Object-Oriented Design

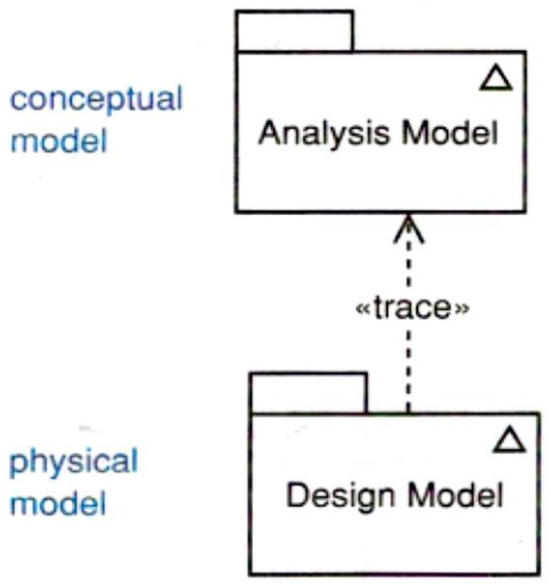
Lecturer: Raman Ramsin

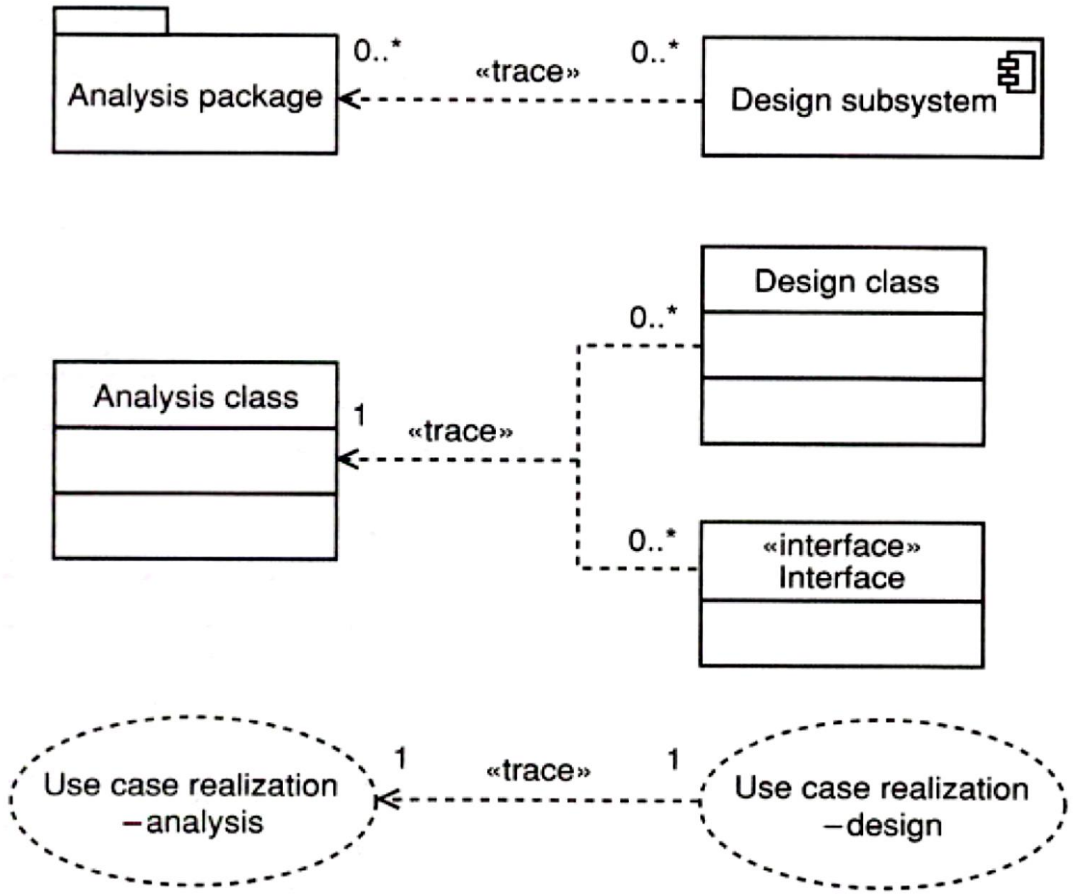
Lecture 16: Design Workflow

Design Workflow

- The design workflow is about determining how the functionality specified in the analysis model will be implemented.
- The design workflow is the primary modeling activity in the last part of the Elaboration phase and the first part of the Construction phase.
- The design model contains:
  - design subsystems;
  - design classes;
  - interfaces;
  - use case realizations-design;
  - a deployment diagram (first-cut).

Trace Relationships





Design Workflow: *Design a Class*

- The *Design Workflow* consists of the following activities:
  - Architectural Design
  - Design a Use Case
  - **Design a Class**
  - Design a Subsystem

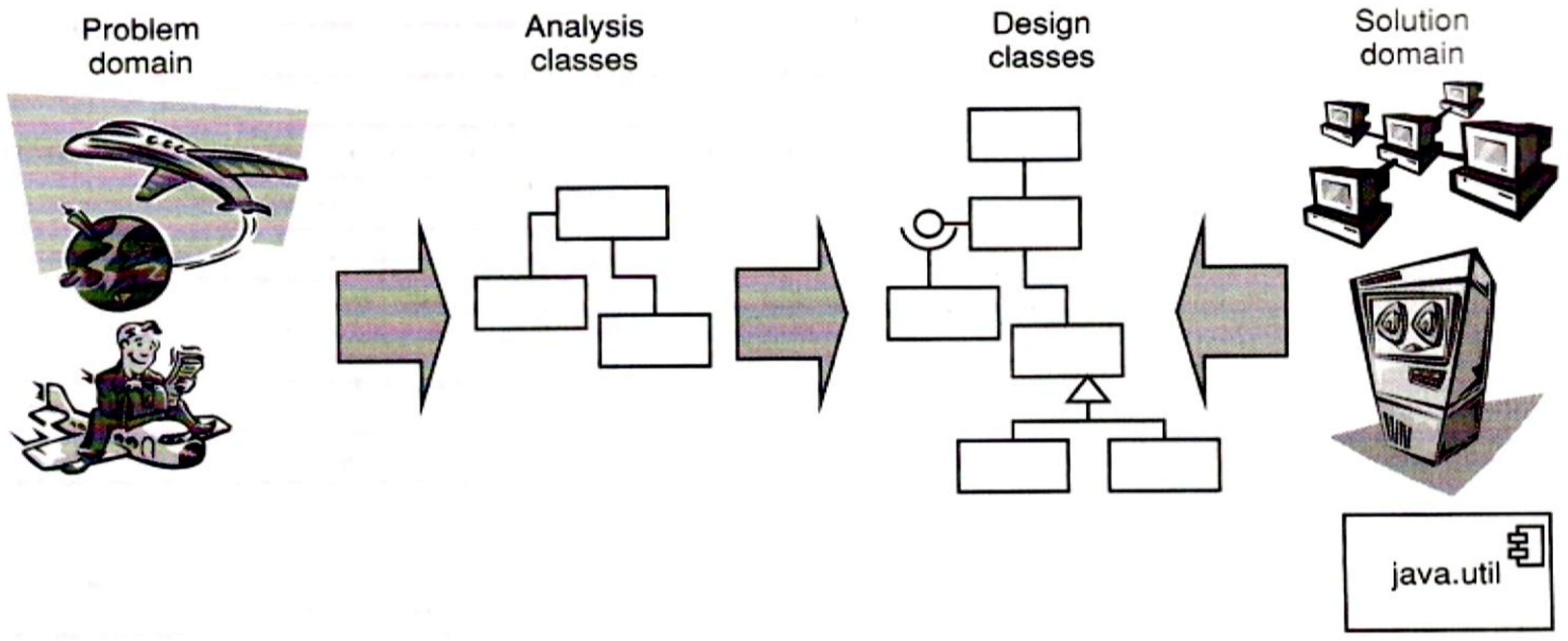
Design Classes

- Design classes are the building blocks of the design model.
- Design classes are developed during the USDP activity *Design a Class*.
- Design classes are classes whose specifications have been completed to such a degree that they can be implemented.

Design Classes: Sources

- Design classes come from two sources:
  - the problem domain:
    - a refinement of analysis classes;
    - one analysis class may become one or more design classes;
  - the solution domain:
    - utility class libraries;
    - middleware;
    - GUI libraries;
    - reusable components;
    - implementation-specific details.

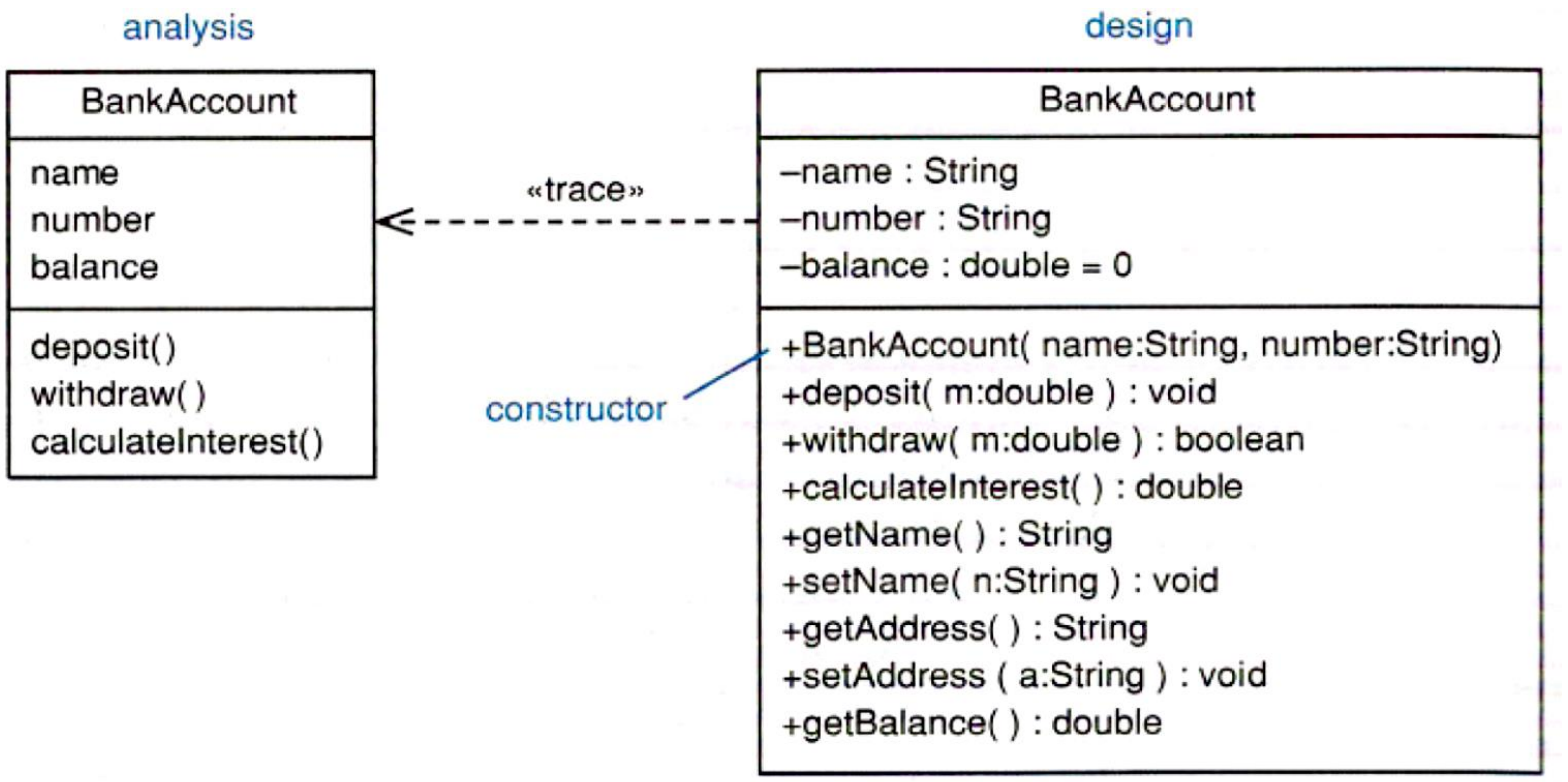
Design Classes: Sources



Design Classes: Anatomy

- Design classes have complete specifications:
  - complete set of attributes including:
    - name;
    - type;
    - default value when appropriate;
    - visibility;
  - operations:
    - name;
    - names and types of all parameters;
    - optional parameter values if appropriate;
    - return type;
    - visibility.

Design Classes: Anatomy



Design Classes: Well-formedness

- The public operations of the class define a contract with its clients.
- **Completeness** - the class does no less than its clients may reasonably expect.
- **Sufficiency** - the class does no more than its clients may reasonably expect.
- **Primitiveness** - services should be simple, atomic, and unique.

Design Classes: Well-formedness (Contd.)

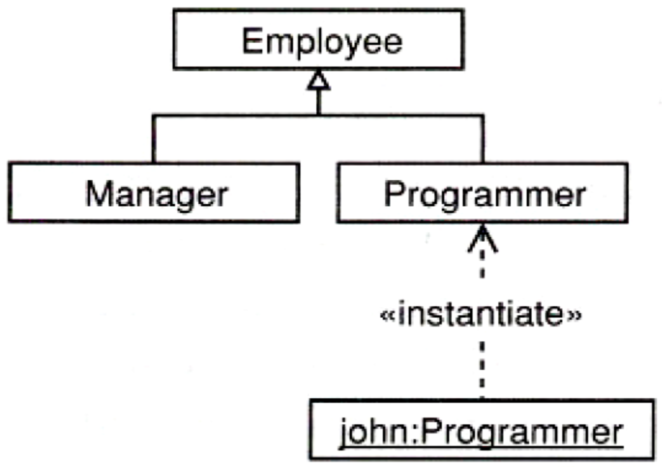
- **High cohesion**:
  - each class should embody a single, well-defined abstract concept;
  - all the operations should support the intent of the class.
- **Low coupling**:
  - a class should be coupled to just enough other classes to fulfill its responsibilities;
  - only couple two classes when there is a true semantic relationship between them;
  - avoid coupling classes just to reuse some code.

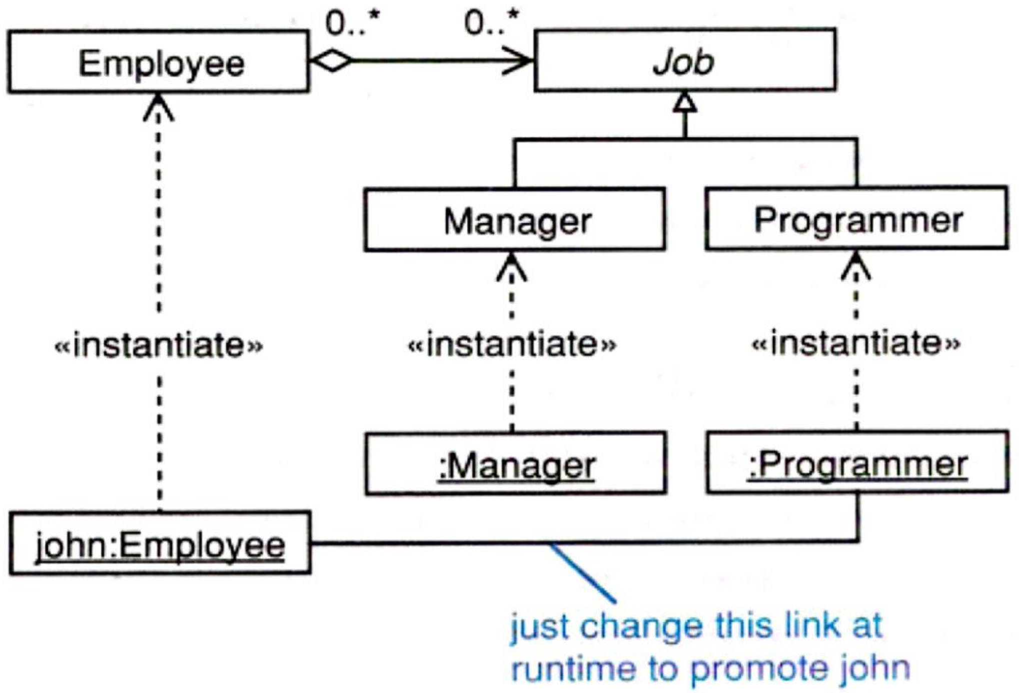
Inheritance

- Only use inheritance when there is a clear "is a" relationship between two classes or to reuse code.
- Disadvantages:
  - it is the strongest possible coupling between two classes;
  - encapsulation is weak within an inheritance hierarchy;
  - very inflexible in most languages - the relationship is decided at compile time and fixed at runtime.

Inheritance and Aggregation

- Subclasses should always represent "is kind of" rather than "is role played by" - always use aggregation to represent "is role played by".





Reference

- Arlow, J., Neustadt, I., *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Ed. Addison-Wesley, 2005.