Design Class Diagrams

PB007 Software Engineering I

Bruno Rossi

28. 11. 2016

Design Class Diagrams

A **Class Diagram** gives a static view of the classes, their attributes, operations and relationships.

Analysis Class Diagram

- business model of the domain - object types and relationships
- the effort is to maintain clarity and simplicity without clogging with implementation details.

Design Class Diagram

- the analysis model classes and the implementation details of the classes.

Design Classes

A **design class** provides a level of abstraction such that it can be easily implemented.

can come from:

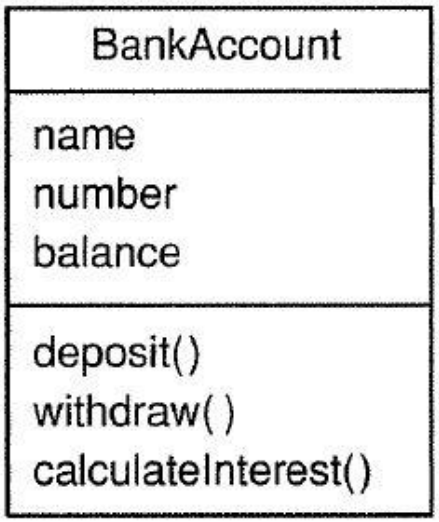
- Business domain - including details at the analysis level (decomposition into more classes, complement implementation details).
- domain technical classes - classes required by the technology used (classes for working with GUI, DB, ...)

Implementation details include:

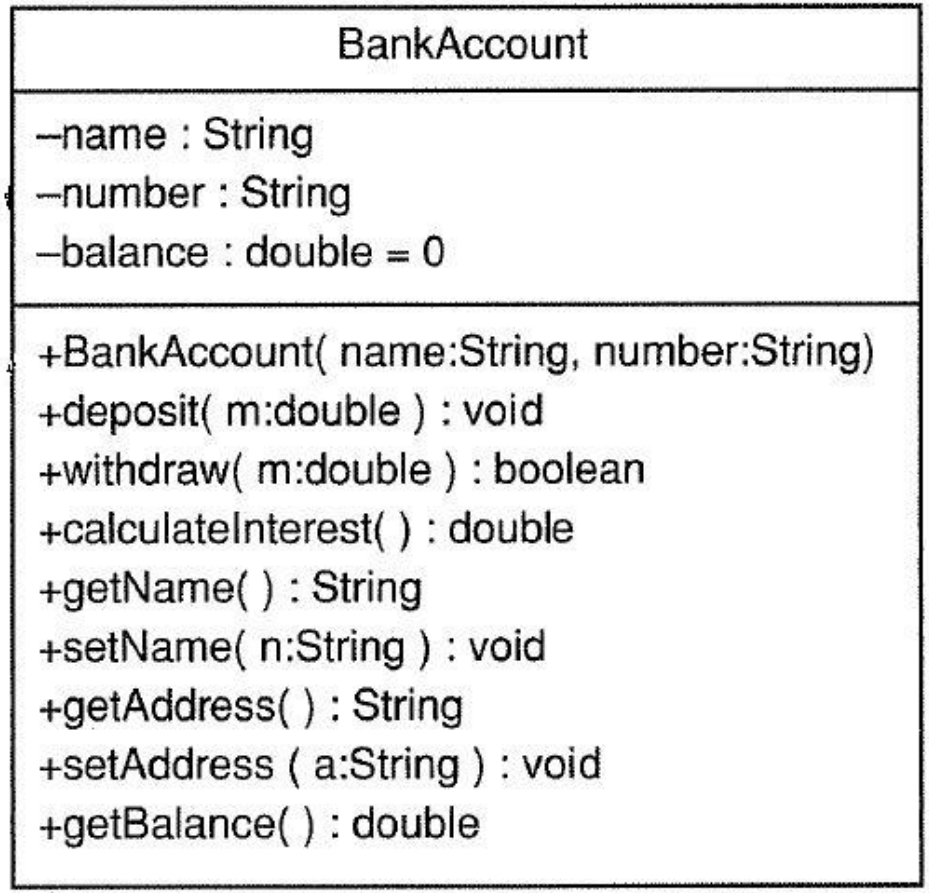
- Attributes and their types/visibility.
- Visibility, arguments, return types from methods.
- Methods added to the analysis operations, such as constructors (destructors), getter/setter methods, implementation methods.

Design Classes - Example

analysis



design



Advanced Analysis Associations

- More advanced association types for implementation details: aggregation or composition.
- Are generally defined with a name, navigability and multiplicity.
- Decomposition of bidirectional associations.
- Type of associations 1:1, 1:M, M:1.
- Decomposition of associations M:N.
- Decomposition of association classes.

Aggregation

Aggregation is a type whole-part relationship.

- The whole may or may not exist without its parts
- Parts can exist independently from the whole
- The whole is in a sense incomplete if some parts are missing.
- Part may theoretically be shared by several units.
- Aggregation is transitive and asymmetric (without cycles).

Composition

Composition is a stronger form of aggregation

- At a specific time parts can only belong to one group (they cannot stand alone).
- The whole is responsible for the creation and deletion of the parts.
- If the whole is deleted, it must either delete all its parts, or shift responsibility for them to another object.
- The composition is asymmetric and transitive (without cycles).

Revision of 1:1 associations

Analysis:

Design:

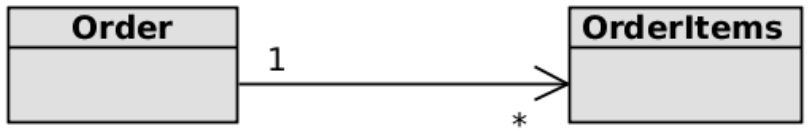
Revision of M:1 associations

Analysis:

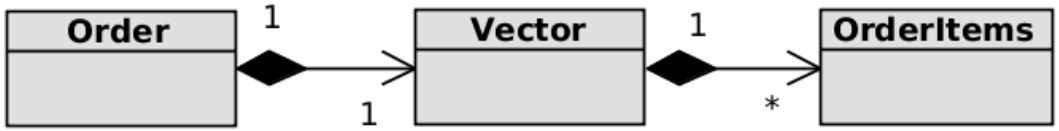
Design:

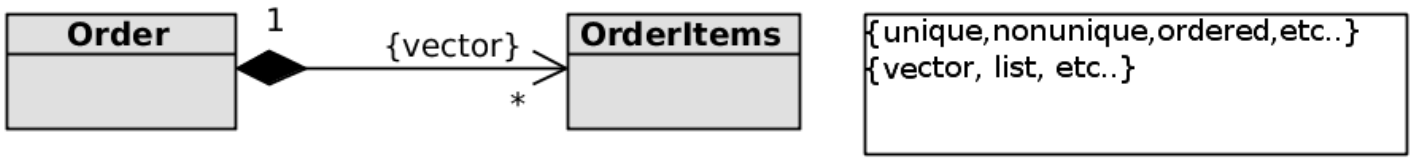
Revision of 1:M associations

Analysis:



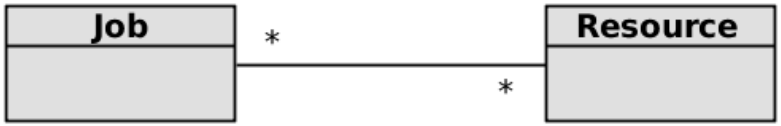
Design:



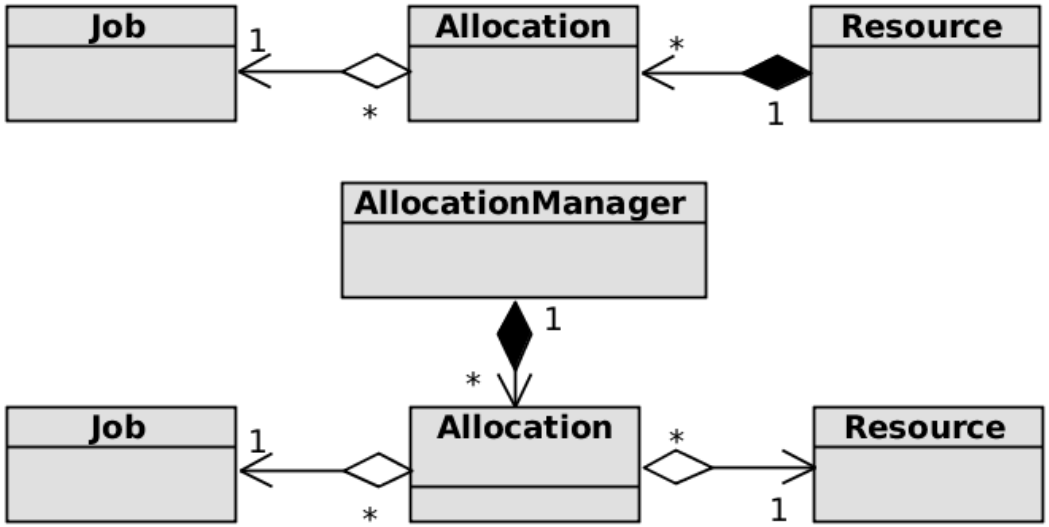


Decomposition of M:N associations

Analysis:

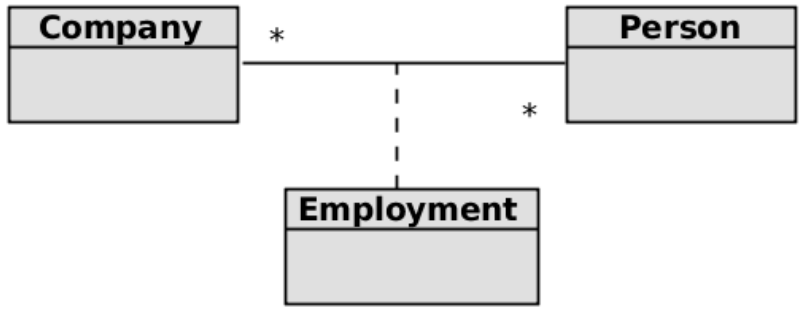


Design:

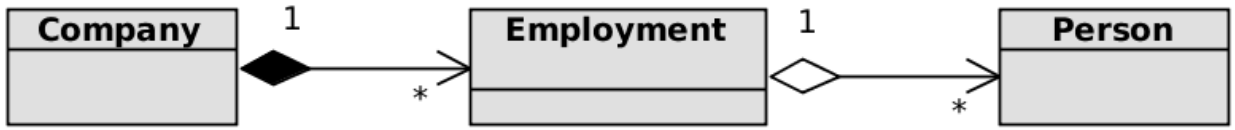


Decomposition of association classes

Analysis:

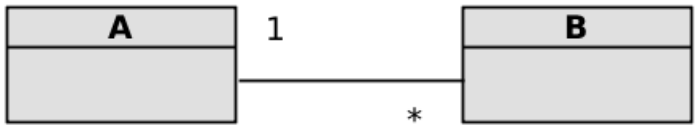


Design:

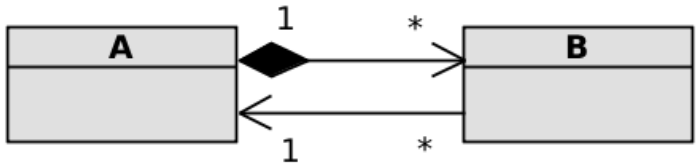


Decomposition of bi-directional associations

Analysis:

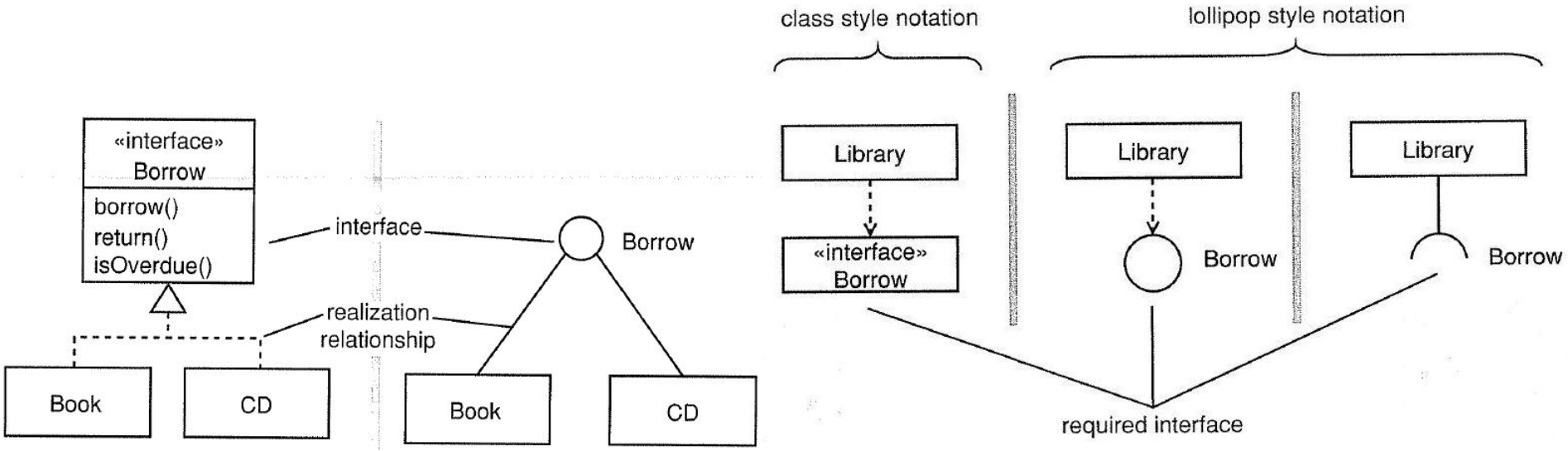


Design:



Interfaces

Interfaces are special classes that define a set of public services, attributes and relationships, but do not implement them. They are used to define the contract that classes provide.



Tasks

- Extend the analysis model into the design model by using class diagrams.
- Specify visibility and type of all attributes.
- Add methods that originated from the decomposition of analysis operations, implementation and support methods (constructors, getter / setter methods, ...), determine their visibility, arguments and return types.
- Please specify further the analysis associations (with naming, multiplicity, navigability, **aggregation / composition**, decomposition of association classes and M: N associations)
- Fill relations of dependencies among classes.
- If necessary, add other implementation classes or interfaces
- Upload the **PDF report** into folder (**Week 09**).
  Deadline: Saturday, 3.12.16 23:59

Customization of PDF Reports