CPSC 310 – Software Engineering

Design

UML (review)

Return of the OOP

An object is an encapsulation of data.

- identity (a unique reference)
- structure/state (variables)
- behavior (methods)

A class is the description of a set of common objects

UML

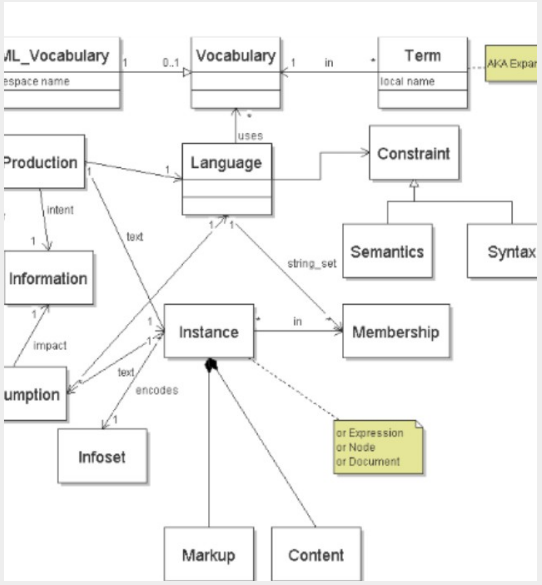
A widespread way of representing/abstracting reality

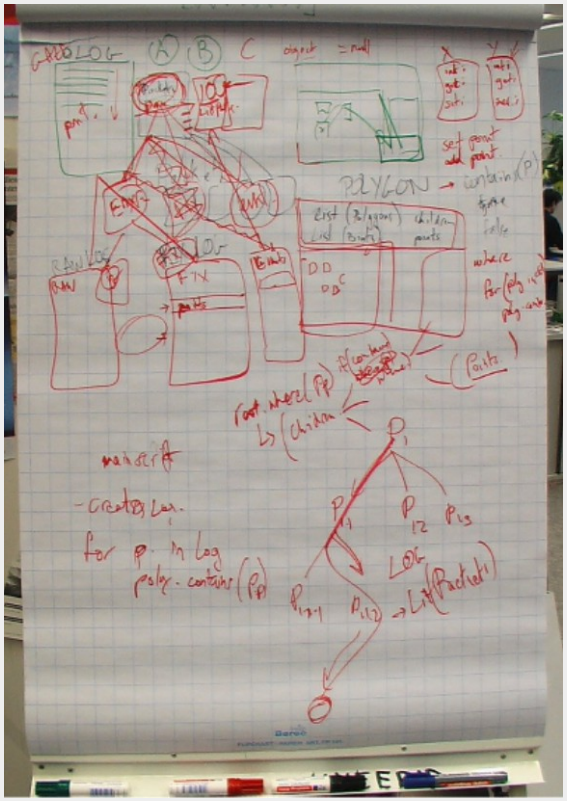
http://www.omg.org/spec/UML/

Diagrams

- Diagrams are a ***communication*** tool (but they can be much more)

- Quality of communication = Quality of design
  - Hence, quality of end product
- Tip for efficient communication:
  - Start light-weight and flexible
  - Then move on to details and more focused
- In terms of diagrams:
  - You may start with draft, hand-written diagrams that can change
  - Towards the end, clean-up and make more readable
  - Use a mutually understood language (eg. UML) that is adapted to your problem





Class Diagrams

- Used to describe the relationships between classes (and/or packages) in the system

Unified Modeling Language part for STRUCTURAL aspects

- Main elements of UML class diagrams
  - Classes
  - Relationships
    - Generalization
    - Association
    - Composition

Class Diagrams: the Class

structure

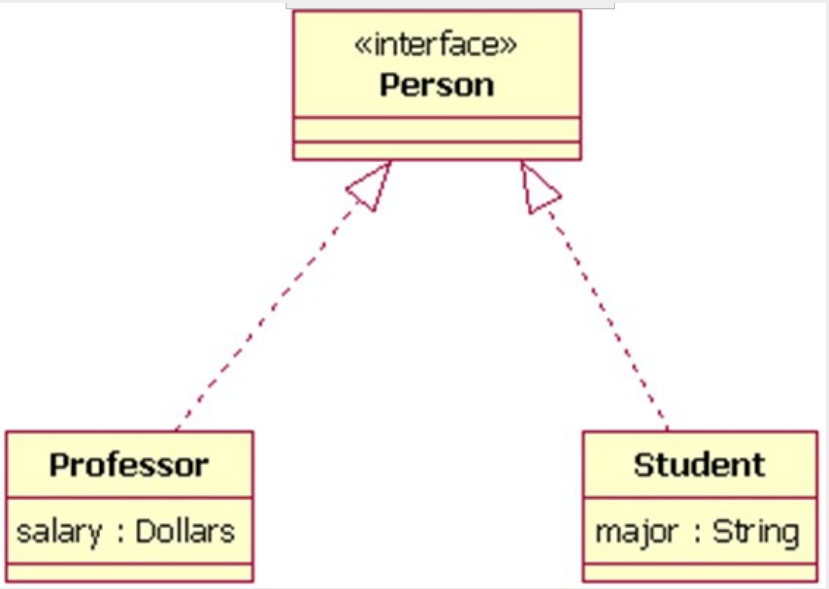
behavior

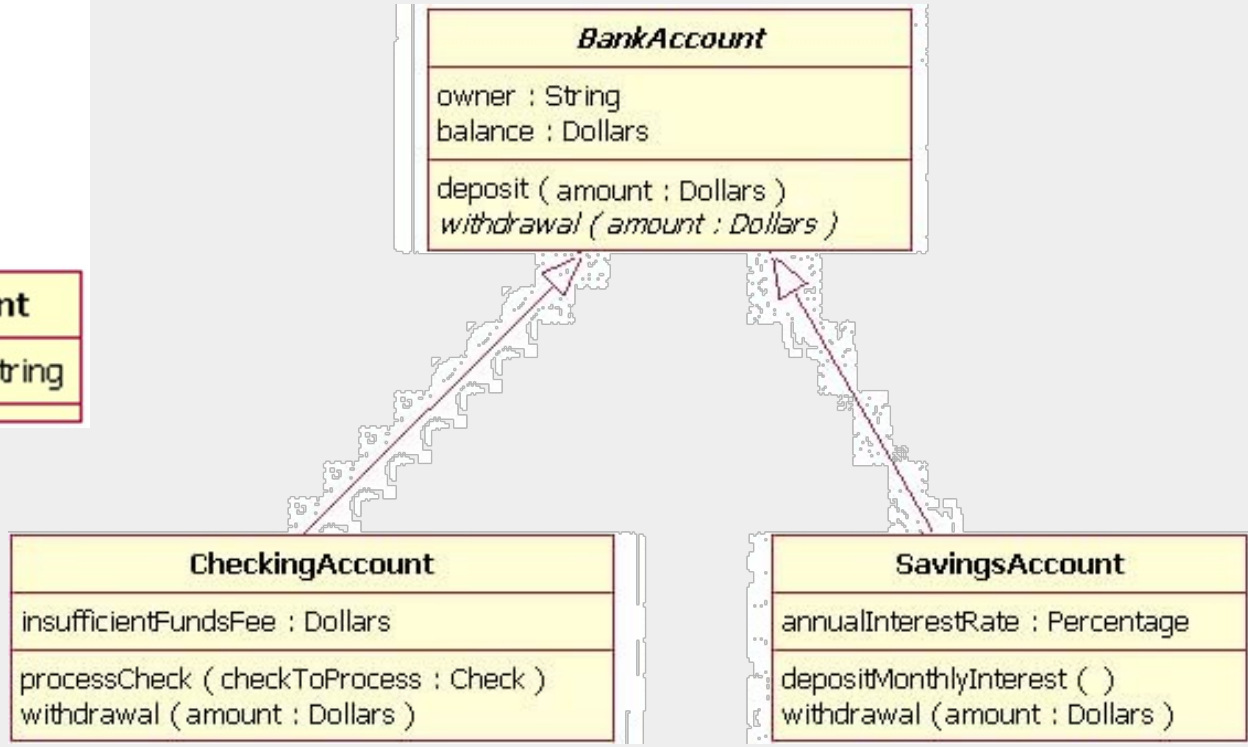
- Class name (*Italics* means abstract)
- Attributes (fields)

  `Name : Type`
- Operations (methods)

  `(parameters) : ReturnType`
- Can also be used for interfaces (without fields)

Class Diagrams: Generalization

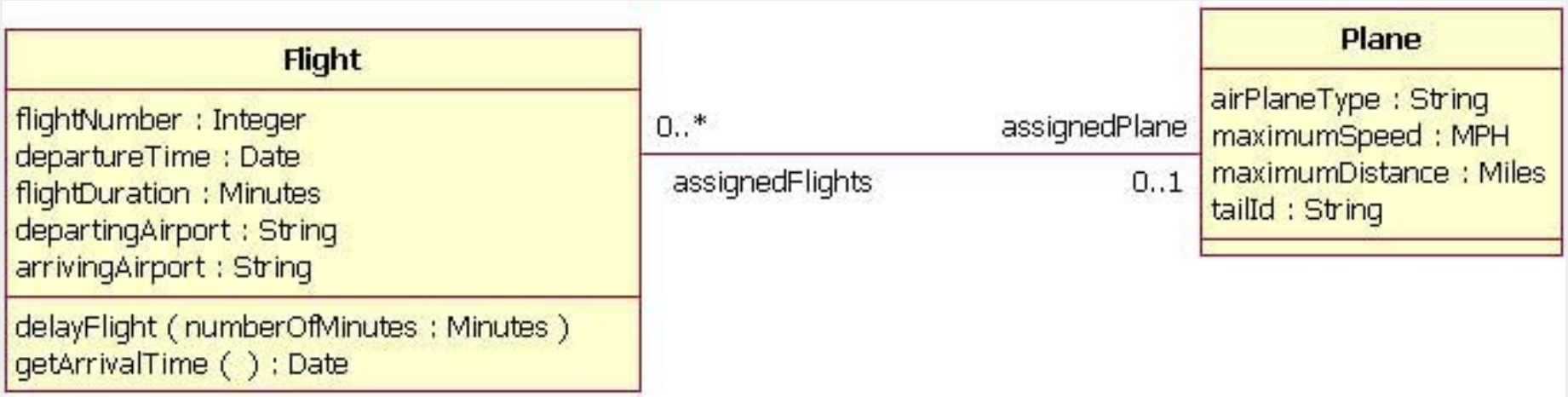




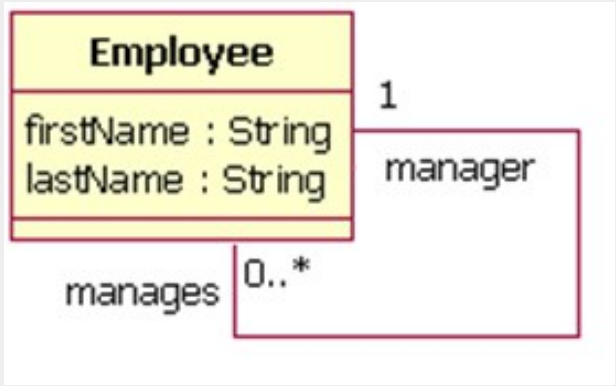
Used for:

- Inheritance
- Interface implementation

Class Diagrams: Association



- Bi-directional
  - Both classes are aware of each other
- Role
  - Usually maps to a field name
- Multiplicity
  - Indicates how many instances can be linked (*i.e.* a list of…)



Class Diagrams: Uni-directional Association

- Only one class knows of the other
- Role
  - Only in one direction
- Multiplicity
  - sometimes shown only on one end (BankAccount doesn't know report)

Class Diagrams: Special Associations

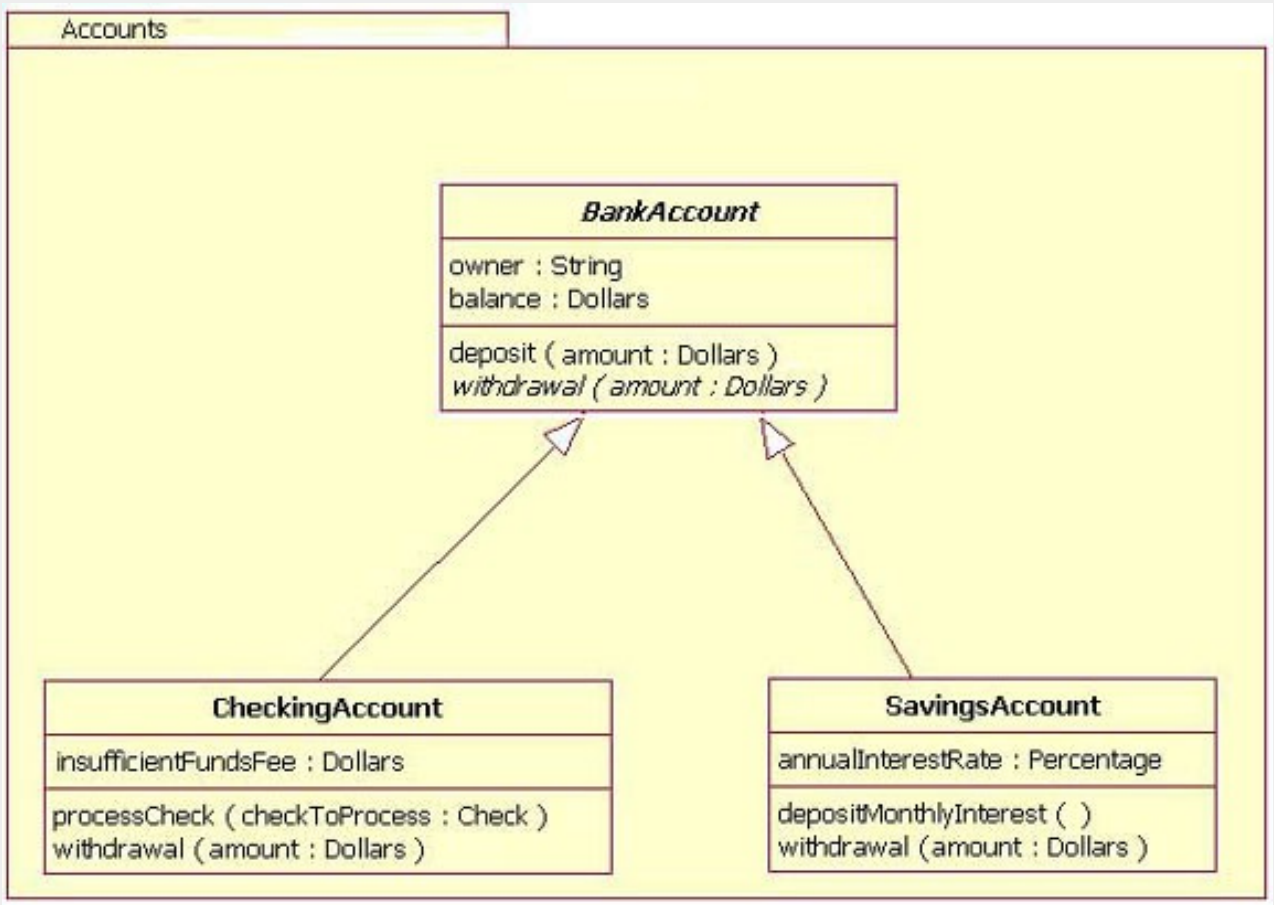
- Constrained associations
- The contained object is *part* of the container
- Two types:
  - ~~Aggregation~~ (forget about that one)
  - Composition: children's life depends on parent

Class exercise

- How would you implement in Java these two simple examples of a normal association and a composition (I will go around picking some of your work)

Class Diagrams: Packages

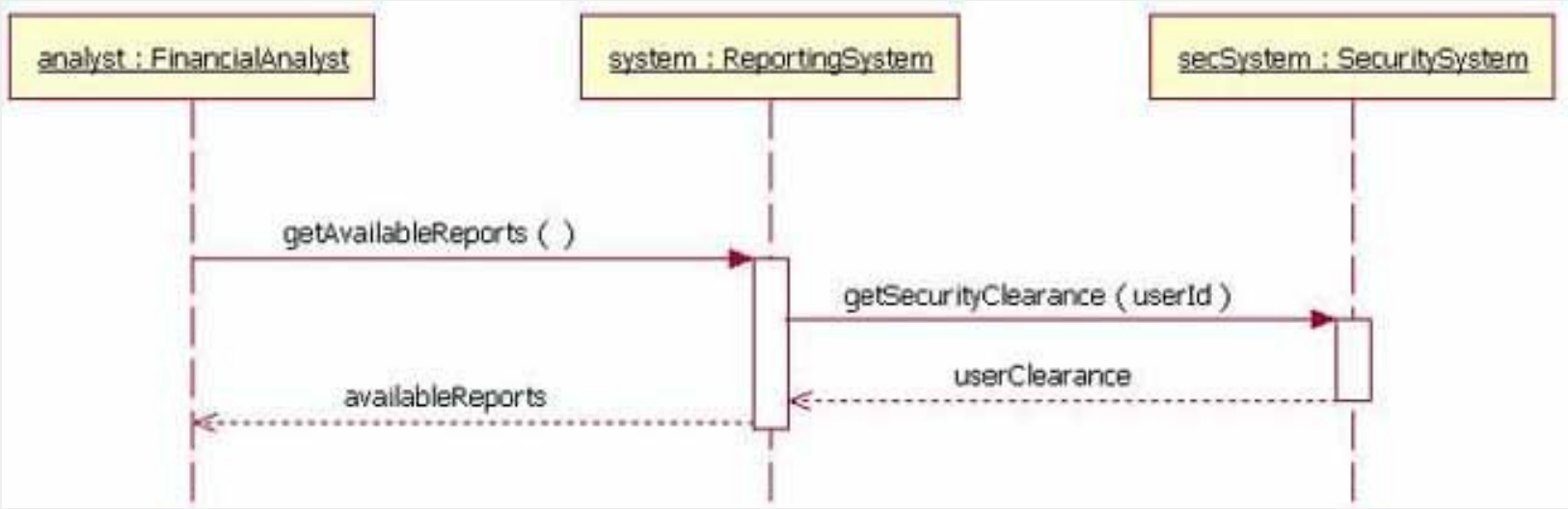
- Group classes together



Sequence Diagrams

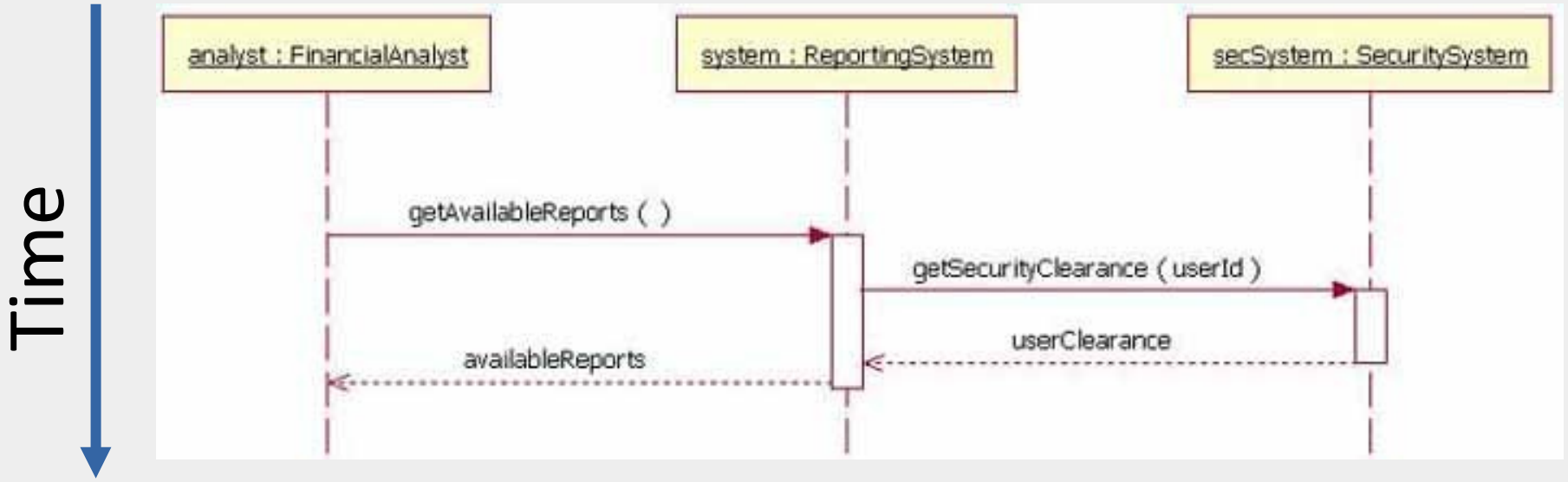
- Used to describe sequences of invocations between the objects that comprise the system
  - Focus less on *type of messages*, more on the *sequence* in which they are received
- Elements of UML sequence diagrams:
  - Lifelines
  - Messages

Sequence Diagrams: Lifeline



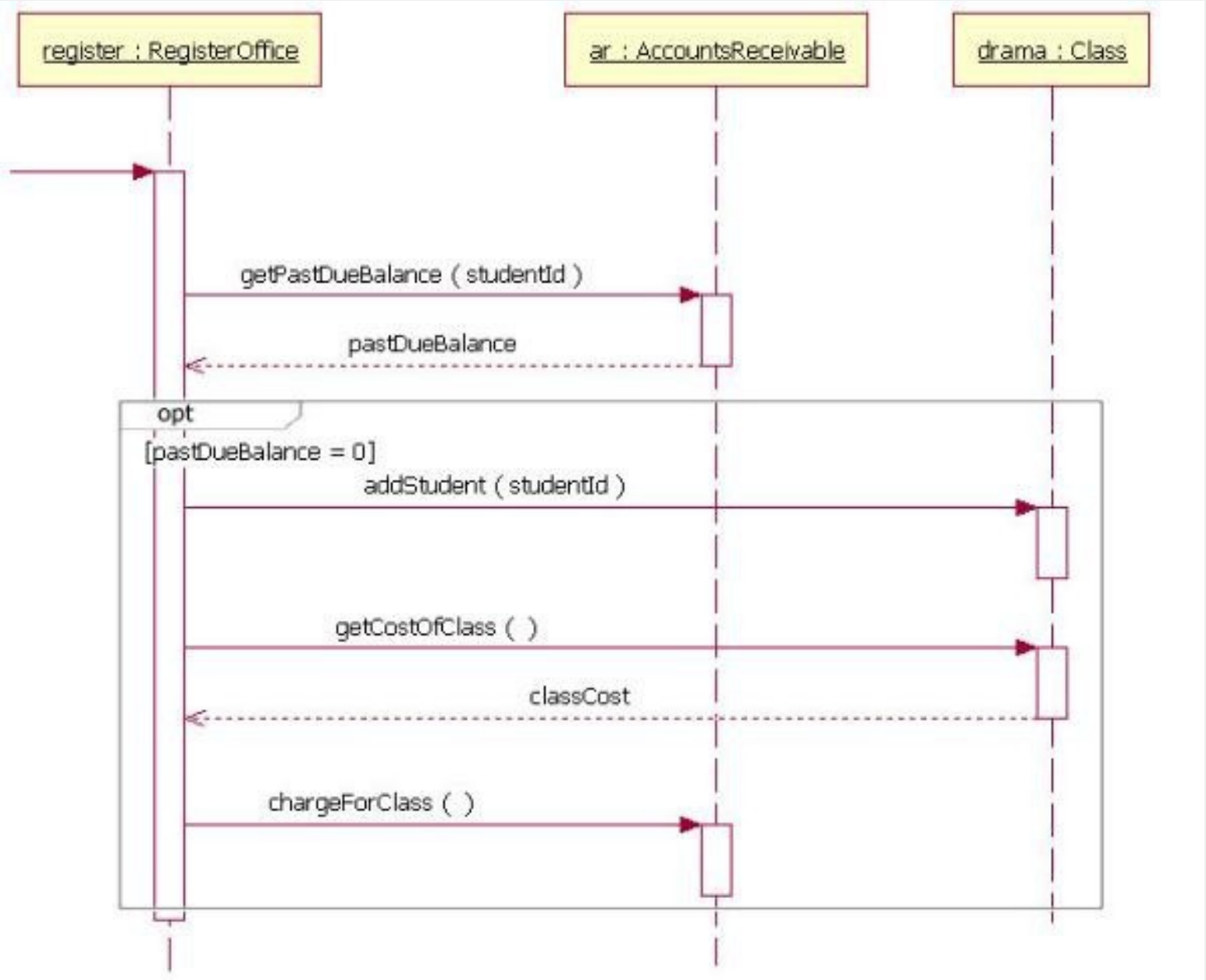
- Roles or object instances
- Participate in the sequence being modeled

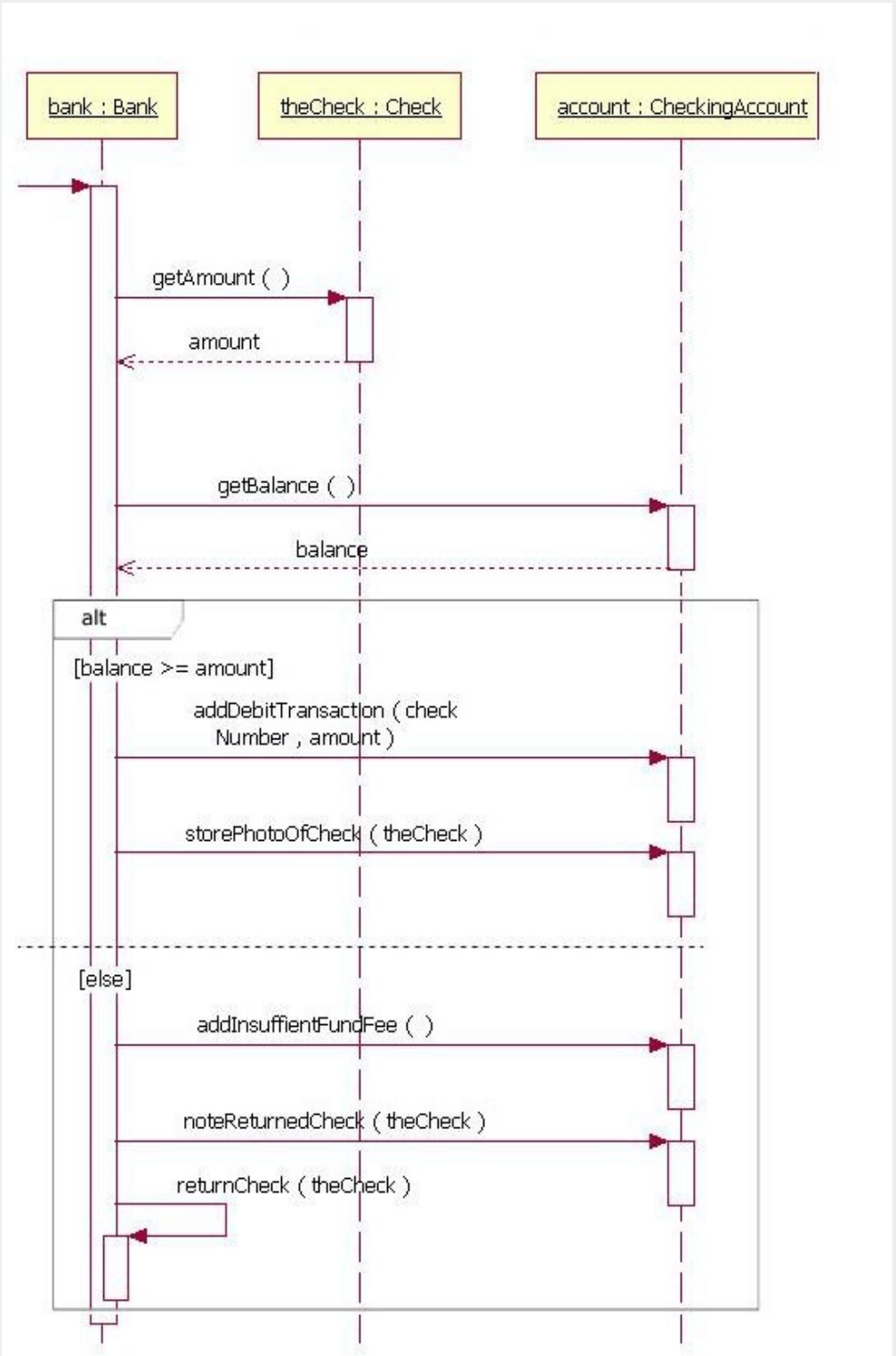
Sequence Diagrams: Messages



- Includes method name
- A box in the receiver's lifeline indicates activation (object's method is on the stack)
- Full arrow: synchronous (blocking)
- Optionally: information returned

Sequence diagram for conditionals





Sequence diagram when some actions are inside an if/else

Loops are similar - put the actions inside a box labeled “loop”