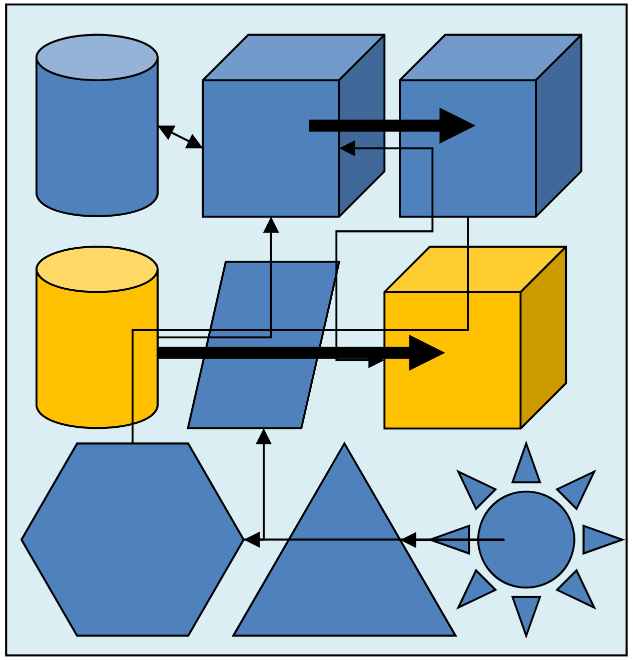
Abstraction - Definitions

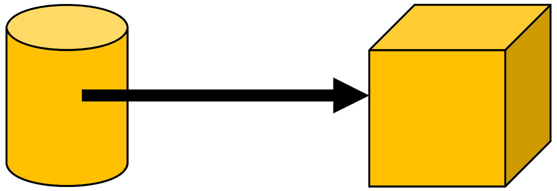
- **Definition 1: Modelling Abstraction**
  - **The process of selecting certain properties (attributes) of an entity to model that entity in say a computer program**
- **Definition 2: Collection Abstraction**
  - **The common properties of and operations on ADTs (set, sequence, tree, graph)**
  - **is_empty(), add(v), find(v), rem(v), cardinality() (size)**
- **Definition 3: ADT (implementation abstraction)**
  - **To implement the ADT as an abstract machine i.e. to hide as many of the implementation details as possible**

Modelling

- reality



- abstraction



Abstract Data Type

- **Abstraction**
  - Name
  - Address
  - P-number
  - Study year
  - Courses

- **Entity** (student) plus **Attributes**
- **Abstract data type**

- **Implementation**

```
Record Student {
    Name        string;
    Address     string;
    P-number    string;
    Study_year  integer;
    Courses     C_list;
    } Student;
```

- **Data type**

Collections

- **Set of students**
  - E.g. 1st year students

- **Properties - collection**
  - Number of entities
  - Empty or not?
  - I.e. Set properties
- **Properties - entity**
  - Name
  - Address
  - P-number
  - Study year
  - Courses

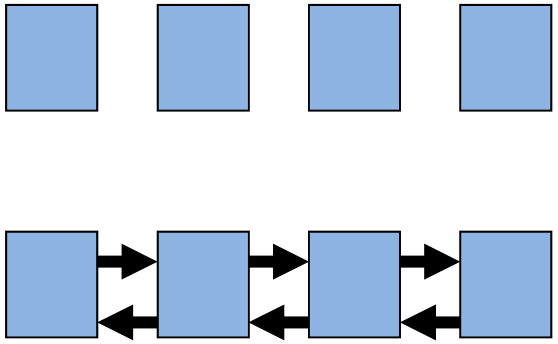
Collections

- In computer science we are often working with a **collection of entities**
- RELATIONSHIPS
  - there is a **relationship** between the entities
- **Collection = entities + relationship**

e.g. SEQUENCE	– successor + predecessor
e.g. TREE	– sub-parts (hierarchy)
e.g. GRAPH	– cost from A to B (general)

ADT - Sequence

- **Collection**



- **Properties**
  - Number of elements
  - Position of entity
    - **ORDER**
  - Successor **(relationship)**
  - Predecessor **(relationship)**

Implementation - Data types

- In most programming languages there are usually **two structures** with which **ADT**s may be implemented
  - **Array**
  - **Record (struct)**

- **Most implementations are based on combinations of these two structures**

Summary

- **Abstraction**
  - **Entity**
    - Attributes
  - **Relationships**
    - Attributes
- **The E/R model**

- **Implementation**
  - **Set**
  - **Sequence**
  - **Tree**
  - **Graph**
- Data types **(in a PL)**
  - **Array**
  - **Record**