Department of Computer Science

Johan Bjärneryd
Jhonny Carvajal

Prototype maker

-Ett prototypverktyg för mjukvaruindustrin

Prototype maker

A prototype tool for the software industry

Computer Science
C-level thesis (15hp)

Datum/Termin:	09-06-05
Handledare:	Martin Blom
Examinator:	Donald F. Ross
Löpnummer:	C2009:08

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

This report is submitted in partial fulfillment of the requirements for the Bachelor's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Johan Bjärneryd

Jhonny Carvajal

Approved, 2009-06-06

Advisor: Martin Blom

Examiner: Donald Ross

Abstrakt

Rapporten handlar om framställningen av ett prototypverktyg för konsultfirman Logica. Marknaden har idag ett stort behov av ett prototypverktyg och det är en lösning på det problemet som vi utvecklat. Utvecklingsarbetet är utfört i Java och har resulterat i ett program som kan rita upp prototyper och ett som kan visa och köra prototypprojekt. Projektets syfte har varit att effektivisera Logicas arbetssätt och tillgodose ett behov av en mjukvara som är avsett för att skapa och hantera prototyper av mjukvarusystem. Därutöver har projektet syftat till att erbjuda Logicas kunder mervärde då de får en körbar prototyp av det system de överväger att köpa.

Abstract

This thesis reports on the work of developing a prototype tool for the consultant firm Logica.Today's software development market is in great need of a prototyping tool and that is the problem we have tried to solve. Our development work that was done in Java has resulted in a piece of software that can draw prototypes and another that can show and run them. The purpose of the project has been to create customer satisfaction and customer value, at the same time as the software tool has made Logicas way to work more efficient.

Sammanfattning

I mjukvarubranschen saknas det idag ett bra prototypverktyg. Detta är det problem som vi sökt en lösning på under vårt examensarbete. Vi har haft den stora konsultfirman Logica som kund vid framställandet av ett sådant verktyg. Sammanfattningsvis anser vi att vi nådde vårt mål att leverera en stabil mjukvara till Logica. Vi lyckades göra detta inom utsatta tidsramar och enligt de uppsatta krav som fanns för projektet. Projektet har dragits med problem i kravställningen, då Logica inte riktigt kunnat möta våra förväntningar på att hålla en bra kravdialog och i tidigt skede komma med en bra kravlista. Trots detta problem så har vi mött alla deadlines. Produkten kommer att bespara Palassogruppen inom Logica mycket tid då de kan effektivisera sin verksamhet. Vi levererar enkelhet där det tidigare varit omständliga arbetsmoment. Till detta har vi utvecklat en mjukvara som ger mervärde åt vår kunds kunder då de får ett helt nytt verktyg att arbeta med vid sidan av sin konsultkontakt. Teknikvalet har varit lyckat då vi med enkelhet har kunnat utveckla mjukvarorna, och då den varit tillräckligt kraftfull för att leverera det önskade slutresultatet. Vi tror ganska starkt att produkten kan vidareutvecklas och säljas till fler kunder än Logica, just med tanke på att marknaden behöver ett bra verktyg, samtidigt som det inte finns någon aktör som erbjuder samma tjänster som vårt verktyg.

1. Figurförteckning

2. Tabellförteckning

3. Ordlista

Scrum[1] – Är en projektledningsmetod anpassad för mjukvaruindustrin som bygger på korta iterationer som körs vanligen i 2-4 veckors intervaller. Tanken med korta intervaller är att man ska få feedback tidigt och kan på så sätt vara i rätt riktning.

RUP[2] – Bygger också på korta iterationer med många steg i varje iteration så som planering, analys, design, implementation m.m. Denna metod tar längre tid än Scrum då det är fler steg och man måste följa modellen.

Vattenfallsmodellen[3] – Allt utförs i fallande ordning som resulterar i en mjukvara. Exempelvis så börjar man med kravanalys, design, implementation.

Prototyp – En modell av en färdig mjukvara som t.ex. vad prototype maker som ritar upp en modell av ett grafikst användargränssnitt.

XP – eXtreme Programming, en samling principer om hur man ska utveckla mjukvara, exempelvis parprogrammering, vilket innebär att man sitter bredvid varandra och hjälper varandra och koda.

JAR – Java Archive är en fil som innehåller klassfilen som antingen kan användas som klassarkiv eller körbart program.

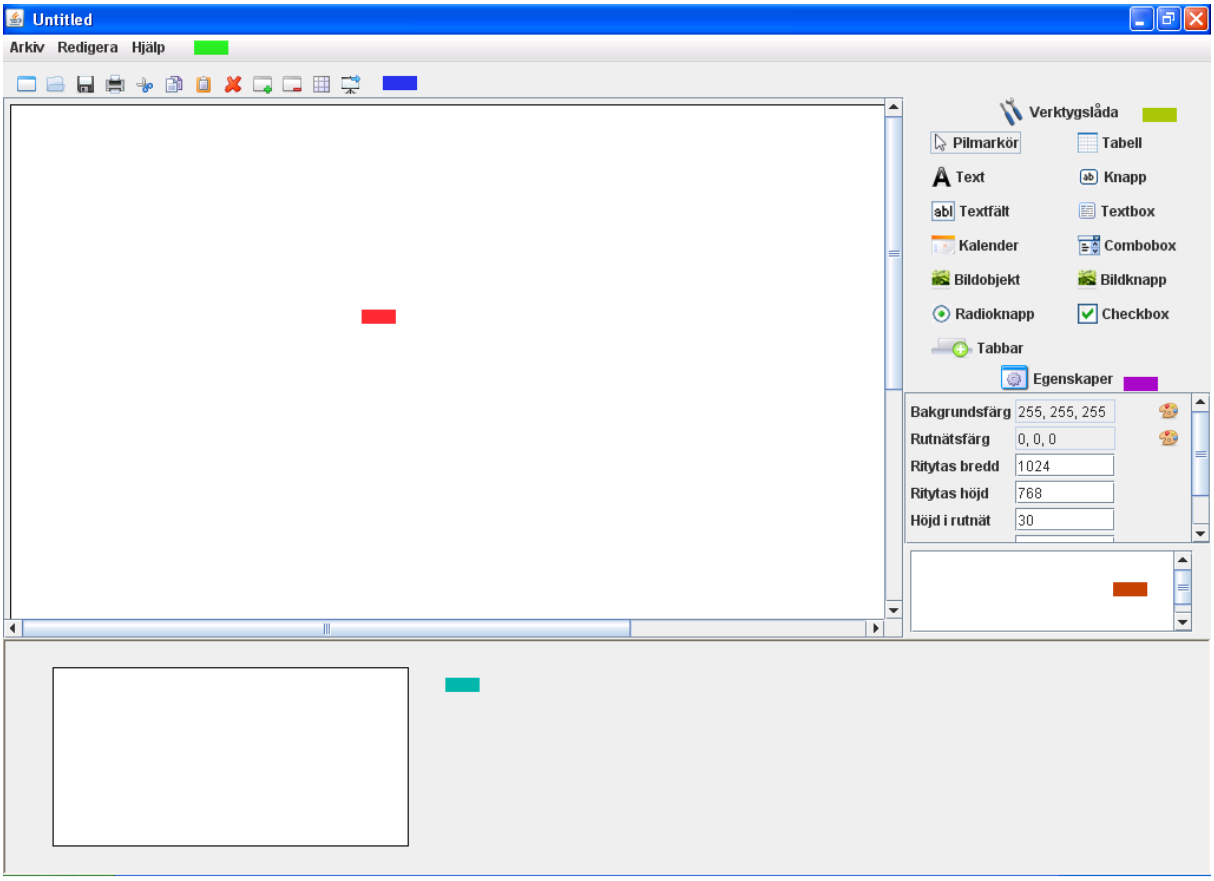
[1] Scrum and XP from the Trenches, Henrik Kniberg, InfoQ, S18
[2] The Rational Unified Process. Svenska Utgåvan, Philippe Kruchten, Addison Wesley, S.17
[3] http://sv.wikipedia.org/wiki/Vattenfallsmodellen

4. Introduktion

4.1. Programöverblick

4.1.1. Prototypverktyg

Nedan finns en skärmdump över hur den överlämnade versionen av vårt prototypverktyg ser ut. Färgmarkeringarna förklaras sedan i en tabell.



Figur 1: Skärmdump av mjukvaran vid leverans

	Huvudmeny.
	Menyrad med genvägar till vanliga använda funktioner.
	Programmets rityta.
	Programmets verktygslåda.
	Egenskapsfält för de olika komponenterna och ritytorna.
	Anteckningsfält. Här kan man lägga en kommentar till en rityta.
	Miniatyrer. En minatyrbild av varje rityta i projektet visas.

Tabell 1: Färgkoder för skärmdump i Figur 1

4.2. Kunden

4.2.1. Logica

WM-data, som fram till 2006 var ett av Sveriges ledande IT-företag[4], förvärvades av Logica, som är ett brittiskt telekom- och IT-företag med säte i London. Logicas verksamhet i Norden utgörs av det uppköpta WM-data. Logica är aktivt i 42 länder och hade 2005 en omsättning på 165 miljarder svenska kronor. Cirka 40 000 arbetar inom Logica, varav cirka 5 500 i Sverige. De tjänster som företaget erbjuder är bland annat konsulttjänster, integration samt outsourcing av it- och affärsprocesser.[5]

4.2.2. Palasso

Human Resources, HR, beskriver hur ett företag hanterar sina mänskliga resurser, dvs. personalen.[6] Palasso är ett datorsystem som hanterar just detta, och via ett och samma gränssnitt kan alla personaladministrativa arbetsprocesser utföras.[7] Palasso ägs i sin helhet av Logica och utvecklingen samt driften av systemet bedrivs vid kontoret i Karlstad. Därifrån erbjuds även konsulttjänster samt support på Palasso.[8] Målgruppen för Palasso är statlig verksamhet, såsom kommuner, universitet, Försvarsmakten och landsting, och är därför specialutformat för den typen av kunder. Palassos gränssnitt specialanpassas för varje kund vilket gör att kunderna kan påverka hur systemet de köper skall se ut. Då Palasso satt fast i ett omständigt, tidskrävande och ineffektivt system för att göra prototyper åt de kunder som köper in Palasso som system, blev de vår kund/produktbeställare då de behövde reformera sitt sätt att skapa och hantera prototyper.

[4] http://sv.wikipedia.org/wiki/WM-data
[5] http://www.logica.se/företaget/15002
[6] http://sv.wikipedia.org/wiki/Human_Resources
[7] http://www.logica.se/palasso/400007833
[8] http://www.palasso.com/

4.3. Metod

Redan på idéstadiet för projektet så bestämde vi inom gruppen att det var Scrum som skulle användas som arbetsmetod under projektets gång. Eftersom vi under flera kurser fått lära oss om olika metoder för mjukvaruutveckling t.ex. vattenfallsmodellen, RUP och Scrum, visste vi att det var Scrum som skulle passa bäst att applicera på projektet. Dels eftersom vi dels hade en fast, orörlig tidsram inom vilken vi var tvungna att bli färdiga, och dels eftersom vi visste att kravbilden riskerade att ändras över tiden. Vi började således begära in en kravspecifikation från Palassogruppen som dock resulterade i att vi fick PowerPoint-slides skickade till oss, vilka innehöll slides över hur Palasso kan se ut. Utifrån dessa ställde vi upp en backlog för produkten som skulle utvecklas. Dock uppkom problem då Logica är i en fas där de precis börjat använda Scrum och därmed inte kunde ta ställning till backlogen som vi satt upp, utan vi fick istället anpassa vår metod efter deras arbetssätt, genom att göra en förstudie och lämna över till dem, och som sedan blev ett levande dokument under hela utvecklingstiden. Förutom problematiken med kravlistan så har valda bitar av Scrum använts genom projektet. Vår sprintlängd har legat på 2 veckor och varannan måndag så har vi haft sprintdemo för projektgruppen, då vi har visat upp produkten så långt vi har kommit och bestämt vad som skall ha blivit utfört fram tills nästa möte. Mellan mötena så har vi haft designmöten när det har varit aktuellt, för att diskutera arkitektur och lösningsförslag, avstämning med hur vi ligger till i utvecklingen och övrig tid har gått till att koda systemet.

5. Bakgrund

Prototypverktyg är ett verktyg som är en efterfrågad produkt på marknaden eftersom det underlättar för företaget att visa systemet innan det är färdigutvecklat. Det hjälper kunden att på ett tidigt stadium se hur den beställda mjukvaran kommer att se ut i vid leverans. Detta hjälper både kunden och leverantören, då rätt produkt kan tas fram redan från början, vilket betyder att man kan komma ifrån dyra ändringar i slutfasen.

När Logica beskrev sitt sätt att arbeta på, insåg vi att behovet av ett prototypverktyg var stort. Dagens arbetssätt för att visa systemet för kunden är med hjälp av en PowerPoint-presentation. De ritar PowerPoint-bilderna genom att infoga bilder av grafiska komponenter, och på så vis ritar de hur mjukvarans olika skärmbilder kommer att se ut. I och med att Palasso anpassas efter sina kunder så krävs det att de ritar upp en ny PowerPoint-presentation för varje kund vilket är tidsödande. Den största nackdelen med PowerPoint-metoden är att man inte kan se ett flöde, vilket exempelvis innebär att inget händer när man trycker på en knapp. Det är ett stort problem då risken finns att kunden inte får rätt uppfattning om hur systemet fungerar när det väl körs.

Vi fick via kontakter reda på att Logica skulle byta arbetssätt till Scrum. Scrum är ett arbetssätt som kör med korta iterationer för att på så sätt kunna öka kvalitén på mjukvara och på ett tidigt stadium kunna uppfatta kravförändringar från kund efter varje produktdemonstration som sker efter en viss tidsintervall.[9] Med vår produktidé som innehöll bland annat ett prototypverktyg kunde deras nya arbetssätt stödjas. Behovet av prototypverktyget ansågs större än resterande delar vilket gjorde att det kunde sättas i fokus och anpassas efter deras behov. Detta skulle stärka konsultorganisationen genom att förbättra kundens syn av systemet, redan innan systemet skapats.

[9] se.wikipedia.org/wiki/scrum

5.1. Projektdiskussion

I det här avsnittet kommer projektet att diskuteras ur ett bakgrundsperspektiv.

5.1.1. Omfattning

Det var från början tänkt att mjukvaran vi skulle leverera skulle stödja mycket mer än bara prototypframställning. Mjukvaran var tänkt att hjälpa deras byte av arbetssätt till Scrum och skulle innehålla stöd för Scrums olika delar, så som backlogs, burndown charts, tasktable och kommunikation mellan utvecklarna. När de beskrev deras stora behov av prototypverktyg beslutades att leverera ett prototypverktyg som en egen mjukvara istället för det tänkta mjukvaran. Allt eftersom projektet fortskred så kom även ett behov av ett visningsprogram för de prototyper man tagit fram. Detta för att kunderna skulle kunna ta med sig prototypen och visa den för sina interna avdelningar, utan möjlighet att kunna ändra i den. Principen med prototypverktyg med tillhörande visare kan liknas vid Adobe Acrobat[10] som används för att skapa PDF-filer och Adobe Acrobat Reader[11] som används för att enbart visa upp filerna[12].

5.1.2. Existerande system

Det finns redan existerande system som kan användas för att skapa prototyper av projekt. De flesta av dessa vänder sig dock till webutvecklare. Vissa har stöd för flöde och andra inte. Exempel på existerande system är Axure[13], Balsamic[14] och Pencil[15]. Balsamic har minst delar som liknar vår produktidé då det endast är inriktat på webb och helt saknar flödeshantering. Axure har många delar man kan jämföra med vårt system i upplägg, flödeshantering och kan köra projektet utifrån det man ritat. Det som skiljer oss från dem är att deras är inriktat mot webutveckling medan vårt är inriktat mot stand-alone-program. I första anblicken av Pencil såg det ut som det vi tänkt utveckla, t.ex. rita upp ett system och tilldela bakgrundsfärger, knappar, namn på komponenter samt att det körs på olika plattformar. Om man tittar närmare på den så

[10] http://www.adobe.com/se/products/acrobat/?promoid=BPCGM
[11] http://www.adobe.com/se/products/acrobat/reader.html
[12] http://www.adobe.com/se/products/acrobat/adobepdf.html
[13] www.axure.com
[14] www.balsamic.com
[15] www.ewolus.vn/pencil

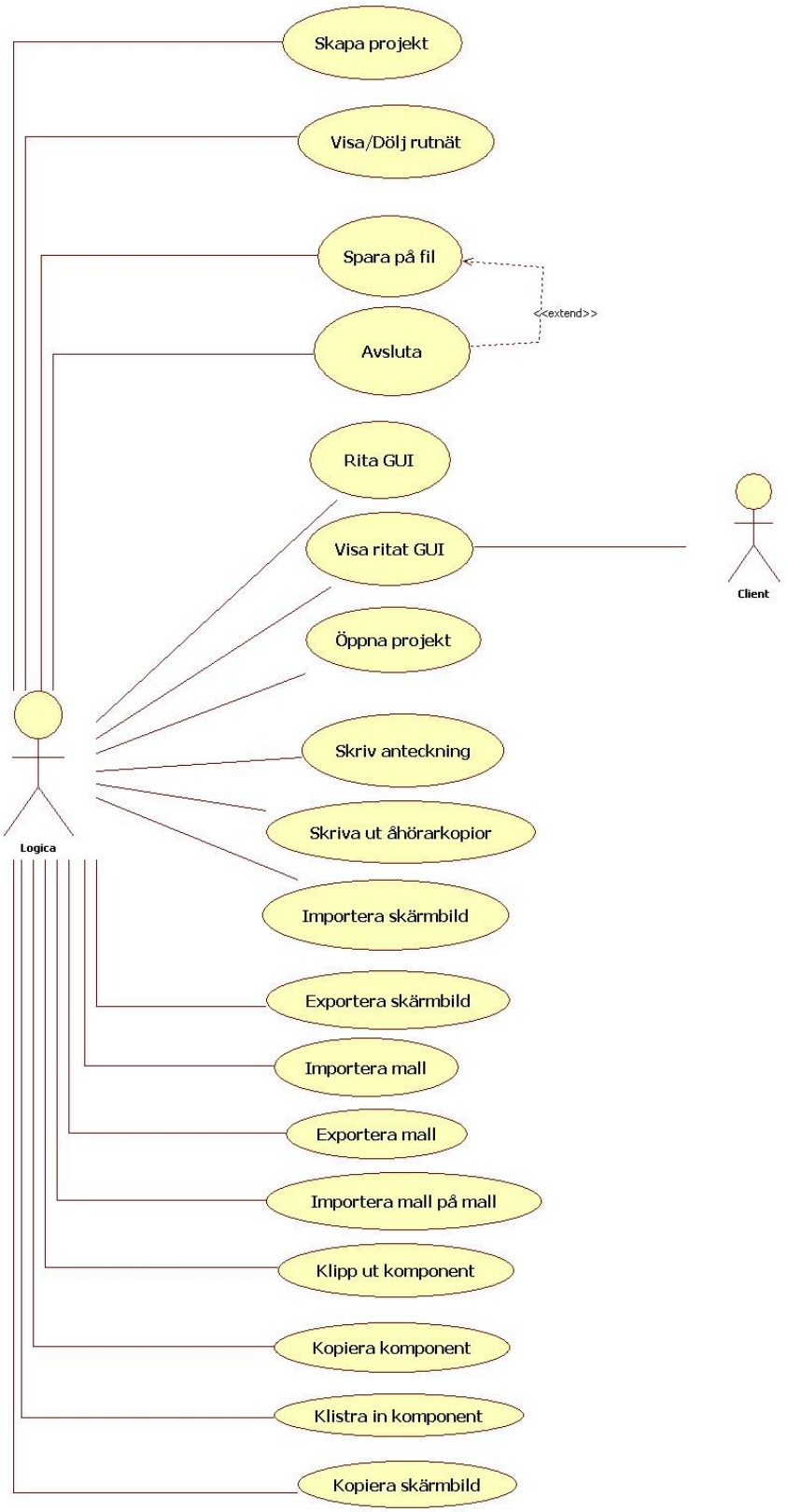
märker man dock att den saknar flödeshantering vilket även medför att man inte kan köra projekt man ritat upp. Något som också skiljer sig åt mellan dessa tre system vi gett exempel på är att de system som är inriktade på webutveckling är kommersiella medan Pencil är open-source. Vår nisch är att erbjuda ett system som inte bara kan rita upp ett fullfjädrat gränssnitt med verktyg såsom rutnät, utan även erbjuder en kraftfull miljö i vilken prototypen kan köras och visas upp för kunder. Detta ger kunder möjlighet att på en tidig nivå testa hur systemet interagerar med de olika delarna och hur man kommer att hantera det.

5.1.3. Prototypskaparen

Prototypverktyget är en Javalösning och distribueras som en JAR-fil[16]. Ledordet vid utvecklingen har varit enkelhet, då användarna sitter på olika delar i organisationen. De olika arbetsrollerna användarna har är bland annat säljare, konsulter och systemutvecklare. Med hjälp av vår flödeshantering kan man lägga till funktionalitet till de olika komponenterna, ex. lägga till hantering för att få knappar att lyssna på klick. Detta finns för att man skall kunna interagera med andra delar i prototypen, exempelvis byta skärmbild vid knapptryck.

För varje skärmbild finns även ett fält för att göra anteckningar till den aktiva skärmbilden för att underlätta när man har många bilder och när man delar på prototypen mellan de olika användarna som ingår i projektet. Denna del med anteckningar är vi, enligt våra efterforskningar, ensamma om på prototypmarknaden. Nedan visas ett usecase-diagram över prototypverktyget.

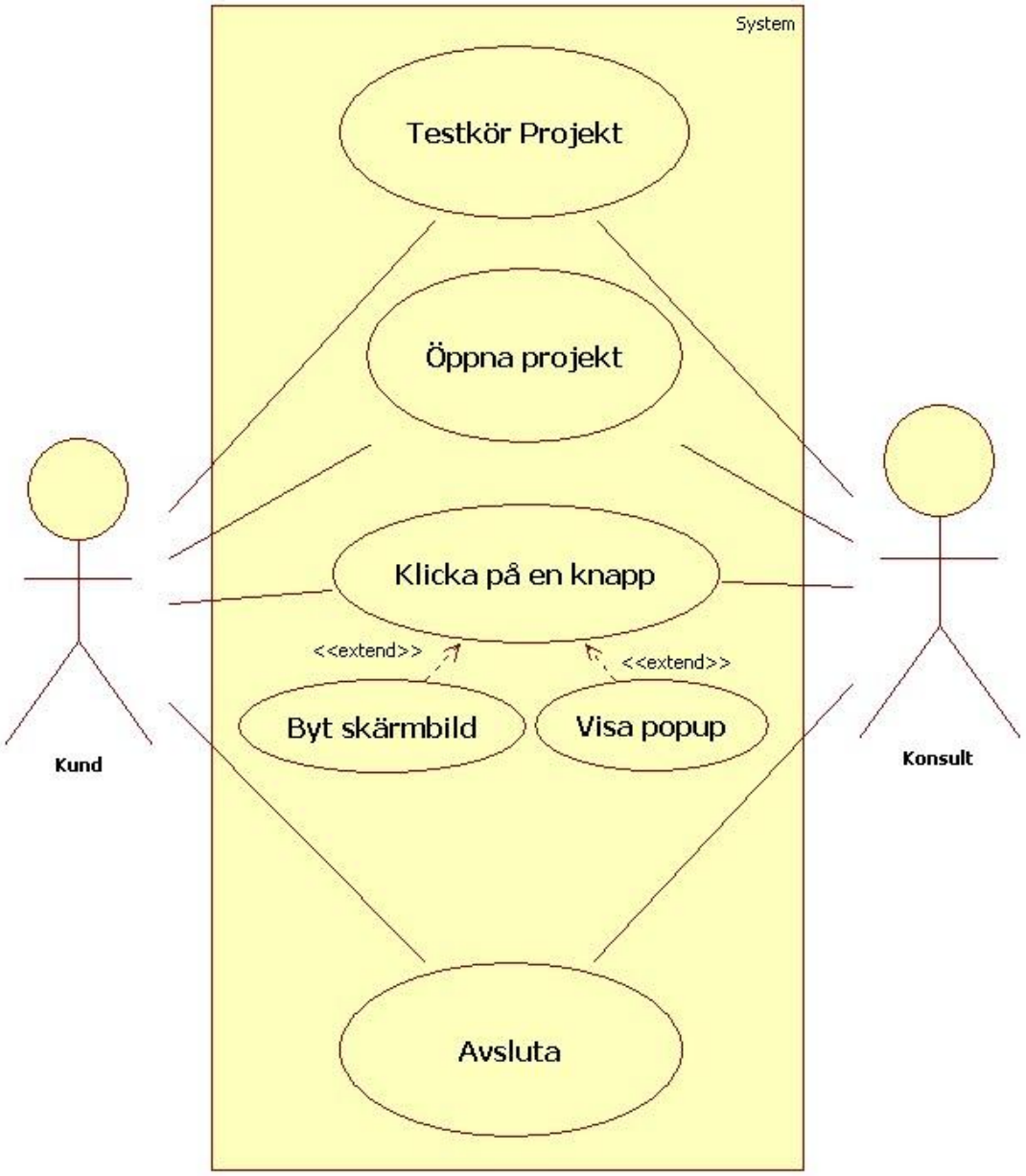
[16] http://en.wikipedia.org/wiki/JAR_(file_format)



Figur 2: UseCase för prototypverktyget

5.1.4. Prototypvisaren

Prototypvisaren är även den byggd på Java och distribueras som en JAR-fil. Programmet kom till som ett led i att erbjuda kundtillfredsställelse. Detta då kunderna erbjuds en möjlighet att få med sig prototypen hem till sitt företag för att kunna utvärdera den grundligt. Ett exempel på nyttan kan vara att kunden behöver rapportera tillbaka till sitt företags styrelse och då kan han/hon visa upp vad det är som blivit erbjudet. Viktigt är dock att ingen ritningsfunktionalitet återfinns i visaren, eftersom tanken är att underlätta för kunden och erbjuda lite mer, men inte ge dem en möjlighet att ta fram eller ändra om i prototypen utan konsultens närvaro. Nedan visas ett usecase-diagram över prototypvisaren.



Figur 3: UseCase för PrototypeViewer

5.1.5. Resurstillgång

Den största resursen vi har att tillgå under projektet är vi själva, då vi är den enda arbetskraften i projektet. Vi har haft full tillgång till Logicas kontor och lokaler, vilka vi utnyttjat under hela projektets gång. Datorer blev vi försedda med av Logica, men vi har även använt våra egna maskiner för projektet, bland annat en servermaskin för att möjliggöra samarbete med hjälp av Subversion. Vidare har vi haft tillgång till kurslitteratur, dels från aktuell kurs, dels från tidigare kurser. En projektgrupp har funnits till hands för att ställa krav på projektet samt ge feedback på regelbundna möten. Gruppen har bestått av vår handledare på Logica, chefen för Palasso, samt två personer ut Palassogruppen. Andra tekniska hjälpmedel vi haft har varit Internet, javas API[17], modelleringsverktyget StarUML[18] samt utvecklingsmiljön Eclipse[19]. Beräknad åtgång av mantimmar ligger på 600 timmar, baserat på att vi arbetar 3 dagar * 8h / vecka med projektet

5.1.6. Framgångsfaktorer

För att lyckas med projektet behövs följande:

- **Nära kundkontakt**

  Vi anser kundkontakten som en nyckelfaktor för att kunna lyckas med projektet, eftersom tidsramen är begränsad, vilket gör att vi behöver utveckla rätt sak redan från början. Det är även viktigt eftersom vi behöver få insikt i hur de arbetar för att kunna stödja deras process på bästa sätt.

- **Miljö fungerar som den ska**

  För att bli klara i tid krävs att utvecklingsmiljön, dvs, utvecklingsdatorerna och Subversionservern fungerar från första början. Tidsmässigt anser vi att ett datorhaveri skulle kunna äventyra hela projektet om data gick förlorad.

- **Kravspecifikationen**

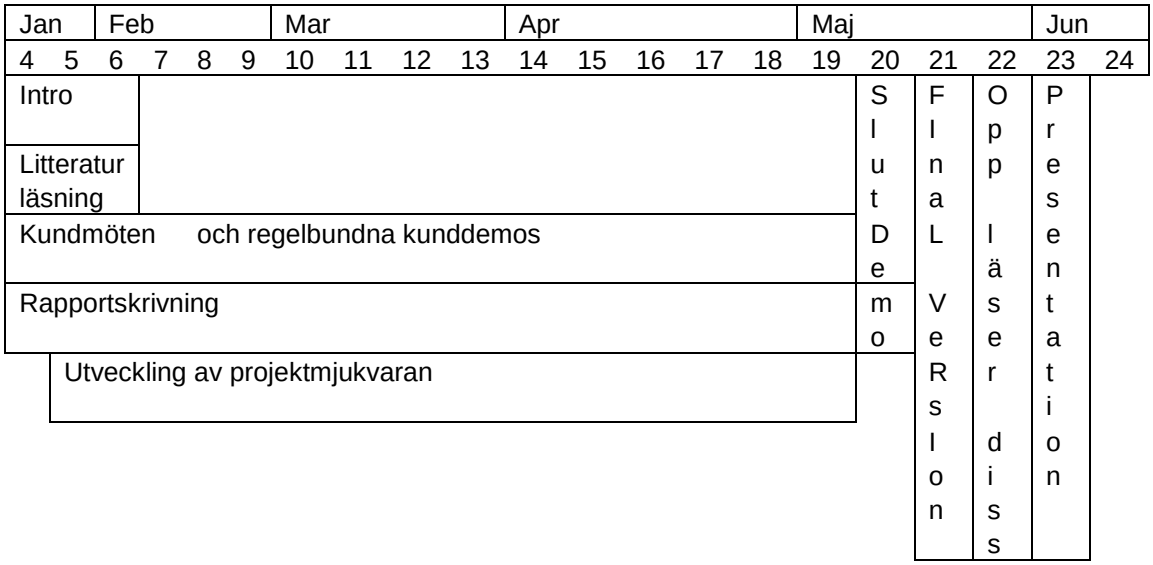
  En kravspecifikation behövs för att kunna utveckla efter kundens prioriteringar i rätt ordning. Då vi planerat att följa Scrums principer så tänker vi inte utveckla saker som ej står i projektets backlog. Detta eftersom vi anser att vi behöver all tid till att utveckla de saker som faktiskt efterfrågas.

[17] http://en.wikipedia.org/wiki/List_of_Java_APIs
[18] http://staruml.sourceforge.net/en/
[19] http://www.eclipse.org/

- **Engagemang**

  För att nå vårt mål behöver vi manifestera ett stort engagemang för uppgiften. Många timmar kommer att behöva sättas in i projektet, för att hinna med att utveckla programmen, rita diagram, dokumentera hela processen samt att hålla kundmöten.

5.1.7. Projektplan



Figur 4: Tidsplanering

5.1.8. Arbetsfördelning

Vi har ingen fast arbetsfördelning inom gruppen, vilket främst beror på att vi endast är två personer i gruppen. Vid designsteg har vi arbetat tillsammans och tänkt ut arkitekturen för vår mjukvara. Vi ansåg det viktigt att båda är med i detta steg och får en djup insikt i hur mjukvaran skall vara uppbyggd. Vid rapportskrivning gäller samma, då vi anser det är viktigt med ”parskrivning” för att få en så korrekt skriven rapport som möjligt, som speglar vårt projekt och gemensamma arbete. Vid programmeringen av mjukvaran har vi suttit i samma rum och hjälpt varandra vid behov. Koden har varit kollektiv och vem som helst av oss har kunnat gå in och modifierat källkoden som den andre har skrivit. Detta är en princip som vi har lånat från XP[20]. En annan metod som har varit viktig för oss är ”Refactoring” som även den är lånad från XP, och

[20] http://www.extremeprogramming.com/

vars syfte är att bygga om strukturen utan att ändra beteendet. Detta resulterar i att kodkvalitén stärks.

5.1.9. Tidsram

Deadline nummer 1 är det vi visar för Logica vecka 20. Detta är vårt slutdemo då ska mjukvaran vara utvecklad. Det är då överlämningen av mjukvaran sker till kunden. Deadline nummer 2 är då rapporten skall lämnas in, detta inträffar vecka 21. Deadline nummer 3 infaller i slutet av vecka 22 då ska vi ha läst en grupps rapport och skrivit en opposition på den.

5.2. Sammanfattning

Då vi sett att det finns ett stort behov av prototypverktyg så blev vårt projekts fokus omarbetat. Detta då det redan finns flertalet system på marknaden som stödjer Scrums alla steg och metoder, men inget prototypverktyg som mappade mot Logicas behov. De existerande prototypverktygen på marknaden är alla utan flödeshantering vilket gör att det går lika bra med en PowerPoint-presentation för att visa upp hur ett GUI kommer att se ut. Vårt jobb går ut på att leverera två olika produkter som stödjer samma filformat, en prototypskapare och en prototypvisare. De båda mjukvarorna är skrivna i programspråket Java. Resurstillgången är två personer som jobbat med projektet 3 dagar i veckan under hela vårterminen. Vi har en jämn arbetsfördelning då vi är lika ansvariga för att resultatet levereras i tid. Den viktigaste framgångsfaktorn är den nära kundrelationen där vi kan få kontinuerlig feedback från kunden vid projektändringar.

6. Design och utveckling

De första två månaderna, från det att idéerna till projektet lagts fram tills designstadiet och förstudien inleddes, så varierade valet av teknik på grund av att projektets omfång och fokus ändrades. Från det initiala systemet med stöd för Scrum, till ett prototypverktyg med databaskoppling, till det slutgiltiga prototypverktyget utan databaskoppling. Vi hade fritt val av teknik, fast slutresultatet var tvunget att följa vissa riktlinjer. Bland annat så skulle mjukvaran vi tog fram, kunna köras utan att behöva installeras på klienterna. Det skulle bara vara för varje användare att hämta programmet från den interna filservern och köra igång. Ett annat krav var att mjukvaran skulle kunna rita ut specifika komponenter som vi fick en lista på. Detta var ett stort krav då om den möjligheten inte fanns så skulle mjukvaran inte kunna uppfylla sitt ändamål, och därmed bli värdelös för Logica och Palassogruppen. Något krav på att systemet skulle gå att köra på olika typer av operativsystem fanns inte, men det var tvunget att fungera i både Windows XP[21] och Windows Vista[22], vilket gör Java idealiskt för uppgiften då det bara krävs en korrekt JVM[23] installerad för det systemet.

6.1. Möjliga lösningar

I det här avsnittet går vi igenom möjliga lösningar vi haft.

6.1.1. .NET

Från allra första början, när idéerna började komma på papper så valdes .NET ut som den teknik som mjukvaran skulle köras med. Tanken var att kärnan i systemet för Scrum som det vid den tidpunkten var målet att byggas skulle kodas i C# och drivas av en SQL-Server. Till detta skulle moduler göras webbaserade genom att använda systemets kärna tillsammans med ASP.NET. När sedan omfattningen och fokuset ändrades, var det fortfarande tänkt att systemet skulle utvecklas i .NET. De webbaserade bitarna var borta eftersom det nu bara gällde ett ritprogram. Systemet skulle byggas med C# och drivas med SQL-Server, alternativt MySql om licenskostnader skulle innebära ett problem för Logica pga. den rådande finanskrisen.

[21] http://sv.wikipedia.org/wiki/Windows_XP
[22] http://sv.wikipedia.org/wiki/Windows_Vista
[23] http://sv.wikipedia.org/wiki/Java_Virtual_Machine

6.1.2. Java

Ett lösningsförslag baserat på Java diskuterades fram tillsammans med projektgruppen. I förslaget skulle mjukvaran vara kodad i Java och sedan vara körbar som en JAR-fil. I botten skulle det finnas en MySql databas. Databasen skulle hålla rätt på projekt som delades inom en utvecklingsgrupp, och en versionshanterare skulle byggas in i prototypverktyget för att säkerställa att inga filer skrevs över, och tillhandahålla så att man kunde se alla versioner som lagrats och av vem.

6.1.3. Webbaserat

Tankar fanns under ett tag på att leverera lösningen som ett webbaserat system. Tanken var då att det skulle köras på en webbserver, byggt med PHP[24] och köras tillsammans med en MySql databas i botten, som hanterade all information och som hade register över användare som fick använda systemet. När fokus lades om till att endast utveckla ett prototypverkyg så lades dock lösningsförslaget på hyllan eftersom vi ansåg att det var för komplicerat och tidskrävande att utveckla produkten baserat på en webblösning.

6.2. Valet av lösning för det här projektet

I den här delen av rapporten presenterar vi vårt val av lösning.

6.2.1. Programspråk

Lösningsförslaget som vann i kampen mellan de tre var Java. Detta berodde främst på att programmet skulle gå att köra utan att behöva installeras. Då vi tänkte dela upp projektet i olika delar, så föll det sig att .NETs system med att hantera programdelar som .DLL-filer inte var tillräckligt smidigt för slutresultatet. Risken för att man skulle glömma kopiera en DLL-fil, och även det faktum att man skulle behöva kopiera ett flertal filer, gjorde att användningen av vår mjukvara hade blivit för komplicerad. I och med att kravbilden ändrades försvann behovet av att ha en MySql databas i botten då produktbeställarna kom på tanken att använda sitt versionshanteringssystem för filer som skapades i verktyget. En annan klar fördel med att använda sig av Java är att få systemet att fungera oavsett om Logica vill köra systemet på Vista,

[24] http://www.php.net/

kommande Windows 7 eller på en dator som kör Linux[25], så länge den har en kompatibel JVM[26] installerad.

6.2.2. Filformat

När vi tänkte i banor gällande filformat för våra sparfiler så ansåg vi att vi behövde egna format för våra filer. Detta då vi inte vill ha kompabilitet mellan andra program, eftersom vi vill knyta kunder till en konsult, och inte göra det möjligt att kunder själva ändrar om i projektet. Detta medför även att användare blir beroende av vårt visningsprogram för att kunna köra våra filer. Valet av egna filer innebar även att vi slapp läsa in oss på andras API[27] för filer vilket sparade tid, och även gjorde det möjligt för oss att fokusera på hur vi ville skapa vår arkitektur och inte behöva bry oss om hur andra system är uppbyggda. En annan bidragande faktor till varför vi valde att införa egna filformat var att vi behövde använda oss av flera olika för att göra det lättare för användarna att skilja på filer. Att hitta unika relevanta filnamnstillägg var dock inte det lättaste, så vi antog en form där vi blandade bokstäver med siffror.

P4P, Project For Prototype: P4P tillägget berättar för användaren att det är en projektfil. En projektfil innehåller hela projektet som användaren skapat och kan innehålla ett godtyckligt antal skärmbilder som i sin tur kan innehålla ett godtyckligt antal grafiska komponenter. En fil av typ P4P kan öppnas av både huvudprogrammet Prototype samt visningsprogrammet Prototype Viewer.

S4P, Screen For Prototype: En fil med filtillägget S4P, innebär att filen innehåller en skärmbild. En skärmbild exporteras från ett projekt, och sparas då som en S4P-fil och innehåller alla de grafiska komponenter som ritats ut på den skärmbilden. Filen innehåller även eventuell anteckning som gjorts till den aktuella skärmbilden. En fil av typen S4P kan sedan importeras till valfritt projekt och användas i det projektet. Detta är en funktionalitet som gör det väldigt enkelt att dela skärmbilder mellan projekt för att slippa ”uppfinna hjulet” flera gånger.

T4P, Template For Prototype: Filtillägget T4P anger att den sparade filen är en mall för mallfil. Mallfunktionen kom till som ett viktigt led att slippa göra om moment som användes på flera

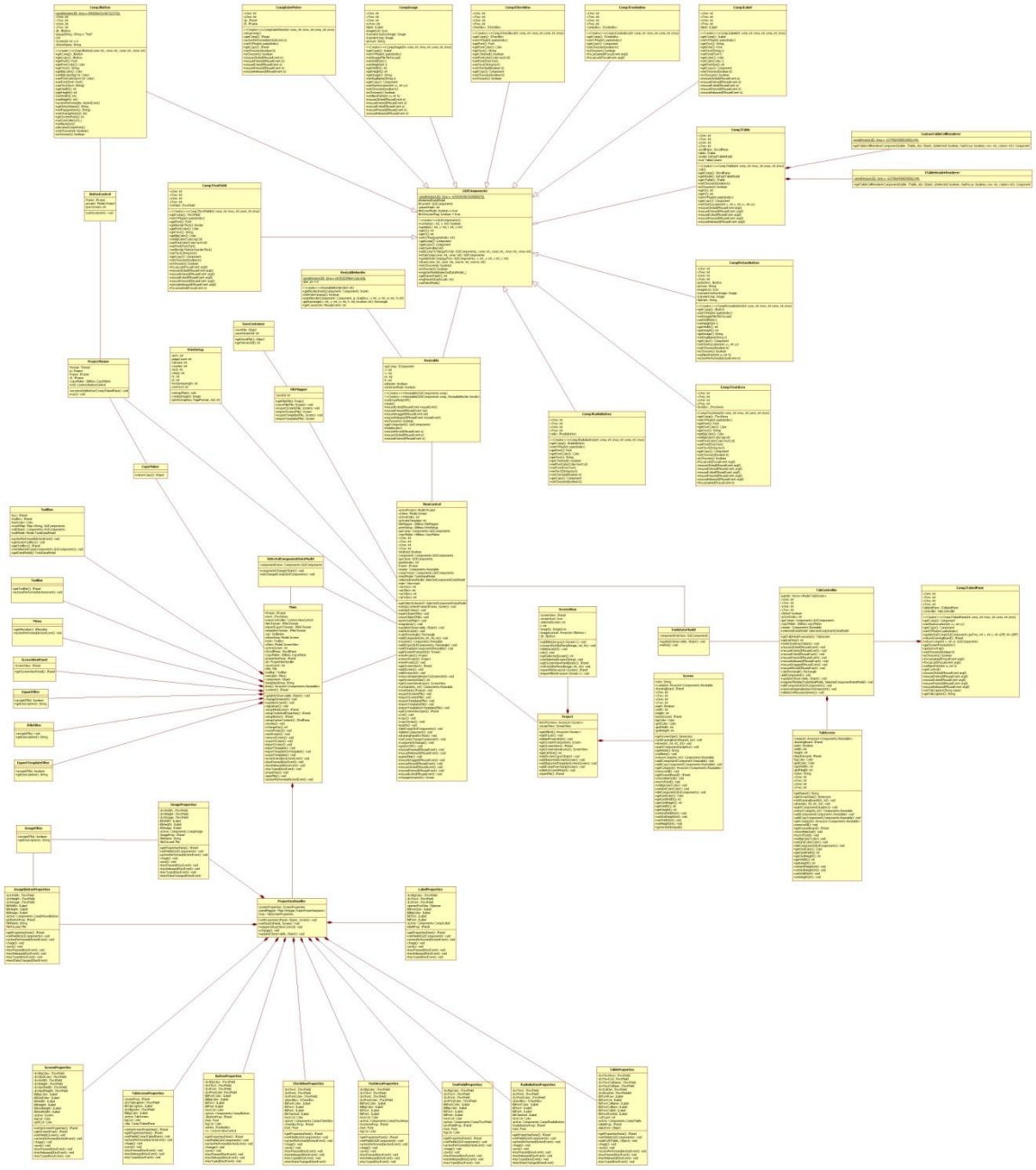
[25] http://sv.wikipedia.org/wiki/Linux
[26] http://sv.wikipedia.org/wiki/Java_Virtual_Machine
[27] http://sv.wikipedia.org/wiki/Application_Programming_Interface

skärmbilder, eller i flera projekt. En mall skapas genom att man ritar på en rityta och därefter väljer att exportera ritytan som en mall. Mallen i fråga kan sedan importeras på valfri rityta i valfritt projekt. En mall kan även importeras över en annan mall vilket gör det möjligt att gruppera en del komponenter i en mall, en grupp i en annan och sedan bygga ihop en helhet utifrån ett arkiv med sparade mallar.

6.3. Prototype beskrivning

Här nedan finns beskrivning om prototype makerns struktur.

6.3.1. Klassdiagram



Figur 5: Klassdiagram prototypverktyget

6.3.2. Designmönster

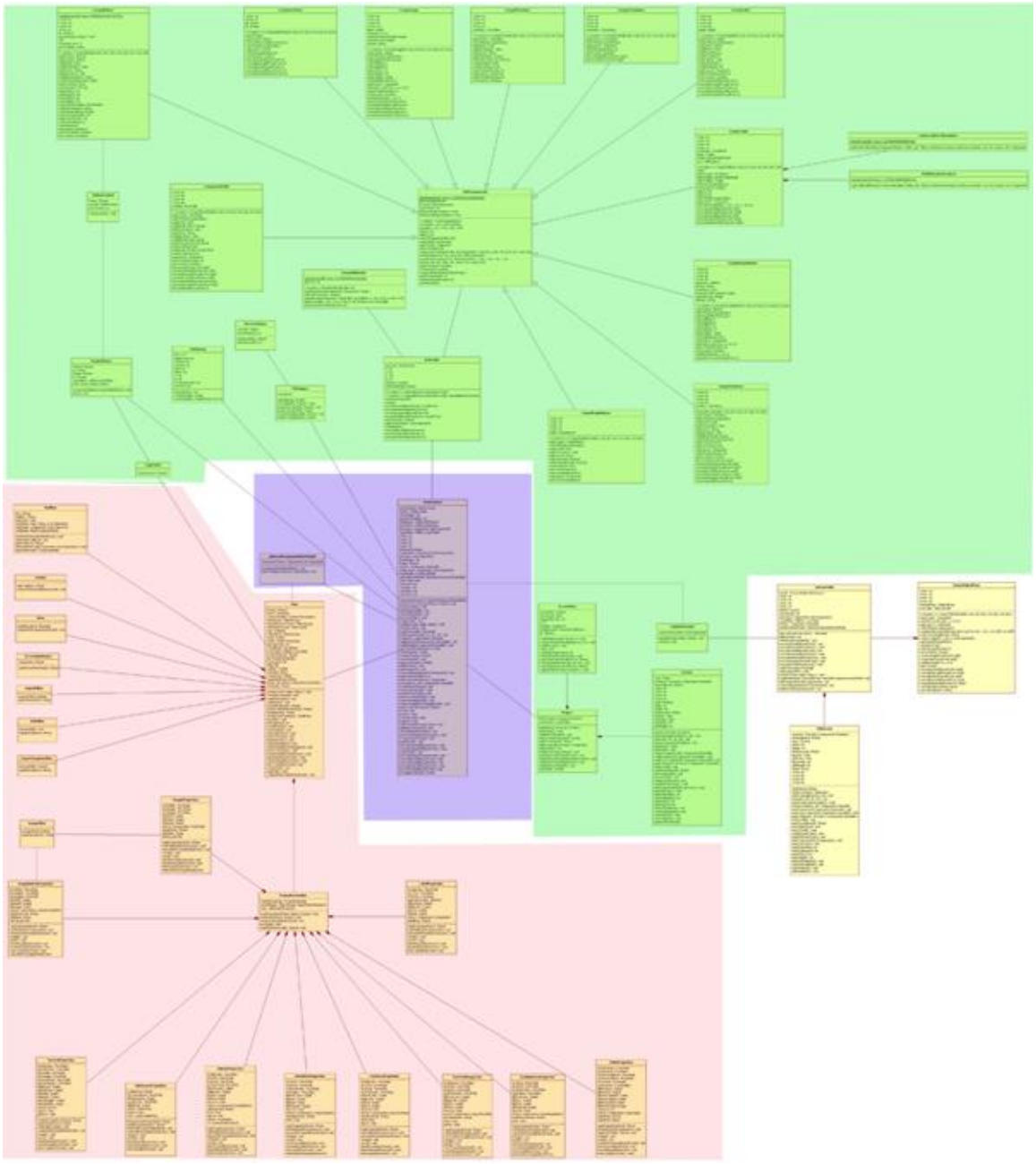
När vi har utvecklat prototypverktyget har vi försökt att använda oss av erkända designmönster där de har varit applicerbara.

- **MVC:** Vi har valt att följa ett av de mer grundläggande designmönstren nämligen MVC, Model View Control, för att få en bra struktur på vår kod. Att använda detta i vårt projekt har medfört att vi med enkelhet har kunnat ändra i gränssnittet över tiden, utan att behöva ändra i implementeringen av den underliggande tekniken. Vi har även försökt att hålla nere antalet beroenden inom de olika delarna för att det skall vara enkelt att byta ut en modul utan att övriga delar av systemet skall påverkas. Nedan visar vi vår struktur med ett klassdiagram, där vi har färgat de olika delarna i olika färger för att göra det lättöverskådligt. Där det saknas bakgrund, är det bara hjälpklasser.

  M=grön

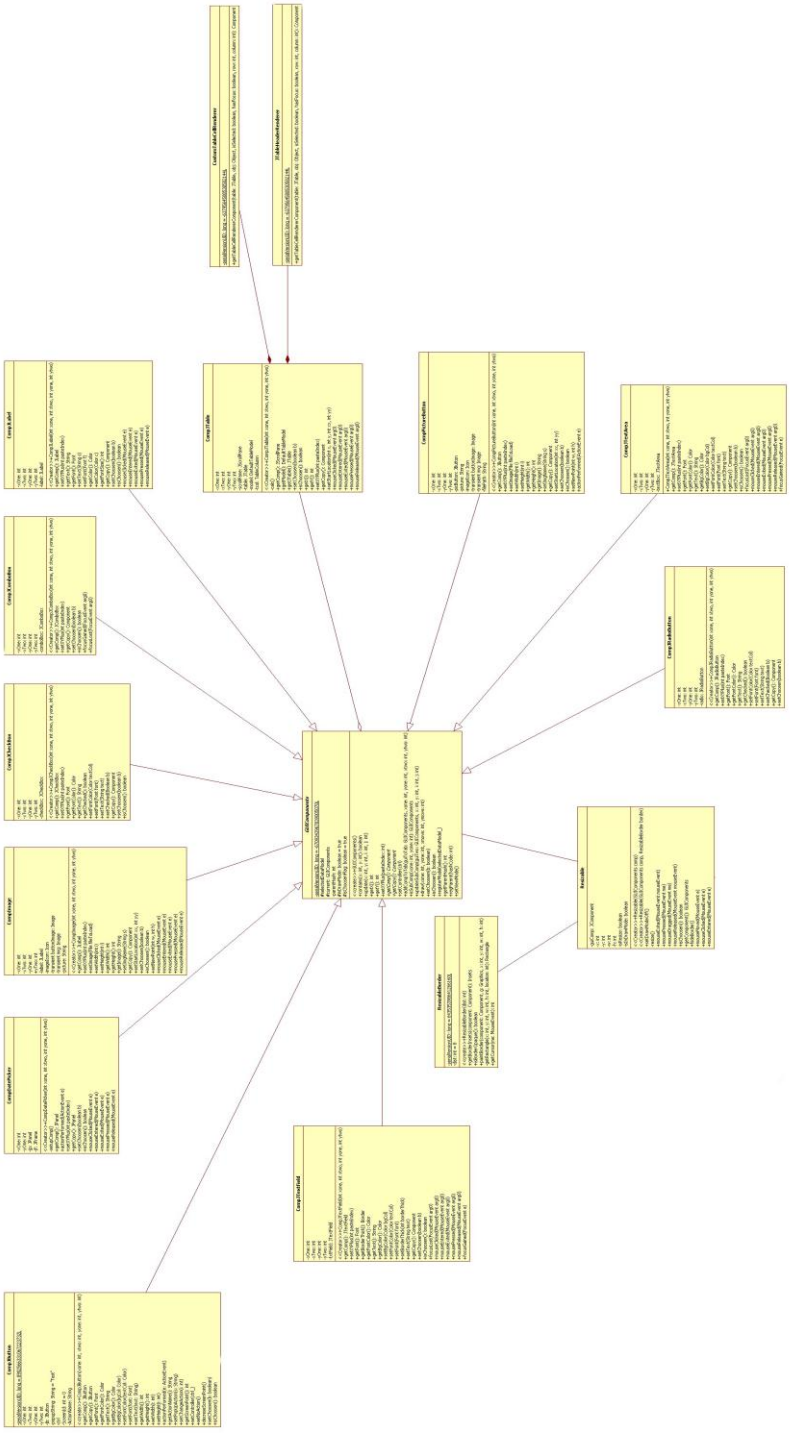
  V=rött

  C=blått



Figur 6 Klassdiagram som visar MVCstrukturen

- **Observer:** Ett mönster som använts flitigt i utvecklingsarbetet är Observer/Observable. Anledningen till detta är för att vi har velat separera beroenden mellan klasser så mycket som möjligt. Vi har använt mönstret för att bland annat tala om för vyn när en annan skärmbild i projektet blivit vald. Mönstret har även använts för att lösa ett problem som uppstod med komponenten JTabbedPane, som är en kontroll för att hantera tabbar i Java, då varje tabb innehåller en rityta och behöver få veta vilken komponent som är vald för att ritas ut. Fördelen blev att vi fick en smidig lösning som hanterar valfritt antal ritytor, men en stor nackdel finns då datamodellen för vilken komponent som är vald för ritning blir en ”Single Point Of Failure”, som med andra ord kan beskrivas att om datamodellen tas bort så upphör ritfunktionen att fungera. Metoden att använda en datamodell och låta klasser lyssna på den har vi även använt då vi har implementerat vilken komponent som blivit klickad på. Detta för att en kontroller skall kunna veta vilket objekt som blivit valt på vilken rityta i hierarkin, för att man skall kunna ta bort, kopiera eller klippa ut det. En nackdel som det dock innebär med att använda sig av många observers är att det blir ett stort dataflöde mellan objekten.
- **Composite:** Vi har en lista som innehåller kompontenter och eftersom vi har flera nivåer i ritytan så har composite-mönstret använts för att kunna hitta rätt komponent i hierarkin samt tilldela knappkontroller till knappar som finns i hierarkin. I figuren nedan ser vi hur vi använder oss utav Compositemönstret då vi har en superklass för alla komponenter som de sedan ärver. På så vis får vi polymorfa objekt vilket gör att vi kan lägga dem i en lista tillsammans.



Figur 7 Klassdiagram över komponenter. Visar hur Composite används

6.3.3. Designprinciper

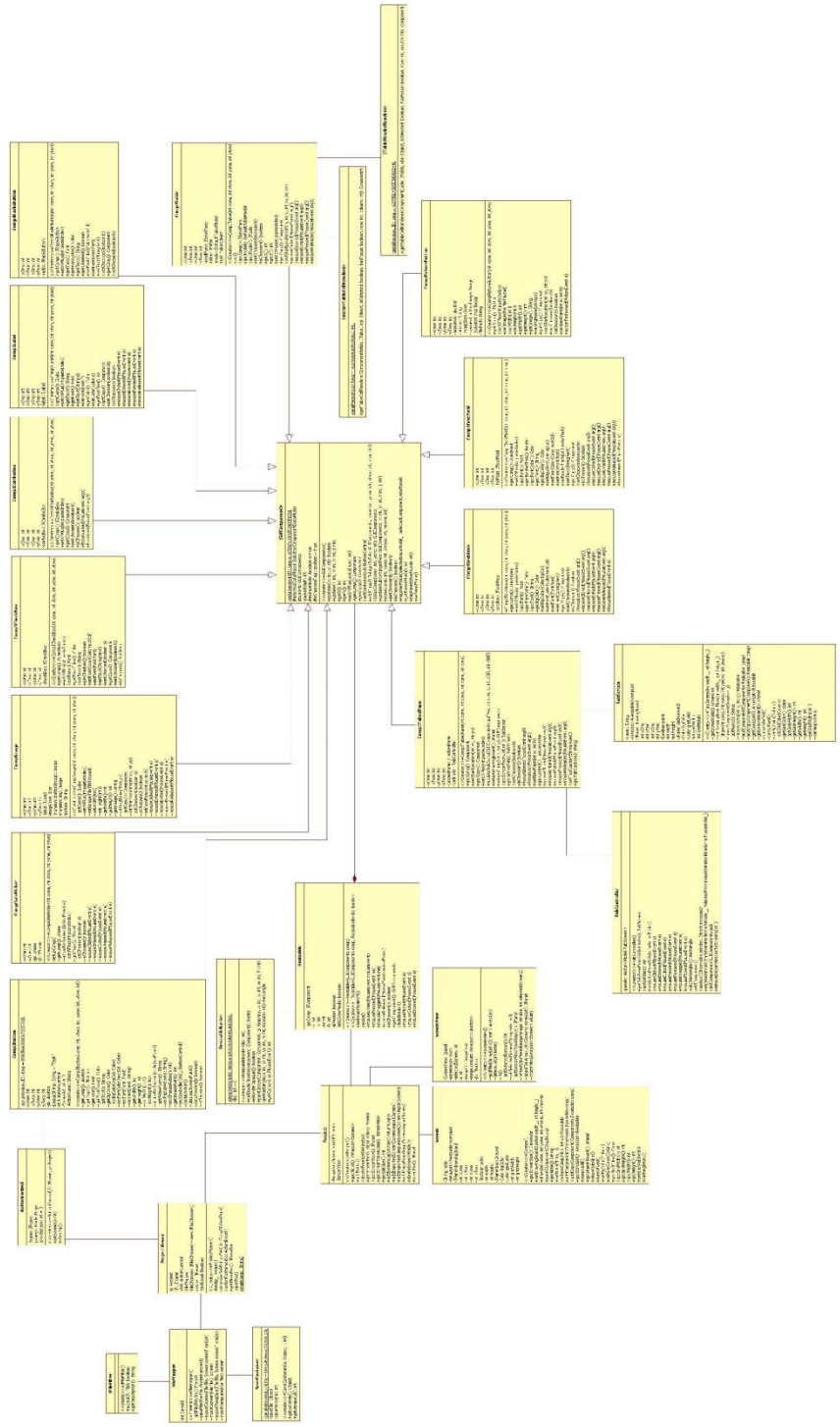
- **YAGNI - ”**You Aint Gonna Need It”, innebär att man inte skall koda saker som man tror sig behöva i framtiden, och inte behöver i nuet. Vi har försökt följa detta under utvecklingstiden för att ha så lite onödig kod som möjligt. Genom att använda denna princip så har vi kunnat reducera koden och fokusera på kraven i förstudien.
- **Refactoring:** Refactoring beytyder att man ändrar om i den interna strukturen utan att påverka programmets funktionalitet. Detta gör man för att bland annat ta bort kodredundans, förbättra koden och öka läsbarheten. Vi har följt det under hela projektet för att minimera kod och få en så läsbar kod som möjlig.
- **High Cohesion:** Att sträva efter hög cohesion innebär att man låter sina klasser ha en väldefinierad uppgift. Vi har delat upp koden så att klasserna inte har flera uppgifter, detta för att ha en så bra kodkvalité som möjlig vilket gör det lättare för framtida ändringar i olika klasserna[28].
- **Low Coupling:** Låg koppling betyder att man skall ha så få kopplingar mellan sina klasser för att de skall vara stabila och enkla att byta ut. Vi har försökt ha så få kopplingar mella klasserna som möjligt för att inte få oönskade förändringar i flera klasser när man ändrar i en klass. Detta minskar tidsåtgången om man ska ändra i en klass då man slipper ändra i flera klasser[29].
- **Förklarande metodnamn:** Vi har försökt hålla principen att metoderna ska vara självförklarande genom sin signatur för att minska dokumentationen i koden och minska tiden på dokumentering.

[28] Barnes David J, Kölling Michael, Pearson, Objects first with Java s 193
[29] Barnes David J, Kölling Michael, Pearson, Objects first with Java s 193

6.4. Prototype Viewer beskrivning

I detta avsnitt går vi igenom prototyp visarens struktur.

6.4.1. Klassdiagram



Figur 8: Klassdiagram PrototypeViewer

6.4.2. Designmönster och designprinciper

Vid arbetet med att ta fram prototypvisaren så har vi använt oss av samma mönster och designprinciper som under utvecklingen av prototypskaparen. För en beskrivning av designmönster se kap. 6.4.2 och för att se en beskrivning av designprinciperna se kap. 6.4.3.

6.4.3. Kodbeskrivning

PrototypeViewern bygger på samma kod som Prototype Maker. Detta har underlättat arbetet eftersom vi kunnat återanvända kod och endast behövt stoppa in en ny klass i programmet, som driver det hela.

6.5. Sammanfattning

Språket som använts för utvecklingen av programvarorna har varit Java. Detta då programmet har haft som krav att vara körbart utan en installation. Vi ansåg att den bästa lösningen för detta var att köra projektet som en exekverbar JAR-fil. Vi har tagit fram egna filformat för att passa vår verksamhet, då vi vill ha en proprietär[30] lösning och på så vis kunna styra vem som får använda produkten. Vi har under hela projektets gång strävat efter att hålla koden i god form och att använda oss av erkända designmönster när så har varit möjligt. Detta har varit ett fokus för oss eftersom vi sett en fortlevnad för programmet, och därmed velat göra det enkelt för framtida modifikationer av källkoden.

7. Resultat

Efter en något skakig inledning på projektet, då vi inte visste var vi skulle sitta och jobba, eller om vi fick fram hårdvara från företaget, så kom projektet på rätt köl. Därefter kom vårt problem med att Logica inte hade kunskapen om hur man använde en produkt backlog enligt Scrum. Vi anpassade oss efter deras sätt att hantera krav, genom att skapa en förstudie, och därefter så kunde arbetet med att ta fram mjukvaran börja. Vi nådde i mål med projektet och kunde avtäcka en färdig mjukvara några veckor innan deadline. En stor framgångsfaktor till detta är att vi haft täta kontakter med vår kund, samt med vår handledare på universitetet, så att projektet hela tiden

[30] http://sv.wikipedia.org/wiki/Propriet%C3%A4r_programvara

har kunnat korrigeras då ändringar gjorts i produktspecifikationen. Vi hade kunnat nå vårt resultat mycket fortare, men en plötslig kravändring mot slutet av projektet gjorde att vi fick bygga om stora delar av arkitekturen för att de nya kraven skulle passa in. Anledningen att vi gick med på kravändringen var för att mjukvaran skulle ha blivit utan värde för kunden utan den nya funktionalliteten som de kom på väldigt sent.

7.1.1. Prototype Maker

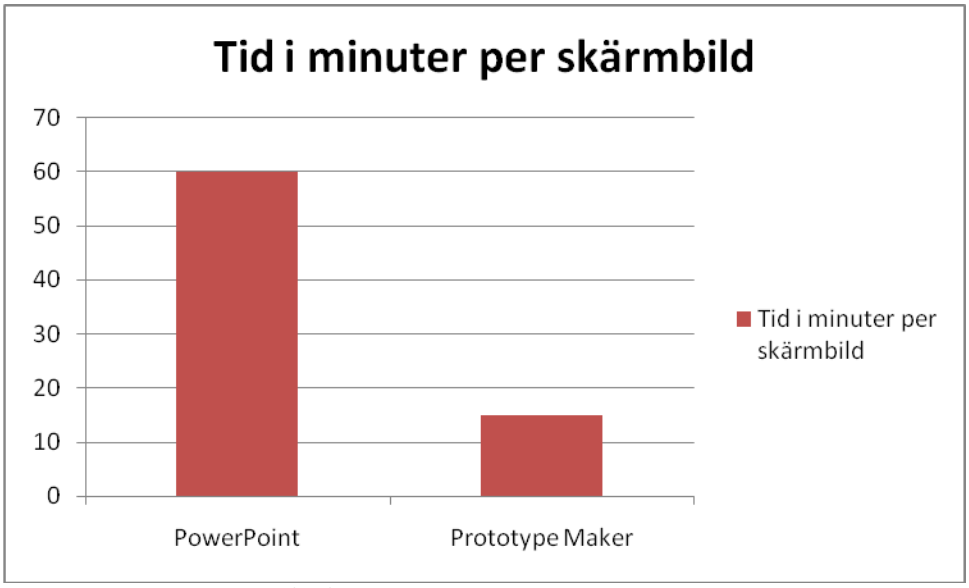
Då vi har färdig produkt som uppfyllt Logicas krav och inom utsatta tidsramar har vi nått i mål. Vi har valt att ge produkten namnet Prototype Maker eftersom det är ett beskrivande namn på mjukvaran. Ett problem vi haft under överlämningsfasen är att Logica lovat att testköra och utvärdera en beta vilket inte gjorts och därmed kan brister finnas i produkten som vi ej upptäckt och som kan ligga kvar när det kommer att användas på riktigt. Detta har vi påpekat och det är inget vi tar ansvar för.

7.1.2. Prototype Viewer

Detta är en bonus från vår sida då idén kom från oss. Detta är ett behov som vi uppfattade fanns som Logica inte tänkt på. Med den här mjukvaran kan Logica erbjuda mervärde till sina kunder då kunden kan ta med sig prototypen hem och utvärdera med t.ex. sin ledning. En begränsning som finns är att Logicas kund inte kan ändra prototypen, utan endast se. Detta för att viewern är tänkt för att spridas fritt medan Prototype Makern kostar pengar.

7.1.3. Tidsbesparningar

Då vi efter intervjuer fått reda på med deras arbetsmetod med PowerPoint är väldigt tidskrävande så var ett av målen att programmet skulle vara användarvänligt och på så vis göra tids- och kostnadbesparningar. Enligt uppgifter tog det 1 timme att göra en PowerPoint-bild vilket går att jämföra med 15 minuter för samma bild i vårt program. Detta innebär att effektiviteten ökar drastiskt då man kan skapa 4 bilder i vårt program på samma tid som de med PowerPoint kan skapa en. Detta ger en användare av vår mjukvara en effektivitetshöjning med 400% jämfört med en användare av Power-Point som arbetsverktyg, vilket vi gör överskådligt i figuren nedan.



Figur 9 Diagram över tidåtgång vid skapande av en skärmbild

7.2. Utvärdering

Nedan presenteras en teknisk utvärdering.

7.2.1. Val av teknik

Java valdes som programmeringsspråk eftersom vi uppfattat att Logica vill kunna köra programmet utan att installera. Vi anser att valet var bra och fungerat bra för det ändamålet vi haft. Javas snabbhet är en begränsande faktor då vi velat ha intensiva funktioner som t.ex. rita miniatyrer. Detta har resulterat i att vi behövt optimera programmet kontinuerligt för att komma till rätta med mindre lagg[31]. Vi har saknat stöd för kalenderfunktionalitet i swing samt funktionalitet för att skriva ut på ett smidigt sätt.

7.2.2. Kodkvalité

Vår intention har hela tiden varit att skapa en mjukvara med hög kodkvalité. Detta var en stor prioritet under hela projektet eftersom vi såg en framtid för mjukvaran. Problem uppstod dock när den sena kravändringen kom och vi fick bygga om en stor del av programmet. Kodkvalién kunde inte längre prioriteras utan allt fokus låg nu på att passa deadline. Koden blev ful och svårtydd, trots stora ansträngningar med att städa upp efter att deadline var mött. De delar som inte

[31] http://sv.wikipedia.org/wiki/Latens

påverkades av ombyggnationen är kvar i sitt gamla stabila tillstånd medans de nyskrivna bitarnas status är okej, men inte bättre. Koden kommer dock att saneras och optimeras framöver, fast det ligger utanför projektets ramar.

7.3. Sammanfattning

Sammanfattningsvis anser vi oss ha nått vårt mål och levererat en stabil mjukvara som möter kundens krav. Trots problem som vi inte kunnat råda över så har alla deadlines blivit mötta och vi har levererat det vi lovat. Från både vårt och Logicas håll anses projektet vara lyckat och i och med att vi sett vilket behov som finns på marknaden kommer produkten att leva vidare samt vidarutvecklas för kommersiella syften. Vi har med vår produkt erbjudit vår kund ett sätt att spara in tid och effektivisera sitt arbete. Vi levererar enkelhet där det tidigare varit omständliga arbetsmoment. Till detta har vi utvecklat en mjukvara som ger mervärde åt vår kunds kunder då de får ett helt nytt verktyg att arbeta med vid sidan av sin konsultkontakt. Teknikvalet har varit lyckat då vi med enkelhet har kunnat utveckla mjukvarorna, och att den varit tillräckligt kraftfull för att leverera det önskade resultatet.

I uppsatsen har vi använt böckerna ”Seminarieboken”[32] och ”A Rulebook for Arguments”[33] för hjälp med rapportbeskrivning samt böckerna ”*Designmönster för programmerare”*[34] *och* “*Object-oriented Modeling and Design with UML Second Edition”*[35] för allmänna programmeringsfrågor.

[32] Seminarieboken, Björklund Maria, Paulsson Ulf, Studentlitteratur
[33] *A Rulebook for Arguments, A Rulebook for Arguments*
[34] *Designmönster för programmerare*, Bilting Ulf, Studentlitteratur
[35] *Object-oriented Modeling and Design with UML Second Edition*, Blaha Michael, Rumbaugh James, Pearson

8. Källförteckning

8.1. Litteratur

1. Kniberg Henrik, InfoQ, "*Scrum and XP from the Trenches"* S18
2. Kruchten Philippe, Addison Wesley, "*The Rational Unified Process"*, Svenska Utgåvan S.17
3. Barnes David J, Kölling Michael, Pearson, "*Objects first with Java"* s 193

8.2. Webbsidor

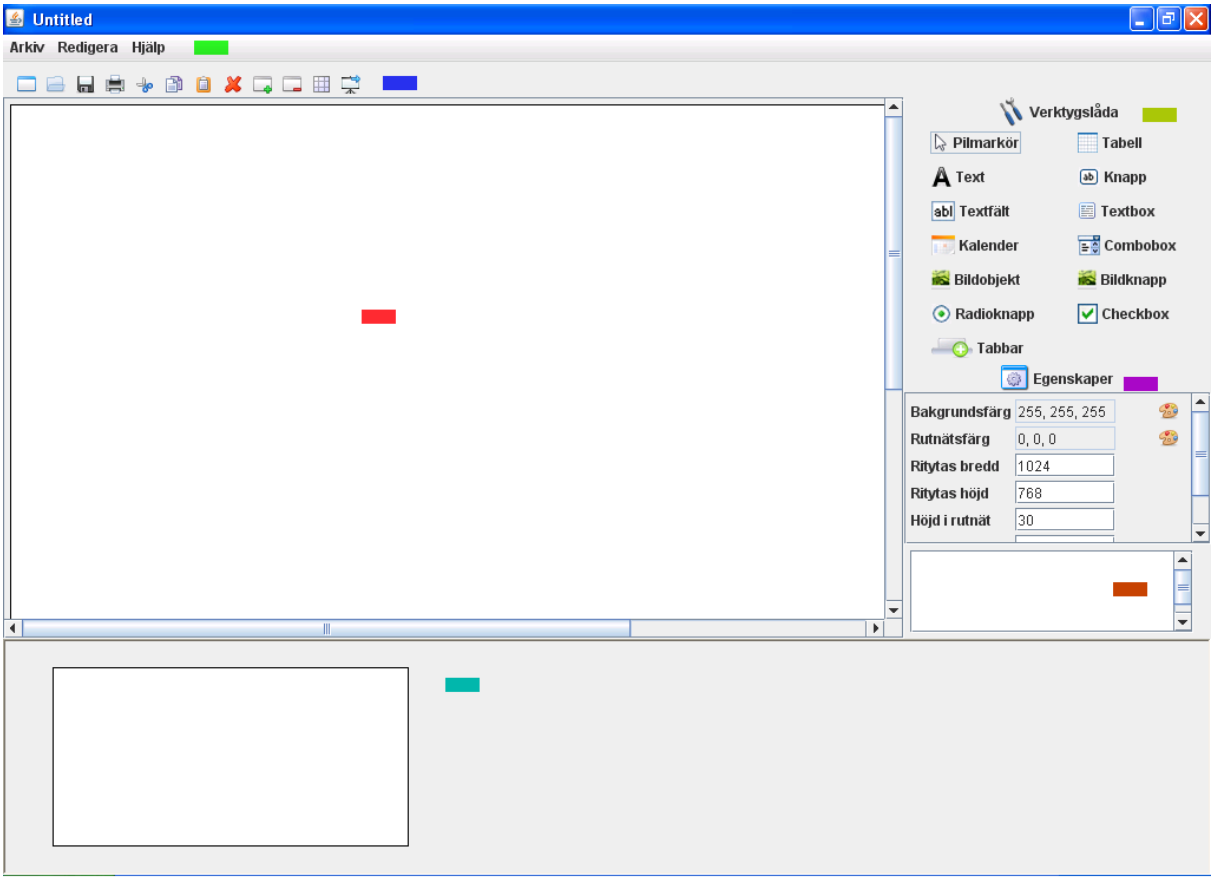
4. http://sv.wikipedia.org/wiki/Vattenfallsmodellen 2009-06-09
5. http://sv.wikipedia.org/wiki/WM-data 2009-06-09
6. http://www.logica.se/företaget/15002 2009-06-09
7. http://sv.wikipedia.org/wiki/Human_Resources 2009-06-09
8. http://www.logica.se/palasso/400007833 2009-06-09
9. http://www.palasso.com/ 2009-06-09
10. se.wikipedia.org/wiki/scrum 2009-06-09
11. http://www.adobe.com/se/products/acrobat/?promoid=BPCGM 2009-06-09
12. http://www.adobe.com/se/products/acrobat/reader.html 2009-06-09
13. http://www.adobe.com/se/products/acrobat/adobepdf.html 2009-06-09
14. www.axure.com 2009-06-09
15. www.balsamic.com 2009-06-09
16. www.ewolus.vn/pencil 2009-06-09
17. http://en.wikipedia.org/wiki/JAR_(file_format) 2009-06-09
18. http://en.wikipedia.org/wiki/List_of_Java_APIs 2009-06-09
19. http://staruml.sourceforge.net/en/ 2009-06-09
20. http://www.eclipse.org/ 2009-06-09
21. http://www.extremeprogramming.com/ 2009-06-09
22. http://sv.wikipedia.org/wiki/Windows_XP 2009-06-09
23. http://sv.wikipedia.org/wiki/Windows_Vista 2009-06-09
24. http://www.php.net/ 2009-06-09
25. http://sv.wikipedia.org/wiki/Linux 2009-06-09
26. http://sv.wikipedia.org/wiki/Java_Virtual_Machine 2009-06-09

27. http://sv.wikipedia.org/wiki/Application_Programming_Interface 2009-06-09
28. http://sv.wikipedia.org/wiki/Propriet%C3%A4r_programvara 2009-06-09
29. http://sv.wikipedia.org/wiki/Latens 2009-06-09

Appendix A

Överblick

Programöverblick



	Huvudmeny.
	Menyrad med genvägar till vanliga använda funktioner.
	Programmets rityta.
	Programmets verktygslåda.
	Egenskapsfält för de olika komponenterna och ritytorna.
	Anteckningsfält. Här kan man lägga en kommentar till en rityta.
	Miniatyrer. En minatyrbild av varje rityta i projektet visas.

Huvudmeny

	I ***"Arkiv"*** hittar vi bland annat filhanteringsfunktionerna. ***"Nytt Projekt"*** Klickar man här så skapas ett nytt projekt som även sätts till det aktiva projektet. ***"Öppna"*** Alternativet används för att öppna en sparad fil. ***"Spara projekt"*** Första gången som användaren trycker på det här alternativet så kommer en dialogruta upp som frågar var användaren vill spara sin fil. Efterföljande gånger sparas projektet till denna valda fil. ***"Spara som…"*** Alternativet används om man vill spara en fil för första gången, eller spara en befintlig fil under annat namn. ***"Skriv ut"*** Funktionen skriver ut miniatyrerna tillsammans med tillhörande anteckningar. ***"Exportera skärmbild"*** Används för att spara ned en rityta till lagringsmedium, för att kunna användas i andra projekt. ***"Importera skärmbild"*** Används för att läsa in en sparad rityta till det aktuella projektet. ***"Importera mall"*** En tidigare sparad mall importeras på den tomma ritytan. ***"Imp…Mall på mall"*** En mall importeras på en annan mall och lägger sig över denne. ***"Exportera mall"*** Ritytan sparas ned som en mall som sedan kan användas i andra projekt. ***"Avsluta"*** Stänger ned programmet.
	Under menyalternativet ***"Redigera"*** finner vi verktyg som har att göra med det aktuella projektet. ***"Ny skärmbild"*** Lägger till en ny skärmbild till projektet. Miniatyren av dess rityta visas i miniatyrfältet. ***"Radera skärmbild"*** Tar bort den aktuella ritytan från projektet. ***"Kopiera skärmbild"*** Kopierar den aktuella ritytan och placerar kopian längst bak i miniatyrlistan. ***"Kopiera"*** Används för att kopiera en ritad komponent. Komponenten markeras genom att klicka på den och därefter kan detta alternativet väljas. ***"Klistra in"*** Efter att en komponent kopierats eller klippts ut så kan detta användas för att klistra in kopian på ritytan.
	Under menyalternativet ***"Hjälp"*** finner vi programinformation. ***"Om"*** Visar en ruta som innehåller licensinformation samt upphovsrättsuppgifter.

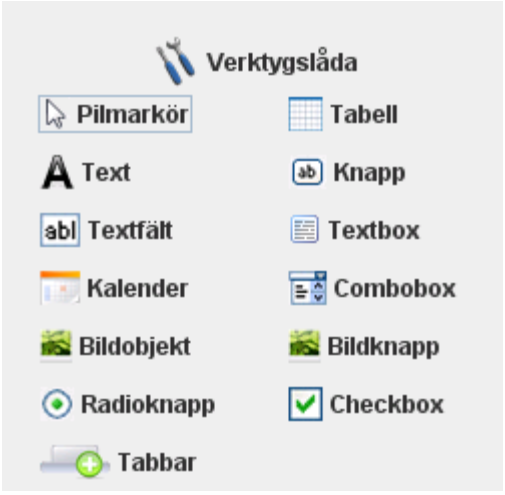
Menyraden

I menyraden hittar vi genvägar till vanliga använda funktioner. Nedan följer en förklaring till ikonerna, skrivna i kommande ordning.

1. Nytt projekt – Skapar ett nytt projekt och sätter det till det aktiva projektet.
2. Öppna projekt – Används för att öppna ett projekt som finns sparat på fil.
3. Spara – Sparar projektet till fil.
4. Skriv ut – Skriver ut miniatyrerna tillsammans med ritytornas anteckningar.
5. Klipp ut – Klipper ut en markerad komponent från ritytan.
6. Kopiera – Kopierar en markerad komponent från ritytan.
7. Klistra in – Klistrar in en kopierad eller utklippt komponent på den aktuella ritytan.
8. Ta bort – Tar bort en markerad komponent från ritytan.
9. Ny skärmbild – Lägger till en ny rityta till projektet.
10. Ta bort skärmbild – Tar bort den aktuella skärmbilden från projektet.
11. Rutnät – Vid klick på rutnät så läggs ett rutnät på ritytan för att man lätt skall kunna rita med valda avstånd.
12. Kör projekt – Används för att köra projektet, med all funktionalitet såsom ex. knapptryckningar.

Verktygsfält

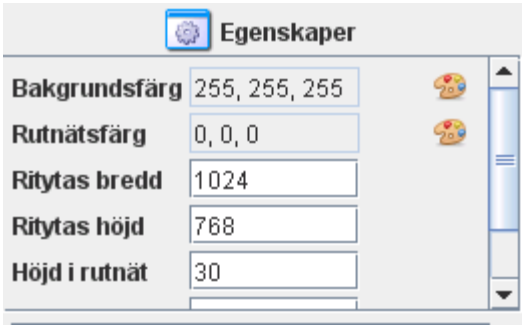
I verktygsfältet återfinns de komponenter som kan ritas ut. Nedan följer en förteckning.

	***Pilmarkör*** – Nollställer ritobjektet vilket innebär att ingen komponent ritas vid drag på ritytan. ***Tabell*** – Ritar ut en tabell. ***Text*** – Ritar en label som kan innehålla text. ***Knapp*** – Ritar en knapp som man sedan kan lägga funktionalitet till, såsom att byta skärmbild. ***Textfält*** – Ett textfält med en rad. ***Textbox*** – En textbox som kan innehålla flera rader text. ***Kalender*** – Innehåller en kalender som visar dagens datum. ***Combobox*** – Är en dropdownlista som kan visa olika värden. ***Bildobjekt*** – Infogar en bild på ritytan. Bild väljs från valfritt lagringsmedium. ***Bildknapp*** – En knapp som använder en bild för att visas. ***Radioknapp*** – En radioknapp kan ritas ut.

	Checkbox – En checkbox kan ritas ut. ***Tabbar*** – En tabbpanel ritas ut på ritytan. Tabbpanelen i sig kan sedan innehålla komponenter av de slag som återfinns i verktygslådan.

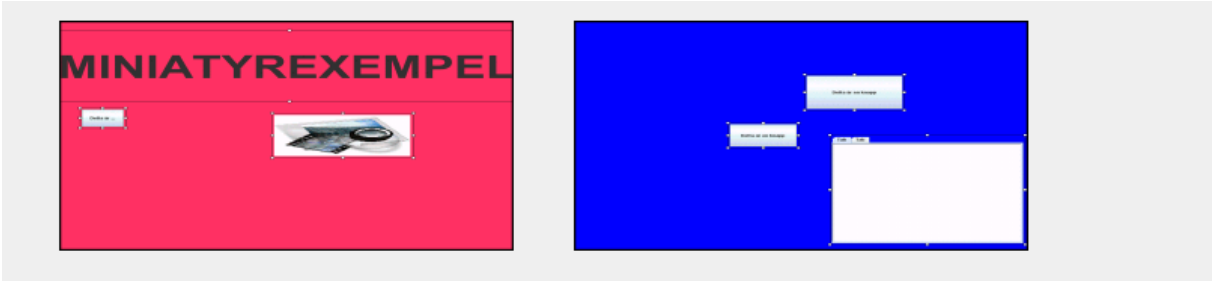
Egenskaper

I egenskapsfältet kan egenskaper för de olika ritytorna och komponenterna ändras. Egenskapsfältet ser olika ut beroende på vad det är för komponent eller rityta som som är vald. Egenskapsfältet uppdateras när en knapp eller rityta blir vald genom att den klickas på.
Nedan är en bild av hur det ser ut när en rityta är vald.



Miniatyrer

Miniatyrerna visar i liten skala hur de olika skrämbilderna som ingår i projektet ser ut. Detta för att användaren enkelt skall få en överblick över sitt projekt. Genom att klicka på miniatyrerna så byter man aktuell rityta. Nedan är ett exempel på hur det kan se ut.

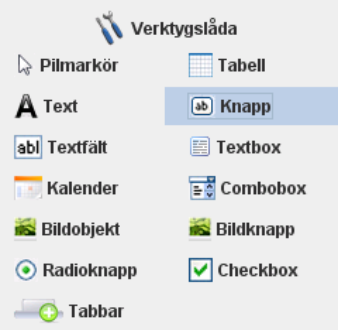


Rita

Rita en komponent

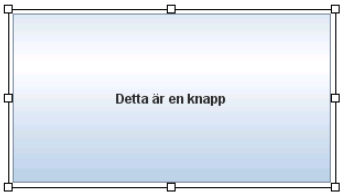
För att rita en komponent på en rityta så börja med att välja en komponent i verktygslådan. Därefter trycks vänster musknapp ned någonstans på ritytan och genom att dra musen åt valfritt håll så bestäms storleken på komponenten. När musknappen sedan släpps upp så ritas komponenten ut på ritytan.

1



2

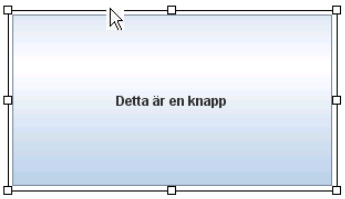
3



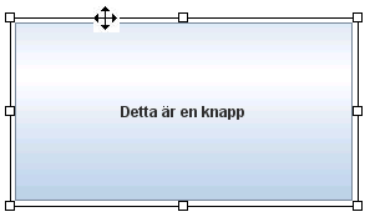
Flytta en komponent

För att flytta en komponent, börja med att markera den. Därefter trycks musknappen ned när markören är över komponentens ram. Därefter dras musen dit komponenten skall placeras. När musknappen släpps upp så uppdateras komponentens position.

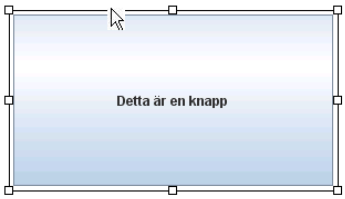
1



2



3



Kopiera, klipp ut, klistra in

För att kopiera en komponent, markera denna och tryck på kopieraknappen.
För att klistra in kopian på valfri rityta, tryck på knappen för klistra in. Komponenten klistras in till den ritytan som är aktiv vid inklistringstillfället.
För att klippa ut en komponent, markera denna och tryck på knappen för klipp ut. För att klistra in utklippt objekt, gör på samma sätt som vid kopiering.

Kopiera:

Klipp ut:

Klistra in:

Appendix B

Kodbeskrivning

Paket View

Klassnamn: Main

Klassbeskrivning: I klassen Main är där programmet startas, då programmets main-metod återfinns här. Klassen genererar det grafiska gränssnittet och presenterar det för användaren. Klassen hanterar de dialogrutor som efterfrågas, t.ex. när man skall öppna en fil eller spara ett projekt.
Implementerar: Serializable, Observer, KeyListener, ActionListener

Metodsignatur: public Main()
Beskrivning: Klassens konstruktor. Det är här som hela prototypverktyget startas igång och det grafiska gränssnittet genereras.

Metodsignatur: public void update(Observable o, Object arg)
Beskrivning: Metoden tvångsimplementeras på grund av att klassen implementerar Observermönstret. Dess uppgift är att ta hand om uppdateringar i datamodellen som observeras och utföra rätt handling baserat på vad det är som har uppdaterats.
Parametrar: o som är av typen Observable och är det objekt som observeras. Arg, som är av typen Object. Arg innehåller meddelandet som skickas från den observerade klassen. När meddelandet tas emot så castas det rätt så att det blir av rätt typ.

Metodsignatur: public void changeScreen(int i)
Beskrivning: Metoden används vid byte av skärmbild så att den valda skärmbilden och dess anteckningar laddas in och den föregående skärmbilden inaktiveras. När metoden anropas stängs muslyssnarna och aktiverar muslyssnare för den nya skärmbilden som är aktiv.
Parametrar: Variabeln i som är av typen integer håller koll på vilken skärmbild som ska aktiveras.
Metodsignatur: public void updateScreen()

Beskrivning: Uppdaterar aktiv skärmbild och miniatyrer

Metodsignatur: private void regListner()
Beskrivning: Registrerar observerare

Metodsignatur: private JPanel setupMiniatures()
Beskrivning: Initierar miniatyrerna som representerar skärmbilderna.

Metodsignatur: private JPanel setupToolsAndProperties()
Beskrivning: Initierar verktygslådan och egenskapernas yta.

Metodsignatur: private JPanel setupNotes()
Beskrivning: Initierar anteckningar i en scrollpane för att kunna scrolla ner ifall man har mer text än vad som kan visas.

Metodsignatur: private JScrollPane setupCenterContent()
Beskrivning: Ritar upp skärmbilden och laddar in egenskapsfälten så att man bl.a. kan kunna ändra storlek på skärmen, ändra bakgrundsfärg, rutnätsfärg.

Metodsignatur: private void saveAs()
Beskrivning: Sätter en int variabel till ett värde så att man får upp en sparafil dialogruta när man trycker på ”Spara som…” knapp.

Metodsignatur: public int changeVar()
Beskrivning: Returnerar en int till en annan klass som tar emot det.

Metodsignatur: public void saveProject()
Beskrivning: Sparar projekt vid behov och beroende på om man trycker på ”Spara projekt” eller ”Spara som” så sker sparningen på olika sätt. Om man sparar för första gången och tar ”Spara projekt” så får man fram en dialogruta för att spara. Nästa gång man trycker på samma knapp så kommer det inte upp en dialogruta men tar man ”Spara som” så kommer det alltid upp en dialogruta för att spara fil till disk.

Metodsignatur: private void newProject()

Beskrivning: När man tar nytt projekt tas allt gammalt bort som finns kvar ifall man har ett projekt öppet för att sedan registrera nya lyssnare och initierar ett nytt projekt som är klar för användning.

Metodsignatur: private void removeScreen()
Beskrivning: Tar bort aktiv skärmbild och tillhörande miniatyr så länge antalet skärmbilder i projektet är större än 1.

Metodsignatur: private void exportScreen()
Beskrivning: Sparar aktiv skärmbild till disk i formatet S4P för att kunna importera sparad skärmbild till andra projekt. Finns det redan en skärmbild med samma namn så sparas den över om man väljer ”Ja” i dialogrutan som dyker upp.

Metodsignatur: private void importScreen()
Beskrivning: Importerar vald skärmbild till aktuellt projekt som läggs sist i listan.

Metodsignatur: private void importTemplate()
Beskrivning: Importerar vald mall från disk.

Metodsignatur: private void importTemplateOnTemplate()
Beskrivning: Importerar mall på mall. Kan användas så länge det finns en mall öppen. Det finns ingen begränsning på hur många mallar som läggs på redan öppna mallar.

Metodsignatur: private void exportTemplate()
Beskrivning: Sparar aktiv skärmbild till disk i formatet M4P för att kunna importera sparad mall till andra projekt. Finns det redan en mall med samma namn så sparas den över om man väljer ”Ja” i dialogrutan som dyker upp.

Metodsignatur: public void eventHandler(ActionEvent e)
Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas t.ex. ”Nytt projekt” så väljs den metoden efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: public static void main(String []args)
Beskrivning: Kör igång klassen.
Parametrar: args används inte.

Metodsignatur: public void keyPressed(KeyEvent e)
Beskrivning: Metoden underlättar kopiering, klistring och borttagning av markerad komponent med hjälp av kortkommandon. Är t.ex. ctrl + c tryckta så kopieras markerad komponent.
Parametrar: e av typen KeyEvent som har en knappkod som jämförs med intryckta knappar för att kunna avgöra vilken metod som ska utföras.

Metodsignatur: private void properties()
Beskrivning: Visar markerad komponents egenskapsfält för att göra ändringar på komponent t.ex. om det är en label som är markerad så kan man ändra storlek, skriva text, byta textfärg m.m.

Metodsignatur: private void openFile()
Beskrivning: Öppnar valt projekt som finns sparat i disk.

Klassnamn: ButtonProperties

Metodsignatur: public ButtonProperties(ViewControl viewController)
Beskrivning: Konstruktor som initierar knappegenskaperna med alla tillhörande fälten.
Parametrar: viewControllern används för att komma åt klassen ViewControl eftersom man kan få tag i antal skärmbilder i projektet.

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel.

Metodsignatur: public void setFields(GUIComponents jb)

Beskrivning: Initierar knappens egenskaper.

Parametrar: jb castas till CompJButton som sedan används för att t.ex. sätta namn på knapp.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas t.ex. ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()

Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på knappen.

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: CheckBoxProperties

Metodsignatur: public CheckBoxProperties()

Beskrivning: Konstruktor som initierar checkboxegenskaperna med alla tillhörande fälten.

Metodsignatur: public JPanel getPropertiesPane()

Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents jchb)

Beskrivning: Initierar checkboxens egenskaper.

Parametrar: castas till CompJCheckBox som sedan används för att t.ex. sätta namn på checkboxen..

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas t.ex. ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()

Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på checkboxen.

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: ExportFilter

Metodsignatur: public boolean accept(File f)

Beskrivning: Filter som kontrollerar att filen är i s4p format.

Parametrar: f av typen File som kollas om det är av rätt filformat.

Metodsignatur: public String getDescription()

Beskrivning: Returnerar en sträng.

Klassnamn: ExportTemplateFilter

Metodsignatur: public boolean accept(File f)
Beskrivning: Filter som kontrollerar om filen är i m4p format.
Parametrar: f av typen File som kollas om det är av rätt filformat.

Metodsignatur: public String getDescription()
Beskrivning: Returnerar en sträng.

Klassnamn: ImageButtonProperties

Metodsignatur: public ImageButtonProperties()
Beskrivning: Konstruktor som initierar bildknappen med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents img)
Beskrivning: Initierar bildknappens egenskaper.
Parametrar: castas till CompPictureButton som sedan används för att ange bl.a. bredd och höjd.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas ladda in bild så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.
Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()
Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()
Beskrivning: Sparar alla ändringar som gjorts på bildknappen

Metodsignatur: public void keyPressed(KeyEvent e)
Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.
Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Metodsignatur: public void itemStateChanged(ItemEvent ie)
Beskrivning: När bildknappen markeras så sparas ändringarna om det gjorts ändringar på bildknappen som inte redan sparats
Parametrar: används inte

Klassnamn: ImageProperties

Metodsignatur: public ImageProperties()
Beskrivning: Konstruktor som initierar bild med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents img)
Beskrivning: Initierar bildens egenskaper.
Parametrar: castas till CompImage som sedan används för att ange bl.a. bredd och höjd.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas ladda in bild så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.
Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()

Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på bildknappen

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Metodsignatur: public void itemStateChanged(ItemEvent ie)

Beskrivning: När bildknappen markeras så sparas ändringarna om det gjorts ändringar på bildknappen som inte redan sparats

Parametrar: används inte

Klassnamn: JFileFilter

Metodsignatur: public boolean accept(File f)

Beskrivning: Filter som kontrollerar om filen är i p4p format.

Parametrar: f av typen File som kollas om det är av rätt filformat.

Metodsignatur: public String getDescription()

Beskrivning: Returnerar en sträng.

Klassnamn: LabelProperties

Metodsignatur: public LabelProperties()

Beskrivning: Konstruktor som initierar label med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents label)
Beskrivning: Initierar labelns egenskaper.
Parametrar: castas till CompJLabel som sedan används för att ange bl.a. text och textstorlek.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Tas t.ex. ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.
Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()
Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()
Beskrivning: Sparar alla ändringar som gjorts på labeln.

Metodsignatur: public void keyPressed(KeyEvent e)
Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.
Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: Menu

Metodsignatur: public JMenuBar getMenuBar()
Beskrivning: Initierar menyfältet med huvudmeny som ”Arkiv” och submenyer.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: meddelar observerararna att ändring skett.
Parametrar: e av typen ActionEvent castas när notifyObservers metoden anropas för att meddela observerarna om att ändring skett.
Klassnamn: PropertiesHandler

Metodsignatur: public PropertiesHandler(ViewController viewController)
Beskrivning: Initierar och registrerar skärmegenskaper.
Parametrar: viewControllern skickas vidare till mapperSetupfunktion.

Metodsignatur: public void setProperties(JPanel propertiesPane, Object gui,Screen drawArea)
Beskrivning: Laddar in rätt egenskapsfält för markerad komponent.
Parametrar: propertiesPane är panelen som egenskapsfälten är i. gui är komponenten som är markerad. drawArea är skärmen som är aktiv där komponenten är markerad.

Metodsignatur: public void setStart(JPanel propertiesPane, Screen drawArea)
Beskrivning: Laddar in skärmens egenskapsfält.
Parametrar: propertiesPan laddar in rätt egenskapsfält och drawArea laddar in rätt skärmbild.

Metodsignatur: private void mapperSetup(ViewControl viewController)
Beskrivning: Registrerar alla komponentegenskaper som lyssnare samt lägger till alla komponentegenskaper i en map.
Parametrar: viewControllern används för att komma åt klassen ViewControl eftersom man kan få tag i antal skärmbilder i projektet.

Metodsignatur: private void change()
Beskrivning: Meddelar observerarna att ändring skett.

Metodsignatur: public void update(Observable o, Object arg)
Beskrivning: Metoden tvångsimplementeras på grund av att klassen implementerar Observermönstret. Dess uppgift är att ta hand om uppdateringar i datamodellen som observeras och utföra rätt handling baserat på vad det är som har uppdaterats.

Parametrar: o som är av typen Observable och är det objekt som observeras. Arg, som är av typen Object. Arg innehåller meddelandet som skickas från den observerade klassen. När meddelandet tas emot så castas det rätt så att det blir av rätt typ.

Klassnamn: RadioButtonProperties

Metodsignatur: public RadioButtonProperties()
Beskrivning: Konstruktor som initierar radiobutton med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents img)
Beskrivning: Initierar radiobuttons egenskaper.
Parametrar: castas till CompJRadioButton som sedan används för att ange bl.a. text, textfärg.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Väljs t.ex. ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.
Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()
Beskrivning: Meddelar observeraren att ändring skett.

Metodsignatur: private void save()
Beskrivning: Sparar alla ändringar som gjorts på radiobutton

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Metodsignatur: public void itemStateChanged(ItemEvent ie)

Beskrivning: När radiobutton markeras så sparas ändringarna om det gjorts ändringar på radiobutton som inte redan sparats

Parametrar: används inte

Klassnamn: ScreenProperties

Metodsignatur: public ScreenProperties()

Beskrivning: Konstruktor som initierar skärm med alla tillhörande fälten

Metodsignatur: public JPanel setUpScreenProperties()

Beskrivning: Returnerar en panel.

Metodsignatur: public JPanel getScreenProp()

Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(Screen screen)

Beskrivning: Initierar skärmens egenskaper.

Parametrar: Används för att ange bl.a skärmens. bredd och höjd.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Väljs ändra höjd så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()
Beskrivning: Meddelar observerararna att ändring skett.

Metodsignatur: private void save()
Beskrivning: Sparar alla ändringar som gjorts på skärmen

Metodsignatur: public void keyPressed(KeyEvent e)
Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.
Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: TableProperties

Metodsignatur: public TableProperties()
Beskrivning: Konstruktor som initierar tabell med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents img)
Beskrivning: Initierar tabellens egenskaper.
Parametrar: castas till CompJTable som sedan används för att bl.a. .

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Händelsehanterare som anropar save metoden efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används vid val av save metoden genom att kolla stränginnehållet.

Metodsignatur: public void addCol(JTable table, Object headerLabel)

Beskrivning: Lägger till en kolumn i markerad tabell.

Parametrar: table är av typen tabell som är tabellen som kommer att modifieras. headerLabel är namnet som sätts på kolumnen.

Metodsignatur: private void change()

Beskrivning: Meddelar observerarna att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på tabellen.

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: TabScreenProperties

Metodsignatur: public TabScreenProperties()

Beskrivning: Konstruktor som initierar tab med alla tillhörande fälten

Metodsignatur: public JPanel setUpScreenProperties()

Beskrivning: Returnerar en panel.

Metodsignatur: public JPanel getPropertiesPane()

Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(CompJTabedPane cjt)

Beskrivning: Initierar tabens egenskaper.

Parametrar: Används för att ange bl.a en tabs text.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Väljs ändra tabtext så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()

Beskrivning: Meddelar observerararna att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på taben.

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: TextAreaProperties

Metodsignatur: public TextAreaProperties()

Beskrivning: Konstruktor som initierar textarea med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()

Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents jtxa)

Beskrivning: Initierar textareans egenskaper.

Parametrar: Används för att ange bl.a text.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Väljs ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.
Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()
Beskrivning: Meddelar observerararna att ändring skett.

Metodsignatur: private void save()
Beskrivning: Sparar alla ändringar som gjorts på textarean.

Metodsignatur: public void keyPressed(KeyEvent e)
Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.
Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: TextFieldProperties

Metodsignatur: public TextAreaProperties()
Beskrivning: Konstruktor som initierar textarea med alla tillhörande fälten

Metodsignatur: public JPanel getPropertiesPane()
Beskrivning: Returnerar en panel

Metodsignatur: public void setFields(GUIComponents jtxt)
Beskrivning: Initierar textfältens egenskaper.
Parametrar: Används för att ange bl.a text.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Händelsehanterare som väljer rätt metod beroende på användarens val. Väljs ändra textfärg så utförs ändringen efter att ha gjort en koll så att det stämmer med valet.

Parametrar: e av typen ActionEvent används när val av rätt metod görs genom att kolla stränginnehållet.

Metodsignatur: private void change()

Beskrivning: Meddelar observerararna att ändring skett.

Metodsignatur: private void save()

Beskrivning: Sparar alla ändringar som gjorts på textfältet.

Metodsignatur: public void keyPressed(KeyEvent e)

Beskrivning: Metoden underlättar sparning av ändring genom att trycka på enterknappen efter ändring av markerad komponent.

Parametrar: e av typen KeyEvent som har en knappnyckelkod som jämförs med enterknappen för att kunna välja rätt operation.

Klassnamn: ToolBar

Metodsignatur: public JPanel getToolBar()

Beskrivning: Initierar verktygfältet med snabbknappar så som ”Nytt projekt”, ”Spara som”, alla med respektive bild.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: meddelar observerararna att ändring skett.

Parametrar: e av typen ActionEvent, castas när notifyObservers metoden anropas för att meddela observerarna om att det tryckts på en av snabbknapparna.

Klassnamn: ToolBox

Metodsignatur: public ToolBox()
Beskrivning: Klassens konstruktor. Det grafiska gränssnittet för verktygslådan genereras.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Hämtar ut rätt komponent ut mapen beroende på användarens val.
Parametrar: e av typen ActionEvent används när val av rätt komponent görs genom att kolla stränginnehållet.

Metodsignatur: privaite void generateToolBox()
Beskrivning: Initierarr och registrerar lyssnare till komponenterna.

Metodsignatur: public JPanel getToolBox()
Beskrivning: Returnerar en panel.

Metodsignatur: public void setSelectedComp(Components.GUIComponents gui)
Beskrivning: meddelar observerararna att ändring skett.
Parametrar: gui är komponenten som meddelar om att det är den komponenten som ska användas.

Metodsignatur: public ToolsDataModel getDataModel()
Beskrivning: Returnerar datamodellen.

Paket Utilities

Klassnamn: CopyMaker

Metodsignatur: public Object returnCopy(Object o)
Beskrivning: Skriver över objekt till en byte array och gör arryen till en input stream för att göra kopia av den till objektet.
Parametrar: Objektet som skickats in för att kopieras.

Klassnamn: FileMapper

Metodsignatur: public Project getFile(File file)
Beskrivning: Hämtar vald projektfil från disk för att öppnas.
Parametrar: filen som innehåller information om valda filen.

Metodsignatur: public void saveFile(File file, Project project)
Beskrivning: Sparar aktivt projekt till disk.
Parametrar: file, innehåller information om filen som ska sparas. project är hela projektet med alla ritytor, komponenter, anteckningar och miniatyrer som kommer att sparas.

Metodsignatur: public void exportScreen(File file,Screen screenFromList)
Beskrivning: Sparar undan aktiv rityta för att ha möjlighet att använda i andra projekt.
Parametrar: file, innehåller information om filen som ska sparas. screenFromList, aktiva ritytan som kommer att sparas.

Metodsignatur: public Screen importScreen(File file)
Beskrivning: Importerar vald skärmbild från disk till aktuellt projekt.
Parametrar: file, innehåller information om filen som ska öppnas.

Metodsignatur: public void exportTemplate(File file, Screen screenFromList)
Beskrivning: Sparar skärmbild som en mall för att kunna använda i andra projekt.
Parametrar: file, innehåller information om filen som ska sparas. screenFromList, aktiva skärmbilden som kommer att sparas som mall.

Metodsignatur: public Screen importTemplate(File file)
Beskrivning: Importerar en skärmbild från disk till aktuellt projekt.
Parametrar: file, innehåller information om filen som ska öppnas.

Klassnamn: ImageFilter

Metodsignatur: public boolean accept(File f)
Beskrivning: Filter som kontrollerar att filen är i .png eller .bmp format.
Parametrar: f av typen File som kollas om det är av rätt filformat.

Metodsignatur: public String getDescription()
Beskrivning: Returnerar en sträng.

Klassnamn: PrintSetup

Metodsignatur: public PrintSetup()
Beskrivning: Klassens konstruktor, initierar variabler.

Metodsignatur: public void Setup(Main m)
Beskrivning: Ger användaren möjlighet att ställa in sin utskrift. T.ex. marginalavstånd.
Parametrar: m, används för att komma åt anteckningar.

Metodsignatur: private Image createImage()
Beskrivning: Tar en skärmdump på aktiv skärmbild för att göra en miniatyr så att det får plats på pappret.

Metodsignatur: public int print(Graphics g, PageFormat pageFormat, int index)
Beskrivning: Skriver ut skärmbilderna som finns i projektet. Skriver ut 3 bilder plus dess anteckningar/sida.
Parametrar: g, objekt som ritar upp komponent efter satta inställningar. pageFormat, har hand om storleken och orienteringen av sidan som ska skriva ut. index som startar på noll som används för att kolla om det finns fler sidor att skriva ut.

Klassnamn: SaveContainer

Klassbeskrivning: Container, som håller koll på sparade projektversioner.
Implementerar: Serializable

Paket Model

Klassnamn: Project

Metodsignatur: public Project()
Beskrivning: Initierar lista med skärmbilder.

Metodsignatur: public void addToList()
Beskrivning: Lägger till en skärmbild och en miniatyr av skärmbilden i projektet.

Metodsignatur: public void deleteFromList(int index)
Beskrivning: Tar bort aktiv skärmbild från projektet.
Parametrar: index, specifierar vilken bils om ska tas bort.

Metodsignatur: public void addScreenCopy(Object returnCopy)
Beskrivning: Lägger till en kopierad skärmbild i projektet.
Parametrar: returnCopy, castas till Screen som läggs till i listan med skärmbilder.

Metodsignatur: public void addImportedScreen(Screen importedScreen)
Beskrivning: Lägger till sparad skärmbild in i projektet.
Parametrar: importedScreen, bilden som ska importeras.

Metodsignatur: public void addImportedTemplateScreen(Screen importScreen)
Beskrivning: Lägger till mall från disk till projekt.
Parametrar: importScreen, skärmbilden som kommer att läggas till i projektet.

Klassnamn: ProjectViewer

Klassbeskrivning: Kör igång projektvisaren i en ny tråd.
Ärver: Thread

Klassnamn: Screen

Metodsignatur: public Screen
Beskrivning: Klassens konstruktor, initierar skärmbild.

Metodsignatur: public void addComponent(Components.Resizable comp)
Beskrivning: Lägger till komponent i listan med komponenter
Parametrar: comp, är en komponent som man kan ändra storlek på.

Metodsignatur: public void showHideGrid()
Beskrivning: Aktiverar/avaktiverar rutnät

Metodsignatur: public void delComponent(GUIComponents gui)
Beskrivning: Tar bort markerad komponent från aktiv skärmbild.
Parametrar: gui, den grafiska komponenten som ska tas bort från skärmbildens lista.

Klassnamn: ScreenView

Metodsignatur: public ScreenView()
Beskrivning: Konstruktorn, initierar miniatyrerna

Metodsignatur: public void ScreenShot(BufferedImage img)
Beskrivning: Skalar ner img till storlek som passar miniatyrfältet och lägger till det i en lista med miniatyrbilder.
Parametrar: img, är bilden som omvandlas till en imageicon.

Metodsignatur: public void deleteList(int listNr)

Beskrivning: Tar bort en minityrbild från listan med miniatyrer.

Parametrar: listNr, specifierar vilken miniatyr som ska tas bort.

Klassnamn: SelectedComponentDataModel

Klassbeskrivning: Datamodell som håller reda på markerad komponent och meddelar observerare.

Ärver: Observable

Implementerar: Serializable

Klassnamn: ToolsDataModel

Klassbeskrivning: Datamodell som håller reda på valt verktyg och meddelar observerare

Ärver: Observable

Implementerar: Observer, Serializable

Klassnamn: TabScreen

Metodsignatur: public TabScreen(int width,int height)

Beskrivning: Klassens konstruktor, initierar en tab.

Parametrar: width, tabens bredd och height, tabens höjd.

Metodsignatur: public Components.Resizable returnComp(int x, int y)

Beskrivning: Hämtar ut rätt komponent.

Parametrar: x och y är koordinaterna på aktiv komponent.

Metodsignatur: public void addComponent(Components.Resizable comp)

Beskrivning: Lägger till komponent, comp i listan

Parametrar: comp av typen Resizable.

Paket Components

Klassnamn: GUIComponents

Klassbeskrivning: Klassens uppgift är att fungera superklass för programmets alla komponenter. Detta för att kunna utnyttja polymorfism, och designmönstret Composite.

Ärver: JComponent

Implementerar: Serializable

Metodsignatur: public abstract Component getComp();

Beskrivning: Funktionen är abstract för att tvinga fram implementation i subklasser. Funktionen syftar till att returnera den grafiska komponent som skapas i en subklass.

Retur: Funktionens retur är av typen Component.

Metodsignatur: public abstract Component getCopy();

Beskrivning: Funktionen är abstract för att tvinga fram implementation i subklasser. Funktionen syftar till att returnera en kopia av den grafiska komponent som skapas i en subklass.

Retur: Funktionens retur är av typen Component.

Metodsignatur: public void registerModel(SelectedComponentDataModel selectedDataModel_)

Beskrivning: Metoden registrerar datamodellen hos komponenten som är aktiv för att möjliggöra att datamodellen kan lyssna på komponenten.

Parametrar: selectedDataModel_ är av typen SelectedComponentDataModel som lyssnar på varje komponent för att veta vilken kontroll som är aktiv för tillfället.

Metodsignatur: public int getParentHash()

Beskrivning:

Retur: parentHash som är av typen int och innehåller hashCoden för ramobjektet som ligger som förälder för den grafiska komponenten.

Metodsignatur: public void regParent(int hashCode)

Beskrivning: Funktionen sparar undan hashCoden för det ramobjekt som ligger som förälder för den grafiska komponenten.

Parametrar: hashCode som är en int och innehåller hashCoden för ramobjektet som ligger som förälder för den grafiska komponenten.

Metodsignatur: public void setViewMode()
Beskrivning: Funktionens uppgift är att ändra en komponents körläge från ritning till körning för att ändra uppförandet för komponenten.

Klassnamn: CompDatePicker

Klassbeskrivning: Klassens uppgift är att skapa en kalenderkomponent som kan ritas ut på en rityta.
Ärver: GUIComponents
Implementerar: Serializable, ActionListener, MouseListener

Metodsignatur: public CompDatePicker(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: private void setupComp()
Beskrivning: Funktionen skapar en kalenderkomponent med tillhörande händelsehantering. Därefter sparas komponenten undan i ett privat fält.

Metodsignatur: public JPanel getComp()
Beskrivning: Funktionen returnerar kalenderkomponenten till en rityta.
Retur: Returen består av variabeln jp, som innehåller kalenderkomponenten. Typen på kalendern är en JPanel.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Funktionen tar hand om events som kommer från knapptryckning inom kalenderkomponenten. I funktionen avgörs vad som skall göra baserat på vilken information eventet som skickas med som parameter innehåller.

Parametrar: e som är av typen ActionEvent. Parametern innehåller ett event som blivit aktiverat för att en knapp blivit klickad på.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker.

Metodsignatur: public JPanel getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.
Retur: Returen består av variabeln jp, som innehåller kalenderkomponenten. Typen på kalendern är en JPanel. Komponentens värden uppdateras med nya värden innan den skickas iväg.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void mouseClicked(MouseEvent e)
Beskrivning: Funktionen är tvingad att implementeras eftersom klassen implementerar muslyssnarinterfacet. Funktionen i fråga tar hand om musklick som sker på komponenten.I metoden avgörs det vad som klickats på och därefter utförs korrekt handling för klicket.
Parametrar: MouseEvent e som innehåller all information om eventet som skett.

Klassnamn: CompImage

Klassbeskrivning: Klassens uppgift är att skapa en komponent som innehåller ett bildobjekt som kan ritas ut på en rityta. Klassen innehåller även metoder för att byta- och modifiera den.
Ärver: GUIComponents
Implementerar: Serializable, MouseListener

Metodsignatur: public CompImage(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd. En kontroll görs på inparametrarna för att se så att någon av dem inte är noll. Om så är fallet så sätts variabeln som är noll till ett högre värde. Detta för att bildens skalningsalgoritm skall fungera då den kräver positiva tal över noll.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public JLabel getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.
Retur: Returen av typen JLabel består av en JLabel som innehåller bilden i form av en ImageIcon.

Metodsignatur: public JLabel getComp()
Beskrivning: Funktionen returnerar bildobjektet till en rityta. Bildobjektet i sig ligger i en JLabel.

Retur: Returen av typen JLabel består av en JLabel som innehåller bilden i form av en ImageIcon.

Metodsignatur: public void setImage(File fileToLoad)
Beskrivning: Funktionen är till för att byta ut bilden på ritytan. Den JLabel som innehåller bildobjektet uppdaterar sin ImageIcon med den nya filen som laddats in.
Parametrar: fileToLoad som är av typen File. Innehåller information om vilken bildfil det är som skall laddas in och ersätta befintlig bild.

Metodsignatur: public void setWidth(int i)
Beskrivning: Funktionen hanterar förändringar i bildens bredd och skalar om bilden baserat på det nya breddvärdet.
Parametrar: i som är av typen int innehåller den nya bredden som bildobjektet skall skalas till.

Metodsignatur: public void setHeight(int i)
Beskrivning: Funktionen hanterar förändringar i bildens höjd och skalar om bilden baserat på det nya höjdvärdet.
Parametrar: i som är av typen int innehåller den nya höjden som bildobjektet skall skalas till.

Metodsignatur: public String getImage()
Beskrivning: Funktionen returnerar bildobjektets filnamn.
Retur: Bildobjektets filnamn som en sträng.

Metodsignatur: public void setImgName(String s)
Beskrivning: I den här metoden så sätts filnamnsvariabeln till ett nytt värde. Detta görs när en ny bild laddas in.
Parametrar: s som är en sträng och innehåller filnamnet.

Metodsignatur: public void setStartLocation(int xx, int yy)
Beskrivning: Vid utritningen av ett bildobjekt så körs metoden för att uppdatera komponenten med nya värden för att göra en omskalning av bilden.
Parametrar: xx som är av typen int och som innehåller det nya breddvärdet för bilden. yy som är av typen inte och som innehåller det nya höjdvärdet för bilden.

Metodsignatur: public void setChoosen(boolean b)

Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.

Parametrar: Boolean b som innehåller information om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()

Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void mouseClicked(MouseEvent e)

Beskrivning: Funktionen är tvingad att implementeras eftersom klassen implementerar muslyssnarinterfacet. Funktionen i fråga tar hand om musklick som sker på komponenten.I metoden avgörs det vad som klickats på och därefter utförs korrekt handling för klicket.

Parametrar: MouseEvent e som innehåller all information om eventet som skett.

Klassnamn: CompJButton

Klassbeskrivning: Klassens uppgift är att skapa en knapp som kan ritas ut på en rityta. Klassen tillhandahåller även funktionalitet för att ge knappar möjlighet att hantera events.

Ärver: GUIComponents

Implementerar: Serializable, ActionListener

Metodsignatur: public CompJButton(int xone, int xtwo, int yone, int ytwo)

Beskrivning: Klassens konstruktor som ser till att en knapp skapas och sätts i ett rätt tillstånd.

Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public JButton getCopy()

Beskrivning: Metoden returnerar en kopia av den skapade knappen. Används vid kopiering av en komponent.

Retur: Returen är av typen JButton och det är en kopia av knappen som returneras.

Metodsignatur: public JButton getComp()

Beskrivning: Funktionen returnerar en knapp till en rityta

Retur: Returen av typen JLabel består av en JLabel som innehåller bilden i form av en ImageIcon.

Metodsignatur: public void setChoosen(boolean b)

Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.

Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()

Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public Font getFont()

Beskrivning: Metoden används för att få tag på typsnittet som knappen använder sig av.

Retur: Metodens returtyp är Font och innehåller information om knappens typsnitt.

Metodsignatur: public Color getFontColor()

Beskrivning: Metoden gör det möjligt att få fram knappens valda textfärg.

Retur: Colorobjekt som innehåller information om knappens textfärg.

Metodsignatur: public String getText()

Beskrivning: Metoden används för att komma åt en knapps text.

Retur: String som innehåller knappens text.

Metodsignatur: public Color getBgColor()

Beskrivning: Funktionen tillhandahåller knappens bakgrundsfärg.

Retur: Colorobjekt innehållandes knappens bakgrundsfärg.

Metodsignatur: public void setBgColor(Color bgCol)

Beskrivning: Funktionen sätter en ny bakgrundsfärg till knappen.

Parametrar: bgCol av typen Color som innehåller färginformation om vald färg.

Metodsignatur: public void setFontColor(Color textCol)

Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.

Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)

Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på knappen.

Parametrar: font av typen Font som innehåller det nya typsnittet som knappen skall använda sig av.

Metodsignatur: public void setText(String text)

Beskrivning: Med hjälp av den här funktionen kan knappens text ändras.

Parametrar: text som är en sträng och som innehåller den nya texten.

Metodsignatur: public void actionPerformed(ActionEvent e)

Beskrivning: Funktionen hanterar klick som sker på knappen. I funktionen görs en kontroll på vilket läge knappen befinner sig i, och därefter undersöks eventet som skedde och rätt åtgärd utförs därefter.

Parametrar: e av typen ActionEvent som innehåller informationen om eventet som skett.

Metodsignatur: public String getActionName()

Beskrivning: Funktionen returnerar vilket namn som är satt för action. Detta för att man i vyn skall kunna se vad knappen är inställd till att utföra.

Retur: En sträng som innehåller action namnet.

Metodsignatur: public void setPopUpAction(String s)

Beskrivning: Den här metoden körs om användaren vill koppla en popupruta till knappen. Funktionen kommer att skapa rutan och sätta den i rätt läge för att i det körbara skicket, vara redo att visas när en knappen trycks ned.

Parametrar: s som är en String och som innehåller texten som skall visas i popuprutan.

Metodsignatur: public void setChangeAction(int i)

Beskrivning: Funktionen anropas om användaren vill att knappen skall utföra ett bildbyte vid klick. Knappen blir satt att peka på en skärmbild i projektet.

Parametrar: i av typen int som är det skärmbildsnummer som den har i listan över skärmbilder.

Metodsignatur: public int getScreenPoint()
Beskrivning: Funktionen har till uppgift att ge systemet information om vilken skärmbild knappen pekar på, utifall knappen är inställd på att byta bild.
Retur: Returnerar en int som innehåller vilken plats i skärmbildslistan knappen pekar på.

Metodsignatur: public void setController(ButtonControl ctrl_)
Beskrivning: När ett projekt byter till körbart läge, så sätts en kontrollklass till varje knapp i projektet, för att hantera bildbyte som skall göras av knapparna. Detta för att knappen i sig själv inte skall ha någon uppfattning om systemet runt sig och inte få lov att interagera med listorna av skärmbilder.
Parametrar: ctrl_ av typen ButtonControl som sätts som kontroller.

Metodsignatur: public void setNoAction()
Beskrivning: Gör så att en knapp inte utför någon uppgift när den klickas på. Används för att ta bort tidigare valda actions, så som bildbyte eller popups.

Metodsignatur: public void decreseScreenPoint()
Beskrivning: Minskar skärmpekningen med ett. Detta görs bara ifall knappen pekar på en annan skärm och en skärmbild som har ett lägre index tas bort.

Klassnamn: CompJCheckBox

Klassbeskrivning: Klassens uppgift är att skapa en JCheckBox som kan ritas ut på en rityta.
Ärver: GUIComponents
Implementerar: Serializable

Metodsignatur: public CompJCheckBox (int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd.

Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public JCheckBox getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.
Retur: Metodens retur består av en kopia av den skapade komponenten av typen JCheckBox.

Metodsignatur: public JCheckBox getComp()
Beskrivning: Funktionen returnerar komponenten, som i det här fallet är en JCheckBox, till en rityta. Metoden registrerar även en itemlistener på checkboxen.
Retur: Metodens retur består av den skapade komponenten av typen JCheckBox.
Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public Font getFont()
Beskrivning: Metoden används för att få tag på typsnittet som checkboxen använder sig av.
Retur: Metodens returtyp är Font och innehåller information om checkboxens typsnitt.

Metodsignatur: public Color getFontColor()

Beskrivning: Metoden gör det möjligt att få fram checkboxens valda textfärg.

Retur: Colorobjekt som innehåller information om checkboxens textfärg.

Metodsignatur: public String getText()

Beskrivning: Metoden används för att komma åt en checkbox text.

Retur: String som innehåller checkboxens text.

Metodsignatur: public void setFontColor(Color textCol)

Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.

Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)

Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på checkboxen.

Parametrar: font av typen Font som innehåller det nya typsnittet som checkboxen skall använda sig av.

Metodsignatur: public void setText(String text)

Beskrivning: Med hjälp av den här funktionen kan checkboxens text ändras.

Parametrar: text som är en sträng och som innehåller den nya texten.

Metodsignatur: public Boolean getChecked()

Beskrivning: Metodens uppgift är till för att erbjuda andra delar av systemet, en möjlighet för att kontrollera om checkboxen är ikryssad eller ej.

Retur: Returtypen är en Boolean som kommer vara true utifall checkboxen är ikryssad, och false om den inte är det.

Metodsignatur: public void setChecked(Boolean b)

Beskrivning: Metoden ändrar checkboxens läge från ikryssad till inte, och tvärt om beroende på parametern som kommer in.

Parametrar: b är av typen Boolean och är true om checkboxen är ikryssad, om inte är den false.

Klassnamn: CompJRadioButton

Klassbeskrivning: Klassens uppgift är att skapa en JRadioButton som kan ritas ut på en rityta.
Ärver: GUIComponents
Implementerar: Serializable

Metodsignatur: public JRadioButton (int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public JRadioButton getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.
Retur: Metodens retur består av en kopia av den skapade komponenten av typen JRadioButton.

Metodsignatur: public JRadioButton getComp()
Beskrivning: Funktionen returnerar komponenten, som i det här fallet är en JRadioButton, till en rityta. Metoden registrerar även en itemlistener på radioknappen.
Retur: Metodens retur består av den skapade komponenten av typen JRadioButton.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()

Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur: True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public Font getFont()

Beskrivning: Metoden används för att få tag på typsnittet som radioknappen använder sig av.

Retur: Metodens returtyp är Font och innehåller information om radioknappen typsnitt.

Metodsignatur: public Color getFontColor()

Beskrivning: Metoden gör det möjligt att få fram radioknappen valda textfärg.

Retur: Colorobjekt som innehåller information om radioknappen textfärg.

Metodsignatur: public String getText()

Beskrivning: Metoden används för att komma åt en radioknappen text.

Retur: String som innehåller radioknappen text.

Metodsignatur: public void setFontColor(Color textCol)

Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.

Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)

Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på radioknappen.

Parametrar: font av typen Font som innehåller det nya typsnittet som radioknappen skall använda sig av.

Metodsignatur: public void setText(String text)

Beskrivning: Med hjälp av den här funktionen kan radioknappen text ändras.

Parametrar: text som är en sträng och som innehåller den nya texten.

Metodsignatur: public Boolean getChecked()

Beskrivning: Metodens uppgift är till för att erbjuda andra delar av systemet, en möjlighet för att kontrollera om radioknappen är vald eller ej.

Retur: Returtypen är en Boolean som kommer vara true utifall radioknappen är vald, och false om den inte är det.
Metodsignatur: public void setChecked(Boolean b)
Beskrivning: Metoden ändrar radioknappen läge från vald till inte vald, och tvärt om beroende på parametern som kommer in.
Parametrar: b är av typen Boolean och är true om radioknappen är vald, om inte är den false.

Klassnamn: CompJComboBox

Klassbeskrivning: Klassens uppgift är att skapa en JComboBox som kan ritas ut på en rityta och som innehåller funktionalitet för att den skall vara användbart i ett körbart läge.
Ärver: GUIComponents
Implementerar: Serializable , FocusListener

Metodsignatur: public CompJComboBox(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public JComboBox getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.
Retur: Metodens retur består av en kopia av den skapade komponenten av typen JComboBox.
Metodsignatur: public JComboBox getComp()

Beskrivning: Funktionen returnerar komponenten, som i det här fallet är en JComboBox, till en rityta. Metoden ser även till att förse comboboxen med en korrekt strängarray så att den fylls med valbara alternativ.
Retur: Metodens retur består av den skapade komponenten av typen JComboBox.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om komponenten är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur: True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void focusGained(FocusEvent arg)
Beskrivning: Metoden lyssnar på fokus och är när komponenten hamnar i fokus så skickas ett meddelande om detta till datamodellen som har hand om vilken komponent som är vald.
Parametrar: arg är av typen FocusEvent och innehåller all information om eventet som skett när komponenten fick fokus.

Metodsignatur: public void focusLost(FocusEvent arg)
Beskrivning: Metoden ansvarar för vad som skall hända när komponenten tappar fokus. Och det som sker är att den sätter sin ”är vald”-flagga till false.
Parametrar: arg är av typen FocusEvent och innehåller all information om eventet som skett när komponenten tappat fokus.

Klassnamn: CompJLabel

Klassbeskrivning: Klassens uppgift är att skapa en JLabel som ritas ut på en rityta. Klassen innehåller även funktionalitet för att man skall kunna ändra på texten så som färg och font.
Ärver: GUIComponents
Implementerar: Serializable, MouseListener

Metodsignatur: public CompJLabel(int xone, int xtwo, int yone, int ytwo)

Beskrivning: Klassens konstruktor som ser till att en label skapas och sätts i ett rätt tillstånd.

Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public JLabel getCopy()

Beskrivning: Metoden returnerar en kopia av den skapade texten. Används vid kopiering av en komponent.

Retur: Returen är av typen JLabel och det är en kopia av labelen som returneras.

Metodsignatur: public JLabel getComp()

Beskrivning: Funktionen returnerar en JLabel till en rityta

Retur: Returen av typen JLabel och innehåller en text som säger JLabel.

Metodsignatur: public void setChoosen(boolean b)

Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.

Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()

Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public Font getFont()

Beskrivning: Metoden används för att få tag på typsnittet som den aktuella labelen använder sig av.

Retur: Metodens returtyp är Font och innehåller information om textens typsnitt.

Metodsignatur: public Color getFontColor()

Beskrivning: Metoden gör det möjligt att få fram komponentens valda textfärg.

Retur: Colorobjekt som innehåller information om komponentens textfärg.

Metodsignatur: public String getText()

Beskrivning: Metoden används för att komma åt den aktuella labelens text.

Retur: String som innehåller texten för den aktuella labelen.

Metodsignatur: public void setFontColor(Color textCol)

Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.

Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)

Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på labelen.

Parametrar: font av typen Font som innehåller det nya typsnittet som skall användas.

Metodsignatur: public void setText(String text)

Beskrivning: Med hjälp av den här funktionen kan texten på komponenten ändras.

Parametrar: text som är en sträng och som innehåller den nya texten.

Metodsignatur: public int getFontSize()

Beskrivning: Funktionen finns till för att övriga delar av systemet skall kunna ta reda på vilken storlek på text som labelen använder sig av.

Retur: Returen består av en int som håller textstorleken.

Klassnamn: CompJTabedPane

Klassbeskrivning: Klassens uppgift är att skapa en JTabedPane som ritas ut på en rityta. Klassen erbjuder även funktionalitet för att kunna hålla en kontrollerklass för tabbarna samt håller indexet av tabbar.

Ärver: GUIComponents

Implementerar: Serializable, FocusListener, MouseListener

Metodsignatur: public CompJTabedPane(int xone, int xtwo, int yone, int ytwo)

Beskrivning: Klassens konstruktor som ser till att komponenten skapas och sätts i ett rätt tillstånd. I konstruktorn skapas även en instans av klassen TabController som kontrollerar ritytorna som befinner sig inuti tabbarna.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public Component getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Funktionen används vid kopiering av en komponent.
Retur: Returen är av typen Component och det är en kopia av tabbkomponenten som returneras.

Metodsignatur: public Component getComp()
Beskrivning: Funktionen sätter upp en JTabedPane med två ritytor som tabbar som default, och returnerar den sedan till en rityta
Retur: Returen av typen Component och består av den skapade JTabedPane komponenten.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public JPanel returnDrawingBoard()

Beskrivning: Funktionen används för att få tag i aktuell rityta från en lista med ritytor som finns lagrad i kontrollklassen. Den returneras dit därifrån funktionen blivit anropad ifrån.

Retur: JPanel som är den aktuella ritytan.

Metodsignatur: public TabScreen getScreenFromList(int i)

Beskrivning: Metoden används för att hämta ut vald rityta från listan med tabbkomponentens ritytor.

Parametrar: i är det heltal som i listan skall användas för att få ut efterfrågad rityta.

Retur: Returen är ett TabScreen-objekt och som innehåller en tabbs rityta.

Metodsignatur: public TabScreen getActiveTab()

Beskrivning: Metoden kan användas för att få fram ritytan för den aktuella tabben.

Retur: Returen är ett TabScreen-objekt och som innehåller en tabbs rityta.

Metodsignatur: public TabController getControl()

Beskrivning: En metod som används för att få tag i kontrollerobjektet för tabbpanelen.

Retur: Tabbpanelens aktuella TabController-objekt.

Metodsignatur: public void setTabCaption(String name)

Beskrivning: Tabbnamn ändras med hjälp av den här metoden. Den ändrar texten från ”Tab” som står som default till det som kommer in som parameter.

Parametrar: name som är en sträng som innehåller det namn som användaren vill ha på den aktuella tabben.

Metodsignatur: public String getTabCaption()

Beskrivning: Används för att komma åt den aktuella tabbens namn.

Retur: En sträng innehållande den aktuella tabbens namn.

Klassnamn: CompJTable

Klassbeskrivning: Klassens uppgift är att skapa en JTable som ritas ut på en rityta. Klassen erbjuder även funktionalitet för att modifiera tabellen och ge den vettiga värden samt kolumnnamn.
Ärver: GUIComponents
Implementerar: Serializable,MouseListener

Metodsignatur: public CompJTable (int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att tabellen skapas och sätts i ett giltigt läge.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public JScrollPane getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Funktionen används vid kopiering av en komponent.
Retur: Returen är av typen JScrollPane och det är en kopia av scrollpanelen, som innehåller tabellen, som returneras.

Metodsignatur: public JScrollPane getComp()
Beskrivning: Funktionen skapar en tabell med tomma fält. Funktionen returnerar sedan tabellen till den rityta som har begärt den.
Retur: Returen av typen JScrollPane som innehåller tabellen.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Klassnamn: CompJTextArea

Klassbeskrivning: Klassens uppgift är att skapa en JTextArea som ritas ut på en rityta. Klassen erbjuder även funktionalitet för att modifiera texten i den.
Ärver: GUIComponents
Implementerar: Serializable, MouseListener, FocusListener

Metodsignatur: public CompJTextArea(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att textarean skapas och sätts i ett giltigt läge.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public JTextArea getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Funktionen används vid kopiering av en komponent.
Retur: Returen är av typen JTextArea och det är en kopia av den skapade komponenten.

Metodsignatur: public JTextArea getComp()
Beskrivning: Funktionen skapar ett textfält. Därefter returneras det till den rityta som har begärt det.
Retur: Returen av typen JTextArea som är den skapade komponenten.

Metodsignatur: public void setChoosen(boolean b)

Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.

Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()

Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.

Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void setXYPlus(int pasteIndex)

Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.

Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public Font getFont()

Beskrivning: Metoden används för att få tag på typsnittet som textfältet använder sig av.

Retur: Metodens returtyp är Font och innehåller information om textens typsnitt.

Metodsignatur: public Color getFontColor()

Beskrivning: Metoden gör det möjligt att få fram komponentens valda textfärg.

Retur: Colorobjekt som innehåller information om komponentens textfärg.

Metodsignatur: public String getText()

Beskrivning: Metoden används för att komma åt textfältets text.

Retur: String som innehåller texten för textfältet.

Metodsignatur: public void setFontColor(Color textCol)

Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.

Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)

Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på komponenten.
Parametrar: font av typen Font som innehåller det nya typsnittet som skall användas.

Metodsignatur: public void setText(String text)
Beskrivning: Med hjälp av den här funktionen kan texten på komponenten ändras.
Parametrar: text som är en sträng och som innehåller den nya texten.

Klassnamn: CompJTextField

Klassbeskrivning: Klassens uppgift är att skapa ett JTextField som ritas ut på en rityta. Klassen erbjuder även funktionalitet för att modifiera texten i den.
Ärver: GUIComponents
Implementerar: Serializable, MouseListener, FocusListener

Metodsignatur: public CompJTextField(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Klassens konstruktor som ser till att komponenten skapas och sätts i ett giltigt läge.
Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public JTextField getCopy()
Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Funktionen används vid kopiering av en komponent.
Retur: Returen är av typen JTextField och det är en kopia av den skapade komponenten.

Metodsignatur: public JTextField getComp()
Beskrivning: Funktionen skapar ett textfält. Därefter returneras det till den rityta som har begärt det.
Retur: Returen av typen JTextField som är den skapade komponenten.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.

Parametrar: Boolean b som innehåller värdet om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void setXYPlus(int pasteIndex)
Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.
Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public Font getFont()
Beskrivning: Metoden används för att få tag på typsnittet som textfältet använder sig av.
Retur: Metodens returtyp är Font och innehåller information om textens typsnitt.

Metodsignatur: public Color getFontColor()
Beskrivning: Metoden gör det möjligt att få fram komponentens valda textfärg.
Retur: Colorobjekt som innehåller information om komponentens textfärg.

Metodsignatur: public String getText()
Beskrivning: Metoden används för att komma åt textfältets text.
Retur: String som innehåller texten för textfältet.

Metodsignatur: public void setFontColor(Color textCol)
Beskrivning: I den här metoden sätts textfärgen till en ny vald färg.
Parametrar: Colorobjekt textCol som innehåller färginformation om den valda färgen.

Metodsignatur: public void setFont(Font font)
Beskrivning: Funktionen har implementerats för att sätta ett nytt typsnitt på komponenten.
Parametrar: font av typen Font som innehåller det nya typsnittet som skall användas.

Metodsignatur: public void setText(String text)

Beskrivning: Med hjälp av den här funktionen kan texten på komponenten ändras.

Parametrar: text som är en sträng och som innehåller den nya texten.

Klassnamn: CompPictureButton

Klassbeskrivning: Klassens uppgift är att skapa en knapp som innehåller ett bildobjekt och som ritas ut på ritytan. Klassen innehåller även metoder för att byta- och modifiera bilden.

Ärver: GUIComponents

Implementerar: Serializable, ActionListener

Metodsignatur: public CompPictureButton(int xone, int xtwo, int yone, int ytwo)

Beskrivning: Klassens konstruktor som ser till att en komponent genereras och sätts i sitt rätta tillstånd. En kontroll görs på inparametrarna för att se så att någon av dem inte är noll. Om så är fallet så sätts variabeln som är noll till ett högre värde. Detta för att bildens skalningsalgoritm skall fungera då den kräver positiva tal över noll.

Parametrar: xone, xtwo, yone, ytwo, som alla är av typen int. De utgör komponentens startvärden för ritning, med xy samt hur hög och bred komponenten skall vara när den ritas ut.

Metodsignatur: public void setXYPlus(int pasteIndex)

Beskrivning: Meningen med den här funktionen är att den skall ändra värdet på komponentens x och y värden, vid inklistring, eftersom det förbättrar överskådligheten då flera komponenter inte kommer att ligga över varandra.

Parametrar: pasteIndex som är en int. Innehåller en siffra beroende på vilken inklistring i ordningen som sker. För varje inklistring ökar pasteIndex med ett.

Metodsignatur: public Component getCopy()

Beskrivning: Metoden returnerar en kopia av den skapade komponenten. Används vid kopiering av en komponent.

Retur: Returen av typen Component består av en kopia av den skapade bildknappen.

Metodsignatur: public Component getComp()

Beskrivning: Funktionen returnerar bildknappen till den rityta som begärt den.

Retur: Returen av typen Component, består av den skapade bildknappen.

Metodsignatur: public void setImage(File fileToLoad)

Beskrivning: Funktionen är till för att byta ut bilden på knappen.

Parametrar: fileToLoad som är av typen File. Innehåller information om vilken bildfil det är som skall laddas in och ersätta befintlig bild.

Metodsignatur: public void setWidth(int i)

Beskrivning: Funktionen hanterar förändringar i bildens bredd och skalar om bilden baserat på det nya breddvärdet.

Parametrar: i som är av typen int innehåller den nya bredden som bildobjektet skall skalas till.

Metodsignatur: public void setHeight(int i)

Beskrivning: Funktionen hanterar förändringar i bildens höjd och skalar om bilden baserat på det nya höjdvärdet.

Parametrar: i som är av typen int innehåller den nya höjden som bildobjektet skall skalas till.

Metodsignatur: public String getImage()

Beskrivning: Funktionen returnerar bildobjektets filnamn.

Retur: Bildobjektets filnamn som en sträng.

Metodsignatur: public void setImgName(String s)

Beskrivning: I den här metoden så sätts filnamnsvariabeln till ett nytt värde. Detta görs när en ny bild laddas in.

Parametrar: s som är en sträng och innehåller filnamnet.

Metodsignatur: public void setStartLocation(int xx, int yy)

Beskrivning: Vid utritningen av ett bildobjekt så körs metoden för att uppdatera komponenten med nya värden för att göra en omskalning av bilden.

Parametrar: xx som är av typen int och som innehåller det nya breddvärdet för bilden. yy som är av typen int och som innehåller det nya höjdvärdet för bilden.

Metodsignatur: public void setChoosen(boolean b)
Beskrivning: Uppgiften för den här metoden är att sätta en komponents valda tillstånd till true eller false. Dvs. om komponenten väljs genom att klickas på, så sätts värdet till true, annars false.
Parametrar: Boolean b som innehåller information om en komponent är vald eller inte.

Metodsignatur: public boolean isChoosen()
Beskrivning: Funktionen returnerar det satta värdet för om en komponent är vald eller ej.
Retur:True eller false beroende på om komponenten är vald eller ej.

Metodsignatur: public void actionPerformed(ActionEvent e)
Beskrivning: Funktionen är tvingad att implementeras eftersom klassen implementerar interfacet ActionListener. Funktionen i fråga tar hand om klick som sker på knappen.
Parametrar: ActionEvent e som innehåller all information om eventet som skett.

Klassnamn: CustomTableCellRenderer

Klassbeskrivning: Klassens uppgift är att skapa en cellrenderare för tabellkomponenten.
Ärver: DefaultTableCellRenderer
Implementerar: Serializable

Metodsignatur: public Component getTableCellRendererComponent(JTable table,Object obj, boolean isSelected, boolean hasFocus, int row, int column)
Beskrivning: Funktionen tar emot tabellen och hanterar bakgrundsfärgerna i raderna på tabellen. Om en rad är vald, så sätts dess bakgrundsfärg till rosa. Annars blir varannan rad vit, och varannan rad ljusgrå.
Retur: Component är returtypen och den innehåller den renderade tabellen.

Klassnamn: JTableHeaderRenderer

Klassbeskrivning: Klassens uppgift är att skapa en renderare för kolumnhuvuden i tabellkomponenten.
Ärver: DefaultTableCellRenderer
Implementerar: Serializable

Metodsignatur: public Component getTableCellRendererComponent (JTable table,Object obj, boolean isSelected, boolean hasFocus, int row, int column)
Beskrivning: Funktionen tar emot tabellen och hanterar bakgrundsfärgerna i kolumnhuvuden på tabellen. Om en kolumn är vald, så sätts dess bakgrundsfärg till rosa. Annars sätts den till grön.
Retur: Component är returtypen och den innehåller den renderade tabellen.

Klassnamn: Resizable

Klassbeskrivning: Klassens uppgift är att tillhandahålla funktionalitet för att ge komponenterna en ram med dragpunkter vilka gör att man kan byta storlek på en komponent genom att dra i dess hörn.
Ärver: JComponent
Implementerar: Serializable, MouseMotionListener, MouseListener

Metodsignatur: public Resizable(GUIComponents comp, ResizableBorder border)
Beskrivning: Detta är klassens konstruktor vilken ansvarar för att en komponent får en ram.
Parametrar: comp som är av typen GUIComponents och är den komponent som ramen skall läggas runt. Border Som är en ResizableBorder, är den ram som läggs runt komponenten.

Metodsignatur: private void resize()
Beskrivning: Funktionen har till uppgift att uppdatera det grafiska systemet när en komponent har ändrats genom ramen.

Metodsignatur: public void mousePressed(MouseEvent me)

Beskrivning: Funktionen hanterar det event som sker när en musknapp trycks ned. Då visas ramen ifall man är i ritläge.
Parametrar: me av typen MouseEvent vilket innehåller eventet som körts.

Metodsignatur: public void mouseDragged(MouseEvent me)
Beskrivning: Funktionen hanterar musdrag när man klickat någonstans på ramen. Antingen ändras komponentens storlek, eller så flyttas komponenten runt på ritytan.
Parametrar: me av typen MouseEvent vilket innehåller eventet som körts.

Metodsignatur: public void mouseReleased(MouseEvent mouseEvent)
Beskrivning: Sköter om det som skall hända när musknappen släpps.
Parametrar: mouseEvent som är det MouseEvent som sker när knappen släpps upp.

Paket Control

Klassnamn: ButtonControl

Klassbeskrivning: ButtonControlklassen används för att tillhandahålla kontrollfunktioner för knappars event, när prototypen är i körbart läge.

Metodsignatur: public ButtonControl(JFrame jf, Project p)
Beskrivning: Klassens konstruktor som sätter klassens privata variabler till vettiga värden.
Parametrar: jf som är en JFrame och är det fönster som prototypen kör i. p av typen Project är det projektet som man kör för tillfället.

Metodsignatur: public void setScreen(int i)
Beskrivning: Metodens uppgift är att hantera skärmbildsbyte när en knapp trycks.
Parametrar: i av typen int är siffran på den skärmbild som kontrollen skall byta till.

Klassnamn: TabController

Klassbeskrivning: Den här klassens uppgift är att tillhandahålla kontrollfunktionalitet till komponenter av typen JTabedPane.

Implementerar: Serializable, MouseListener, MouseMotionListener, Observer

Metodsignatur: public TabController()
Beskrivning: Konstruktorn för klassen skapar de två tabbarna som används som default.

Metodsignatur: public TabScreen getTabPaneFromList(int index)
Beskrivning: Tillhandahåller funktionalitet för att hämta ut en tabpanel från listan av tabpaneler.
Parametrar: index som är ett heltal och specificerar vilket skärmbildsindex man vill hämta ut.
Retur: Skärmbilden av typen TabScreen.

Metodsignatur: public int getListSize()
Beskrivning: Metoden returnerar listans storlek över hur många tabscreens som finns i den.
Retur: int som består av listans storlek.

Metodsignatur: public void setActiveDrawTab(int selectedIndex)
Beskrivning: Metoden sätter vald tab till den aktuella.
Parametrar: selectedIndex är det heltal som användaren valt skall visas.

Metodsignatur: private Rectangle calc(Rectangle r)
Beskrivning: Funktionen inverterar värden ifall man ritar på minussidan från där man tryckte ned musknappen.
Parametrar: r av typen Rectangle, som innehåller den rektangel man ritat upp på ritytan.
Retur: En rektangel som innehåller de inverterade värdena.
Metodsignatur: private void addComponent()
Beskrivning: Funktionen lägger till en ritad komponent på den aktiva ritytan.
Metodsignatur: public void delComponent(GUIComponents gui)
Beskrivning: Tar bort en komponent från den aktuella ritytan.
Parametrar: gui av typen GUIComponents vilket är den komponent som skall tas bort.

Metodsignatur: public void removeDependenciesToScreens(int i)
Beskrivning: Används för att ta bort beroenden till skärmbilder som tas bort. Funktionen går igenom alla knappar på alla ritytor i tabkomponenten och tar bort alla referenser.
Parametrar: i som är det skärmbildsindex som tas bort.

Klassnamn: ViewControl

Klassbeskrivning: Klassen har till uppgift att erbjuda systemets funktionalitet till det grafiska gränssnittet.
Ärver: Observable
Implementerar: Observer,Serializable, MouseListener, MouseMotionListener

Metodsignatur: public void openObject(File file)
Beskrivning: Funktionen har hand om att öppna en fil. Den kontaktar filemappern och ser till att rätt fil blir öppnad.
Parametrar: file av typen File som är den filen som skall öppnas.

Metodsignatur: public void saveObject(File file)
Beskrivning: Funktionen har hand om att spara en fil till disken. Den kontaktar filemappern och ser till att filen blir sparad.
Parametrar: file av typen File som är den fil som skall sparas.

Metodsignatur: public void addComponent(int xone, int xtwo, int yone, int ytwo)
Beskrivning: Funktionen lägger till en komponent till den aktiva ritytan.
Parametrar: Parametrarna är av typen int och beskriver startvärdet för x och y och sedan bredd och höjd för den komponenten som skall ritas ut.
Metodsignatur: public void startDemo(JFrame frame)
Beskrivning: Metoden kör igång prototypen till ett körbart läge.
Parametrar: frame som är det fönster som prototypen skall köras i.

Metodsignatur: public void drawingPanelScrShot()
Beskrivning: Den här funktionen är den som hanterar renderingen av miniatyrbilderna.

Metodsignatur: public Project newProject()
Beskrivning: En funktion som hanterar valet för ett nytt projekt. Skapar ett nytt projekt och sätter alla värden i korrekt läge.
Retur: Det nyskapade projektet.

Metodsignatur: public void update(Observable o, Object arg)

Beskrivning: Funktion som implementerats för att kunna lyssna på andra klassers förändringar. En kontroll görs på vilket objekt som skickar meddelandet och därefter genomförs korrekt uppgift.

Parametrar: o som är det observerade objektet och arg som är meddelandet som skickas.

Klassnamn: ProjectViewer

Klassbeskrivning: Den här klassens uppgift är att driva runt hela prototypvisaren. Här hittar vi bland annat main-metoden, samt funktionalitet som utför nödvändiga saker vid startupfasen då man laddar in en prototyp som man vill köra.

Implementerar: ActionListener

Metodsignatur: public ProjectViewer()

Beskrivning: Klassens konstruktor vars uppgift är att skapa de objekt som visaren behöver för att fungera, bland annat en instans av filemappern. Här skapas även fönstret som skall hantera visningen av prototypen.

Metodsignatur: private void openFile()

Beskrivning: Funktionen hanterar funktionalitet för att öppna en fil och läsa in prototypen till programmet.

Metodsignatur: private void load(Project p_)

Beskrivning: Funktionen anropas från openFile() och det är här som projektet laddas in och börjar köras. Här börjar en kontroll köras på komponenterna för att se ifall de är knappar, och i så fall sätts en buttoncontroller till den komponenten. Om en komponent av typen CompJTabedPane stöts på så anropas recursiveSubButtonSet.

Parametrar: Prototypen av typen Project som skall köras.

Metodsignatur: private void recursiveSubButtonSet(CompJTabedPane cjt)

Beskrivning: Funktionen kör igenom en CompJTabedPanes alla komponenter för att identifiera knappar för att sätta en buttoncontroller till dem. Funktionen fungerar rekursivt och kallar på sig själv om den upptäcker en CompJTabedPane i sig själv.

Parametrar: cjt som är den CompJTabredPane som skall gås igenom.