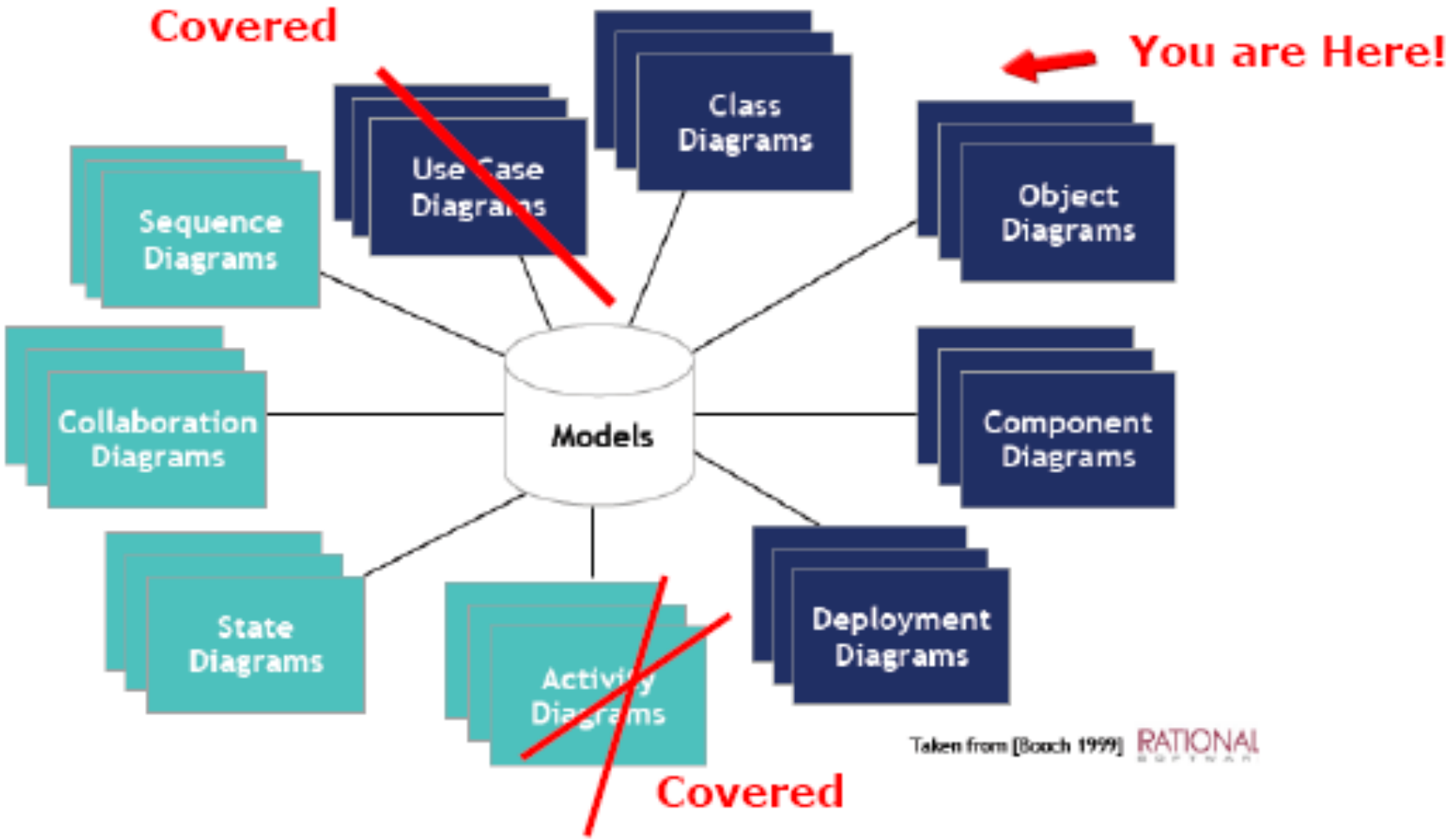
UML Diagrams



General Design Approaches

- Top-Bottom Approach

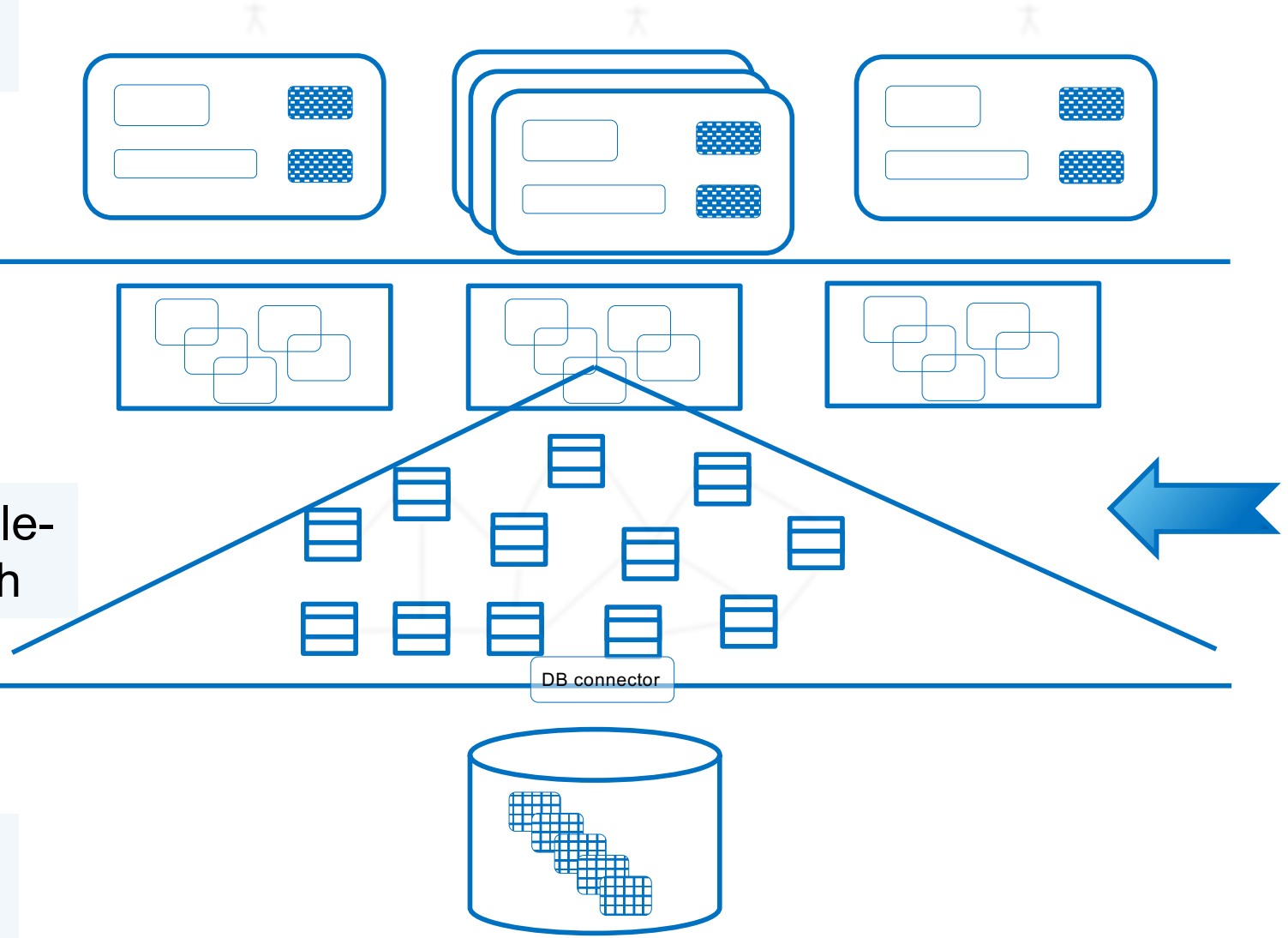
Top

Middle

- Middle-top-Middle-bottom Approach

Bottom

- Bottom-Top Approach



Class diagrams

Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.

An object class can be thought of as a general definition of one kind of system object.

An association is a link between classes that indicates that there is some relationship between these classes.

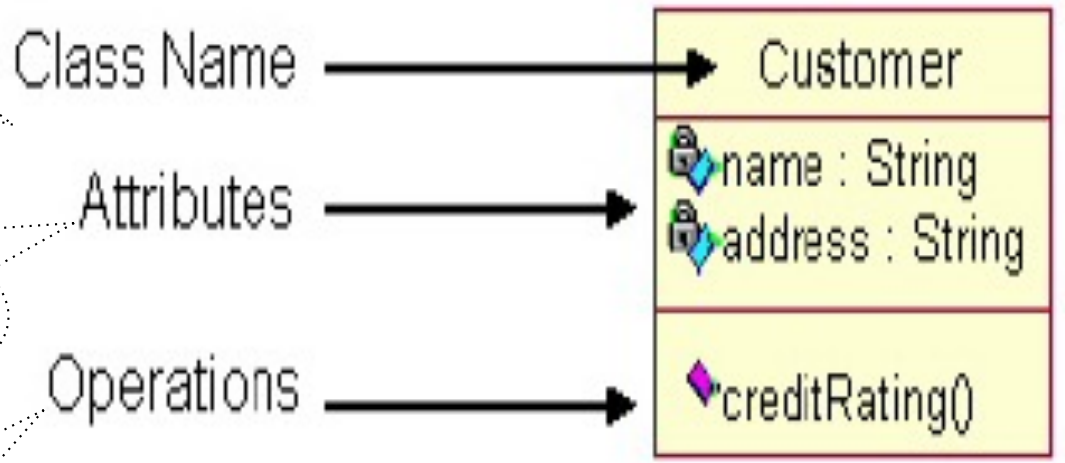
When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

Simple Class Diagram

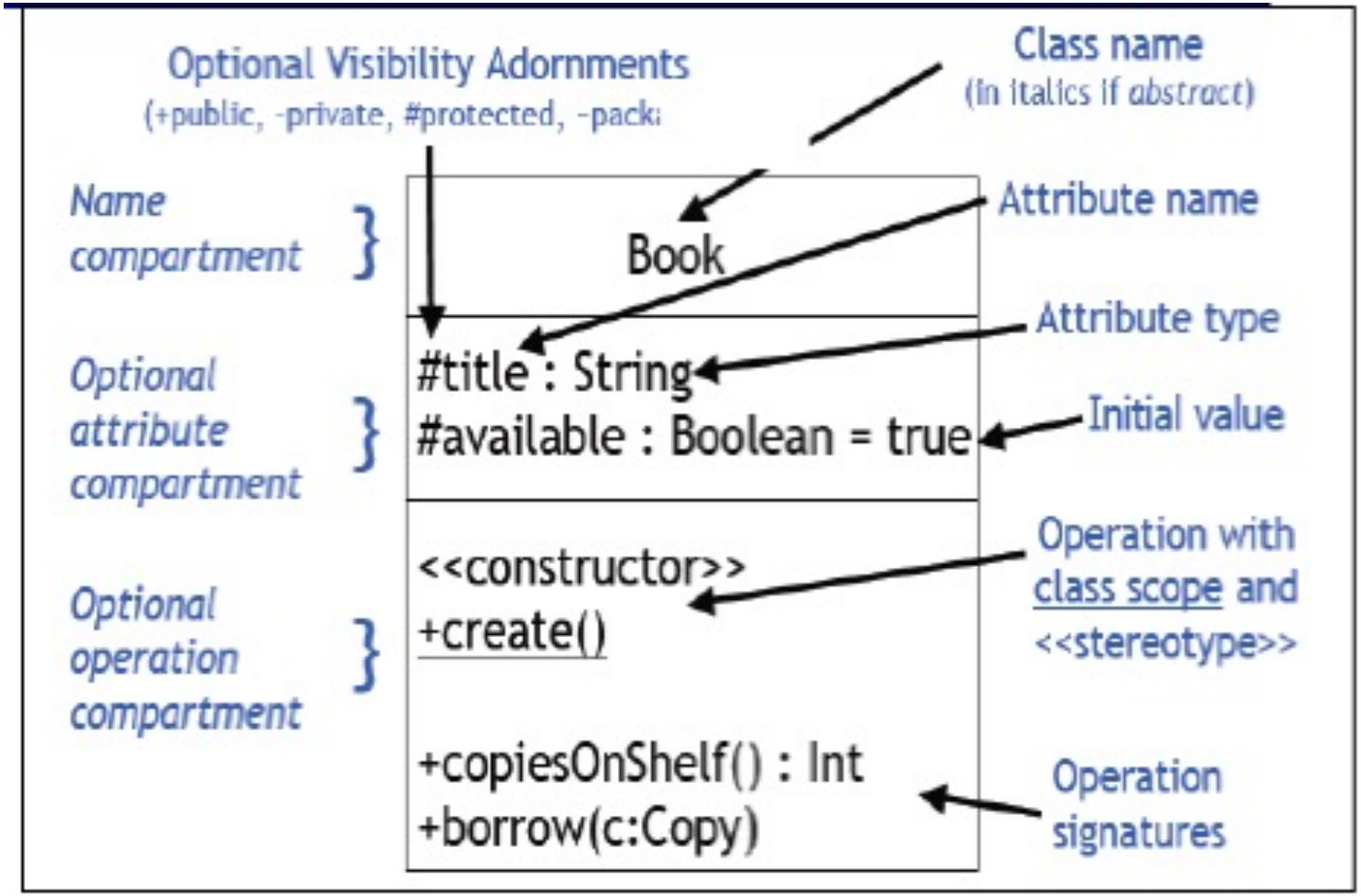
This represents a system object, which may refer to an Actor or entity of the system

These relate to the data that system must store for this object

These relate to the use cases provided by this object



UML Class Icons



Reference: D. Rosenblum, UCL

+, #, -

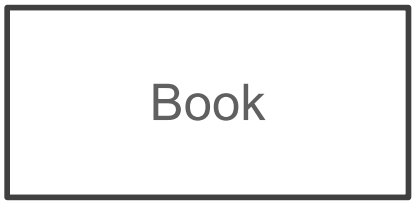
+ means public: public members can be accessed by any client of the class
# means protected: protected members can be accessed by members of the class or any subclass
- means private: private members can only be accessed by members of the same class

Additional:
~ package visibility
/ derived classes visibility

Analysis Class

An analysis class abstracts one or more classes and/or subsystems in the system's design

- Focuses on handling functional requirements
- Defines responsibilities (cohesive subsets of behaviour defined by the class, e.g. use cases or services it provides to other classes)
- Defines attributes
- Expresses relationships the class is involved in



Approach: Data-Driven Design

Identify all the data in the system
Divide into classes before considering responsibilities
Common approach: **noun identification**
- Identify candidate classes by selecting all the nouns and nouns phrases in the requirements document
- Discard inappropriate candidates
  - Redundant or omnipotent entities
  - Vague entities
  - Events or operations
  - Meta-language
  - Entities outside system scope
  - Attributes

Verbs and verb phrases highlight candidate operations!

Data-Driven Design Approach

Some heuristics/hints of what kind of things are classes [Shlaer and Mellor; Booch]:

Tangible or "**real-world**" things – e.g. book, copy, course;

Roles- e.g. library member, student, director of studies,

Events- e.g. arrival, leaving, request;

Interactions- e.g. meeting, intersection

Exercise

Perform **noun-verb** analysis of a requirements document (example text from next slide);
Underline all the noun and noun phrases,
Create a list of candidate classes (in examining the discard criteria, you may also identify some candidate attributes)

Identify all verb and verb phrases
Create a list of candidate operations and assign them to classes

Noun/Verb Analysis

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

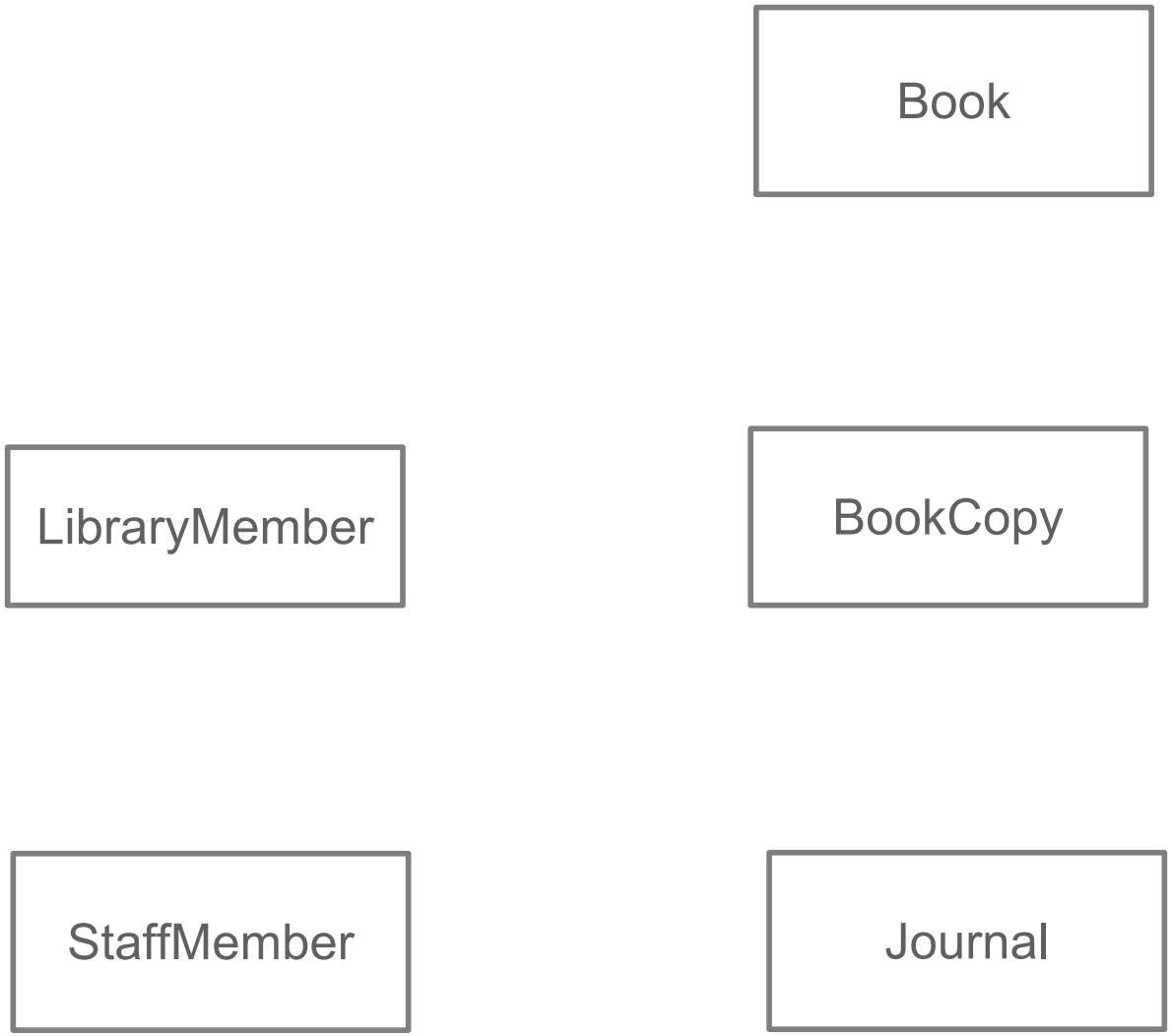
Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

1. Noun Analysis

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

First-Cut Class Diagram: Class Model (Analysis Classes)

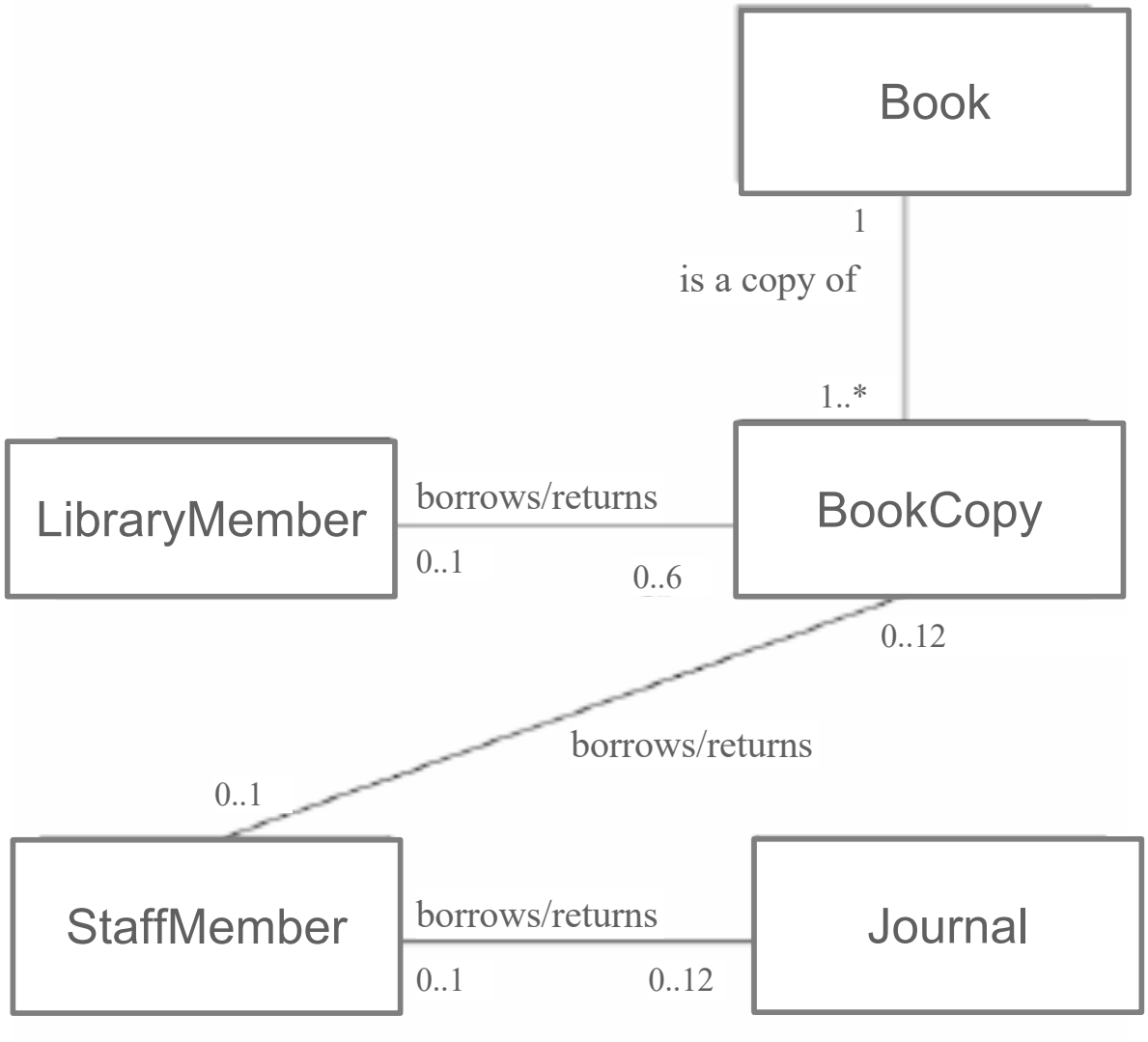


2. Verb Analysis

Books and journals:
The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Borrowing:
The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

First-Cut Class Diagram: Class Model



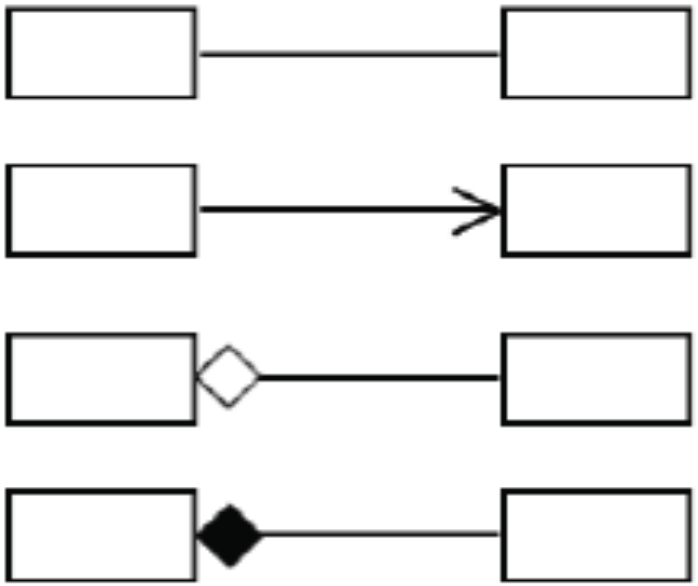
Relationships/Associations

Relationships are connections between modelling elements
Improve understanding of the domain, describing how objects work together
Act as a sanity check for good modelling

Associations are relationships between classes
Examples
- Object of class A sends a message to object of class B
- Object of class A creates an object of class B
- Object of class A has attribute whose values are objects of class B
- Object of class A receives a message with argument of class B

Links are relationships between objects
- Links can be instances of associations (as in UML 1.4)
- Allow one object to invoke operations on another object

UML Relationships Notations



bidirectional / binary

unidirectional

aggregation

composition

Generalization

Inheritance

Composition

Aggregation

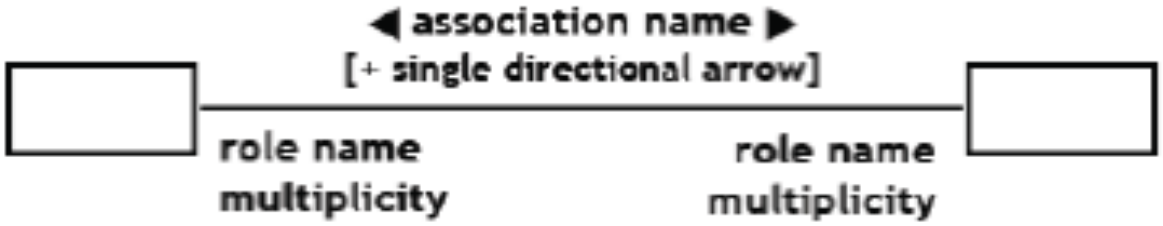
Dependencies

<< >>

Properties

1 *

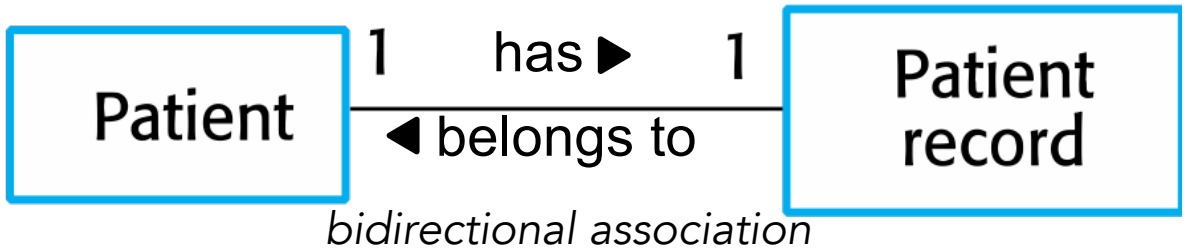
Multiplicity



supplementary characteristics

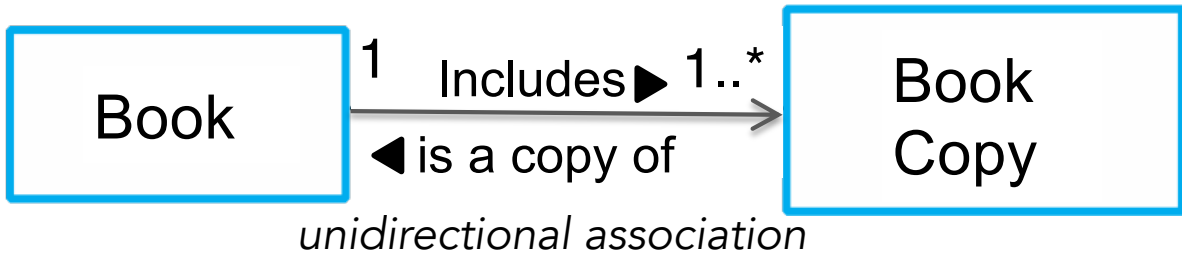
Reference: D. Rosenblum, UCL

UML classes and associations



bidirectional association

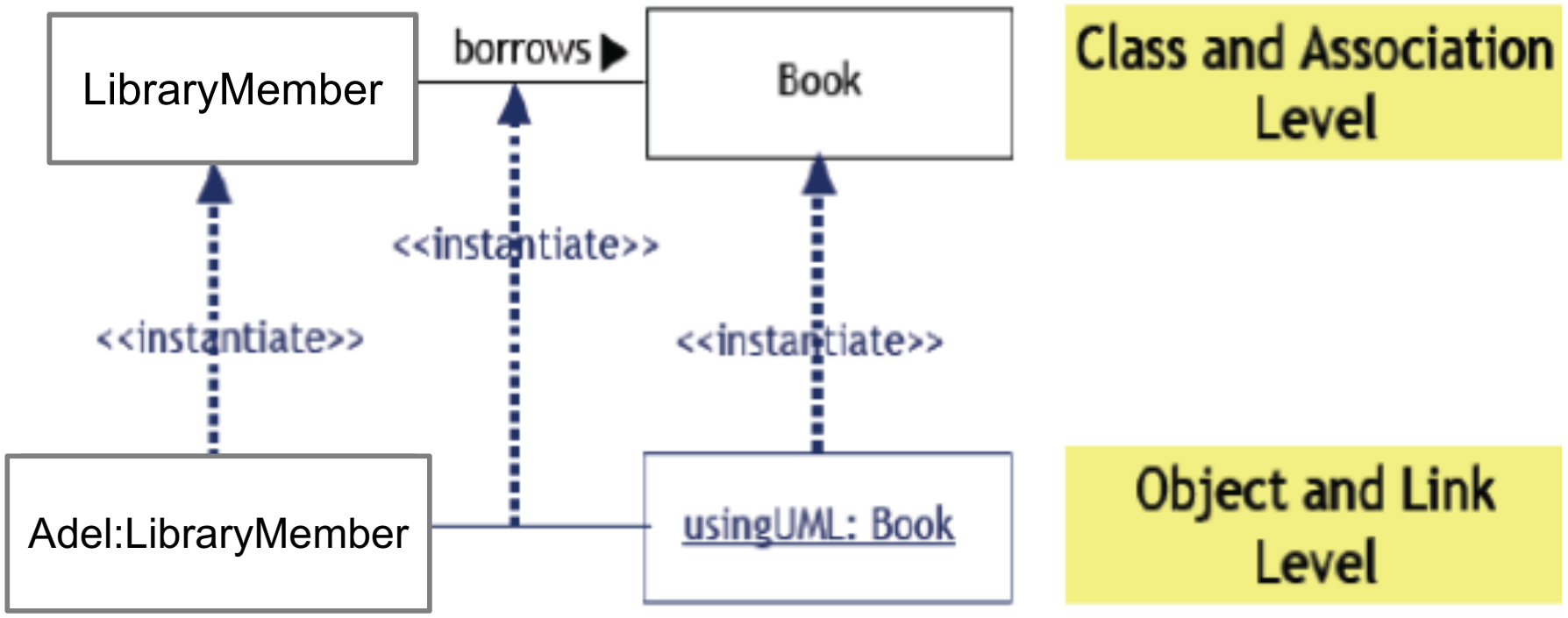
i.e., Both classes are not instances of each other
Access and flow can be from a Patient to a "Patient record" and vice versa



unidirectional association

i.e., a Book contains an instance (or more) of BookCopy
Access and flow from a Book to a "BookCopy"

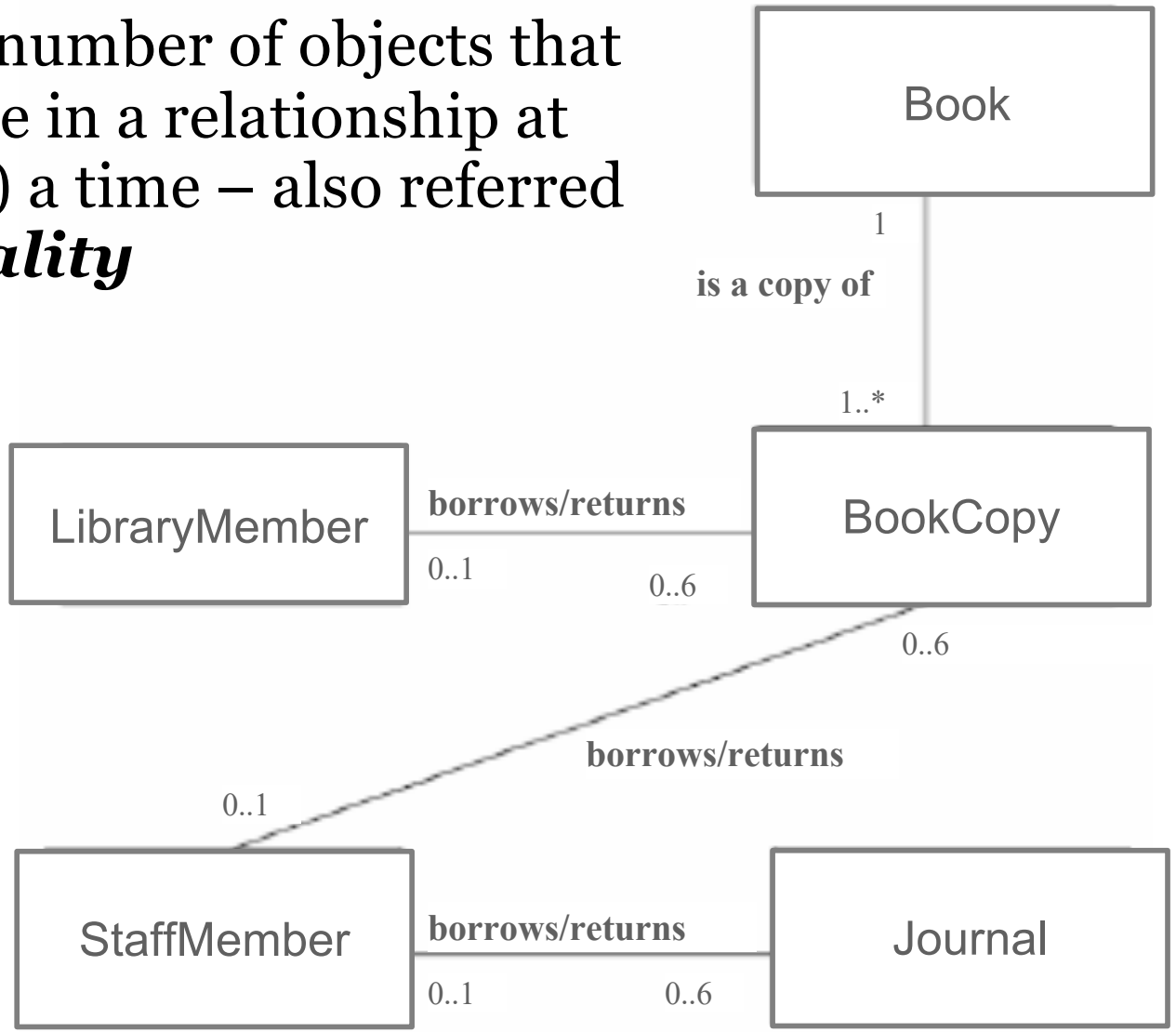
Links Instantiate Associations



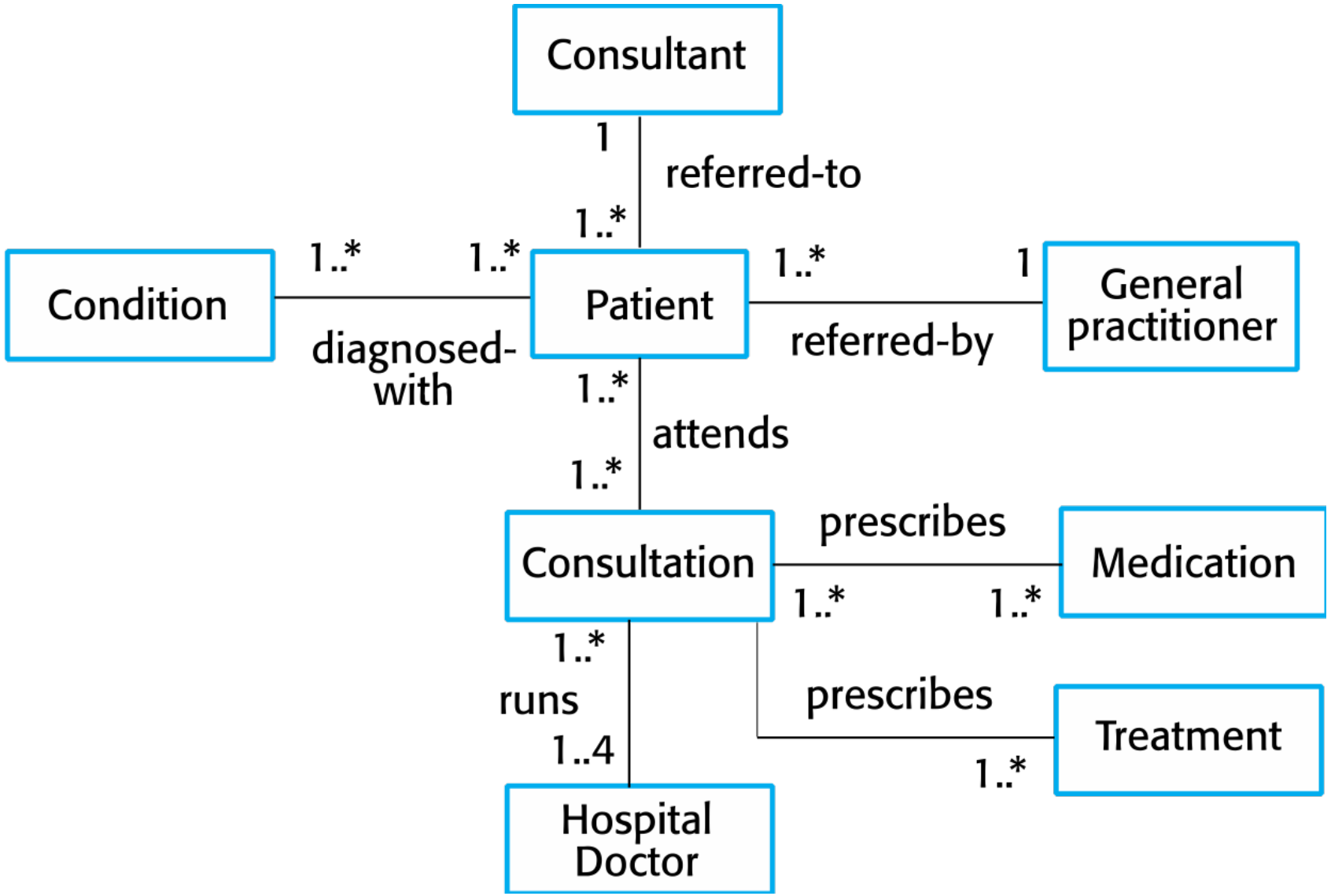
Reference: D. Rosenblum, UCL

Multiplicity of an Association

- Indicates the number of objects that can participate in a relationship at (any point in) a time – also referred to as ***cardinality***

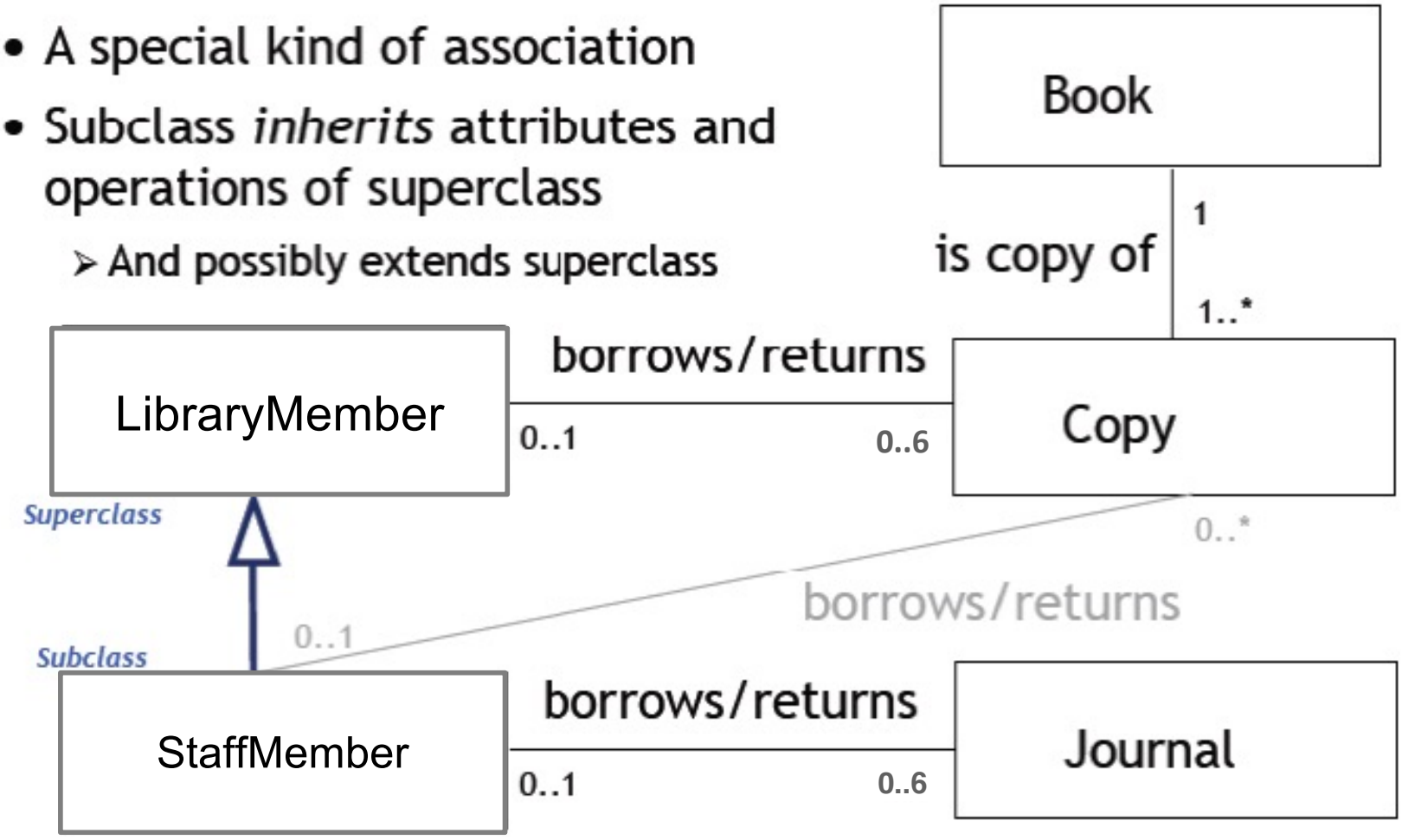


Class diagram/Model of the MHC-PMS



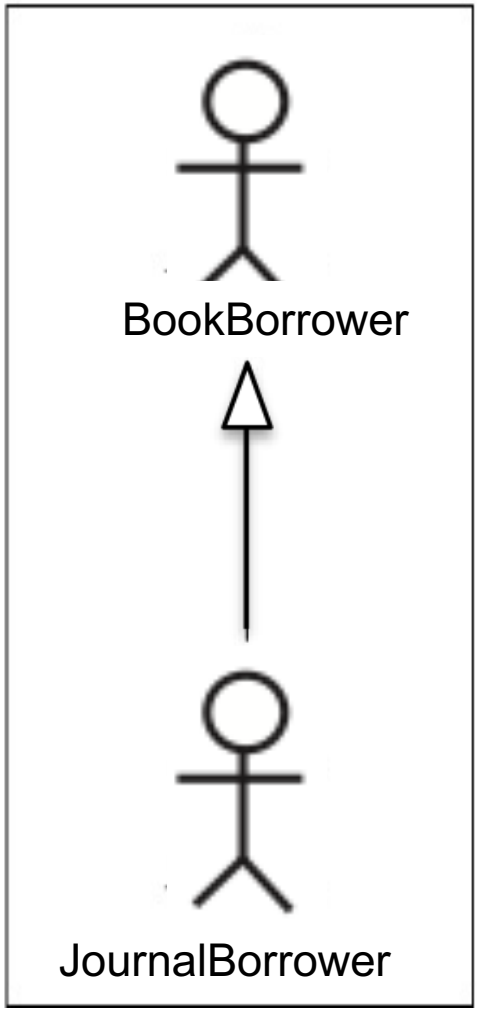
Generalisation (Inheritance)

- A special kind of association
- Subclass *inherits* attributes and operations of superclass
  - And possibly extends superclass

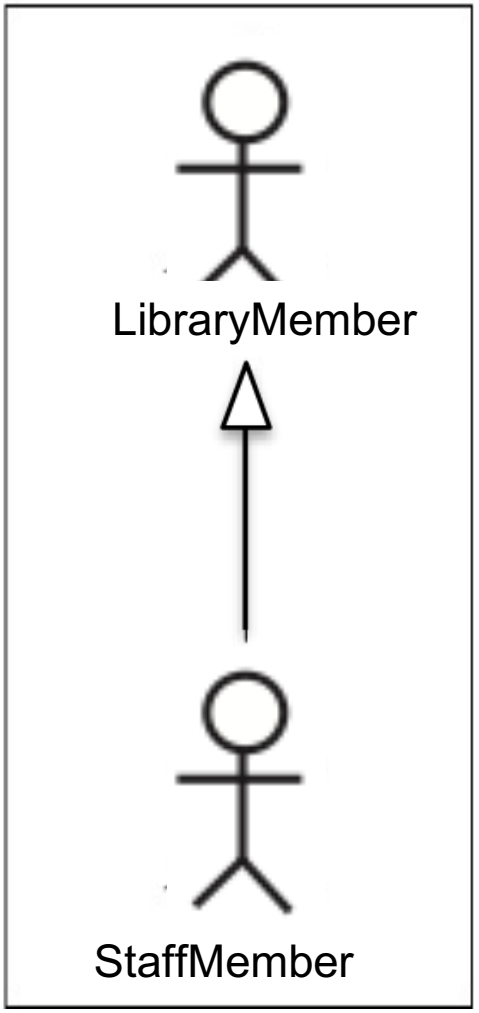


Generalization

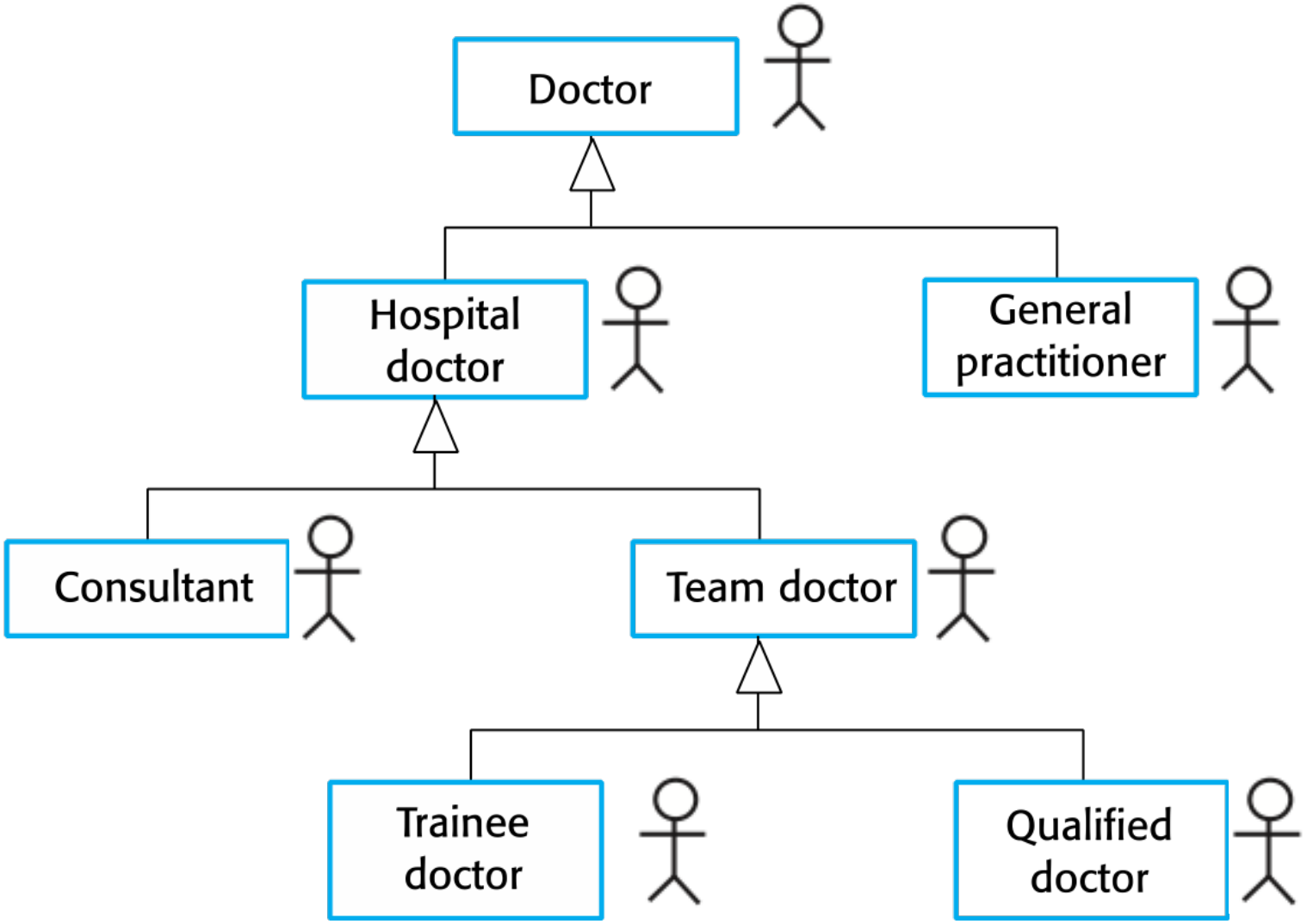
Journal Borrower is a book borrower



Member of Staff is a member of library



A generalization hierarchy



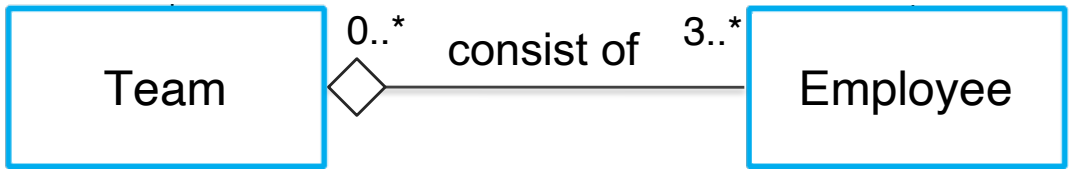
Part/Whole Associations (Aggregation)

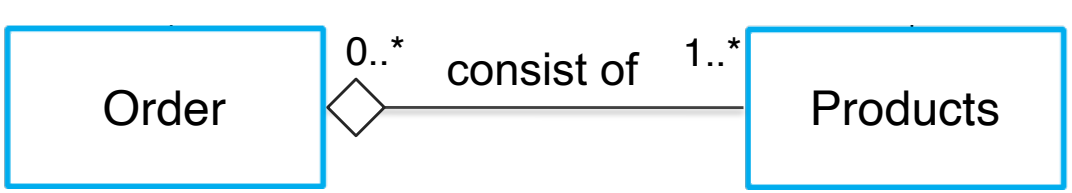
- **Aggregation:** Weak Ownership
  - The part objects can feature simultaneously in any number of other whole objects

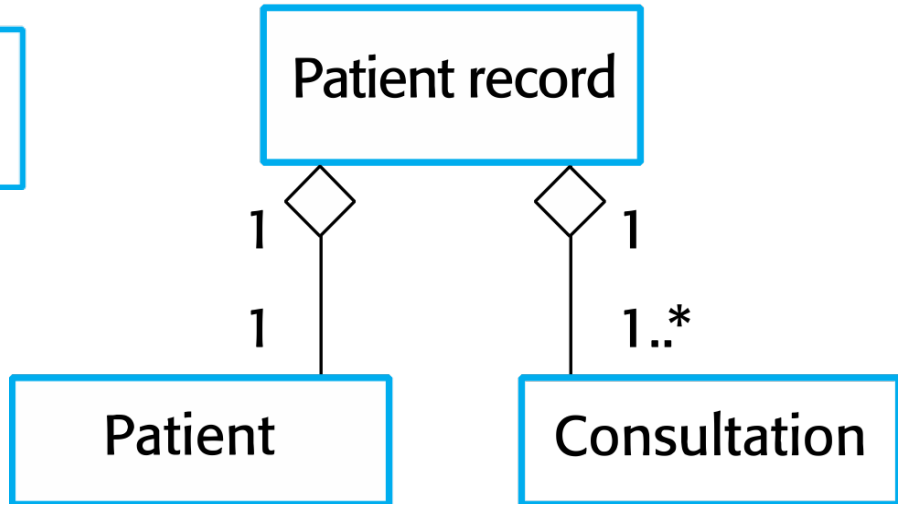
<made-up-of> association
<consist-of> association

a Course is part of a Programme
In fact,
5 or more courses are part of one or more programmes

Aggregation association: Example







Part/Whole Associations: Example

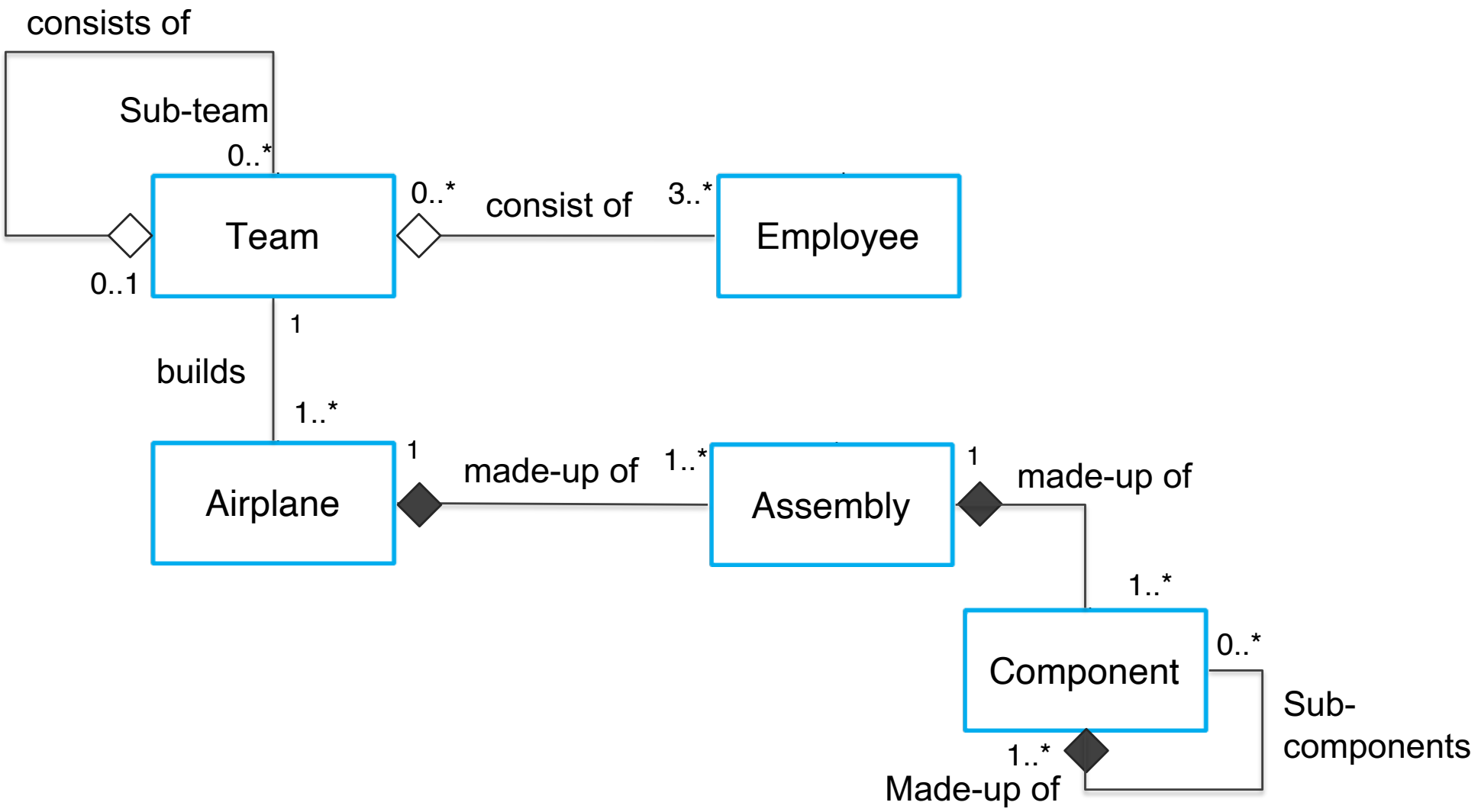
Composed of 64 squares

- **Composition:** Strong Ownership
  - The whole strongly owns its parts, so the parts cannot feature elsewhere

[CheckBoard] is <made-up-of> 64 [Square]

- NOTE: Not all 1-to-* relationships imply ownership

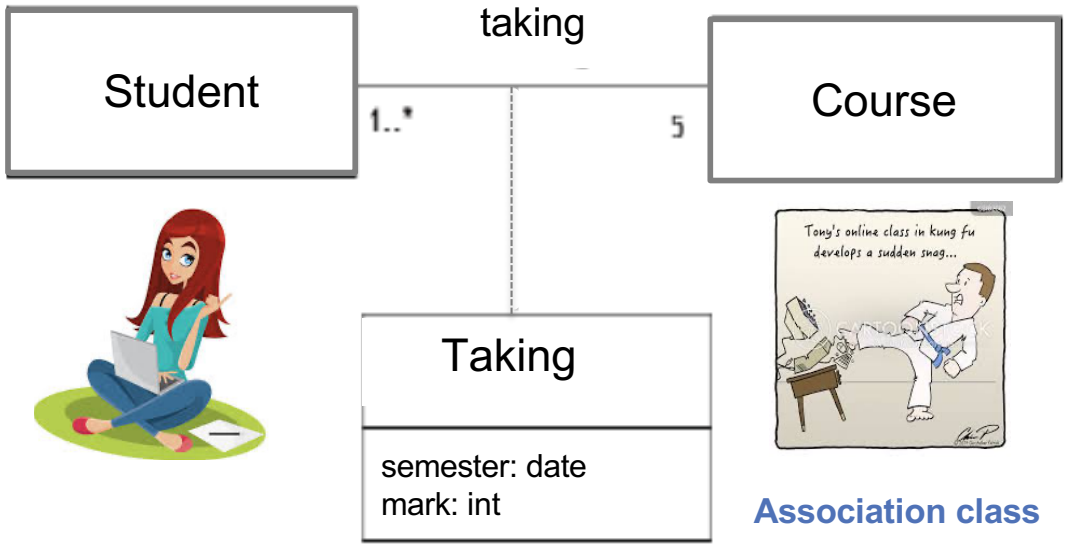
Part/Whole association: Example

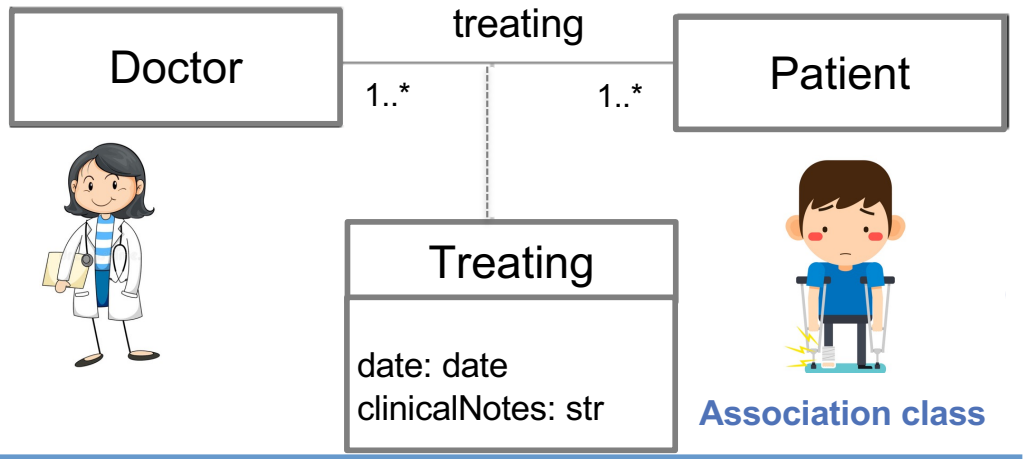


Association Classes

- Used to attach attributes to an association itself rather than the classes themselves

- Class association line must have the same name!

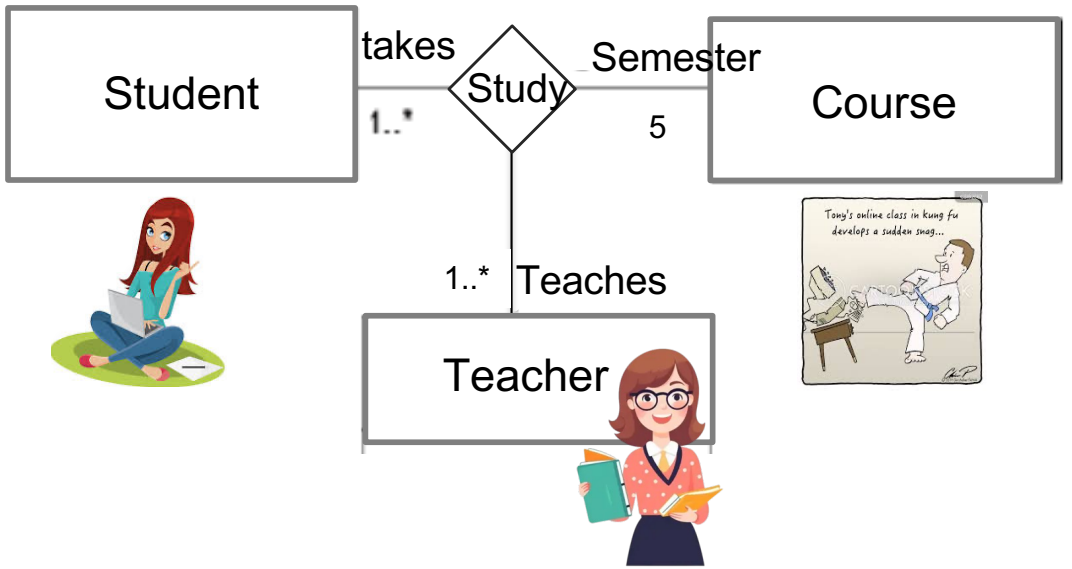


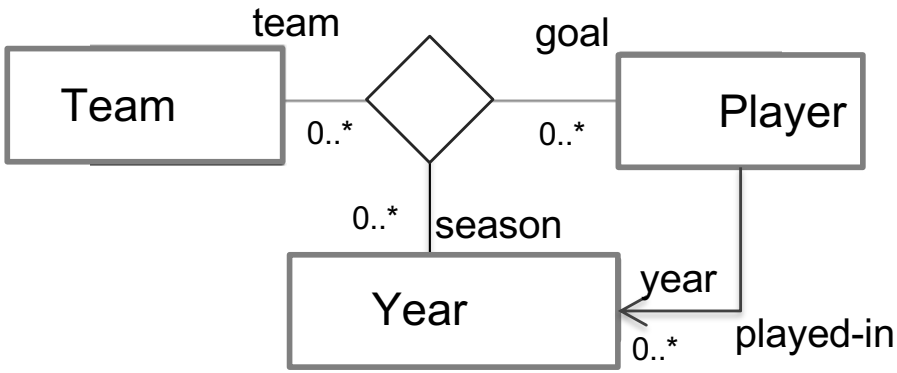


Ternary Association Classes

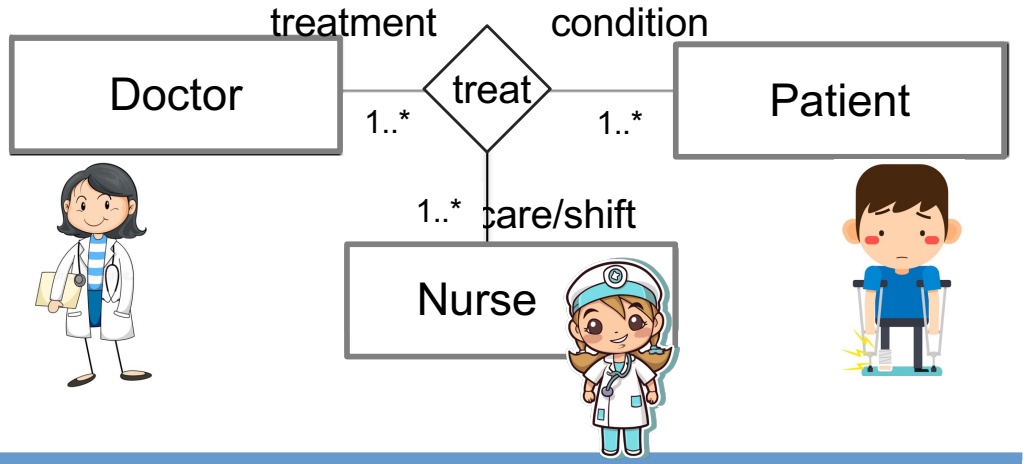
- Three-way association between three entities/classes

-Three objects participate in a ternary association





UMl-OMG-2.5Specs



What Makes a 'Good' Analysis Class..

Its name reflects its intent

It is a crisp abstraction that models one specific element of the problem domain

It has a small but defined set of responsibilities

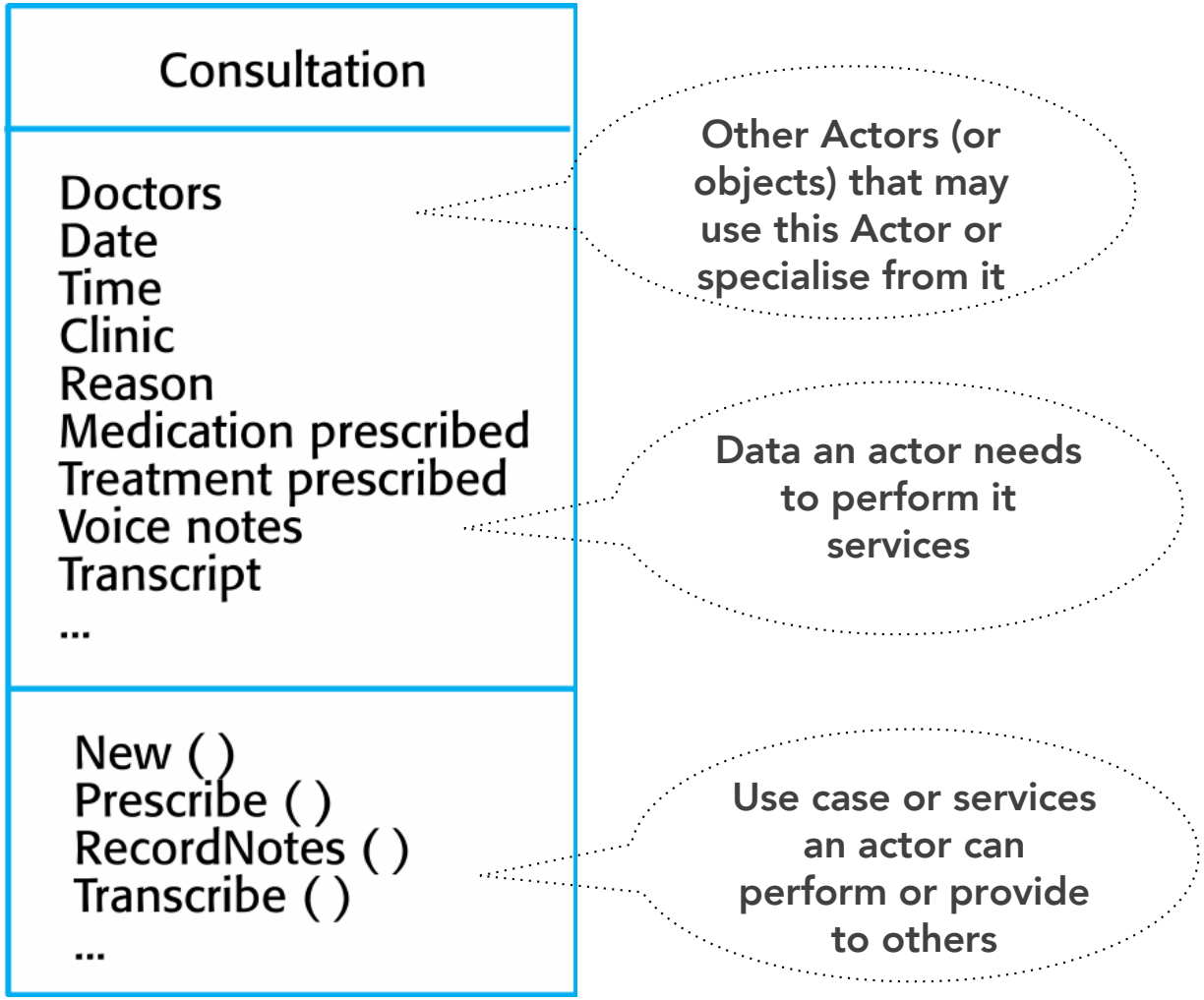
It has high *cohesion*

[i.e., a class has all related necessary attributes and methods or operations (unrelated/unnecessary ones decrease cohesion)]

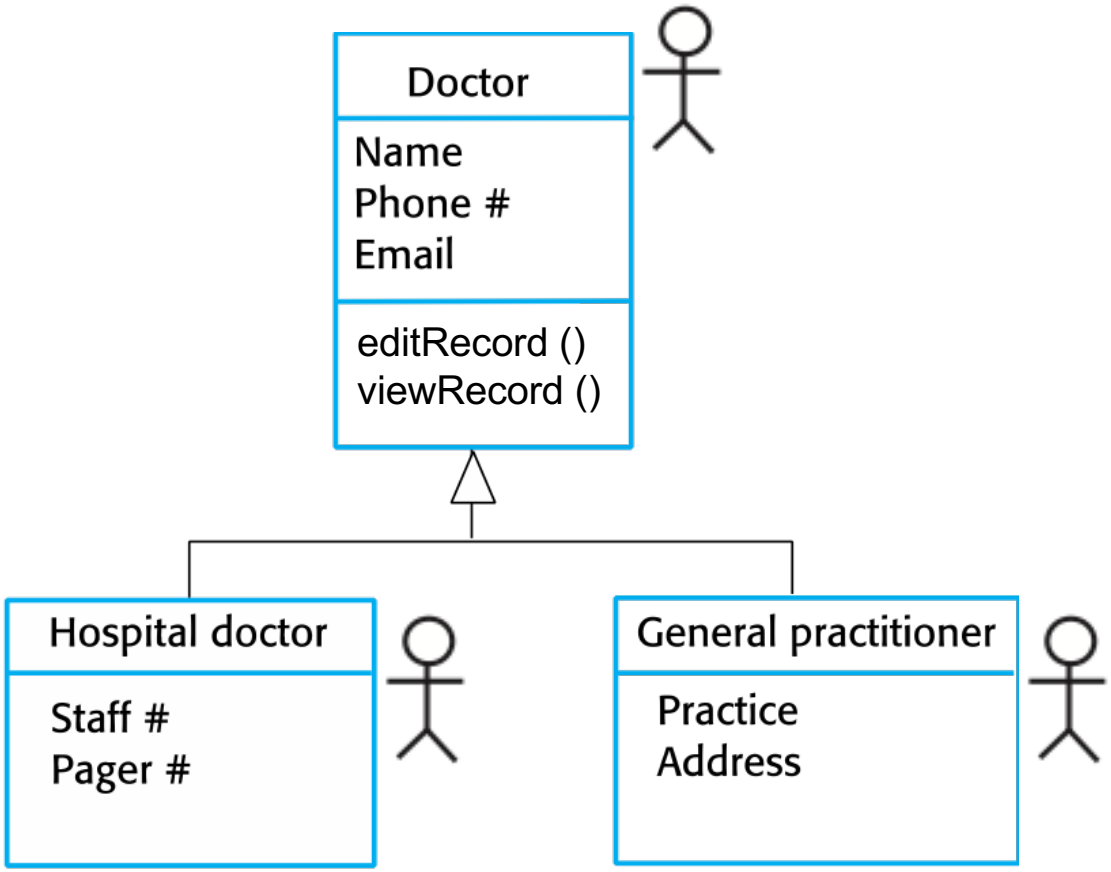
It has low *coupling* with other classes:

[i.e., a class has only the necessary associations with other classes (unnecessary associations increases coupling)]

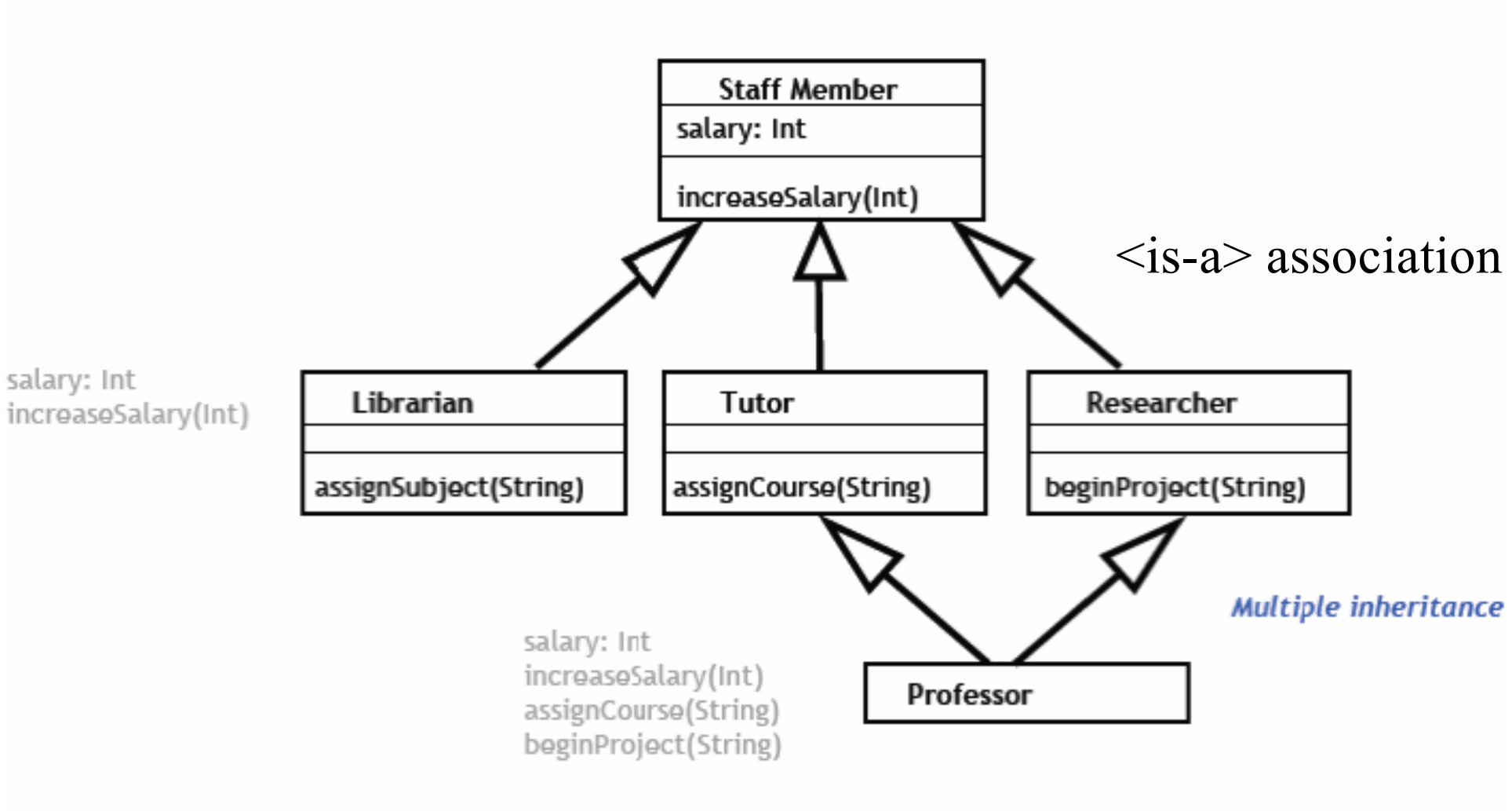
Complete class Description



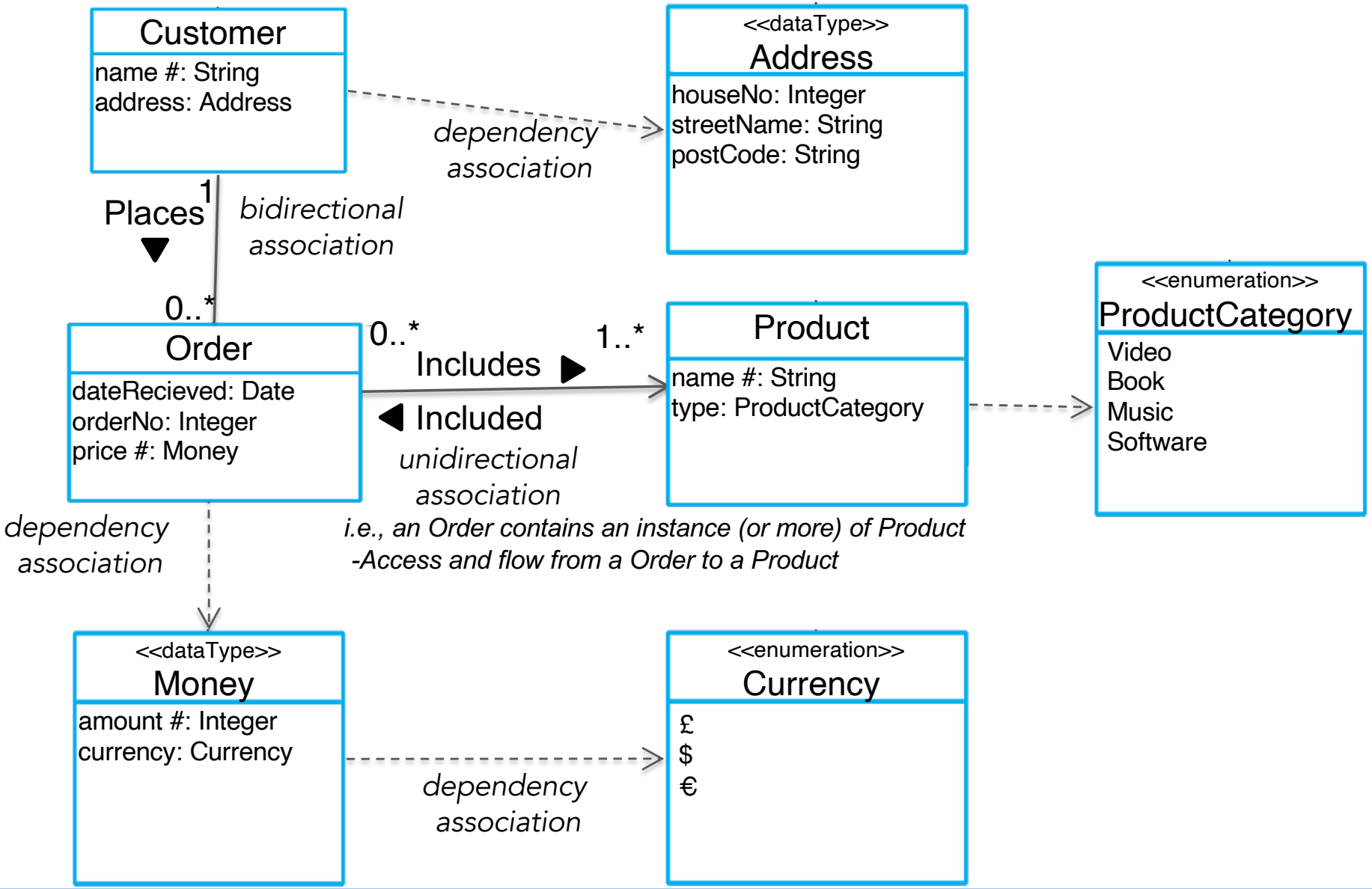
A generalization hierarchy: Details



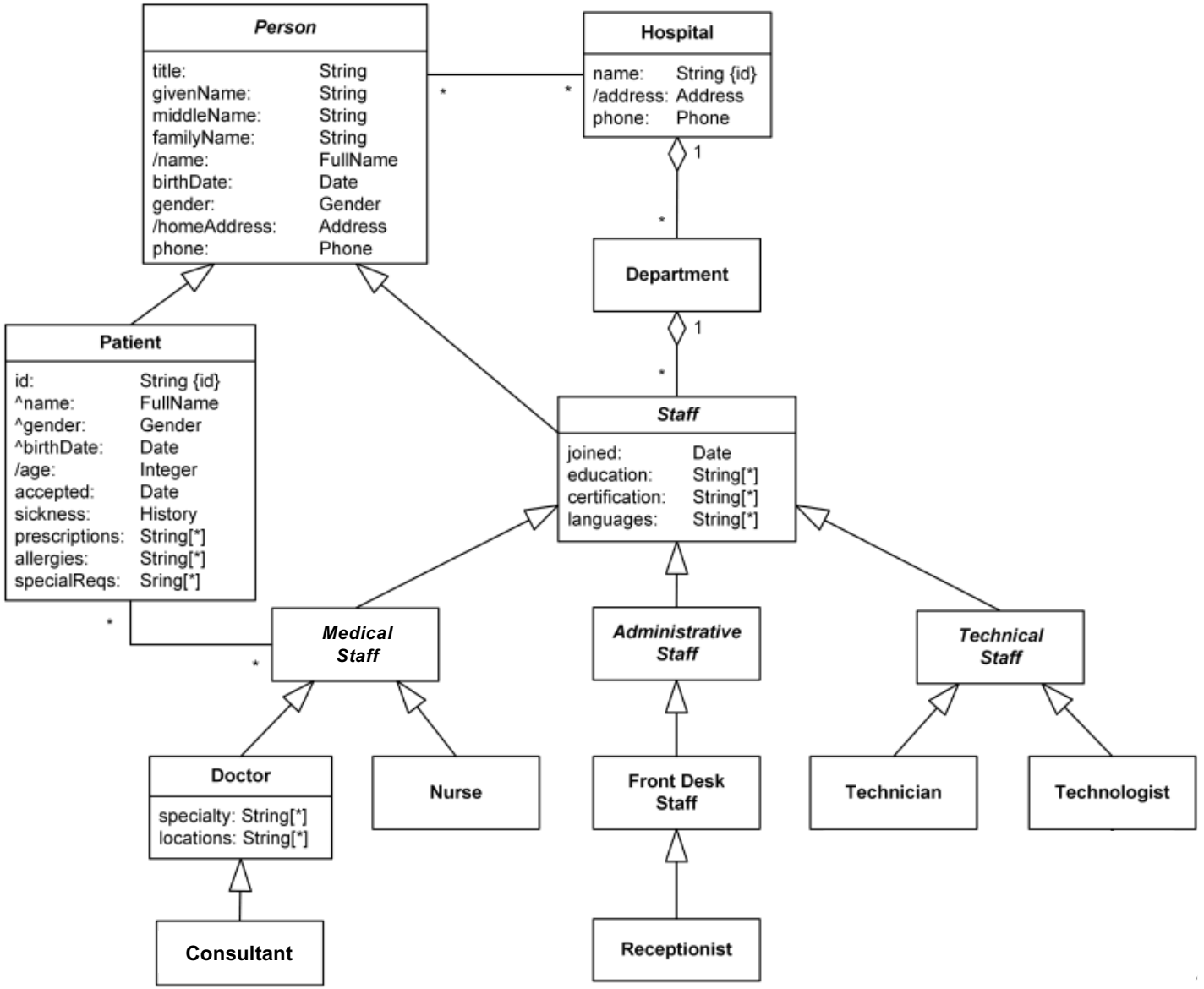
Another Generalisation Example



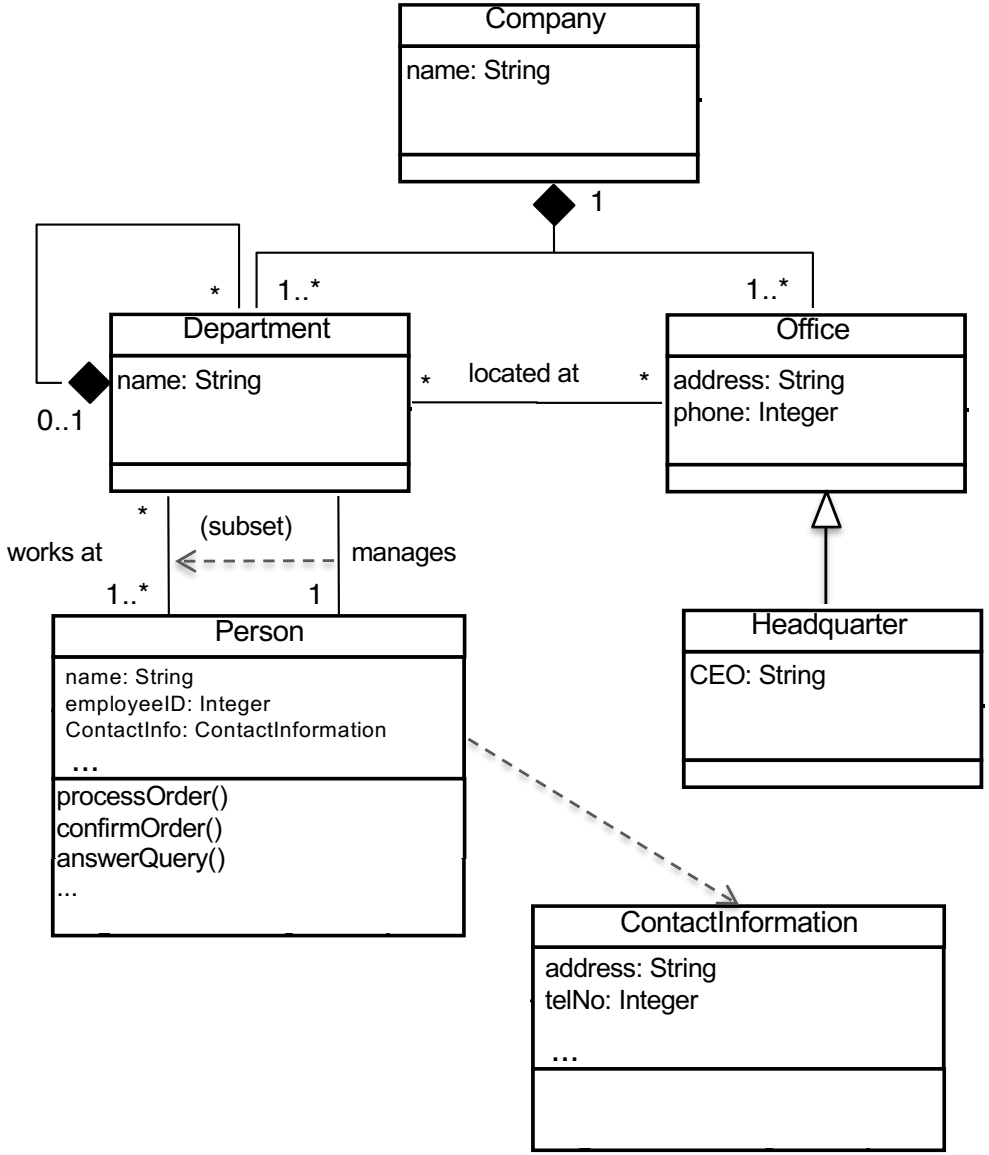
UML classes and associations



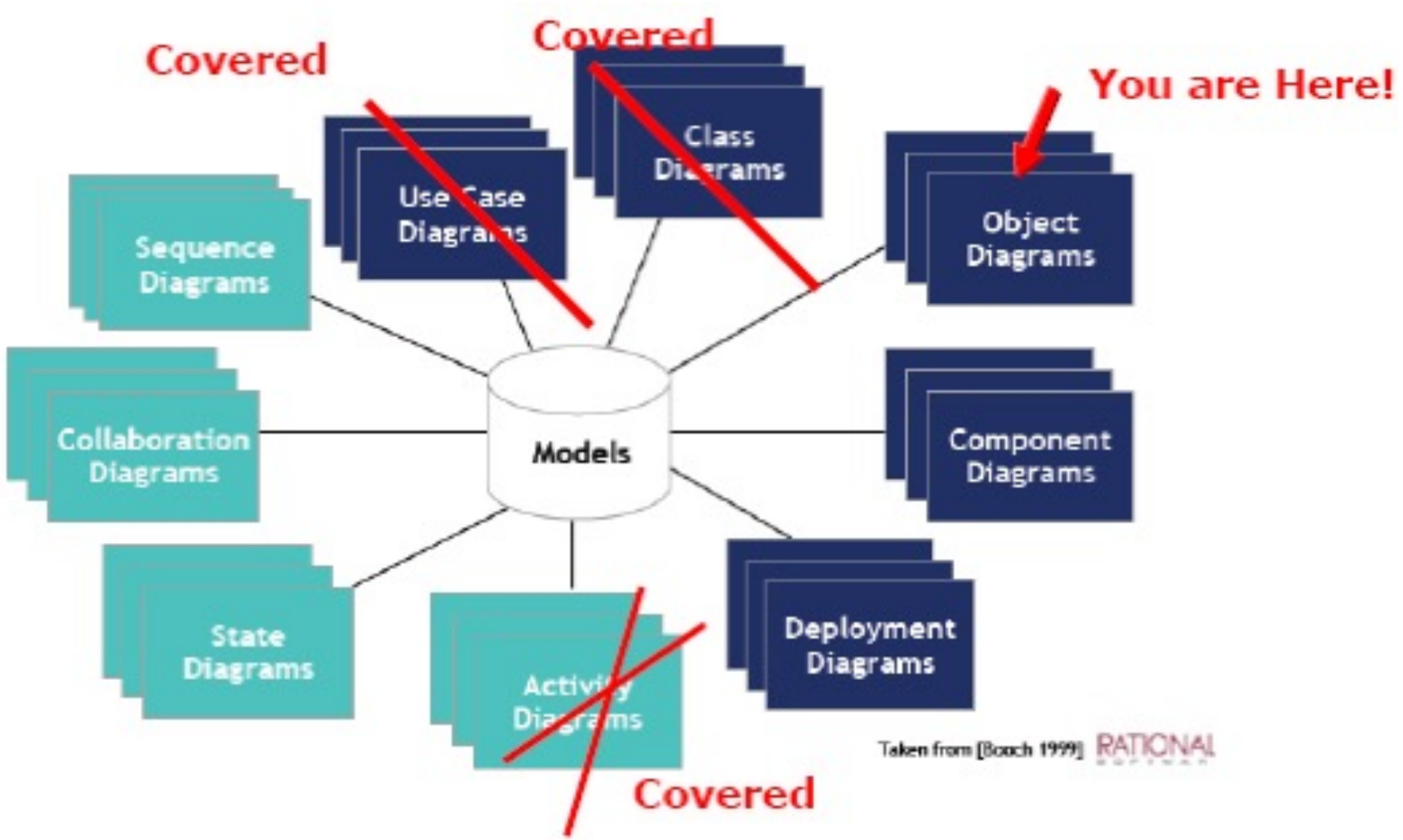
Another Example: Hospital



Example: Detailed Class Diagram



UML Diagrams



Object Diagram

Objects are instances of Classes

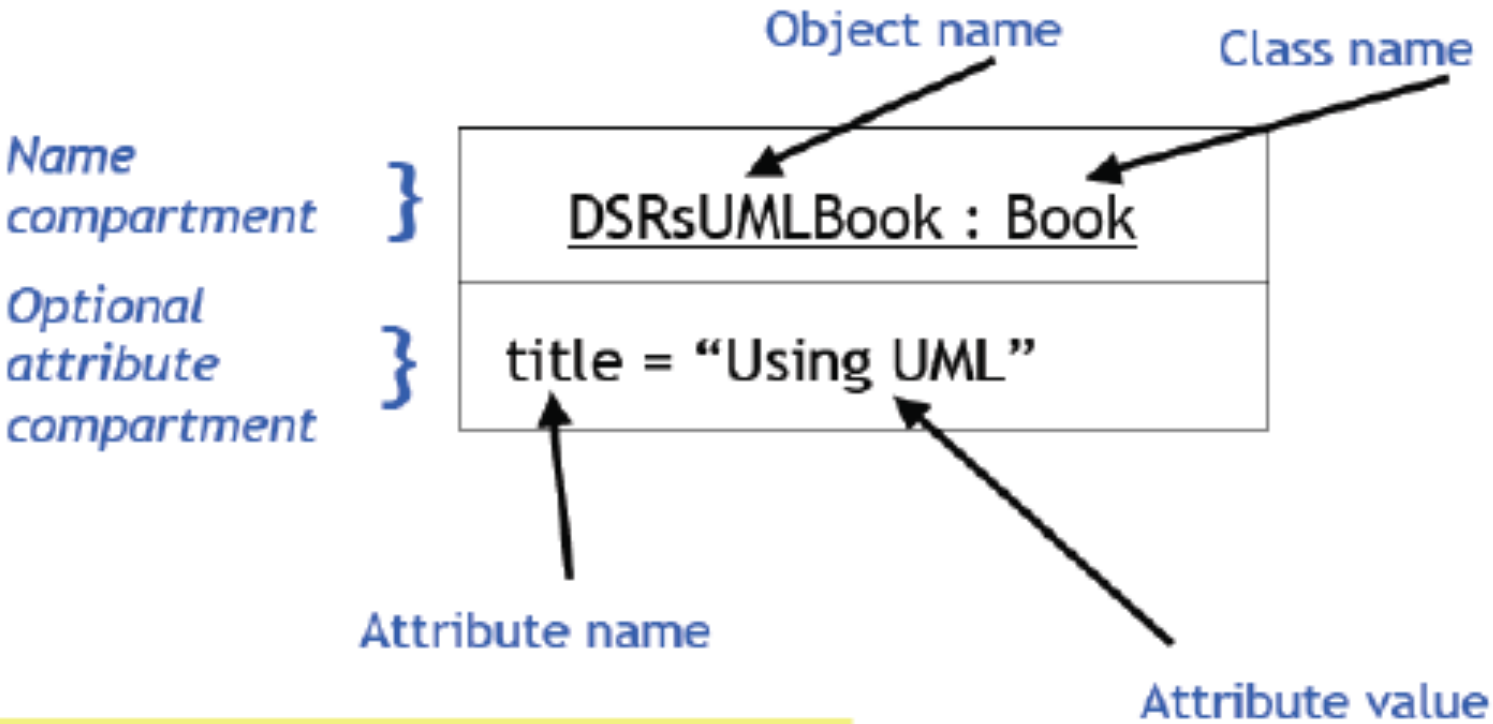
Object Diagram captures objects and relationships between them, in other words, it captures instances of Classes and links/associations between them.

Built during analysis & design

- Illustrate data/object structures
- Specify snapshots
- Validates Class Model, is it sufficient for persistence of data elements and methods.

Developed by analysts, designers and implementers

UML Object Icons

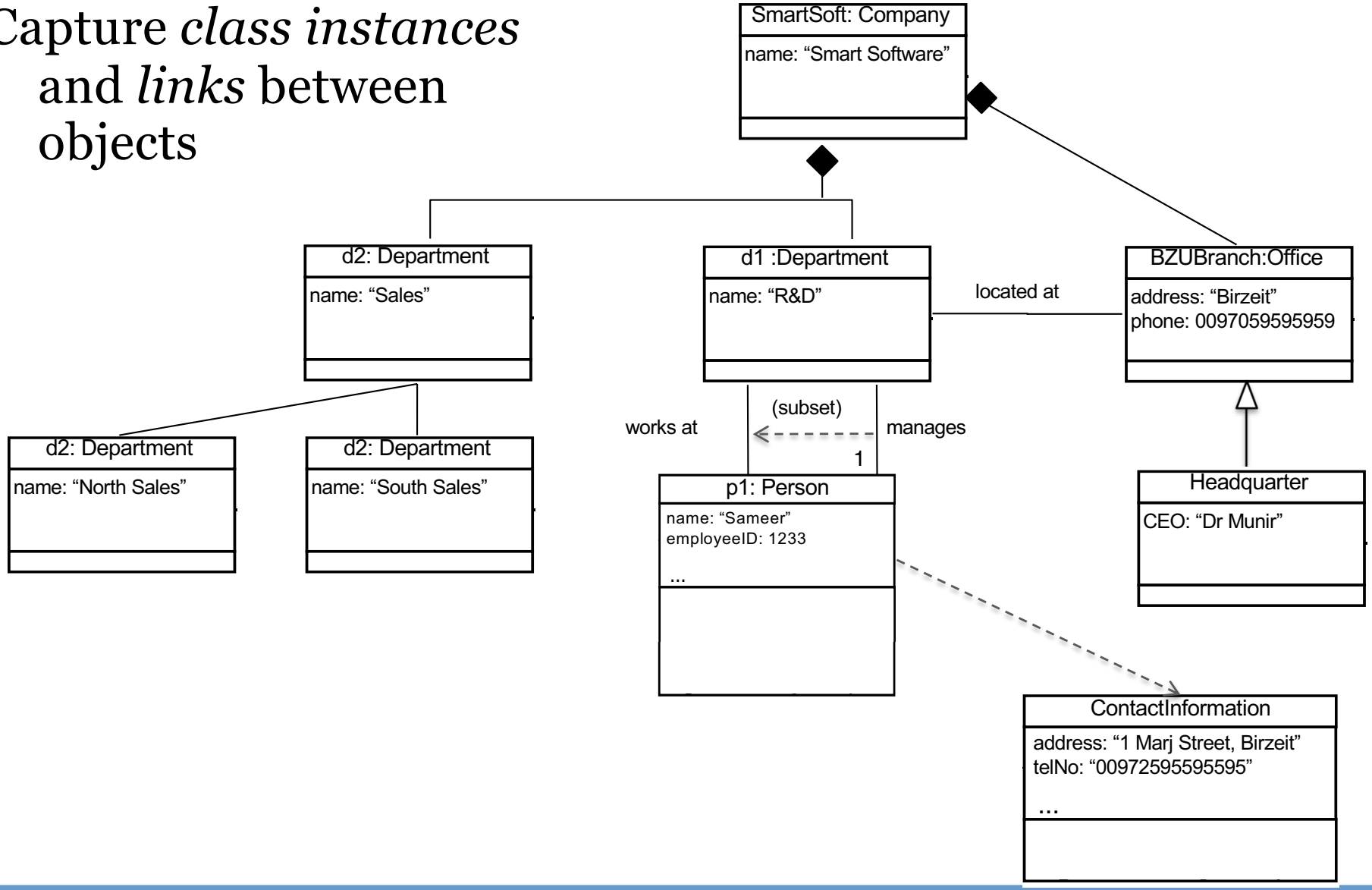


Operations and attribute types are *not* shown on object diagrams!

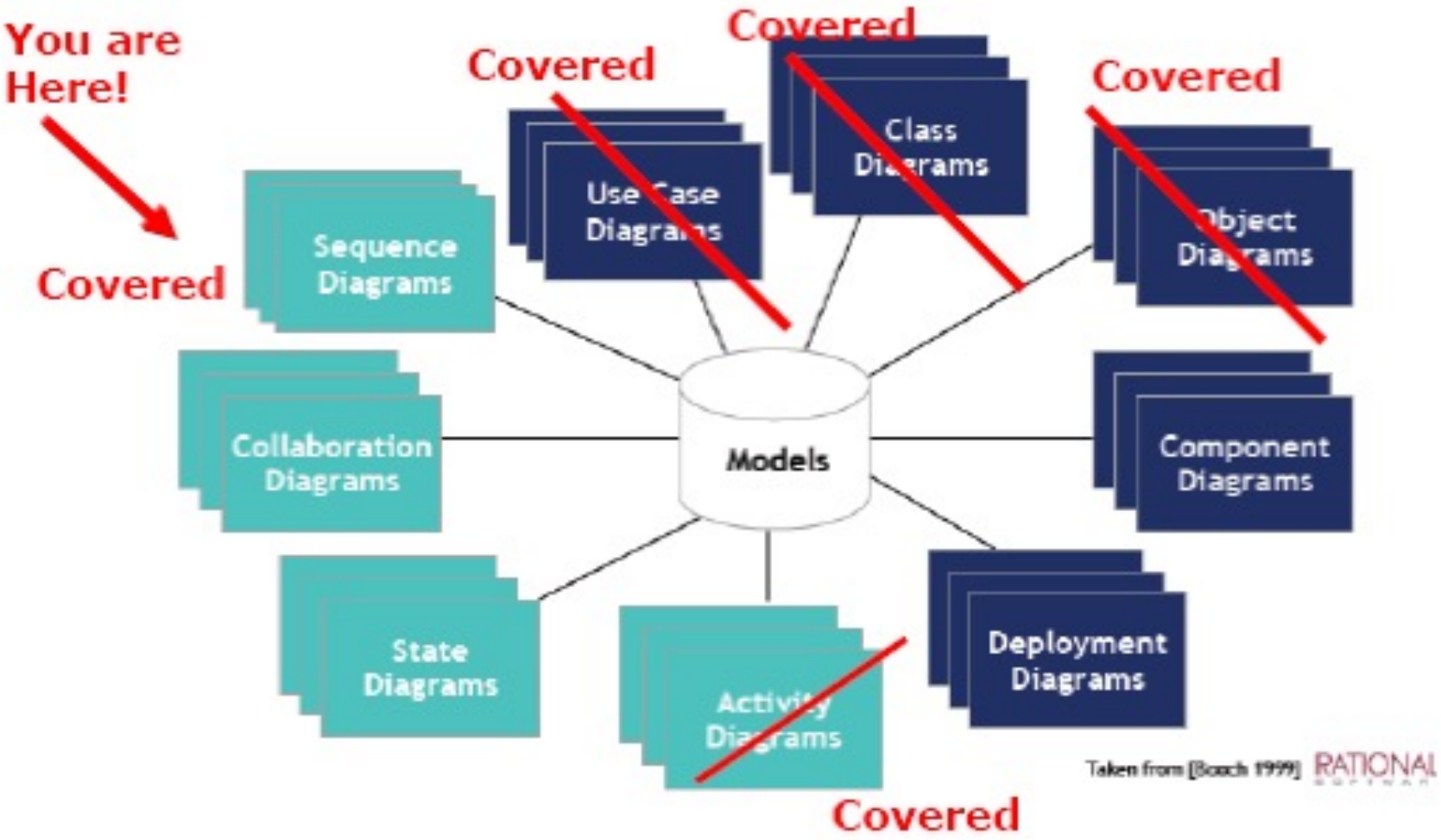
Reference: D. Rosenblum, UCL

Object Diagram

Capture *class instances* and *links* between objects



UML Diagrams



Sequence diagrams

Sequence diagrams are used to model the interactions between the actors and the objects within a system, with a time-oriented view.

A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.

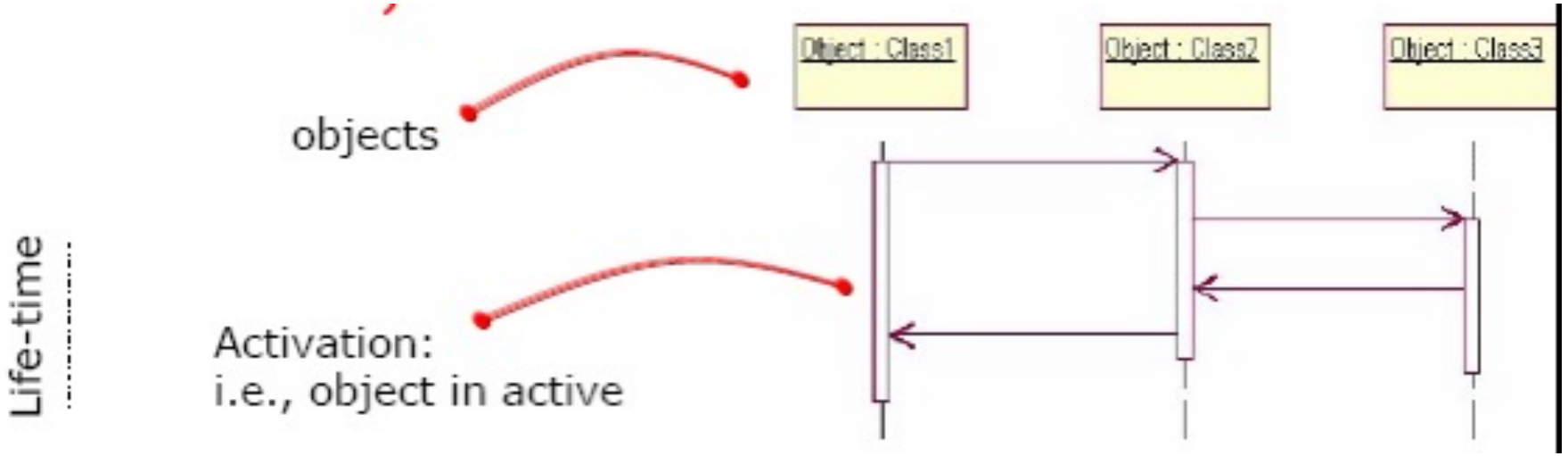
The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.

Interactions between objects are indicated by annotated arrows.

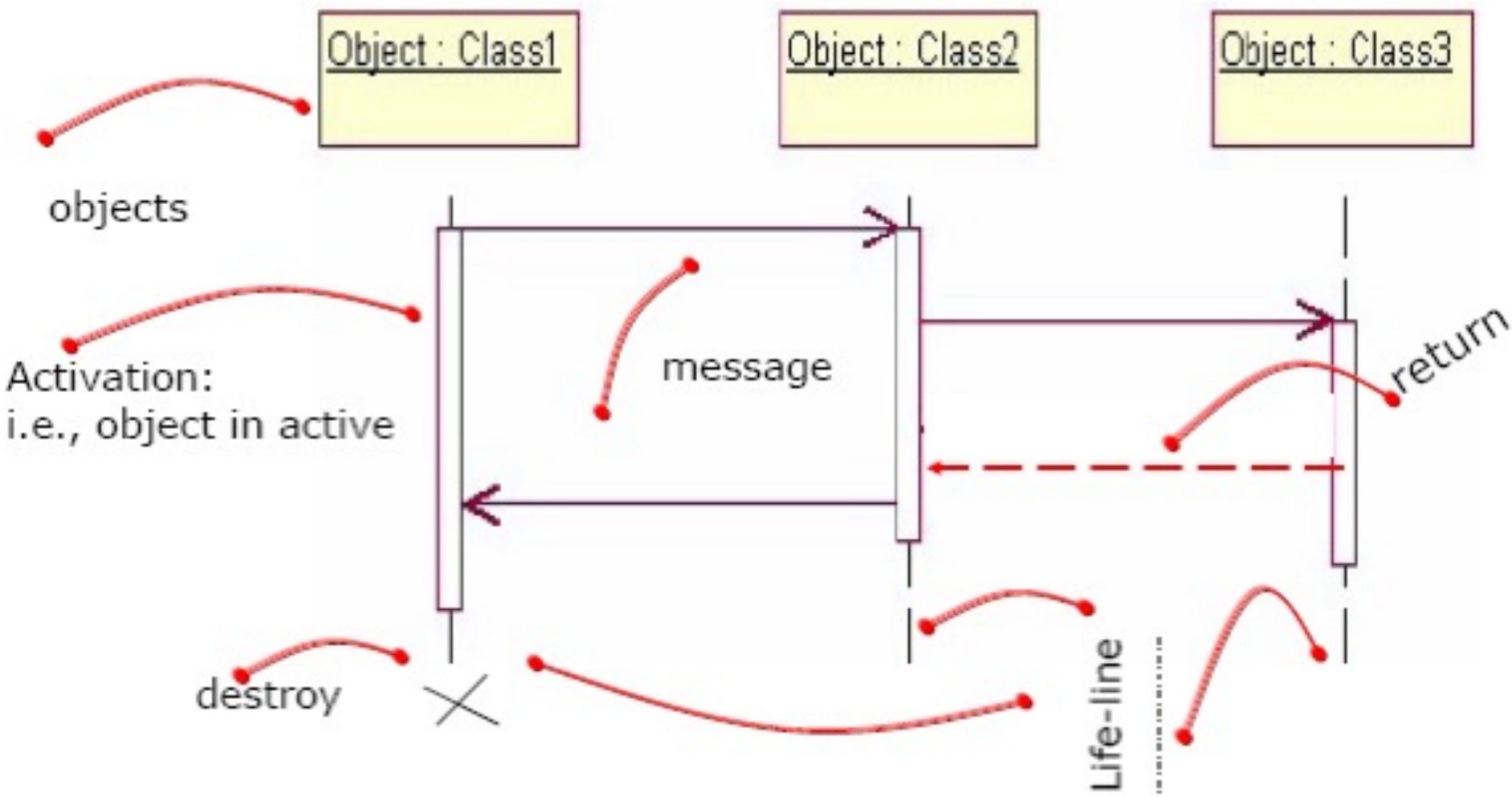
Sequence diagrams

Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass. the diagrams are read left to right and descending.

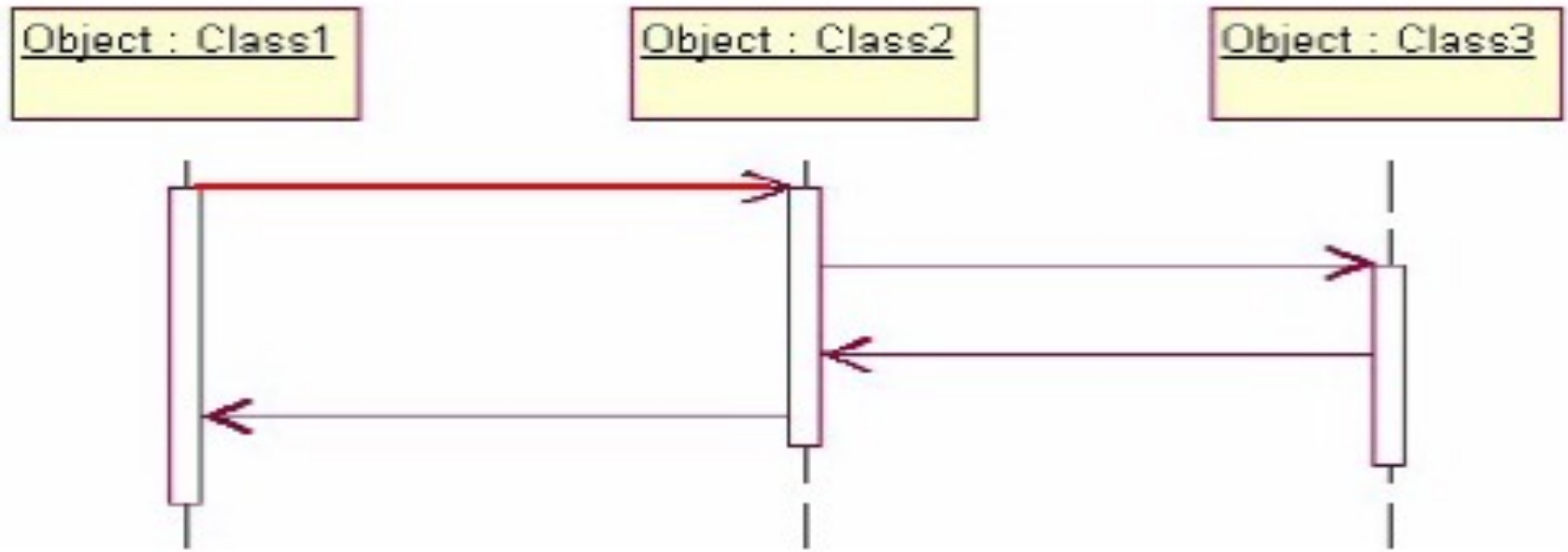
Object interactions are arranged in a time sequence (i.e. time-oriented)



Sequence diagrams



Sequence diagrams



The example shows an object of class 1 start the behaviour by sending a message to an object of class 2. Messages pass between the different objects until the object of class 1 receives the final message

Example

In a self-service, e.g. money (e.g. ATM), machine, three objects do the work we're concerned with:

the front: the interface the self-service machine presents to the customer

the money register: part of the machine where money is collected

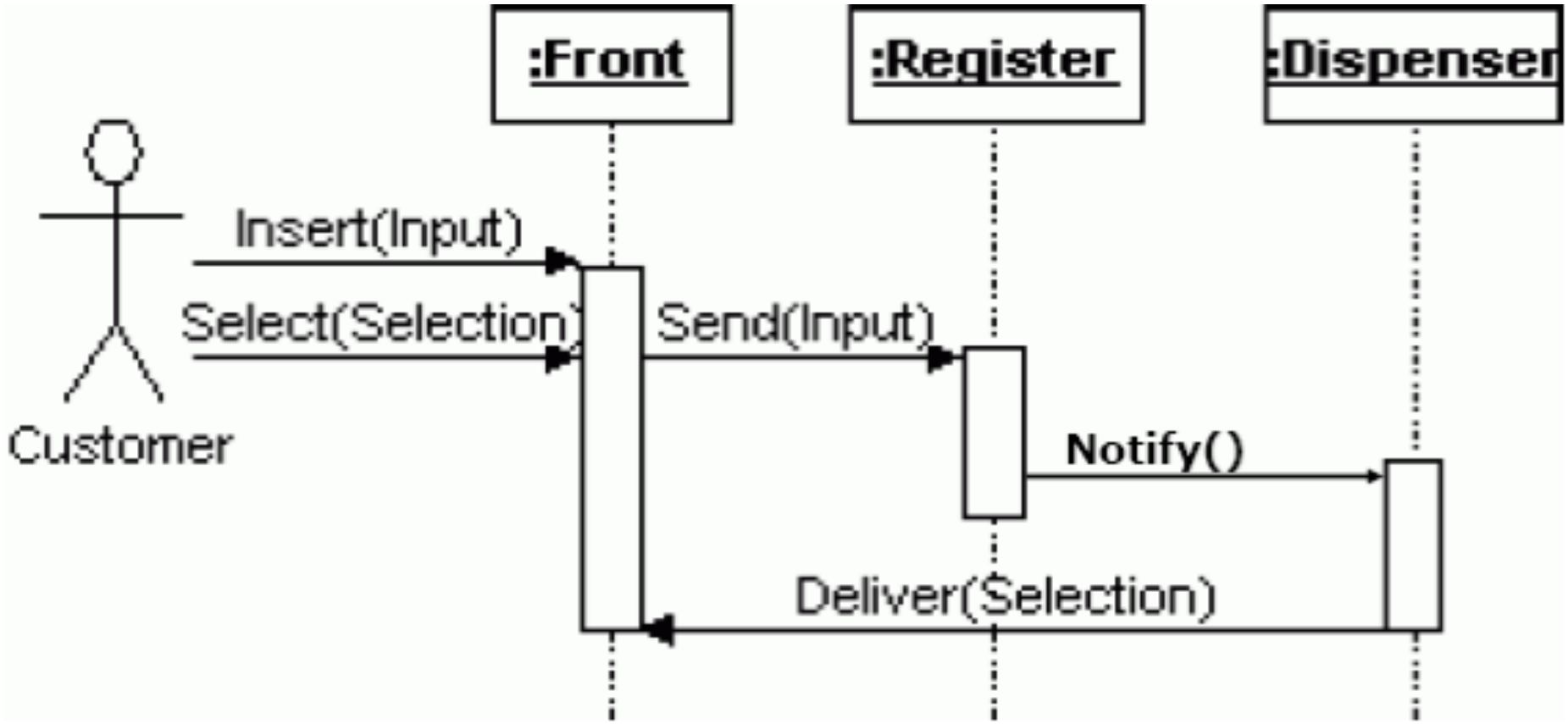
the dispenser: which delivers the selected product to the customer

Example

The instance sequence diagram may be sketched by using this sequence:

1. The **customer** inserts money in the money slot in **front** money collector.
2. The customer makes a selection on the **front** UI (e.g., selects which account to deposit the money in and enters the money amount deposited)
3. The money is sent to the **register**
   - The **register** checks to see whether the correct money is in the money **collector/dispenser**
   - The **register** updates its cash reserve
4. The **register** notifies the **dispenser**.
5. The **dispenser** delivers the product (e.g. receipt) to the front of the machine

Example



The “Deposit Cash” use case.
This is the normal scenario/flow for the use case, alternative or error are not shown

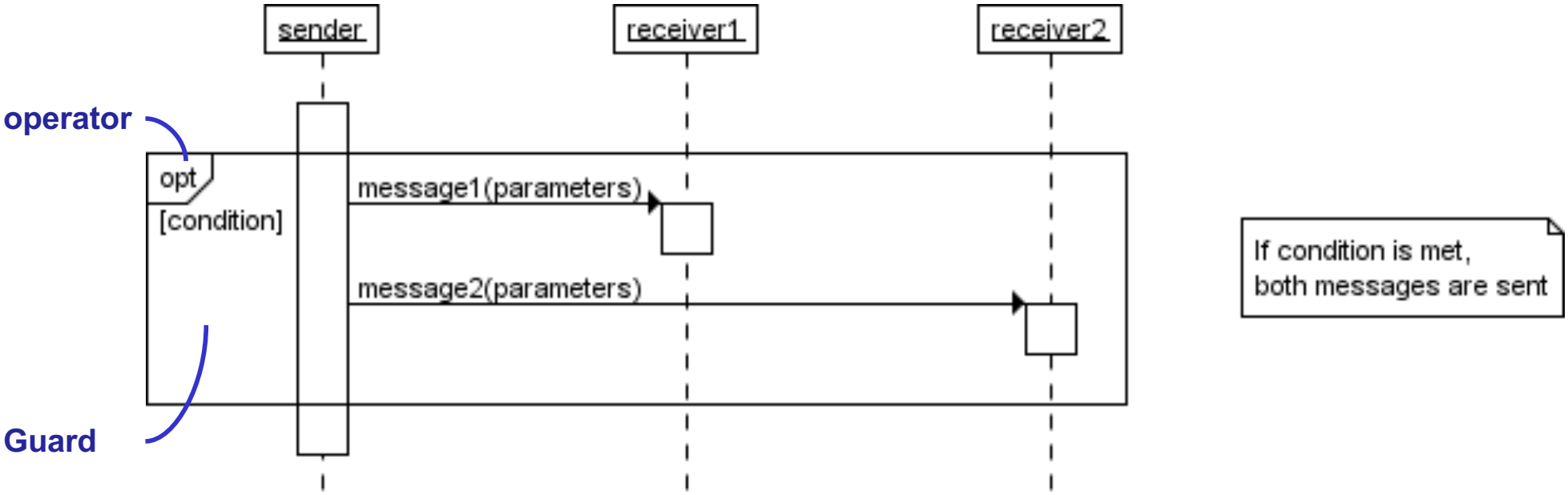
However, note…

We have seen an instance of an interaction diagram- i.e. one possible sequence of messages

Since a use case can include many scenarios
- There is a need to show conditional behaviour
- There is a need to show possible iterations

A generic interaction diagram shows all possible sequences of messages that can occur

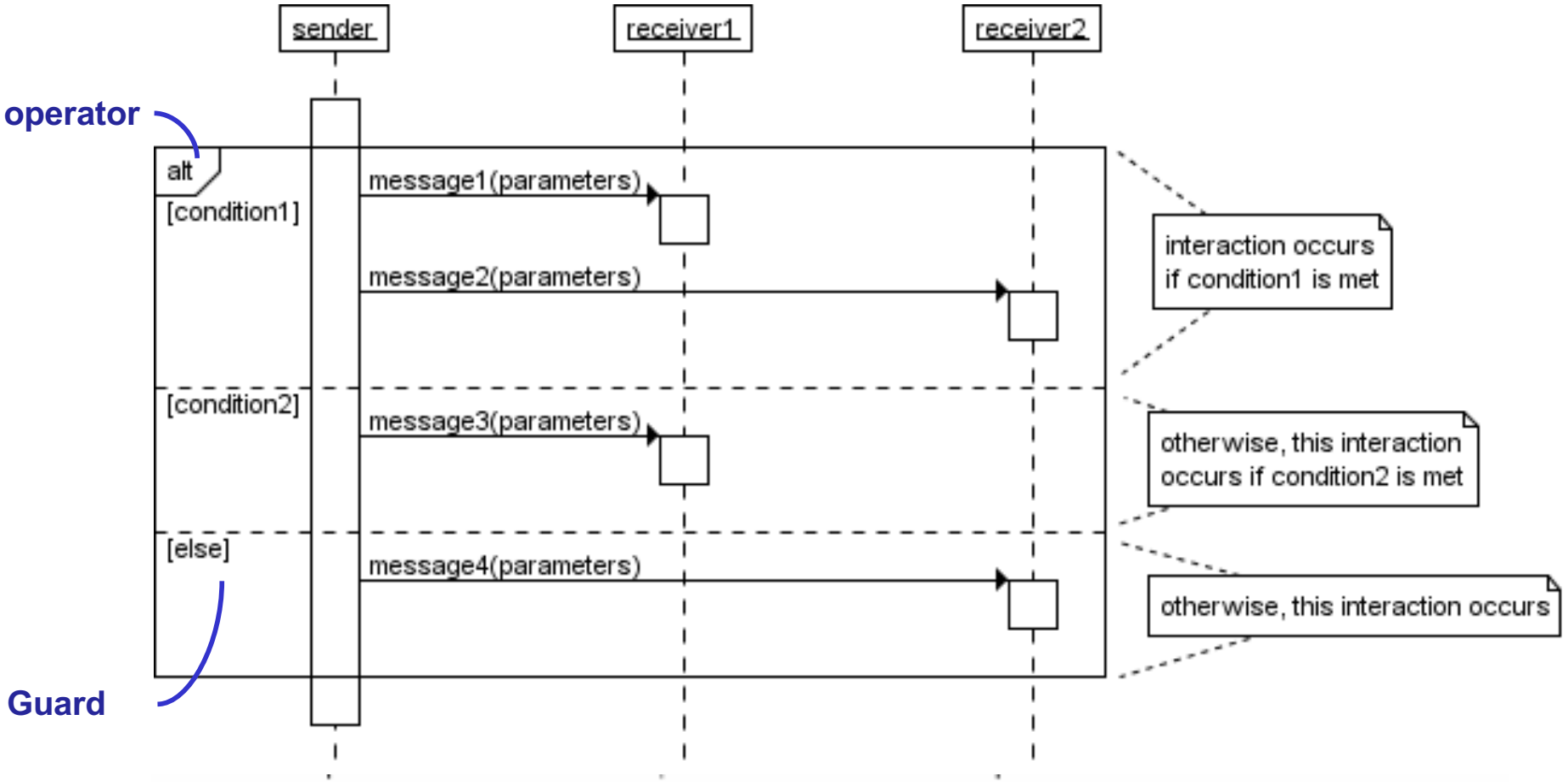
Opt(ional) in UML 2.0



Opt: Optional; the fragment executes only if the supplied condition is true.

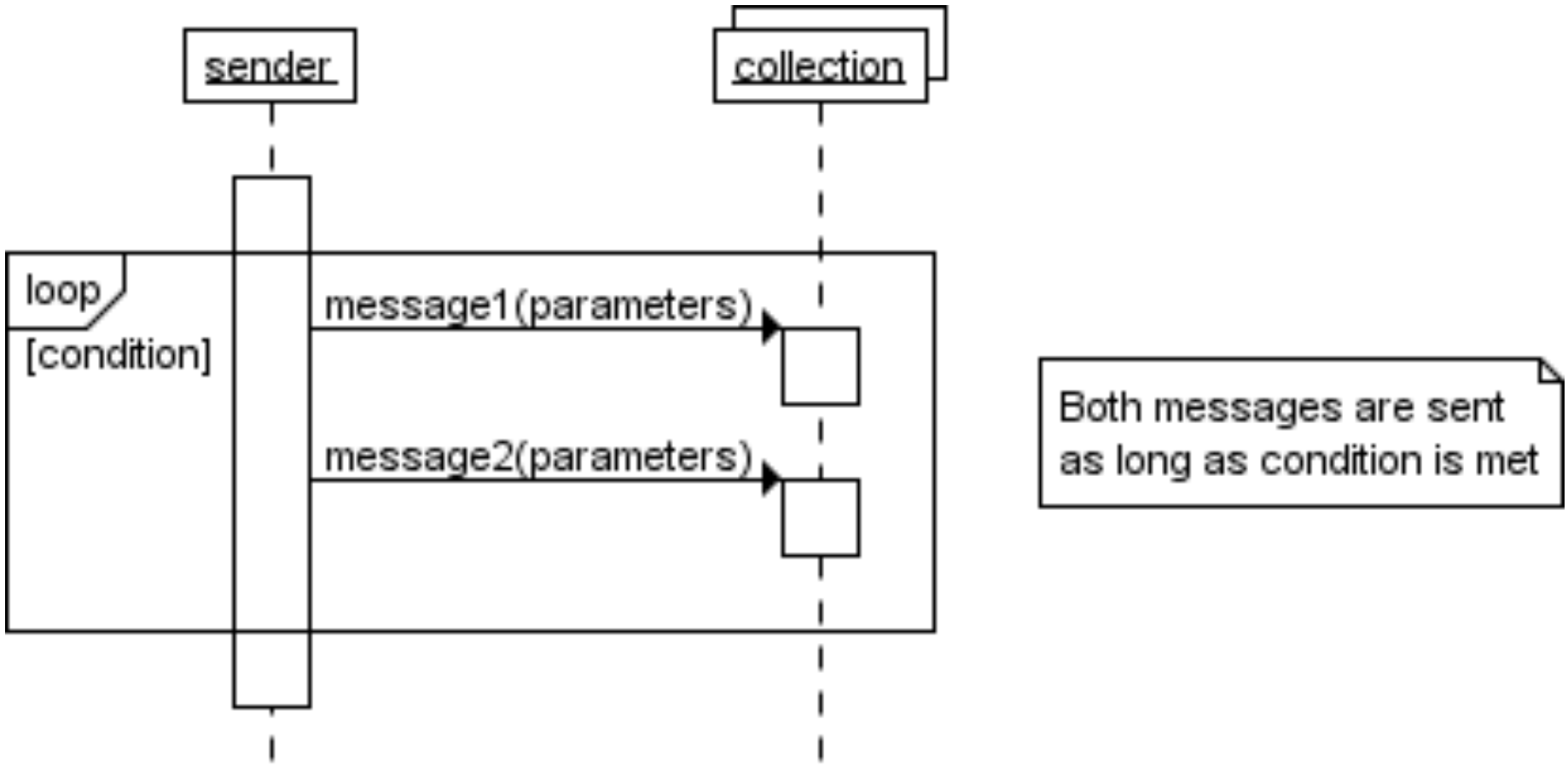
This is equivalent to an "**alt**" with one trace (next slide)

alt(ernative): Operators in interactions frames – UML 2.0



Alternative multiple fragment: only the one whose condition is true will execute

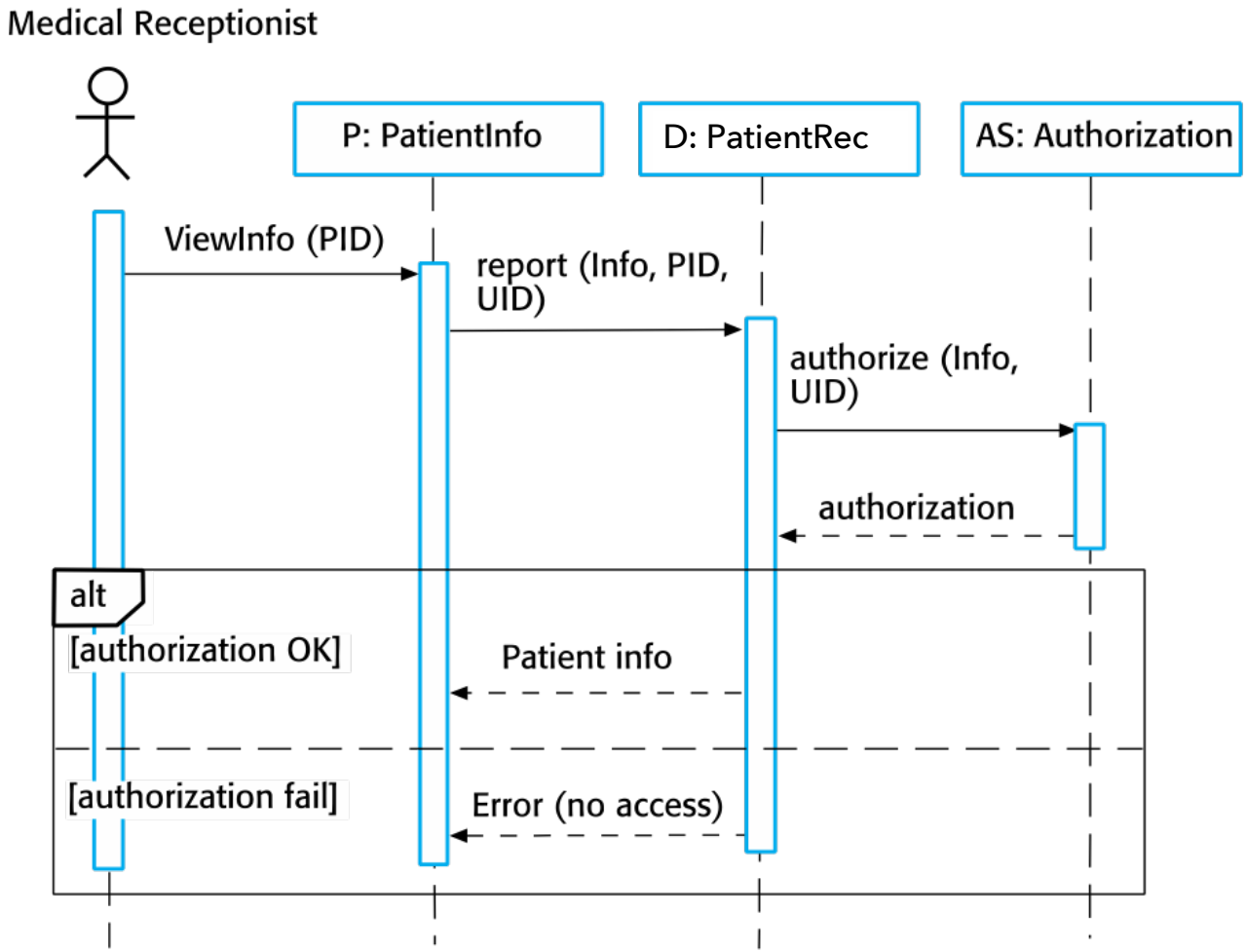
Loops in UML 2.0



Loop: the fragment may execute multiple times, and the guard indicates basis for iterations

Sequence diagram for "View patient information" use case

Use case: View Patient Information – through authorization



Sequence diagram for "Transfer Patient Data" use case

Use case:
Transfer Patient Data