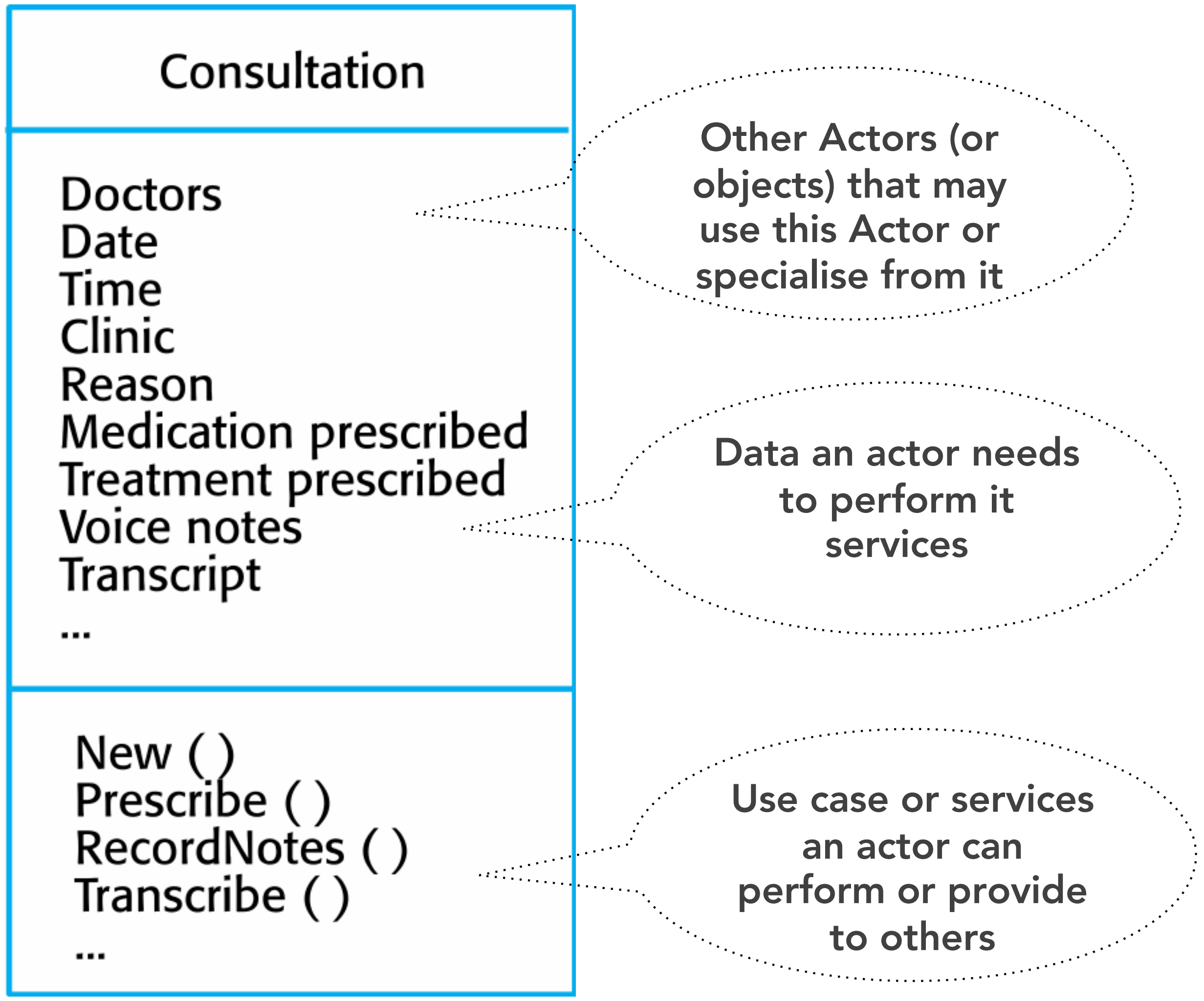
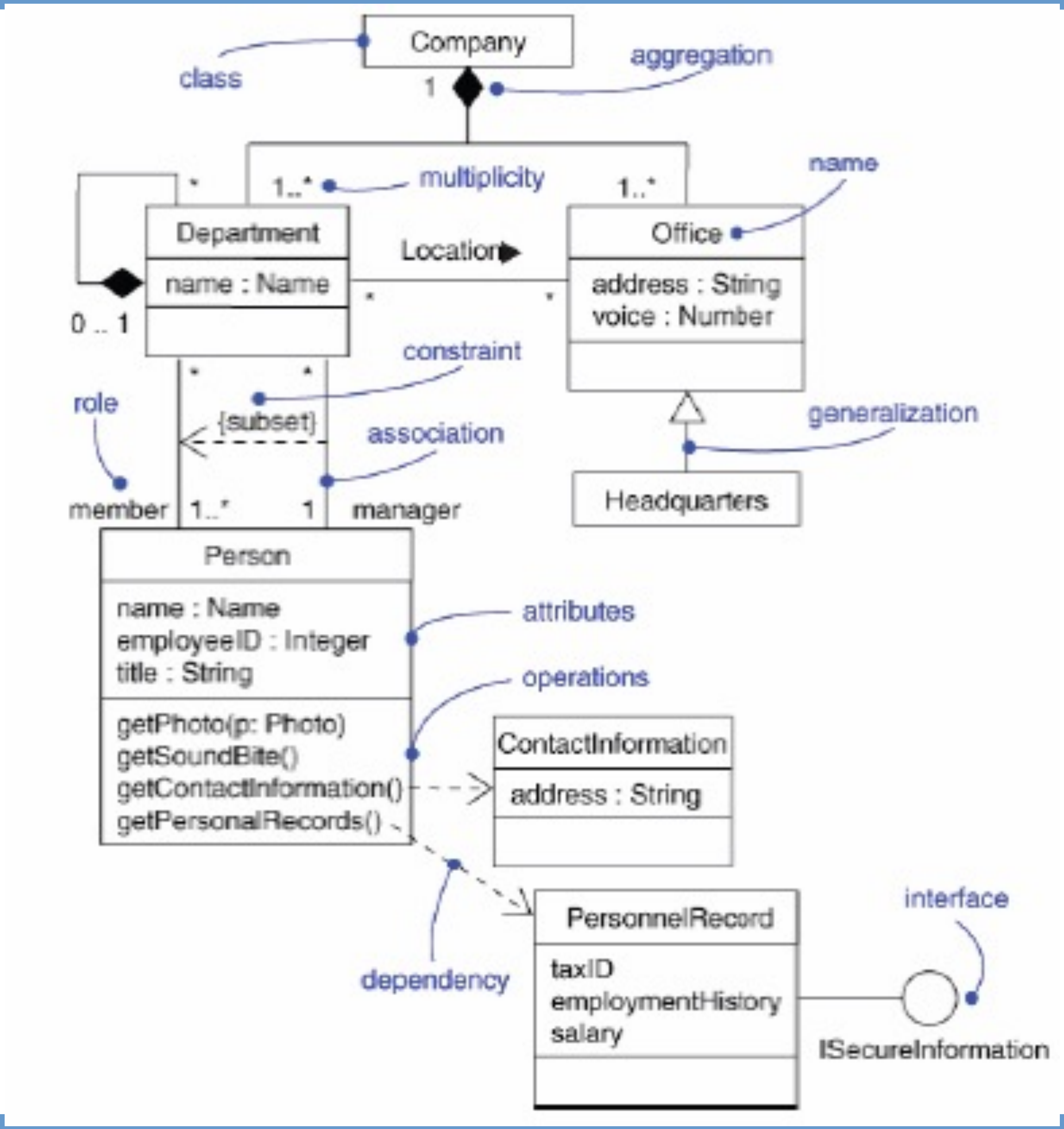
Complete class Description

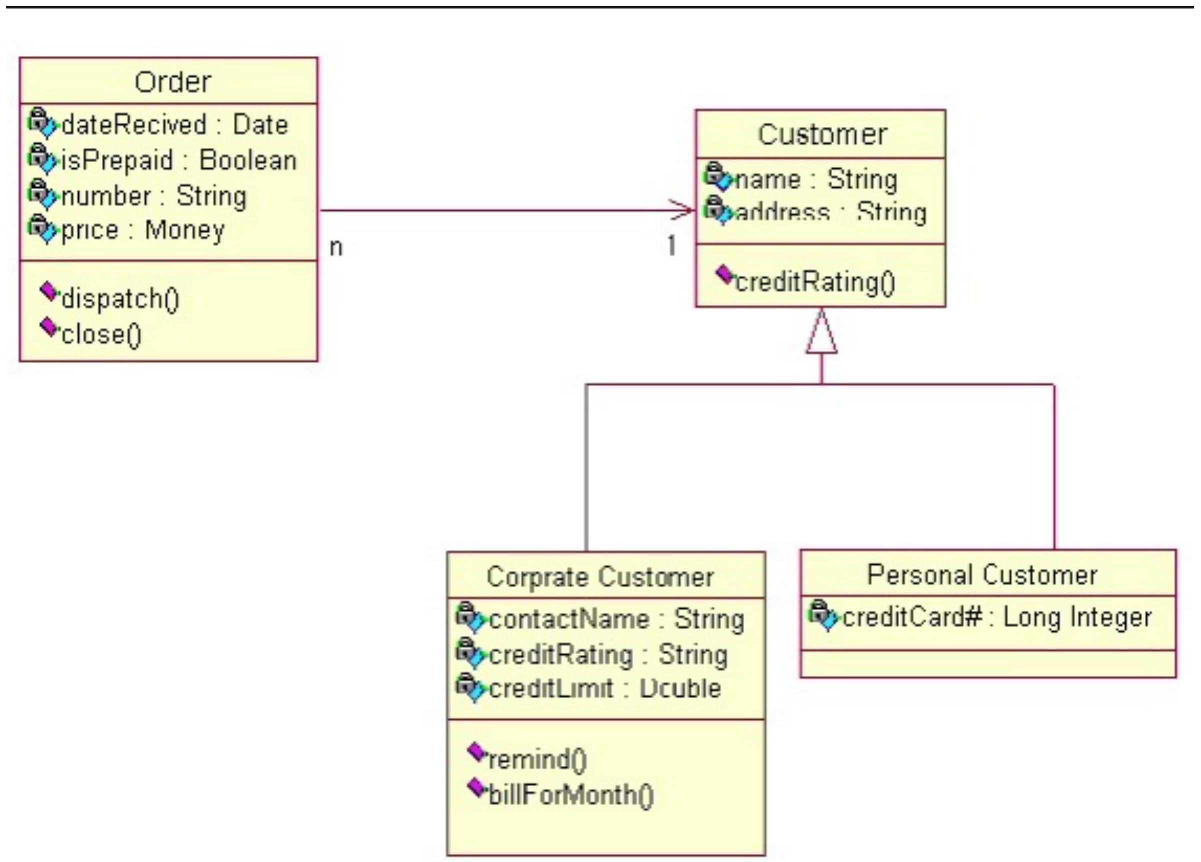

Consultation
Doctors
Date
Time
Clinic
Reason
Medication prescribed
Treatment prescribed
Voice notes
Transcript
...
New ( )
Prescribe ( )
RecordNotes ( )
Transcribe ( )
...
Other Actors (or objects) that may use this Actor or specialise from it
Data an actor needs to perform it services
Use case or services an actor can perform or provide to others

Example: Detailed Class Diagram

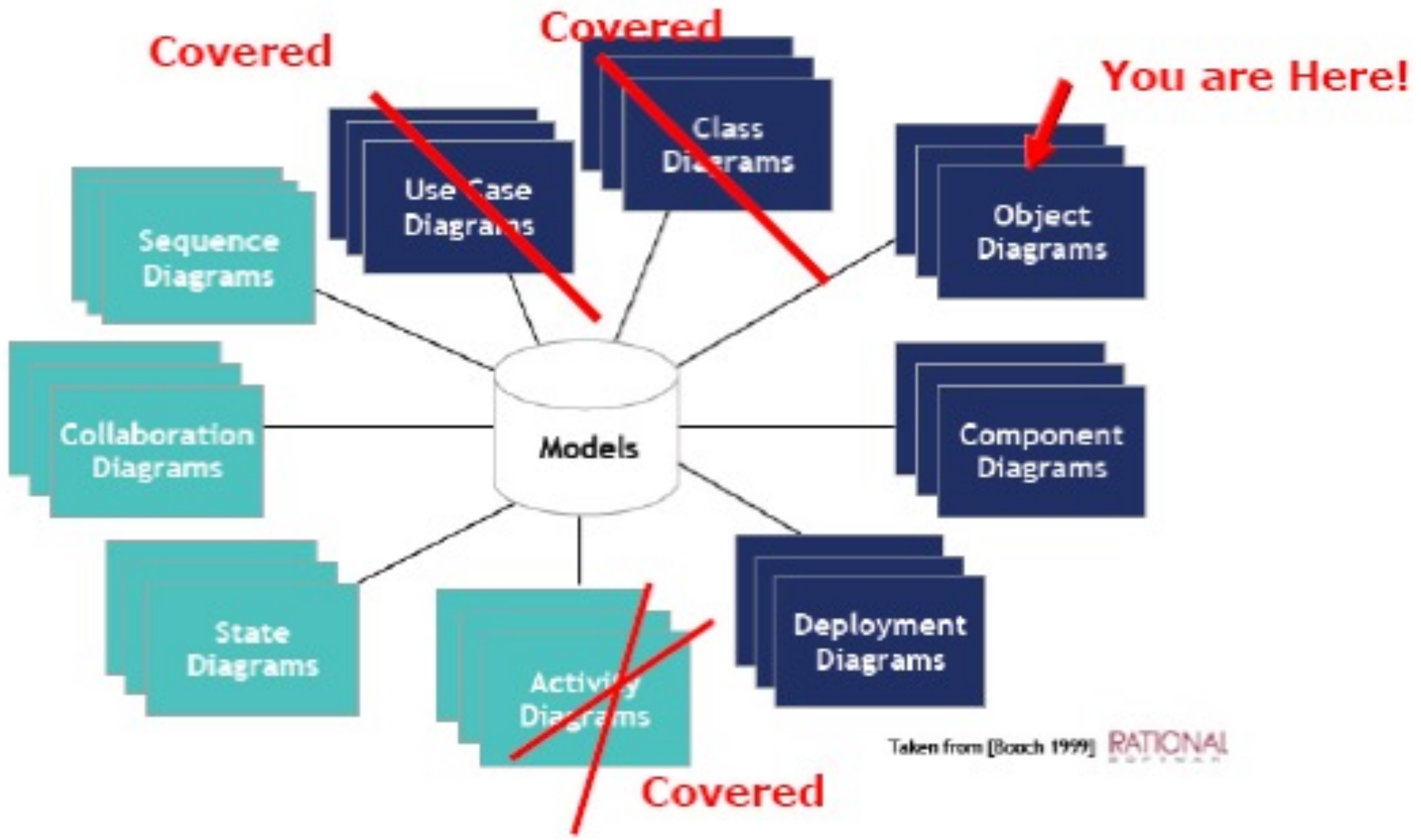


Another Example

Corporate Customer and Personal Customer classes may have some common attributes/operations such as name and address, but each class has its own attributes and operations. The class Customer is a general form of both the Corporate Customer and Personal Customer classes.



UML Diagrams



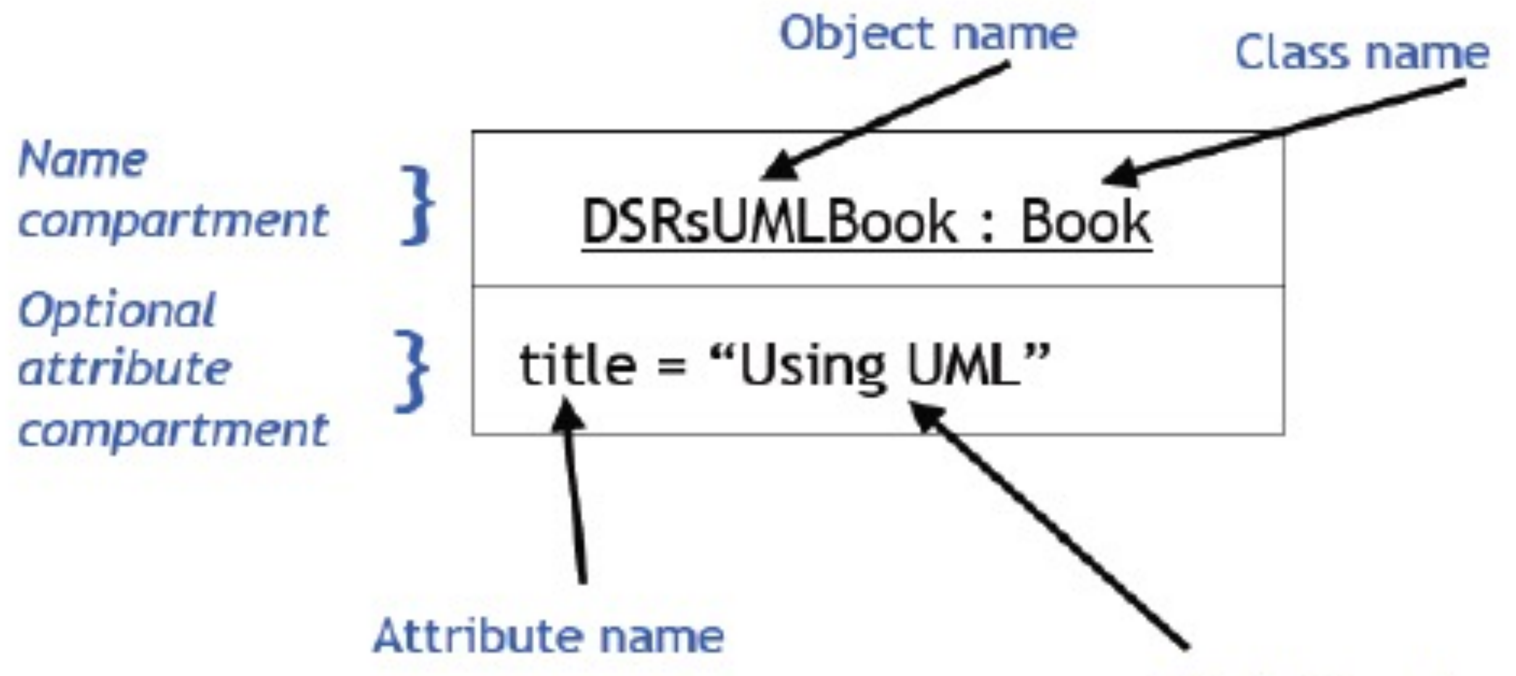
Object Diagram

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

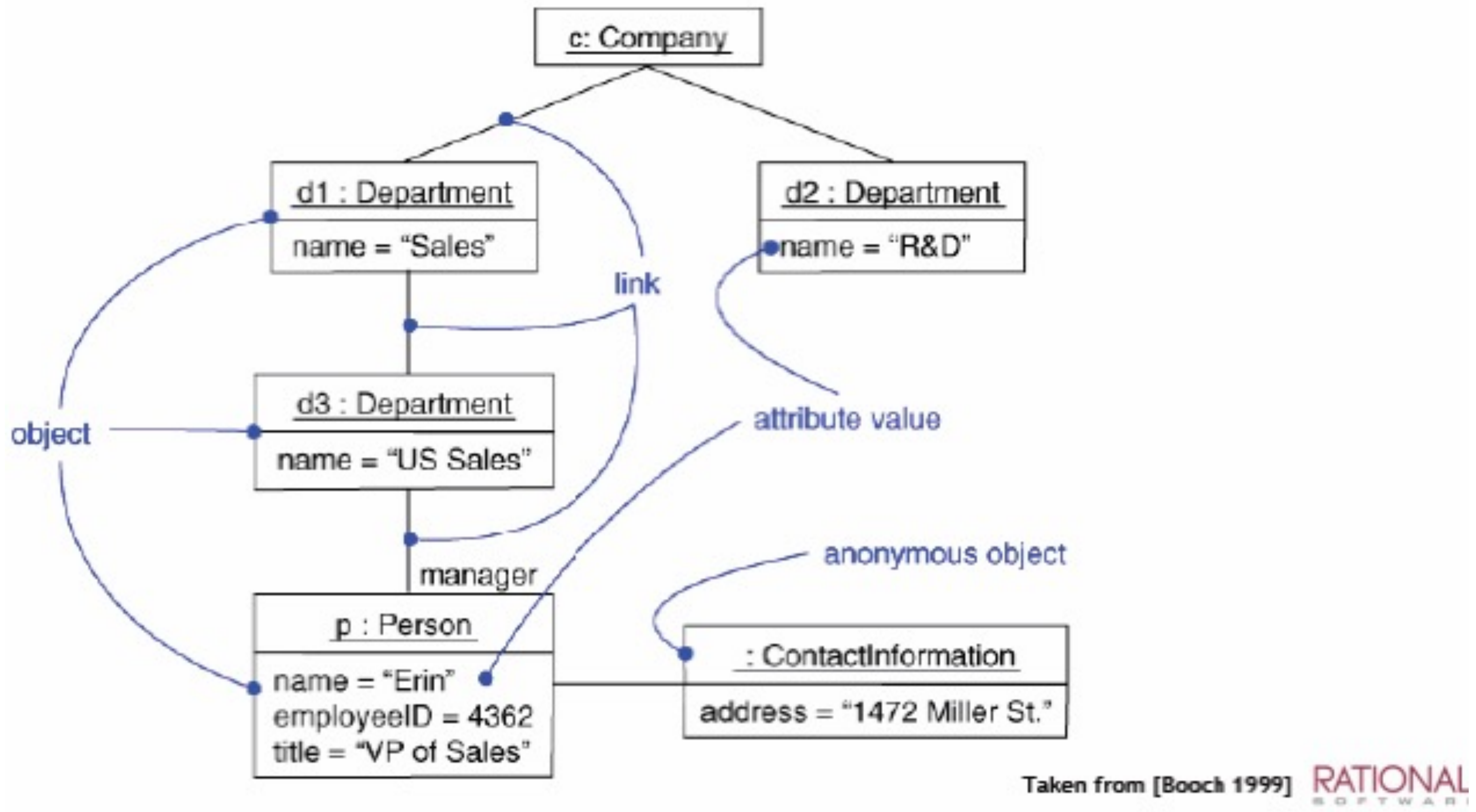


Operations and attribute types are *not* shown on object diagrams!

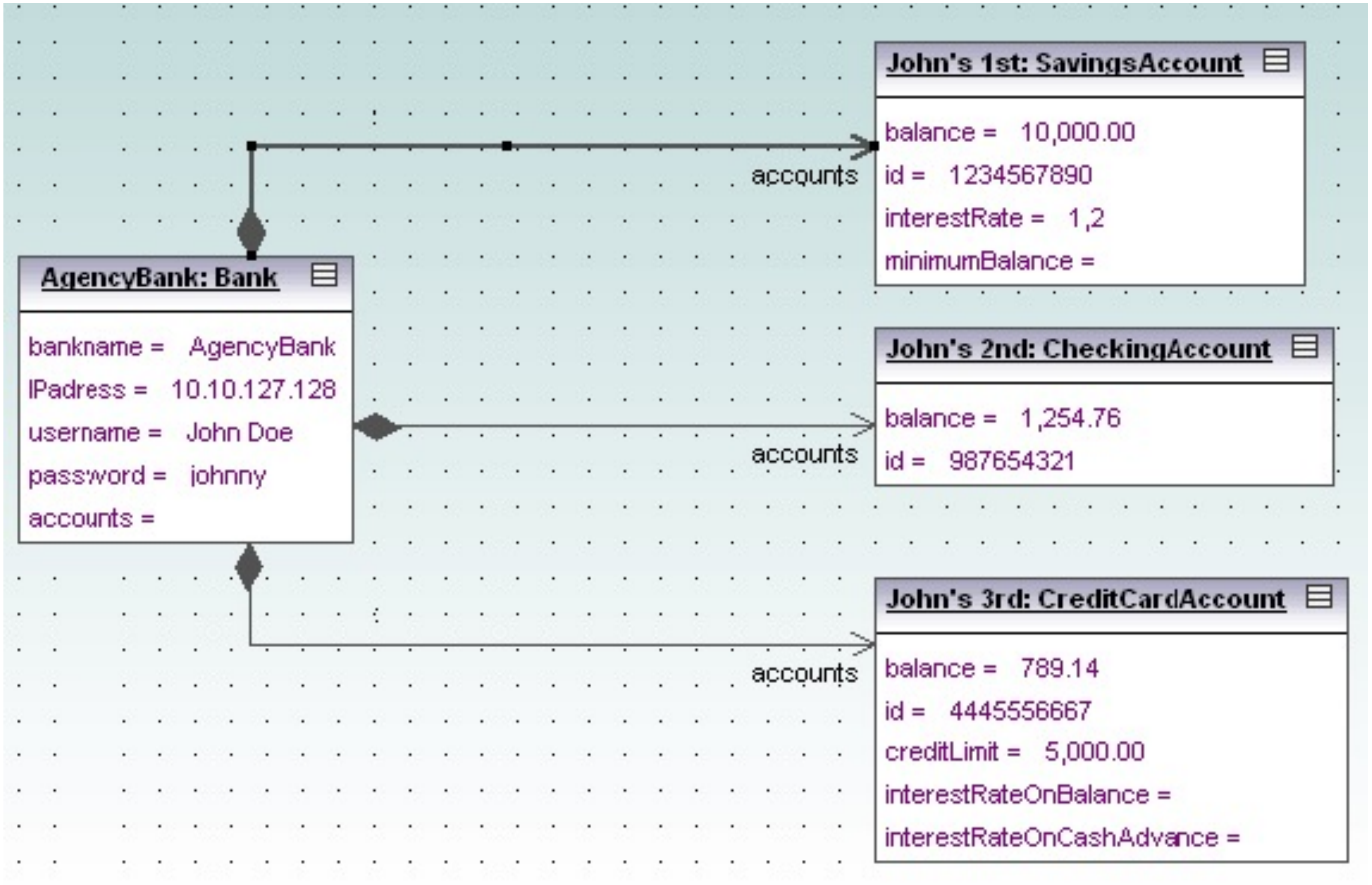
Reference: D. Rosenblum, UCL

Object Diagram

Capture *class instances* and *links* between objects



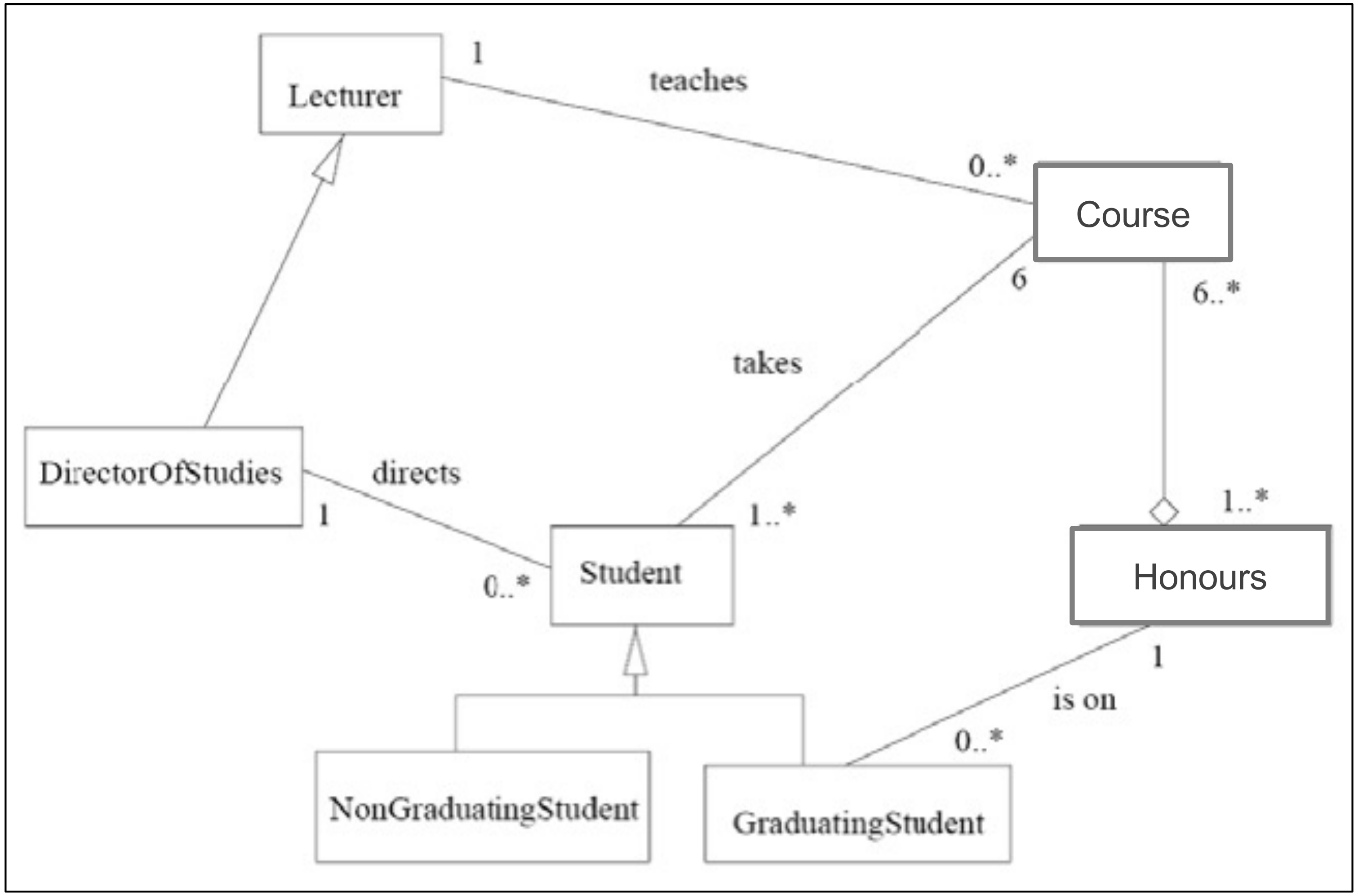
Example: Object Diagram



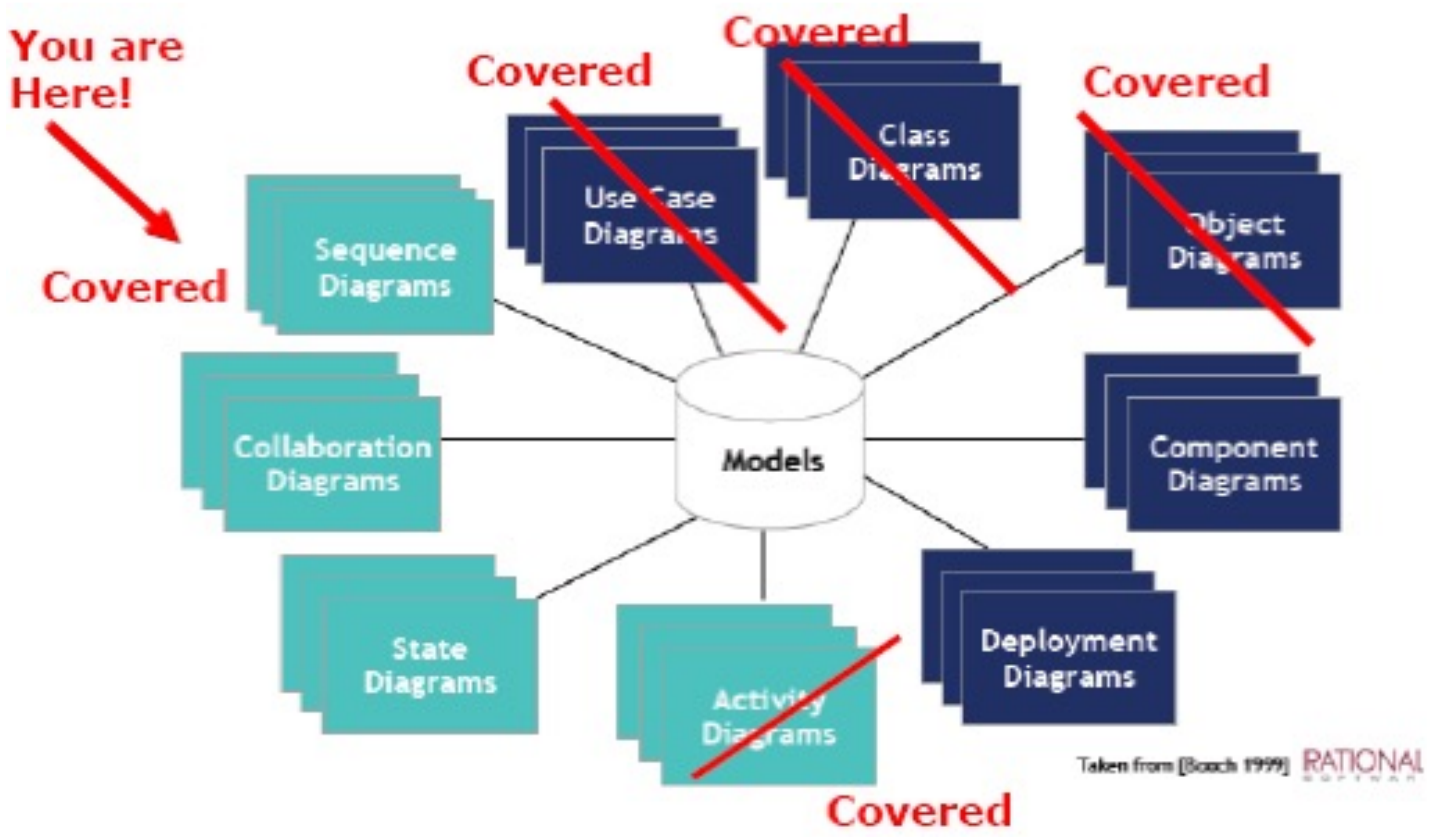
Example: Object Model/Diagram

For the following class model draw:

- a detailed Class Model (or Diagram)
- an Object Model (or Diagram)



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.

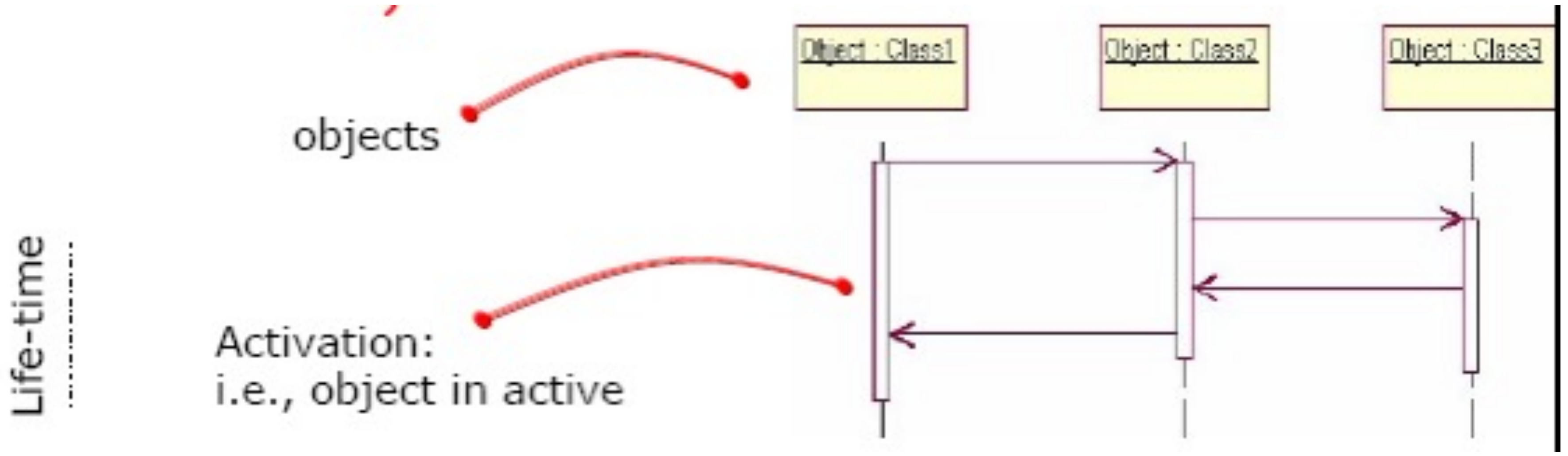
A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.

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

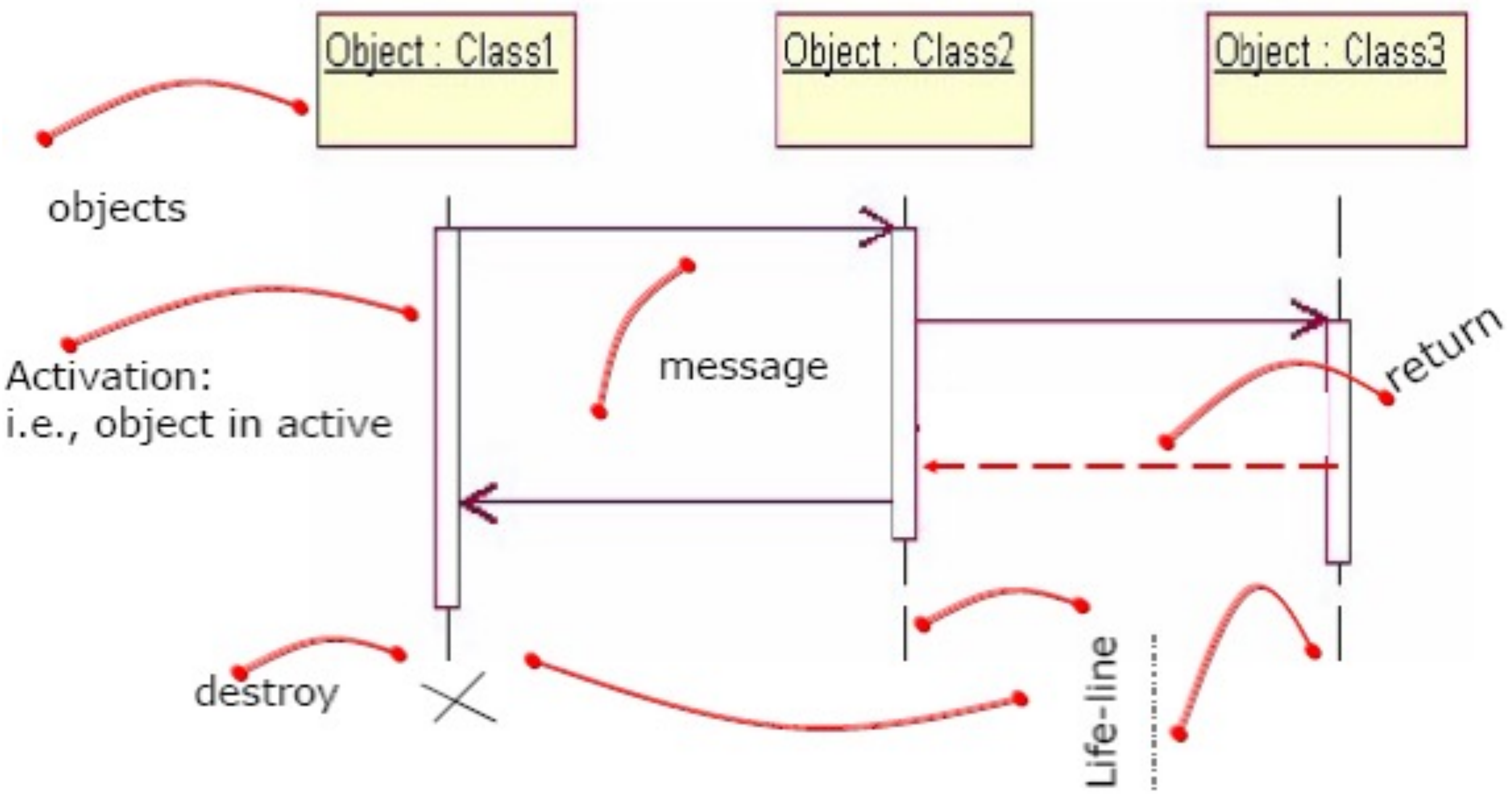
Interactions between objects are indicated by annotated arrows.

Sequence diagrams

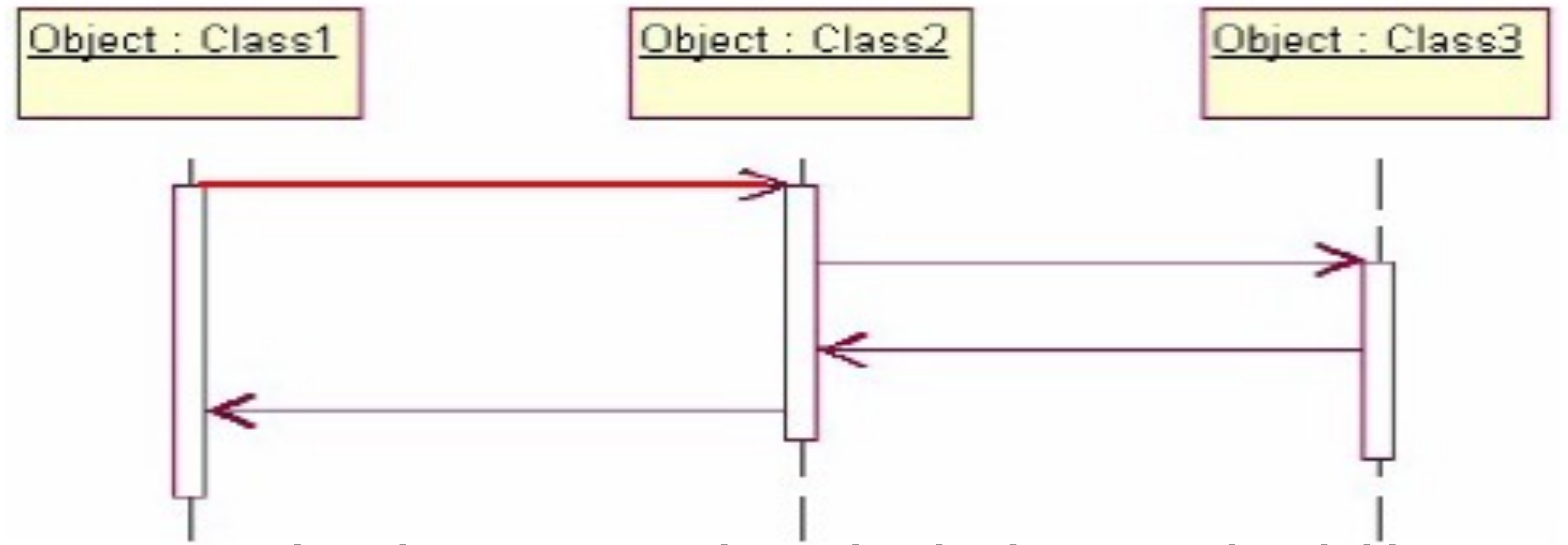
Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass. the diagrams are read left to right and descending.
Object interactions are arranged in a time sequence (i.e. time-oriented)



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

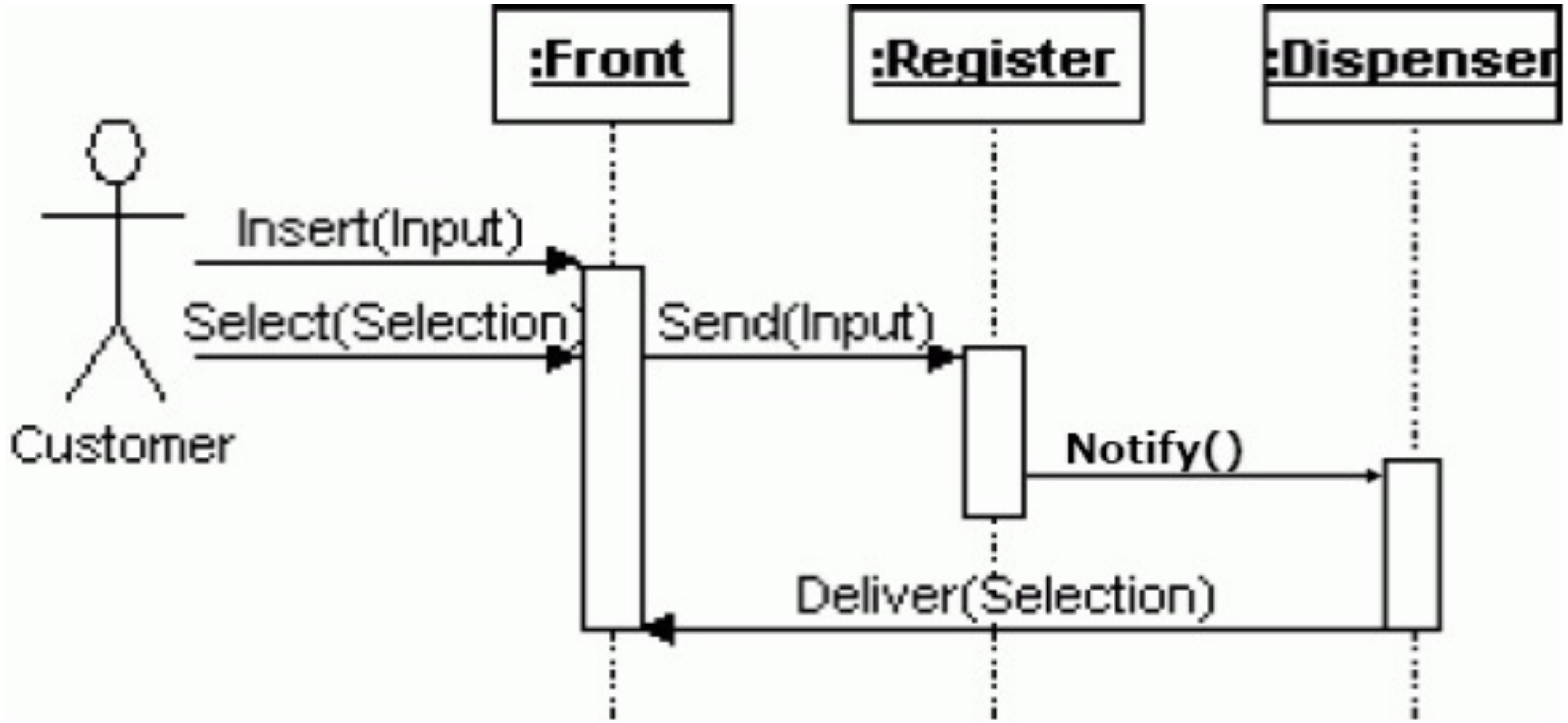
the dispenser: which delivers the selected product to the customer

Example

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

1. The customer inserts money in the money slot in **front** money collector.
2. The customer makes a selection on the **front** UI
3. The money travels to the **register**
4. The **register** checks to see whether the correct money is in the money **collector/dispenser**
5. The **register** updates its cash reserve
6. The **register** notifies the **dispenser** which delivers the product (e.g. receipt) to the **front** of the machine

Example



The "Buy a product" scenario.
Because this is the best-case scenario, it's an instance sequence diagram

However, note…

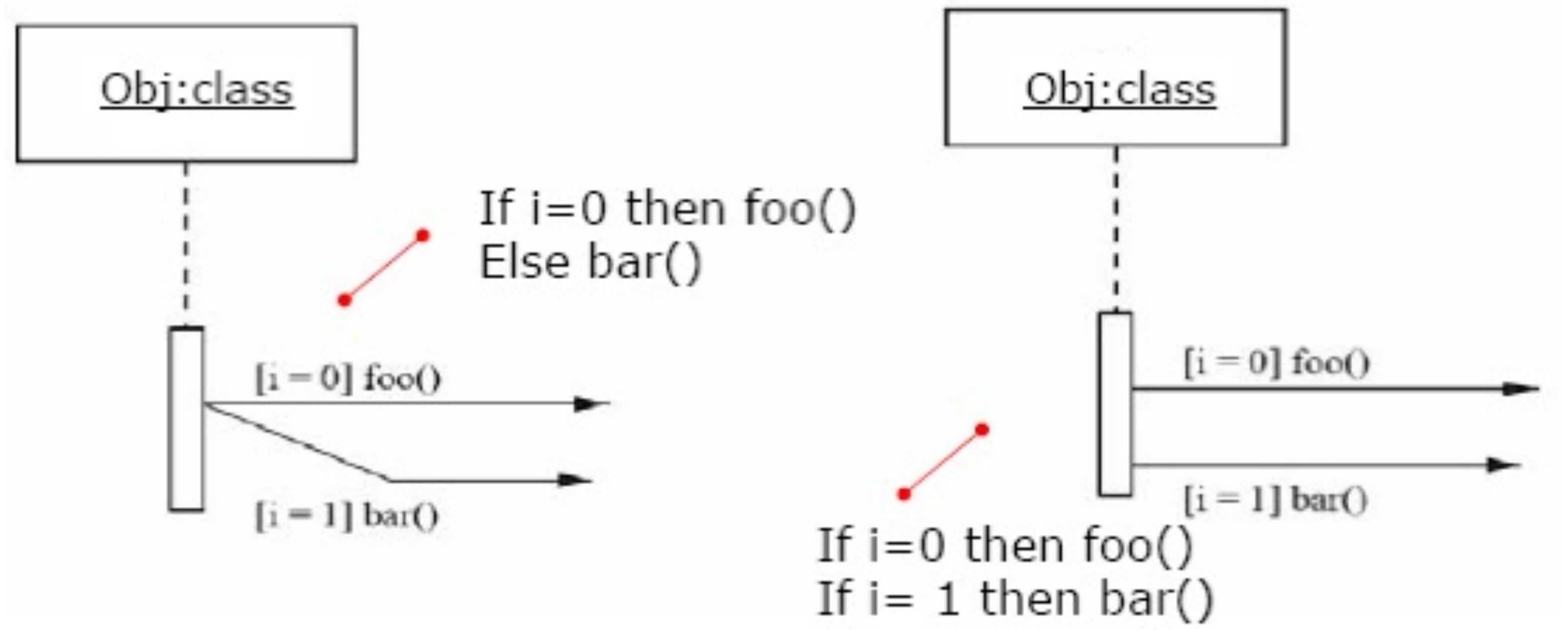
We have seen an instance of an interaction diagram- i.e. one possible sequence of messages

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

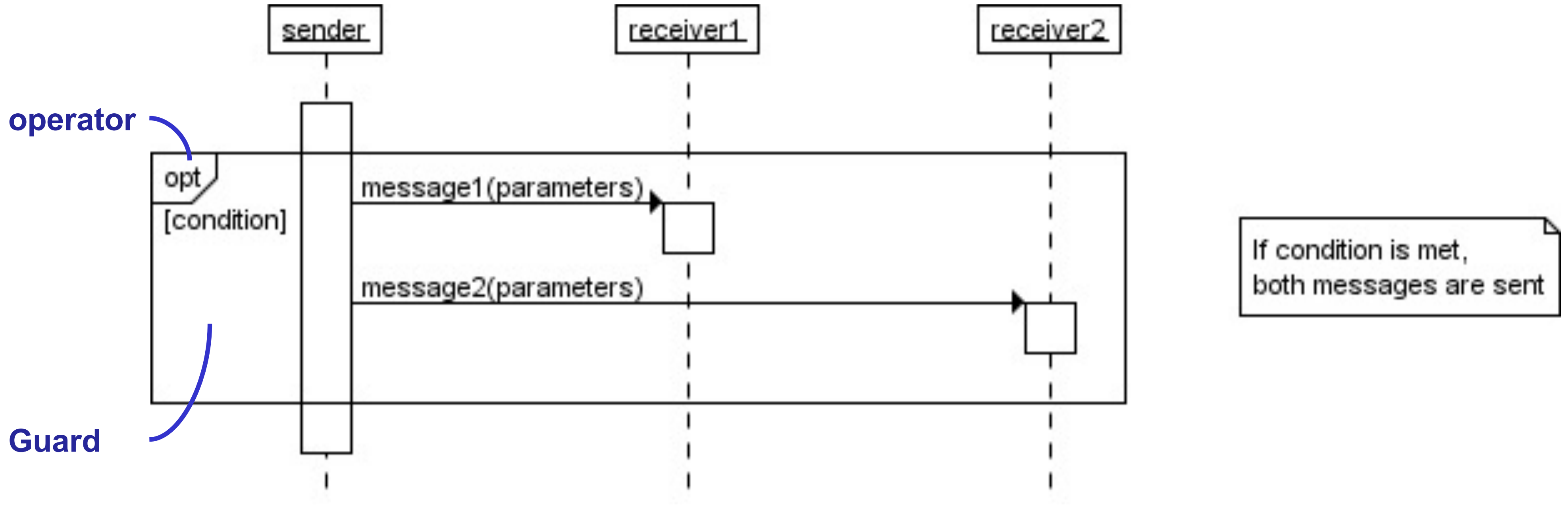
Showing conditional behaviour

- A message may be **guarded** by a condition
- Messages are only sent if the **guard** evaluates to true at the time when the system reaches that point in the interaction



Notation in UML 1.0 and UML 1.4

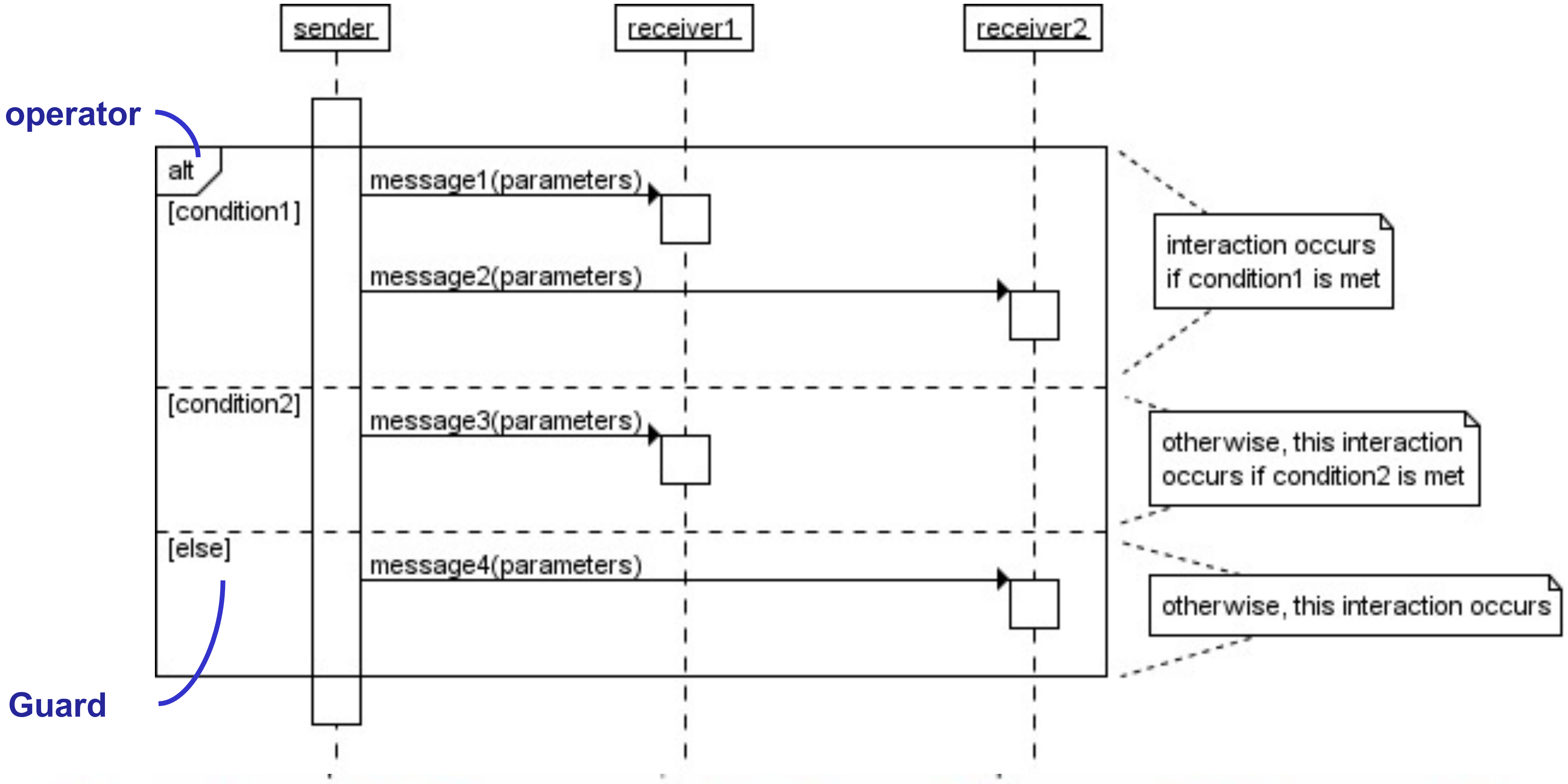
Opt(ional) in UML 2.0



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

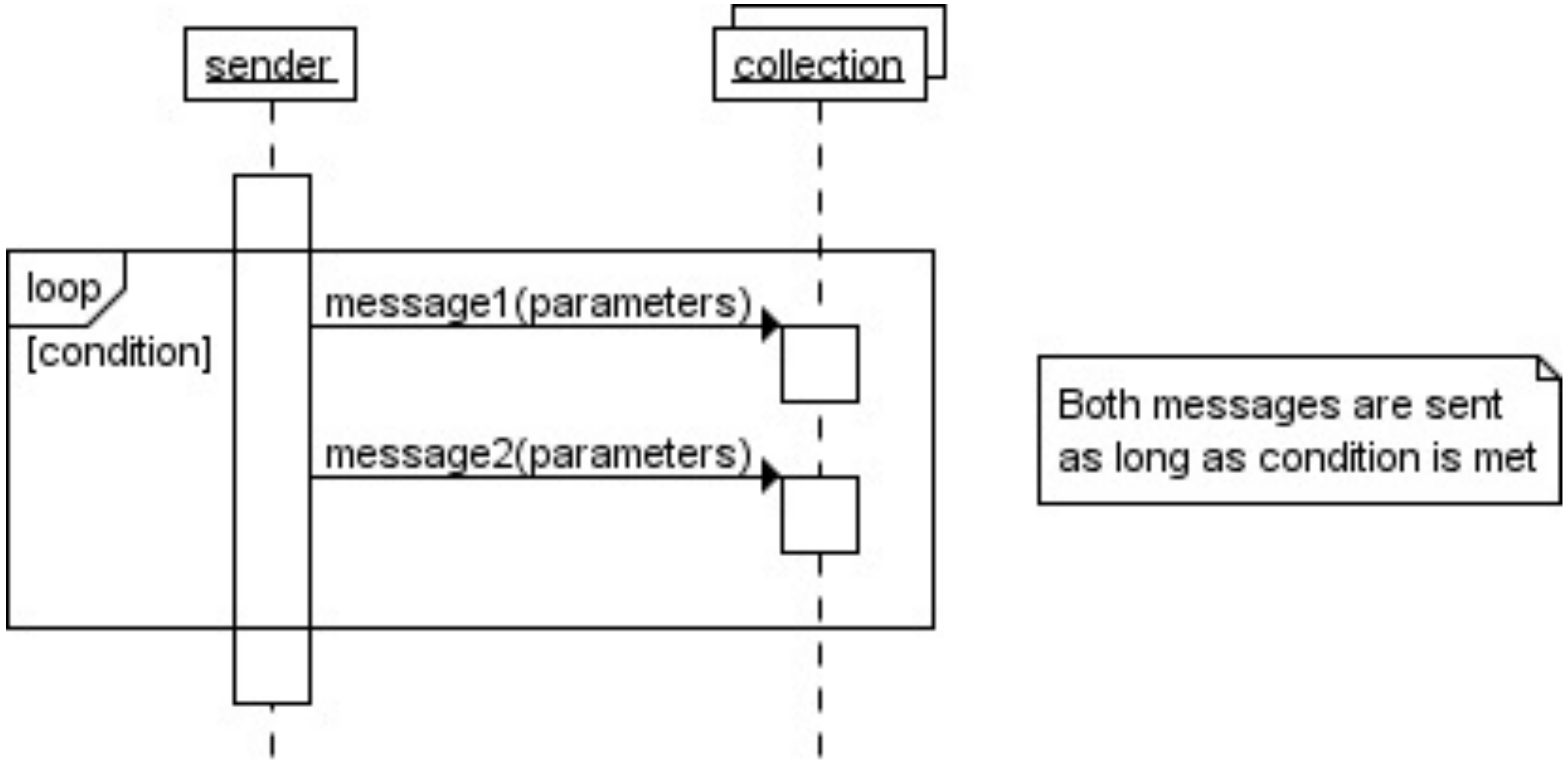
This is equivalent to an "**alt**" with one trace (next slide)

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



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

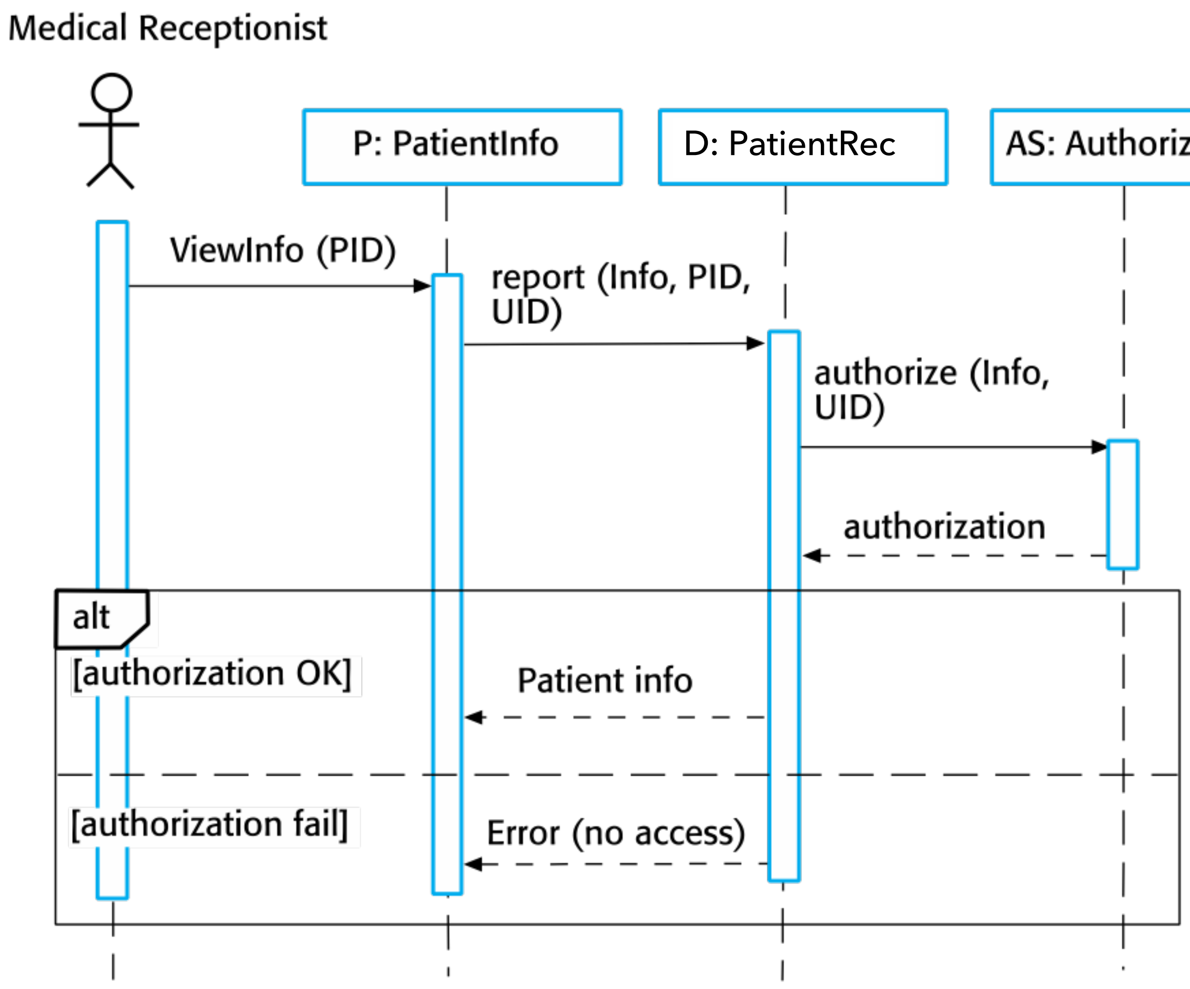
Loops in UML 2.0



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

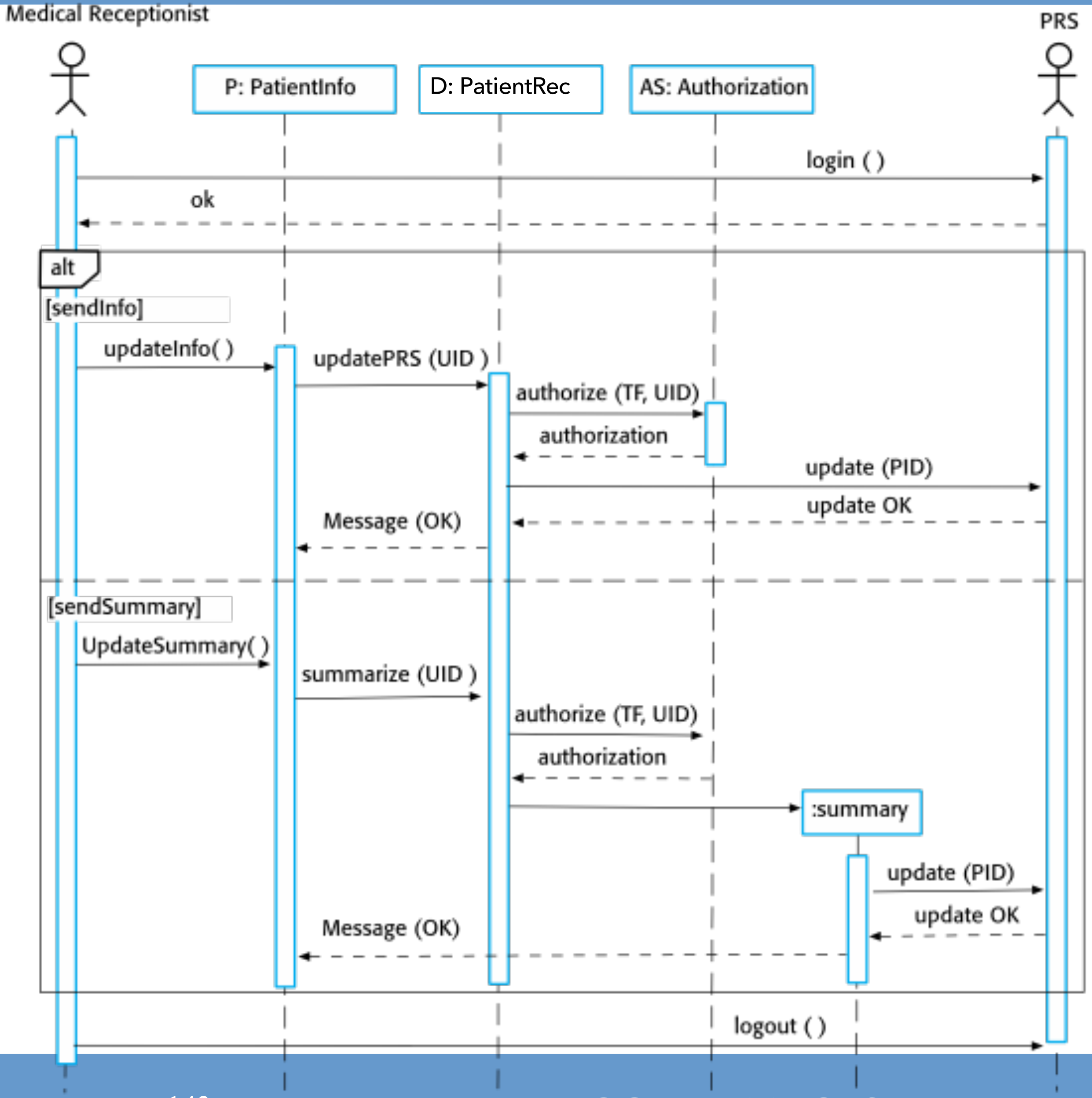
Sequence diagram for View patient information use case

Use case: View Patient Information – through authorization



Sequence diagram for Transfer Data

Use case: Transfer Data- demonstrates interactions between Actors



Exercise: Draw a sequence diagram for the Use-Case "Borrow Copy of a Book"

Library system, four **objects** are involved to do the work to achieve the Use case: (Borrow Copy of a Book)

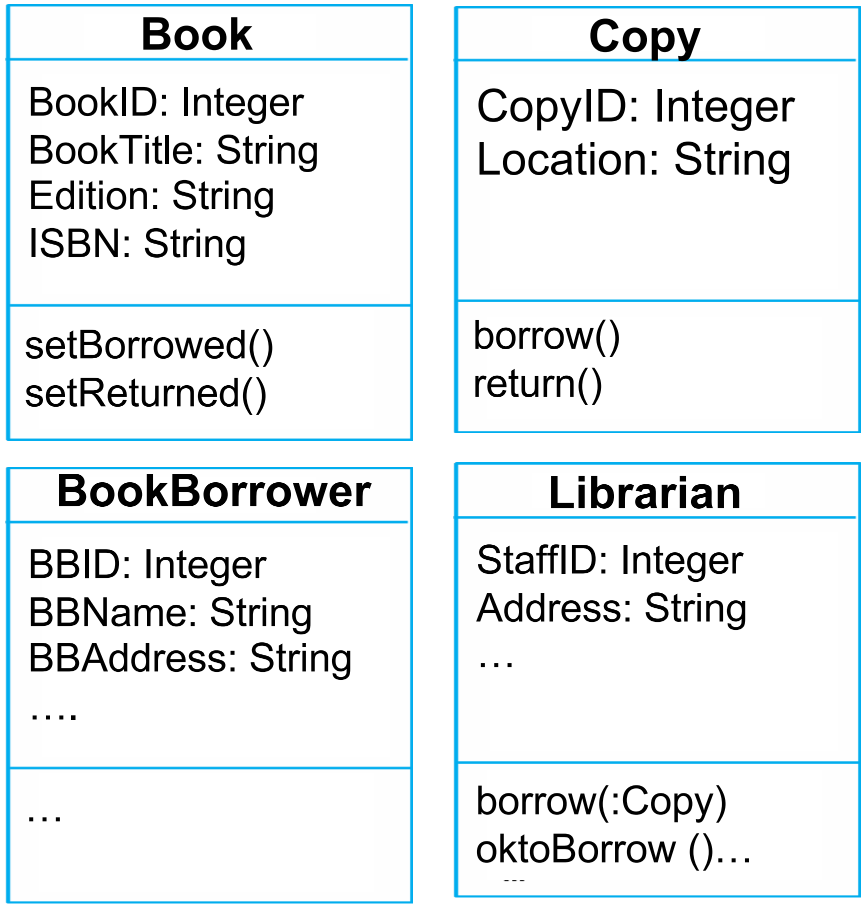
BookBorrower: that will borrow the book

Copy: copy of a book

Book: to which the Copy is of it.

Librarian/LibraryStaff: which authorizes and register the borrowing of the borrowed copy.

Relevant objects: derive from class model, below


Book
BookID: Integer
BookTitle: String
Edition: String
ISBN: String
setBorrowed()
setReturned()
Copy
CopyID: Integer
Location: String
borrow()
return()
BookBorrower
BBID: Integer
BBName: String
BBAddress: String
….
…
Librarian
StaffID: Integer
Address: String
…
borrow(:Copy)
oktoBorrow ()…