SENSITIVE_PROP = "completionCaseSensitive"; //
NOI18N

    public static final String COMPLETION_NATURAL_SORT_PROP = "completionNaturalSort"; //
NOI18N

    public static final String COMPLETION_AUTO_POPUP_DELAY_PROP = "completionAutoPopupDelay";
// NOI18N

    public static final String FORMAT_SPACE_BEFORE_PARENTHESIS_PROP =
"formatSpaceBeforeParenthesis"; // NOI18N

    public static final String FORMAT_COMPOUND_BRACKET_ADD_NL_PROP =
"formatCompoundBracketAddNL"; // NOI18N

    public static final String FAST_IMPORT_SELECTION_PROP = "fastImportSelection"; //NOI18N

    public static final String DISPLAY_GO_TO_CLASS_INFO_PROP = "displayGoToClassInfo"; //
NOI18N

    private static final String HELP_ID = "editing.editor.java"; // !!! NOI18N

    public static final String JAVADOC_BGCOLOR = "javaDocBGColor"; // NOI18N

    public static final String JAVADOC_AUTO_POPUP_DELAY_PROP = "javaDocAutoPopupDelay";
//NOI18N

    public static final String JAVADOC_PREFERRED_SIZE_PROP = "javaDocPreferredSize"; //NOI18N

    public static final String JAVADOC_AUTO_POPUP_PROP = "javaDocAutoPopup"; //NOI18N

    public static final String UPDATE_PD_AFTER_MOUNTING_PROP = "updatePDAfterMounting";
//NOI18N

    public static final String SHOW_DEPRECATED_MEMBERS_PROP = "showDeprecatedMembers";
//NOI18N

    public static final String COMPLETION_INSTANT_SUBSTITUTION_PROP =
"completionInstantSubstitution"; // NOI18N

    public static final String FAST_IMPORT_PACKAGE_PROP = "fastImportPackage"; // NOI18N

    public static final String SHOW_CONTRACT_PROP = "showContract"; //NOI18N

    static final String[] JAVA_PROP_NAMES = new String[] {
                                                COMPLETION_AUTO_POPUP_PROP,
                                                COMPLETION_CASE_SENSITIVE_PROP,
                                                COMPLETION_AUTO_POPUP_DELAY_PROP,
                                                FORMAT_SPACE_BEFORE_PARENTHESIS_PROP,
                                                FORMAT_COMPOUND_BRACKET_ADD_NL_PROP,
                                                DISPLAY_GO_TO_CLASS_INFO_PROP,
                                                JAVADOC_BGCOLOR,
                                                JAVADOC_AUTO_POPUP_DELAY_PROP,
                                                JAVADOC_PREFERRED_SIZE_PROP,
                                                JAVADOC_AUTO_POPUP_PROP,
                                                UPDATE_PD_AFTER_MOUNTING_PROP,
                                                SHOW_DEPRECATED_MEMBERS_PROP,
                                                COMPLETION_INSTANT_SUBSTITUTION_PROP,
                                                COMPLETION_NATURAL_SORT_PROP,
                                                FAST_IMPORT_PACKAGE_PROP,
                                                SHOW_CONTRACT_PROP
                                            };

    static final long serialVersionUID =-7951549840240159575L;

    public JavaOptions() {
        this(JavaKit.class, JAVA);
    }

    public JavaOptions(Class kitClass, String typeName) {
        super(kitClass, typeName);
```

```
    }

/*    public boolean getFormatSpaceBeforeParenthesis() {
        return
((Boolean)getSettingValue(JavaSettingsNames.JAVA_FORMAT_SPACE_BEFORE_PARENTHESIS)).booleanValu
e();
    }
    [Mila] Removed because it was moved to JavaIndentEngine; setter must stay here
*/
    public void setFormatSpaceBeforeParenthesis(boolean v) {
        setSettingBoolean(JavaSettingsNames.JAVA_FORMAT_SPACE_BEFORE_PARENTHESIS, v,
            FORMAT_SPACE_BEFORE_PARENTHESIS_PROP);

        // Need to check whether the service exists and if not, add it
//        checkJavaIndentEngineRegistered();
    }

/*    public boolean getFormatCompoundBracketAddNL() {
        return
((Boolean)getSettingValue(JavaSettingsNames.JAVA_FORMAT_NEWLINE_BEFORE_BRACE)).booleanValue();
    }
    [Mila] Removed because it was moved to JavaIndentEngine; setter must stay here
*/
    public void setFormatCompoundBracketAddNL(boolean v) {
        setSettingBoolean(JavaSettingsNames.JAVA_FORMAT_NEWLINE_BEFORE_BRACE, v,
            FORMAT_COMPOUND_BRACKET_ADD_NL_PROP);

        // Need to check whether the service exists and if not, add it
//        checkJavaIndentEngineRegistered();
    }

/*    private void checkJavaIndentEngineRegistered() {
        ServiceType.Registry sr = TopManager.getDefault().getServices();
        Enumeration en = sr.services(JavaIndentEngine.class);
        if (!en.hasMoreElements()) {
            // Need to add
            List l = sr.getServiceTypes();
            l.add(new JavaIndentEngine());
            sr.setServiceTypes(l);
        }
    }
 */

    public boolean getCompletionAutoPopup() {
        return getSettingBoolean(ExtSettingsNames.COMPLETION_AUTO_POPUP);
    }
    public void setCompletionAutoPopup(boolean v) {
        setSettingBoolean(ExtSettingsNames.COMPLETION_AUTO_POPUP, v,
            COMPLETION_AUTO_POPUP_PROP);
    }

    public boolean getCompletionCaseSensitive() {
        return getSettingBoolean(ExtSettingsNames.COMPLETION_CASE_SENSITIVE);
    }
    public void setCompletionCaseSensitive(boolean v) {
        setSettingBoolean(ExtSettingsNames.COMPLETION_CASE_SENSITIVE, v,
            COMPLETION_CASE_SENSITIVE_PROP);
    }

    public boolean getCompletionInstantSubstitution() {
        return getSettingBoolean(ExtSettingsNames.COMPLETION_INSTANT_SUBSTITUTION);
    }
    public void setCompletionInstantSubstitution(boolean v) {
        setSettingBoolean(ExtSettingsNames.COMPLETION_INSTANT_SUBSTITUTION, v,
            COMPLETION_INSTANT_SUBSTITUTION_PROP);
    }

    public int getCompletionAutoPopupDelay() {
        return getSettingInteger(ExtSettingsNames.COMPLETION_AUTO_POPUP_DELAY);
    }
    public void setCompletionAutoPopupDelay(int delay) {
        if (delay < 0) {
            NbEditorUtilities.invalidArgument("MSG_NegativeValue"); // NOI18N
            return;
        }
        setSettingInteger(ExtSettingsNames.COMPLETION_AUTO_POPUP_DELAY, delay,
            COMPLETION_AUTO_POPUP_DELAY_PROP);
```

```
}

public int getJavaDocAutoPopupDelay() {
    return getSettingInteger(ExtSettingsNames.JAVADOC_AUTO_POPUP_DELAY);
}
public void setJavaDocAutoPopupDelay(int delay) {
    if (delay < 0) {
        NbEditorUtilities.invalidArgument("MSG_NegativeValue"); // NOI18N
        return;
    }
    setSettingInteger(ExtSettingsNames.JAVADOC_AUTO_POPUP_DELAY, delay,
        JAVADOC_AUTO_POPUP_DELAY_PROP);
}

public boolean getDisplayGoToClassInfo() {
    return getSettingBoolean(ExtSettingsNames.DISPLAY_GO_TO_CLASS_INFO);
}
public void setDisplayGoToClassInfo(boolean v) {
    setSettingBoolean(ExtSettingsNames.DISPLAY_GO_TO_CLASS_INFO, v,
        DISPLAY_GO_TO_CLASS_INFO_PROP);
}

public boolean getJavaDocAutoPopup() {
    return getSettingBoolean(ExtSettingsNames.JAVADOC_AUTO_POPUP);
}
public void setJavaDocAutoPopup(boolean auto) {
    setSettingBoolean(ExtSettingsNames.JAVADOC_AUTO_POPUP, auto,
        JAVADOC_AUTO_POPUP_PROP);
}

public Color getJavaDocBGColor() {
    return (Color)getSettingValue(ExtSettingsNames.JAVADOC_BG_COLOR);
}
public void setJavaDocBGColor(Color c) {
    setSettingValue(ExtSettingsNames.JAVADOC_BG_COLOR, c,
        JAVADOC_BGCOLOR);
}

public Dimension getJavaDocPreferredSize() {
    return (Dimension)getSettingValue(ExtSettingsNames.JAVADOC_PREFERRED_SIZE);
}
public void setJavaDocPreferredSize(Dimension d) {
    setSettingValue(ExtSettingsNames.JAVADOC_PREFERRED_SIZE, d,
        JAVADOC_PREFERRED_SIZE_PROP);
}


public String getUpdatePDAfterMounting(){
    return (String)getSettingValue(ExtSettingsNames.UPDATE_PD_AFTER_MOUNTING);
}

public void setUpdatePDAfterMounting(String update){
    setSettingValue(ExtSettingsNames.UPDATE_PD_AFTER_MOUNTING, update,
    UPDATE_PD_AFTER_MOUNTING_PROP
    );
}

public boolean getShowDeprecatedMembers() {
    return getSettingBoolean(ExtSettingsNames.SHOW_DEPRECATED_MEMBERS);
}
public void setShowDeprecatedMembers(boolean v) {
    setSettingBoolean(ExtSettingsNames.SHOW_DEPRECATED_MEMBERS, v,
        SHOW_DEPRECATED_MEMBERS_PROP);
}

public boolean getCompletionNaturalSort() {
    return getSettingBoolean(ExtSettingsNames.COMPLETION_NATURAL_SORT);
}
public void setCompletionNaturalSort(boolean v) {
    setSettingBoolean(ExtSettingsNames.COMPLETION_NATURAL_SORT, v,
        COMPLETION_NATURAL_SORT_PROP);
}

public int getFastImportSelection() {
    return getSettingInteger(ExtSettingsNames.FAST_IMPORT_SELECTION);
}
public void setFastImportSelection(int sel) {
```

```
        setSettingInteger(ExtSettingsNames.FAST_IMPORT_SELECTION, sel,
            FAST_IMPORT_SELECTION_PROP);
    }

    public boolean getFastImportPackage() {
        return (getFastImportSelection() == JavaFastImport.IMPORT_PACKAGE);
    }
    public void setFastImportPackage(boolean v) {
        if (v) setFastImportSelection(JavaFastImport.IMPORT_PACKAGE);
    }

    public boolean getShowContract() {
        return getSettingBoolean(ExtSettingsNames.SHOW_CONTRACT);
    }
    public void setShowContract(boolean auto) {
        setSettingBoolean(ExtSettingsNames.SHOW_CONTRACT, auto,
            SHOW_CONTRACT_PROP);
    }

    protected Class getDefaultIndentEngineClass() {
        return JavaIndentEngine.class;
    }

    public HelpCtx getHelpCtx () {
        return new HelpCtx (HELP_ID);
    }

}
```

E.14 org.netbeans.modules.editor.options.JavaOptionsBeanInfo.java

```
/*
 *                 Sun Public License Notice
 *
 * The contents of this file are subject to the Sun Public License
 * Version 1.0 (the "License"). You may not use this file except in
 * compliance with the License. A copy of the License is available at
 * http://www.sun.com/
 *
 * The Original Code is NetBeans. The Initial Developer of the Original
 * Code is Sun Microsystems, Inc. Portions Copyright 1997-2003 Sun
 * Microsystems, Inc. All Rights Reserved.
 *
 * Contributor(s): Håkan Bergmark, Tony Bergh
 */

package org.netbeans.modules.editor.options;

import java.beans.*;
import java.awt.Image;
import org.netbeans.editor.ext.ExtSettingsNames;
import org.netbeans.modules.editor.java.PDCustomizerEditor;

import org.openide.util.NbBundle;

/** BeanInfo for plain options
*
* @author Miloslav Metelka
* @version 1.00
*/
```

```
   public class JavaOptionsBeanInfo extends BaseOptionsBeanInfo {

       /** Propertydescriptors */
       private static PropertyDescriptor[] descriptors;
       /** Additional beaninfo */
       private static BeanInfo[] additional;

       private static final String[] EXPERT_PROP_NAMES = new String[] {
           JavaOptions.JAVADOC_BGCOLOR,
           JavaOptions.JAVADOC_AUTO_POPUP_DELAY_PROP,
           JavaOptions.JAVADOC_PREFERRED_SIZE_PROP,
           JavaOptions.JAVADOC_AUTO_POPUP_PROP,
           JavaOptions.COMPLETION_CASE_SENSITIVE_PROP,
           JavaOptions.UPDATE_PD_AFTER_MOUNTING_PROP,
           JavaOptions.SHOW_DEPRECATED_MEMBERS_PROP,
           JavaOptions.COMPLETION_INSTANT_SUBSTITUTION_PROP,
           JavaOptions.COMPLETION_NATURAL_SORT_PROP,
           JavaOptions.FAST_IMPORT_PACKAGE_PROP,
           JavaOptions.SHOW_CONTRACT_PROP
               };


       public JavaOptionsBeanInfo() {
           super("/org/netbeans/modules/editor/resources/javaOptions"); // NOI18N

       }

       protected String[] getPropNames() {
           return OptionSupport.mergeStringArrays(BaseOptions.BASE_PROP_NAMES,
JavaOptions.JAVA_PROP_NAMES);
       }

       protected void updatePropertyDescriptors() {
           super.updatePropertyDescriptors();

           setExpert(EXPERT_PROP_NAMES);
           setPropertyEditor(JavaOptions.UPDATE_PD_AFTER_MOUNTING_PROP,
UpdatePDAfterMountingEditor.class);
           setHidden(new String[] {
               JavaOptions.FORMAT_SPACE_BEFORE_PARENTHESIS_PROP,
               JavaOptions.FORMAT_COMPOUND_BRACKET_ADD_NL_PROP,
               JavaOptions.FAST_IMPORT_PACKAGE_PROP
           });
       }

       protected Class getBeanClass() {
           return JavaOptions.class;
       }

       public static class UpdatePDAfterMountingEditor extends PropertyEditorSupport {
```

```
        private static String[] tags = new String[] {
                                           ExtSettingsNames.ALWAYS,
                                           ExtSettingsNames.NEVER
                                       };

        private static String[] locTags = new String[] {
                                              getString("ALWAYS"), // NOI18N
                                              getString("NEVER") // NOI18N
                                          };

        public String[] getTags() {
            return locTags;
        }

        public void setAsText(String txt) {
            for (int i = 0; i < locTags.length; i++) {
                if (locTags[i].equals(txt)) {
                    setValue(tags[i]);
                    break;
                }
            }
        }

        public String getAsText() {
            String val = (String) getValue();
            for (int i = 0; i < tags.length; i++) {
                if (tags[i].equals(val)) {
                    return locTags[i];
                }
            }
            throw new IllegalStateException();
        }

        static String getString(String s) {
            return NbBundle.getMessage(JavaOptionsBeanInfo.class, s);
        }

    }

}
```

F SUN Public License

```
SUN PUBLIC LICENSE Version 1.0

1. Definitions.

1.0.1. "Commercial Use" means distribution or otherwise making the
```

Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof and corresponding documentation released with the source code.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and

apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated documentation, interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1 The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters.

(a) Third Party Claims.

If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL'' which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.

If Contributor's Modifications include an application programming interface ("API") and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section

3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.

You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions.

Sun Microsystems, Inc. ("Sun") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Sun. No one other than Sun has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.

If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must: (a) rename Your license so that the phrases "Sun," "Sun Public License," or "SPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Sun Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A

shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS'' BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that

Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions.

With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as ?Multiple-Licensed?. ?Multiple-Licensed? means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

Exhibit A -Sun Public License Notice.

The contents of this file are subject to the Sun Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. A copy of the License is available at http://www.sun.com/

The Original Code is _____. The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright (C)_____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms of the _____ license (the ?[___] License?), in which case the provisions of [_____] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [____] License and not to allow others to use your version of this file under the SPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [___] License. If you do not delete the

provisions above, a recipient may use your version of this file under either the SPL or the [___] License."

[NOTE: The text of this Exhibit A may differ slightly from the text of the notices in the Source Code files of the Original Code. You should use the text of this Exhibit A rather than the text found in the Original Code Source Code for Your Modifications.]