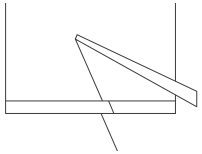
Bring ideas to life
VIA University College



CRC, Interaction- & Class Diagrams

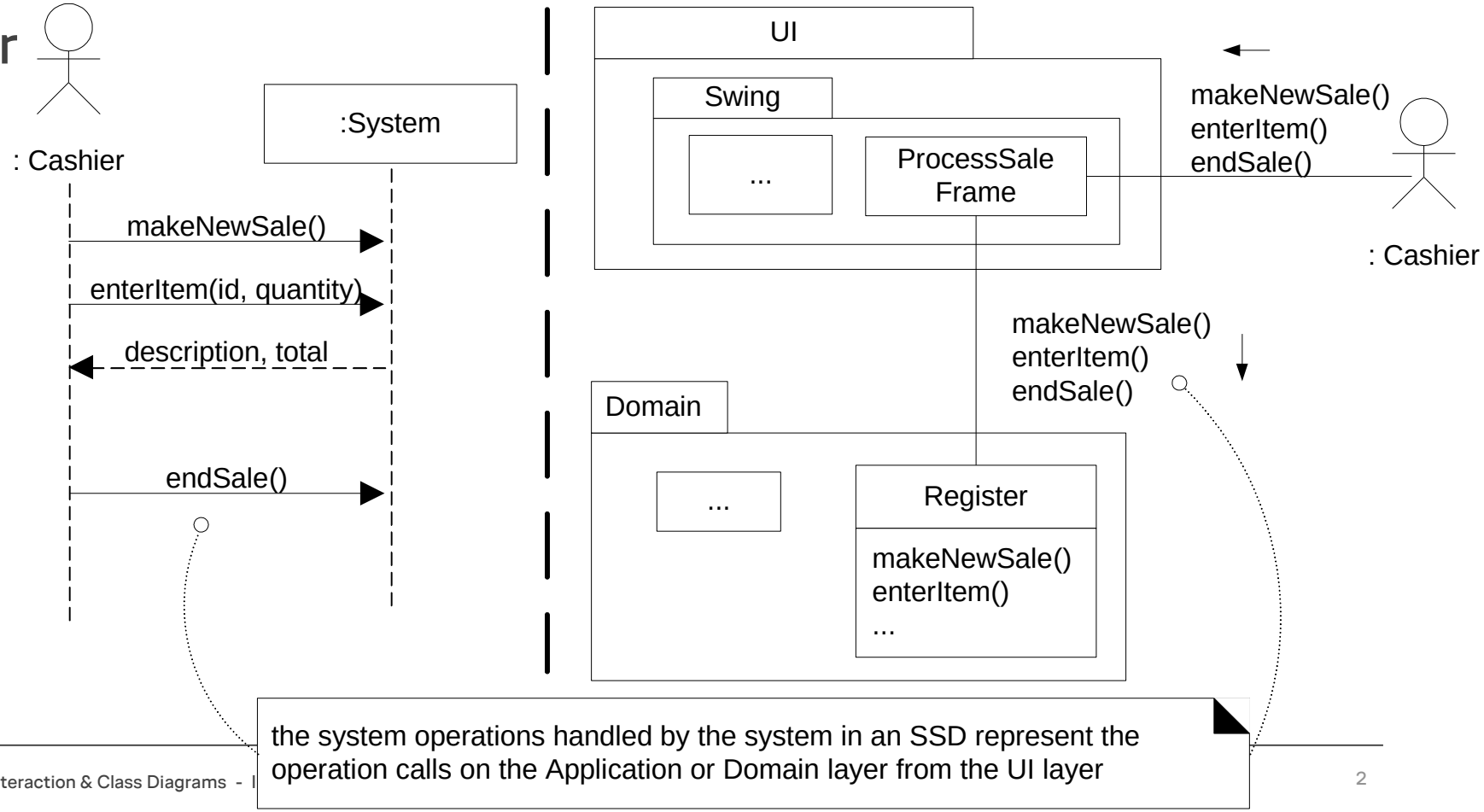
SWE1

System Sequence Diagrams (SSDs) System Operations and Layers

[Larman, 2005] Fig. 13.8

System operations are normally captured by objects in UI/View Layer

These UI objects forwards the request to objects in the Domain/Model layer for handling



The three ways to design Objects

1. **Just Code**
   - Design while coding
   - From mental model to code
   - Very difficult to do in a group
   - OK for prototyping of small parts
2. **Draw, then Code**
   - Drawing UML on paper, board or CASE tool
   - Don’t overdo it!
   - Switch to 1.
   - Easy to share work in a group
3. **Only draw**
   - Draw very specific UML diagrams and the do automatic code generation
   - Typically as hard as writing the code by hand

CRC-Cards

Class name:	
Responsibilities:	**Collaborators:**

Class **R**esponsibilities **C**ollaboration – Card

- Use one card per class

Class name: Student	
Responsibilities: 1) To study during the semester 2) Deliver answers to exercises	**Collaborators:** School Teacher Project team

CRC-Cards

Class name describes the class that the CRC card represents

Responsibilities each distinct responsibility is on its own row

Collaborators some responsibilities will collaborate with one or more other classes to fulfil one or more *Scenarios*

- Collaborators are listed on the right hand side of the CRC card, next to the responsibilities that they are helping to realize

Play CRC-Cards around a table

CRC-Cards - how-to

Find Classes start finding the main classes of the system

- typically coming from the Domain Model
- Student, Class, Course, Teacher

Find Responsibilities what must the class do as well as what information you wish to maintain about it

- You will often identify a responsibility for a class to fulfil a collaboration with another class

CRC-Cards - how-to

Collaborators a class often does not have sufficient information to fulfil its responsibilities

- it must collaborate (work) with other classes to get the job done
- collaboration will be in one of two forms
  - a request for information
  - a request to perform a task

Identify the collaborators of a class for each responsibility by asking

- "does the class have the ability to fulfil this responsibility?“

If not then look for a class that either has the ability to fulfil the missing functionality or the class which should fulfil it

In doing so you'll often discover the need for new responsibilities in other classes and maybe even the need for a new class or two

Classes

Rules of thumb (1):

- The class should be kept as simple as possible
  - Three to five responsibilities/methods per class
- Classes always collaborate with other classes
  - Associates with a small number of other classes (low coupling)
  - Can delegate some of the responsibilities to helper classes

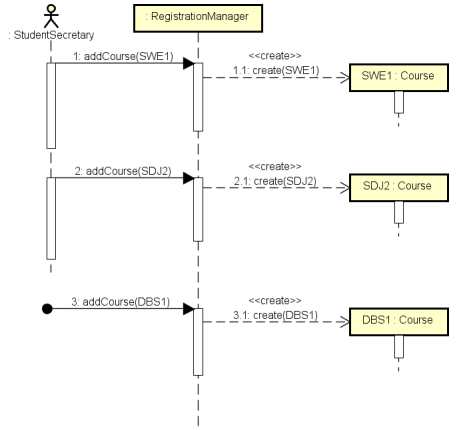
Classes

Rules of thumb (2):

- If there are to many small classes with just one or two responsibilities it's time to look on consolidating some of these into larger classes
- If there are a few large classes with many (>5) responsibilities or omnipotent classes – it's time to see if they can be decomposed into smaller classes
- Don't make classes that only wraps a single function
- Avoid deep inheritance trees

UML-Interaction Diagrams

A interaction diagram shows how objects/instances are working together (collaborate with each other) to realize a Use Case scenario

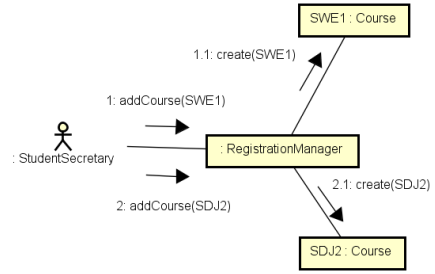


Sequence Diagrams

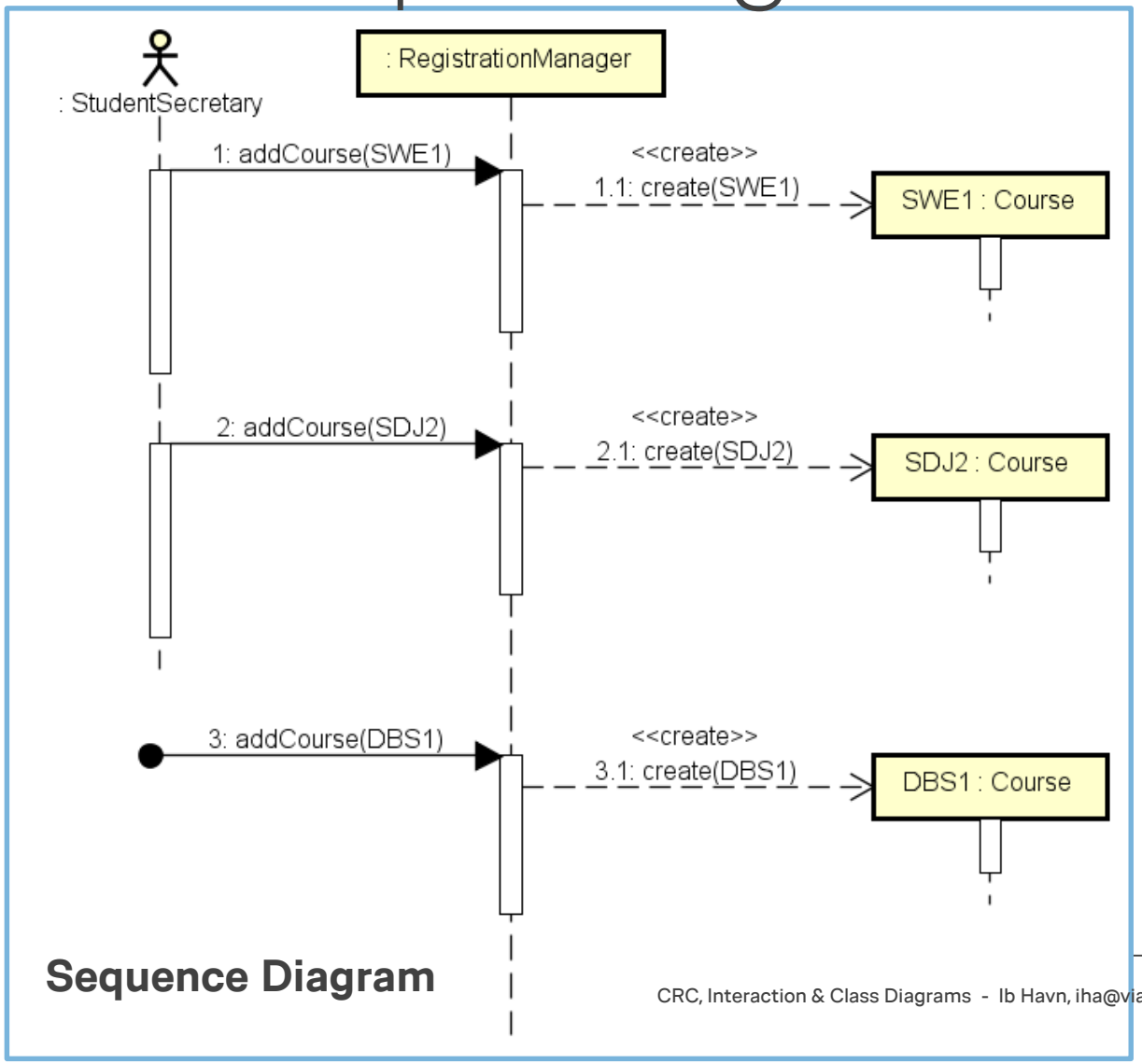
- You already seen System Sequence Diagrams (SSDs)
- Shows the time dimension
- Most used

Communication Diagrams (Collaboration Diagrams in UML 1.x)

- A network of classes showing how they are collaborating
- Space efficient
- Good for sketches "UML as a sketch"

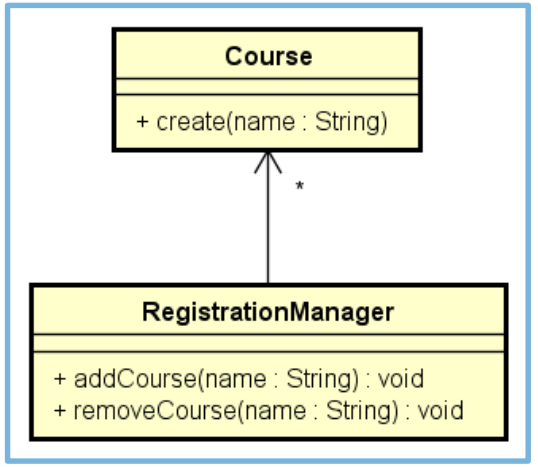


UML-Interaction Diagrams Example – register courses

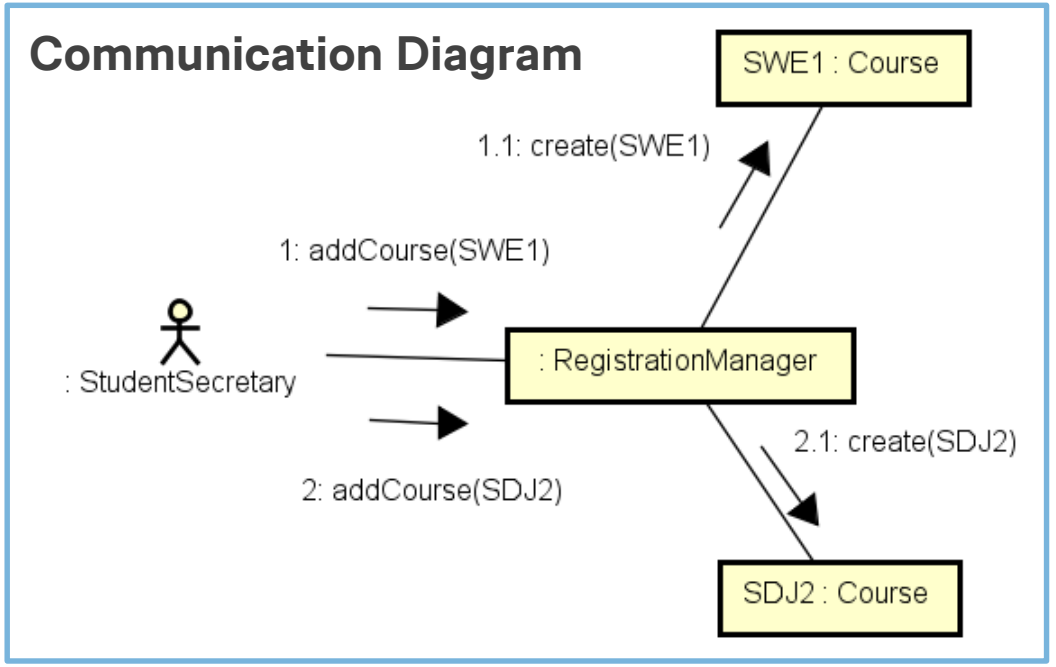


Sequence Diagram

Class Diagram

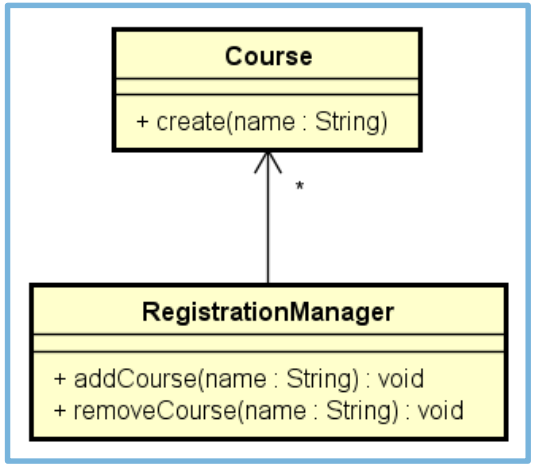


Communication Diagram

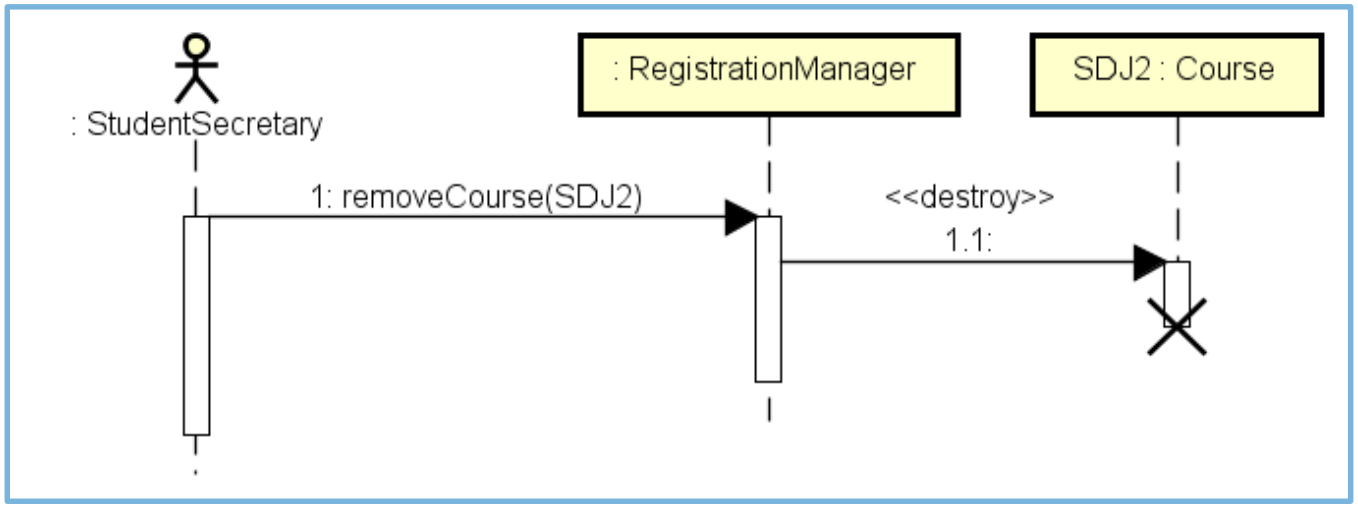


UML-Interaction Diagrams Example – remove courses

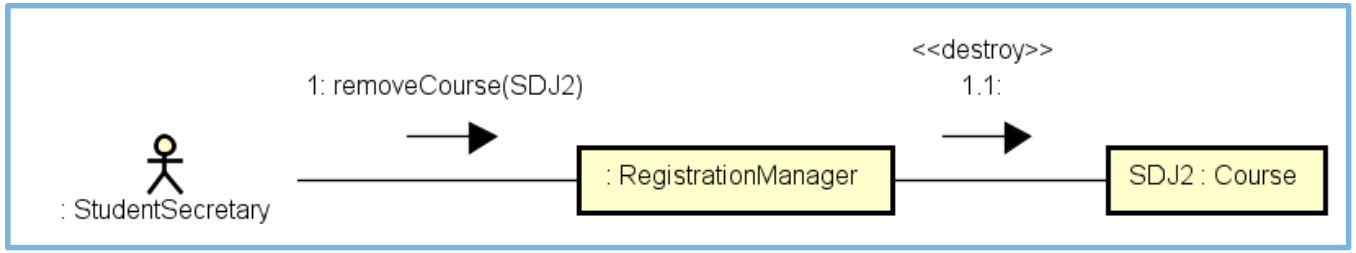
Class Diagram



Sequence Diagram

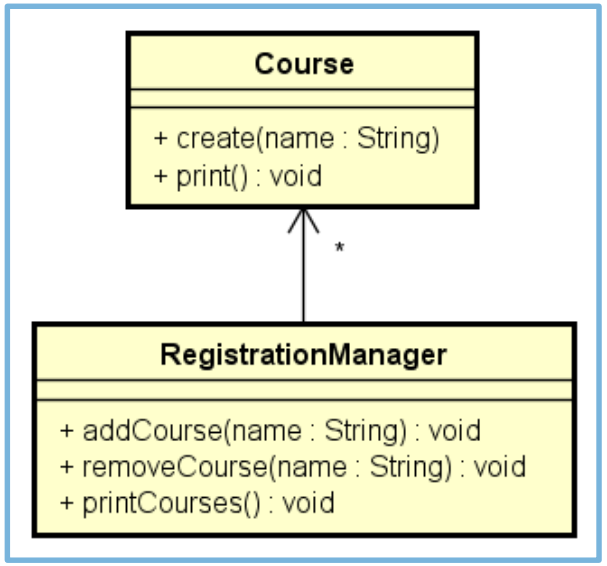


Communication Diagram

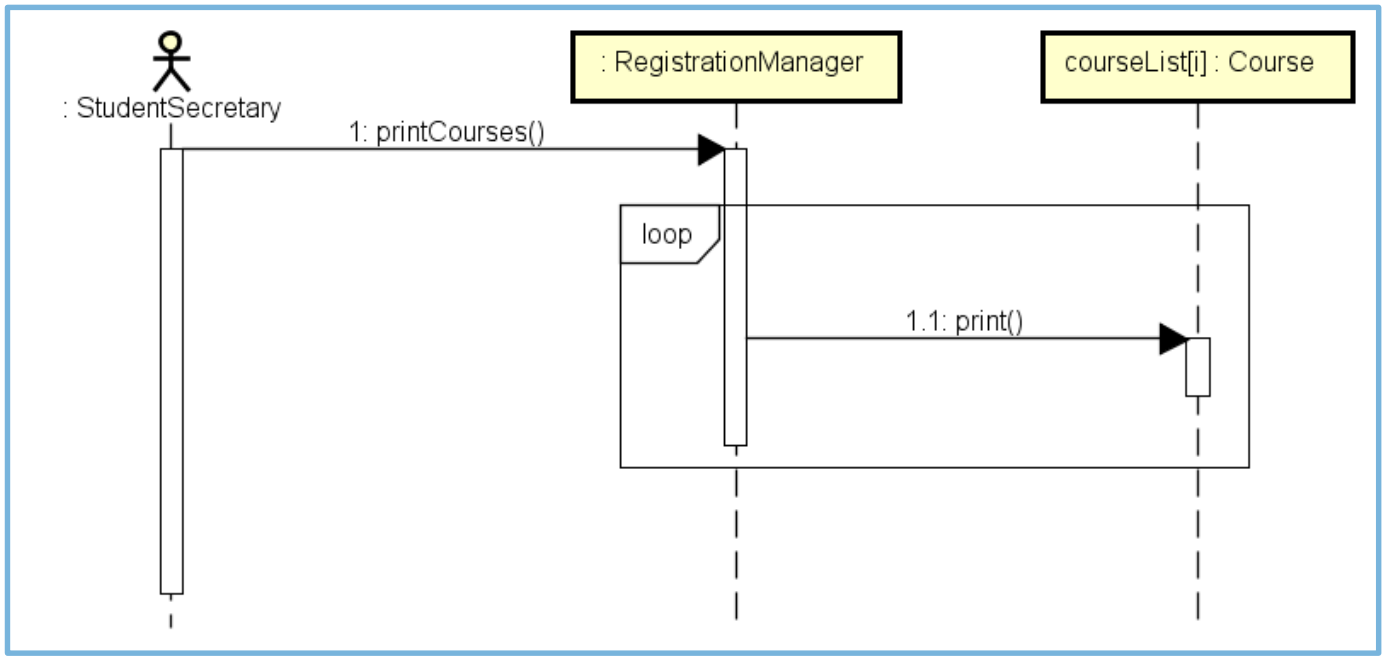


UML-Interaction Diagrams Example – print courses

Class Diagram



Sequence Diagram



Communication Diagram

Itteration not available in Astah, but see [Larman, 2005] figure 15.32

UML-Interaction Diagrams vs. Class Diagrams

Start out with a Class diagram with the classes from the Domain Model

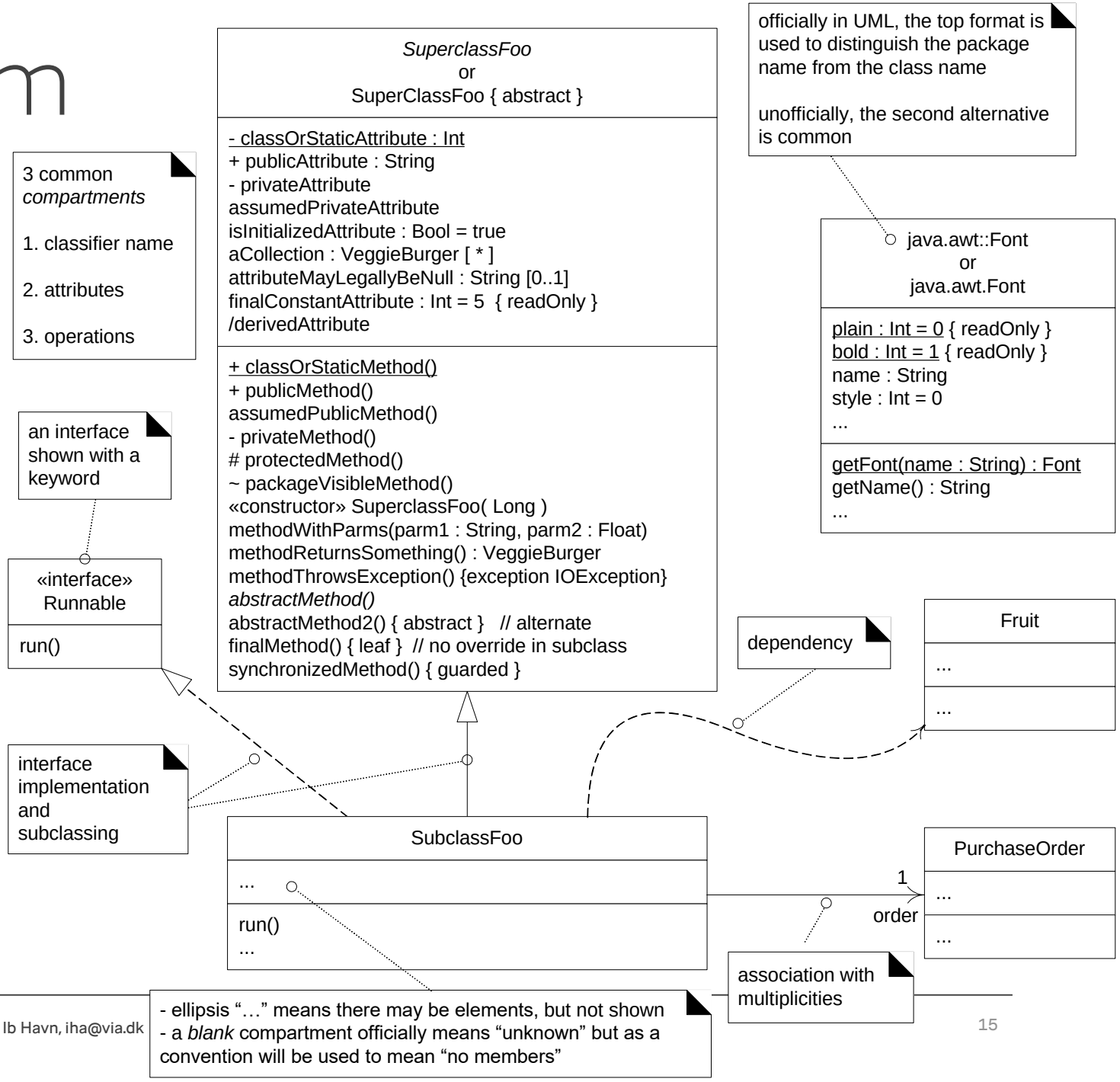
Continue with dynamic modelling (Interaction Diagrams)

- Details of what objects are responsible for, and how they collaborates via messages/methods and parameters

Classes typically evolves automatically

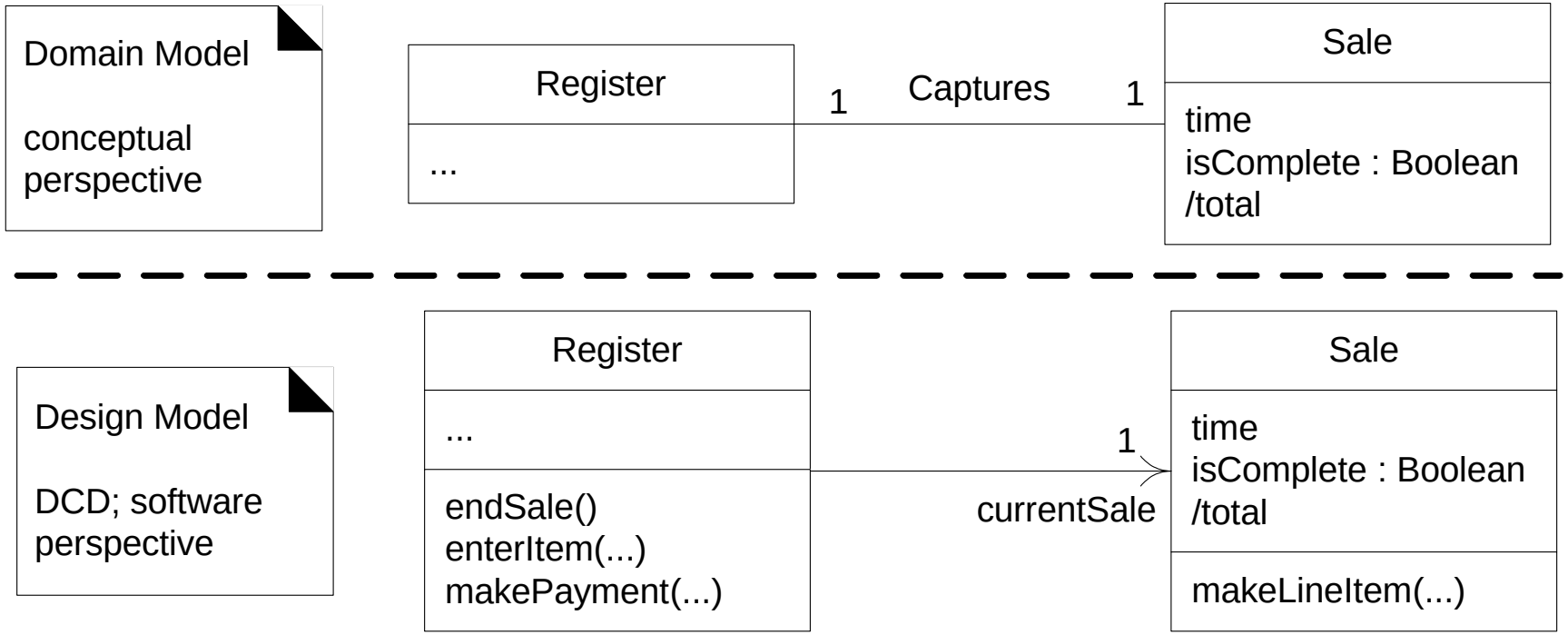
UML-Class Diagram

[Larman, 2005] Fig. 16.1



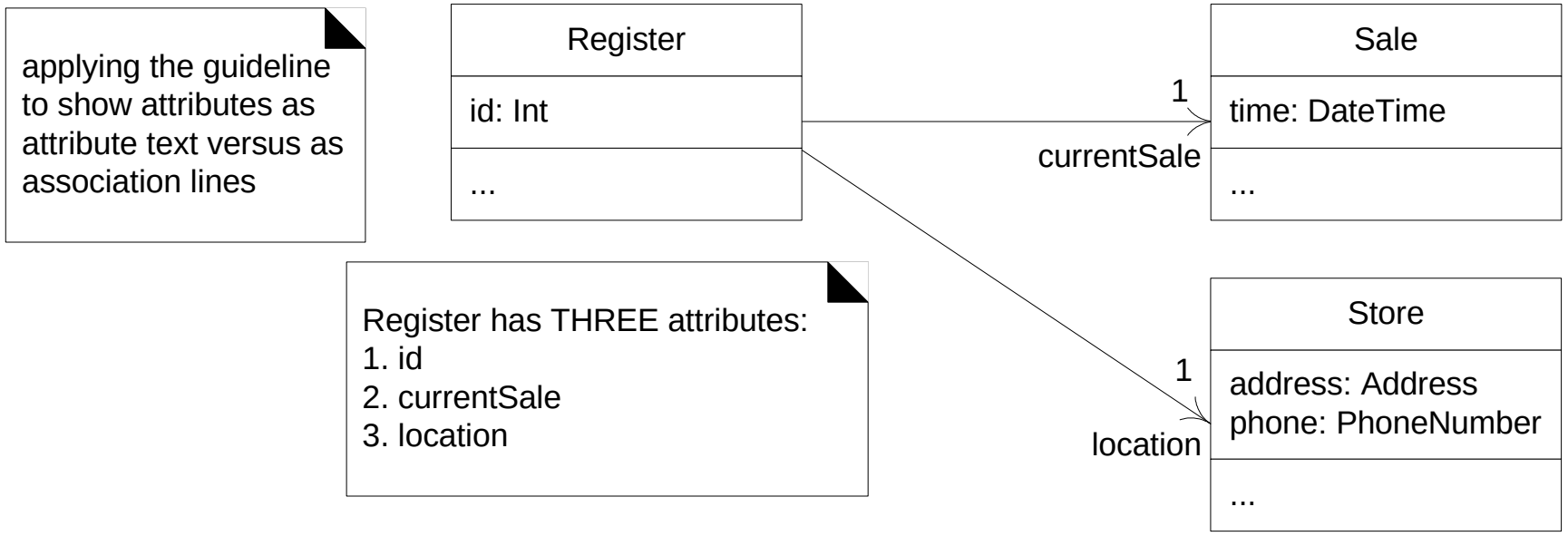
UML-Class Diagram

[Larman, 2005] Fig. 16.2



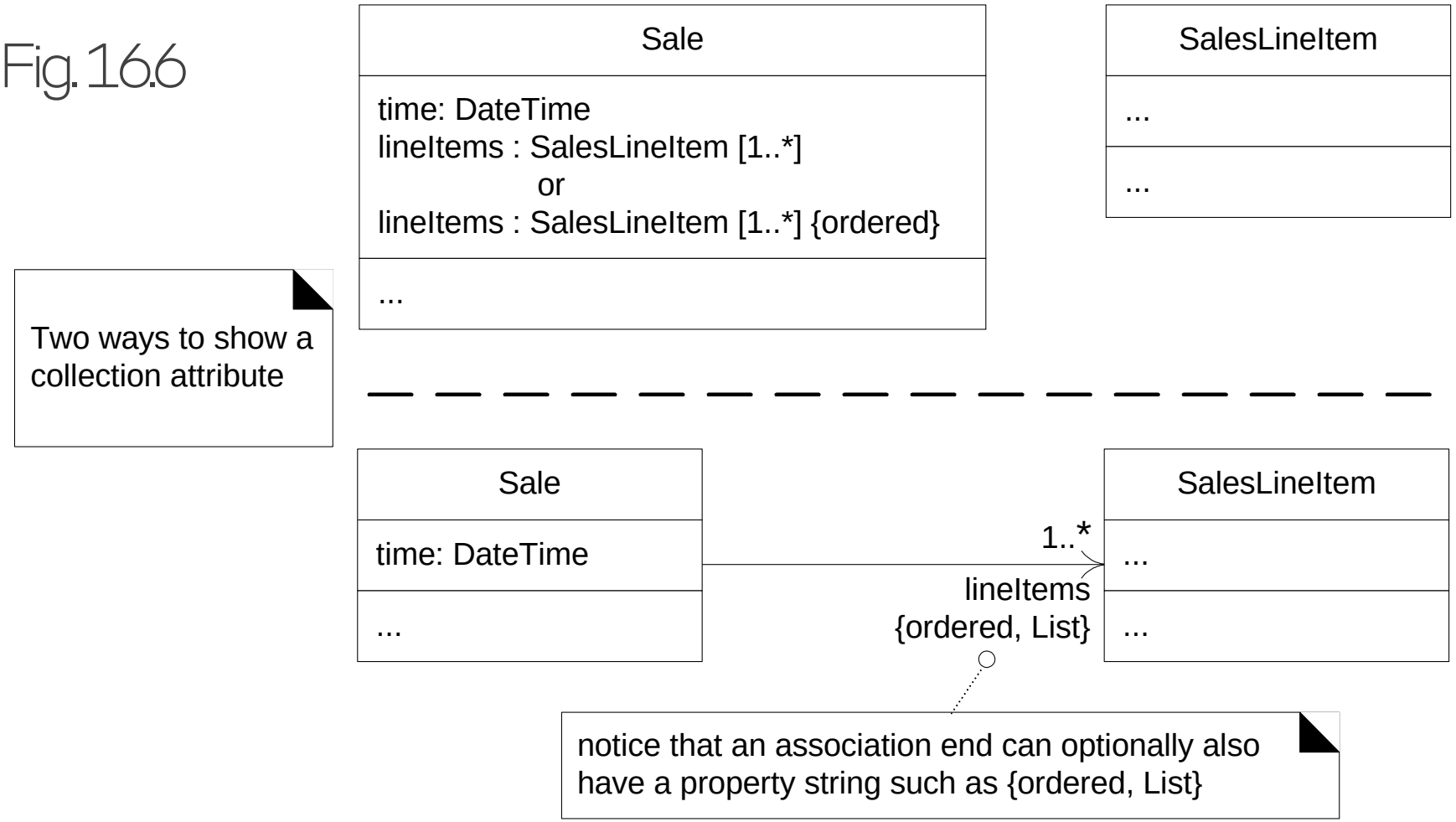
UML-Class Diagram

[Larman, 2005] Fig. 16.5



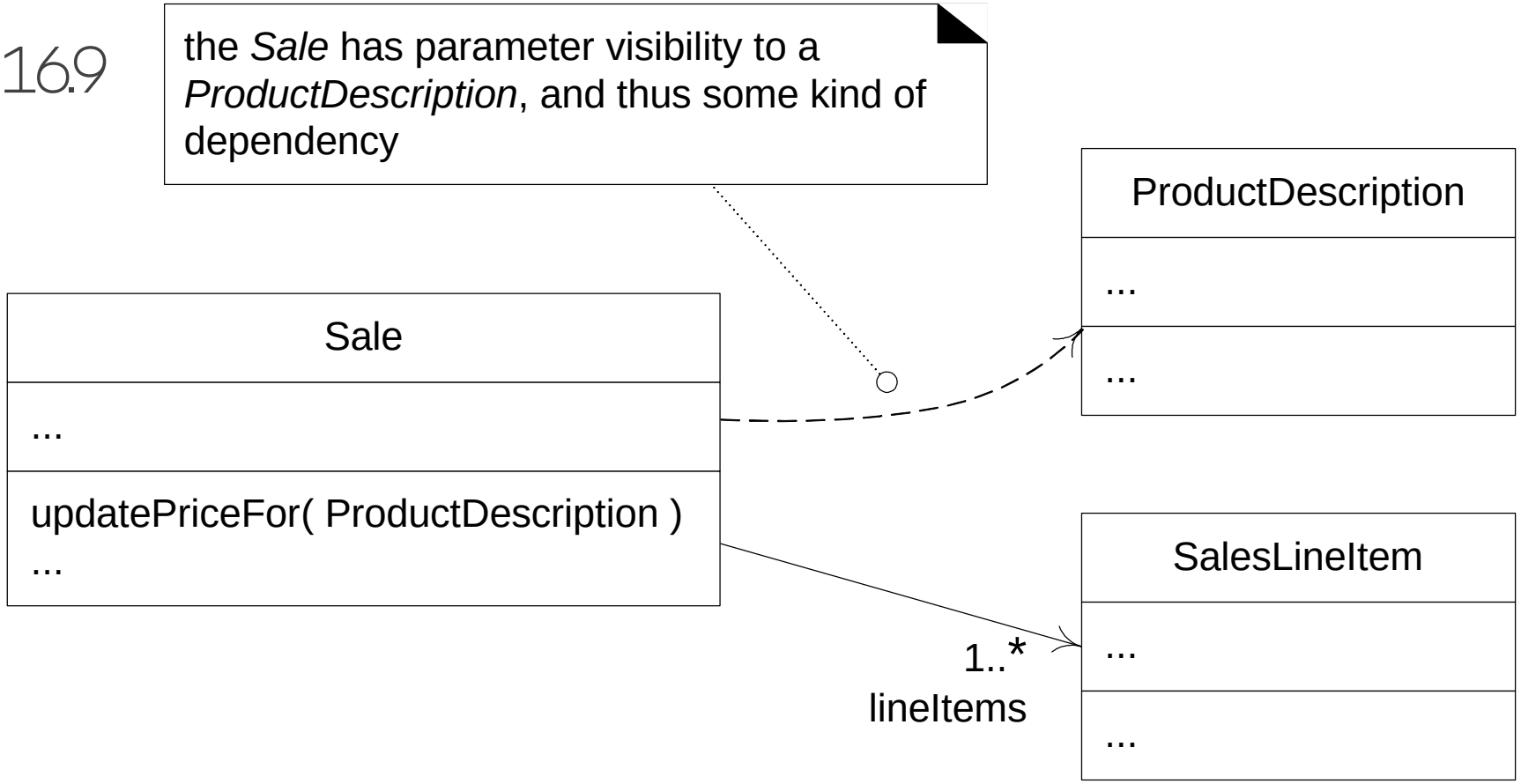
UML-Class Diagram

[Larman, 2005] Fig. 16.6



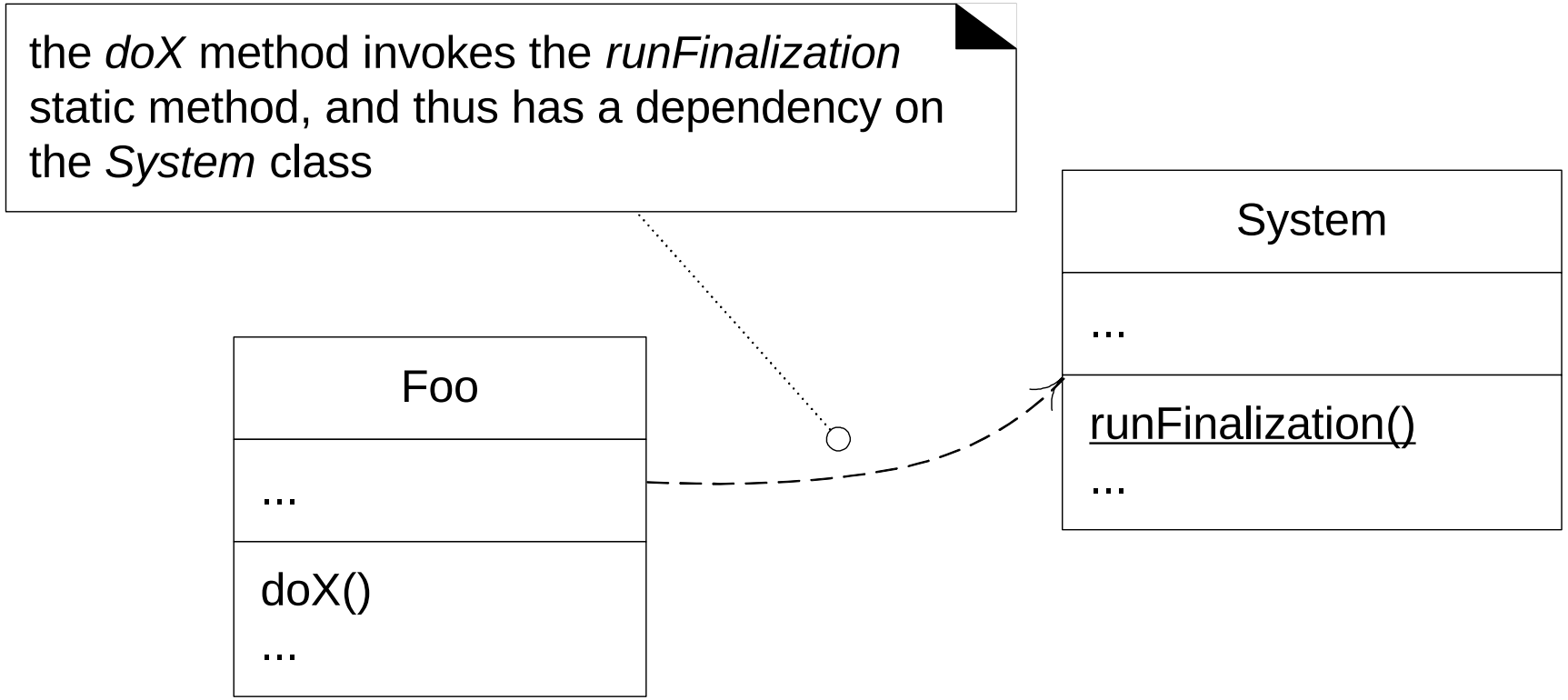
UML-Class Diagram

[Larman, 2005] Fig. 16.9



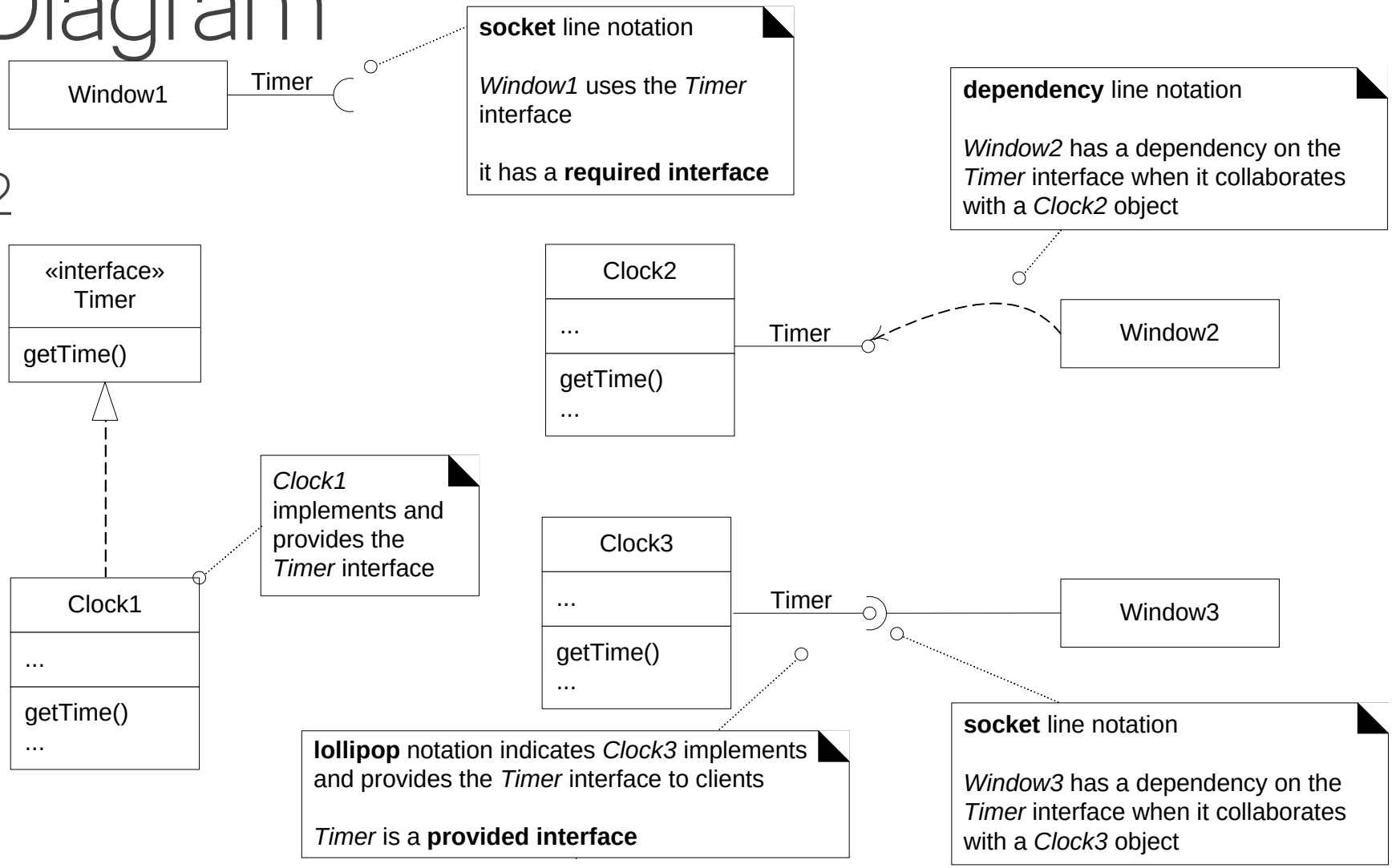
UML-Class Diagram

[Larman, 2005] Fig. 16.10



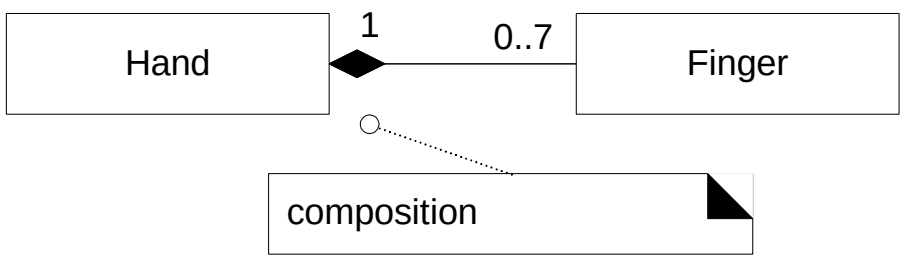
UML-Class Diagram

[Larman, 2005] Fig. 16.12



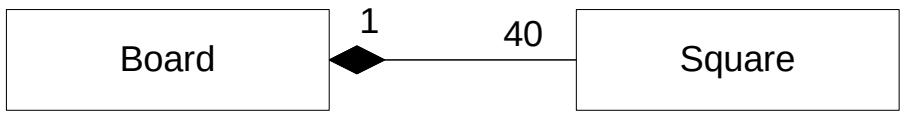
UML-Class Diagram

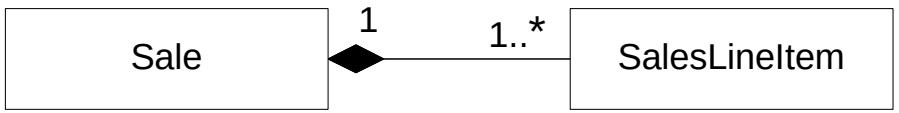
[Larman, 2005] Fig. 16.13



composition means
-a part instance (*Square*) can only be part of one composite (*Board*) at a time

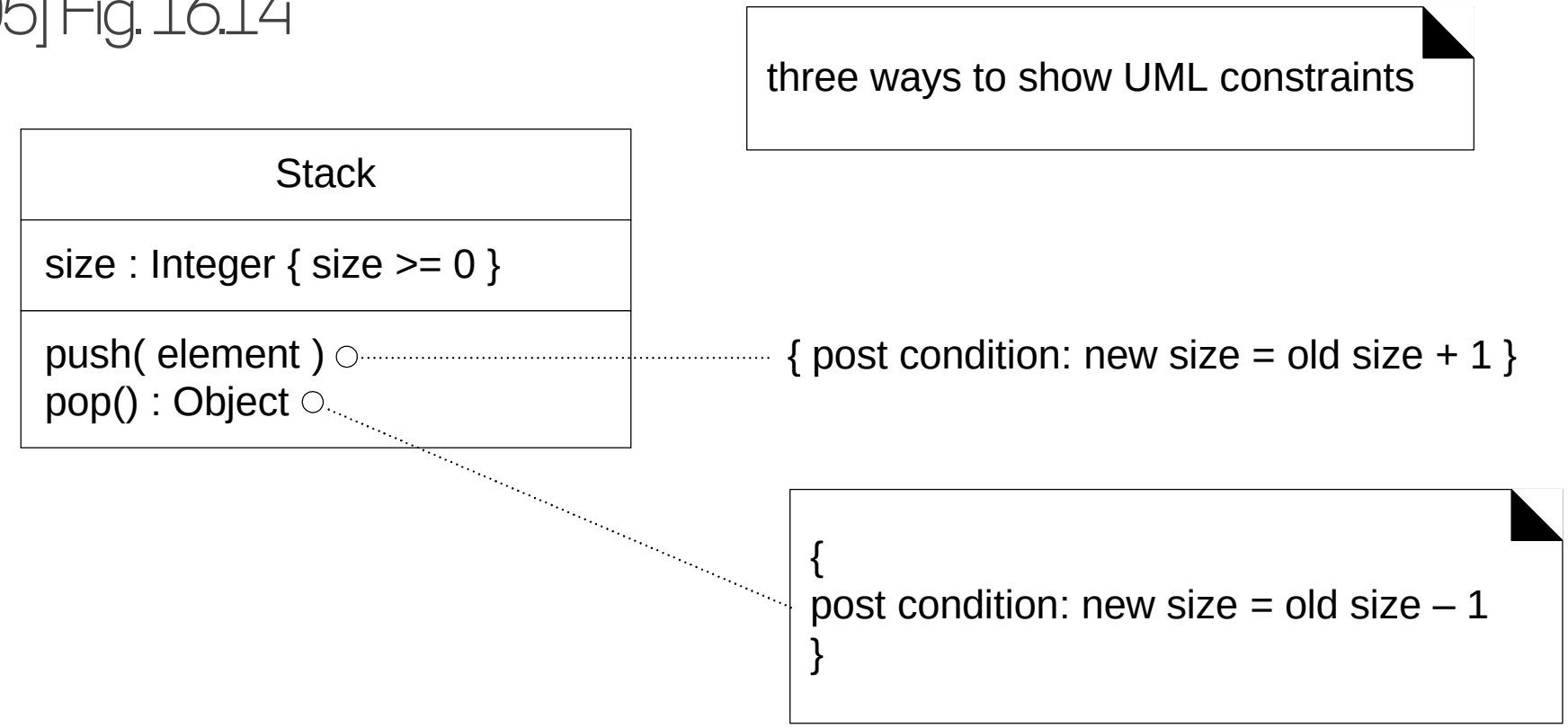
-the composite has sole responsibility for management of its parts, especially creation and deletion





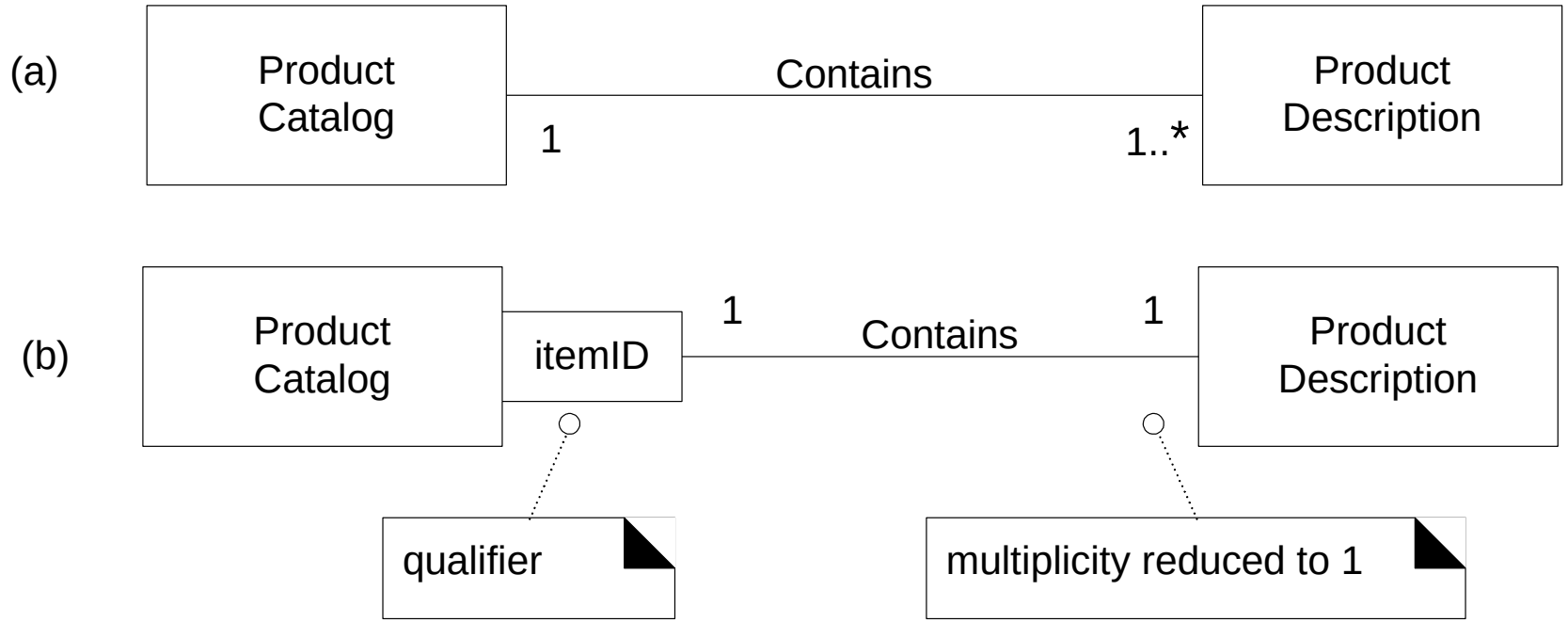
UML-Class Diagram

[Larman, 2005] Fig. 16.14



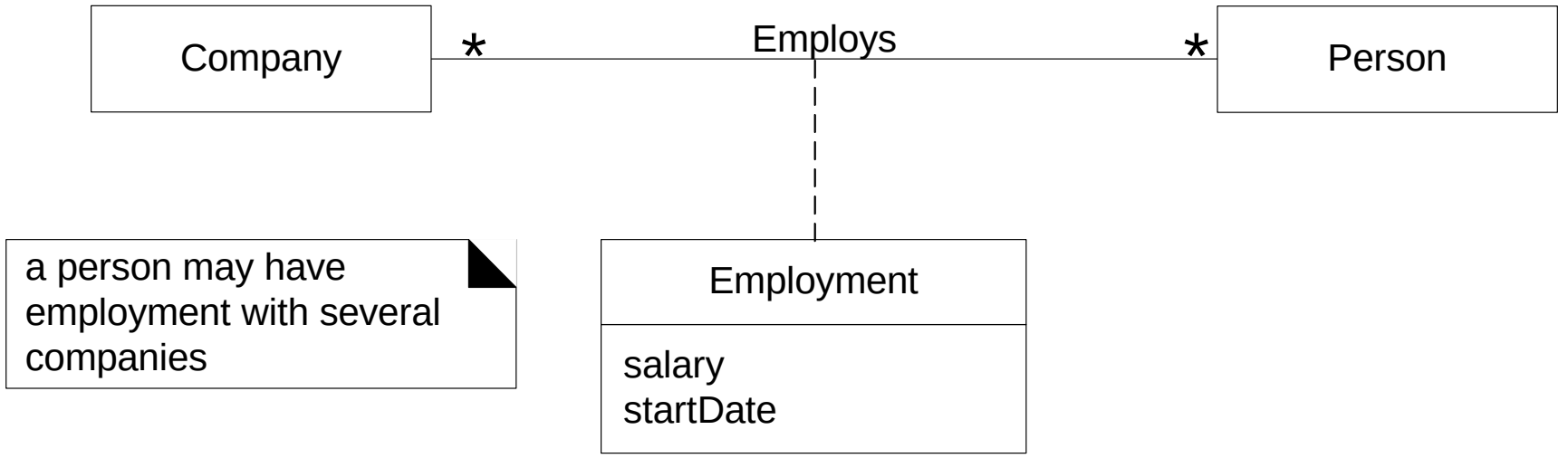
UML-Class Diagram

[Larman, 2005] Fig. 16.15



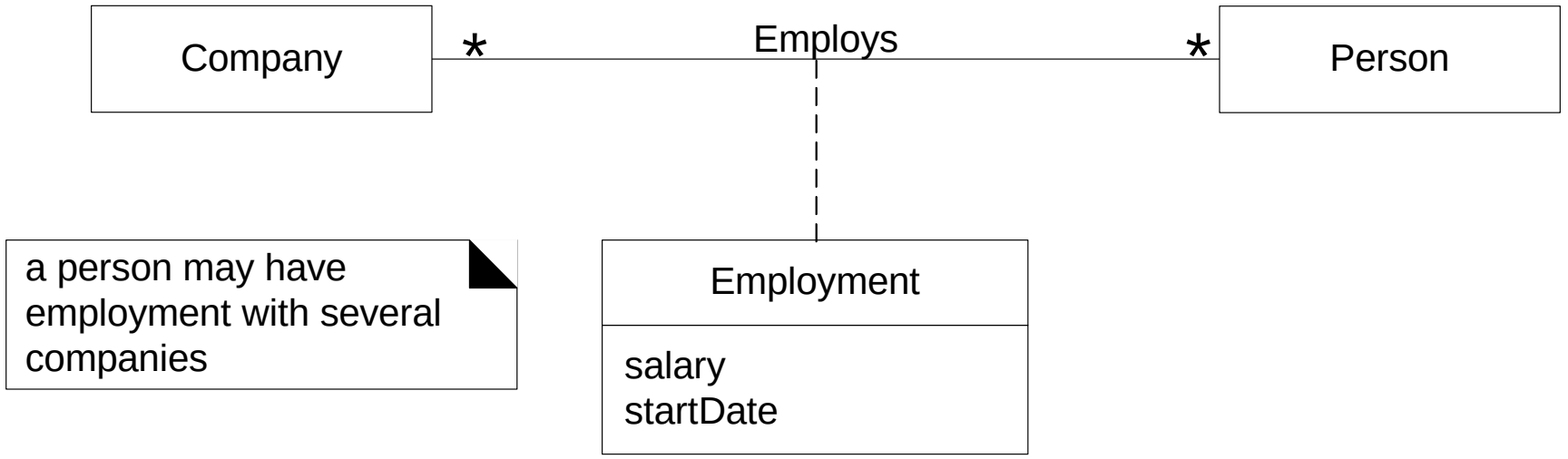
UML-Class Diagram

[Larman, 2005] Fig. 16.16



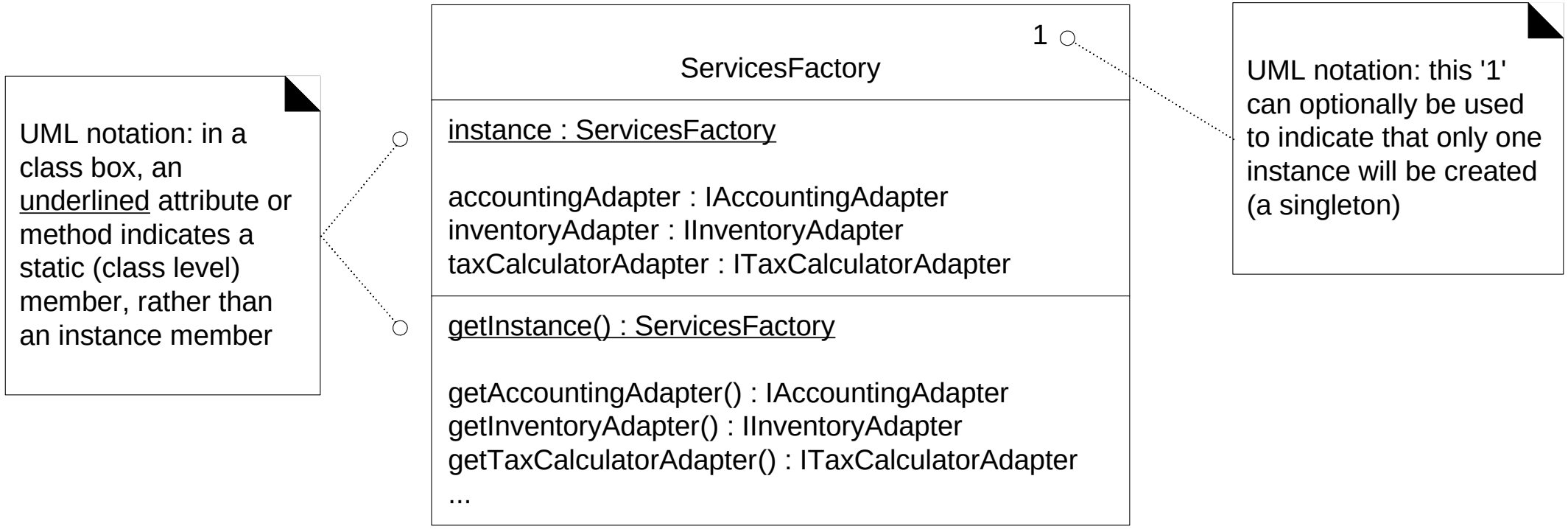
UML-Class Diagram

[Larman, 2005] Fig. 16.16



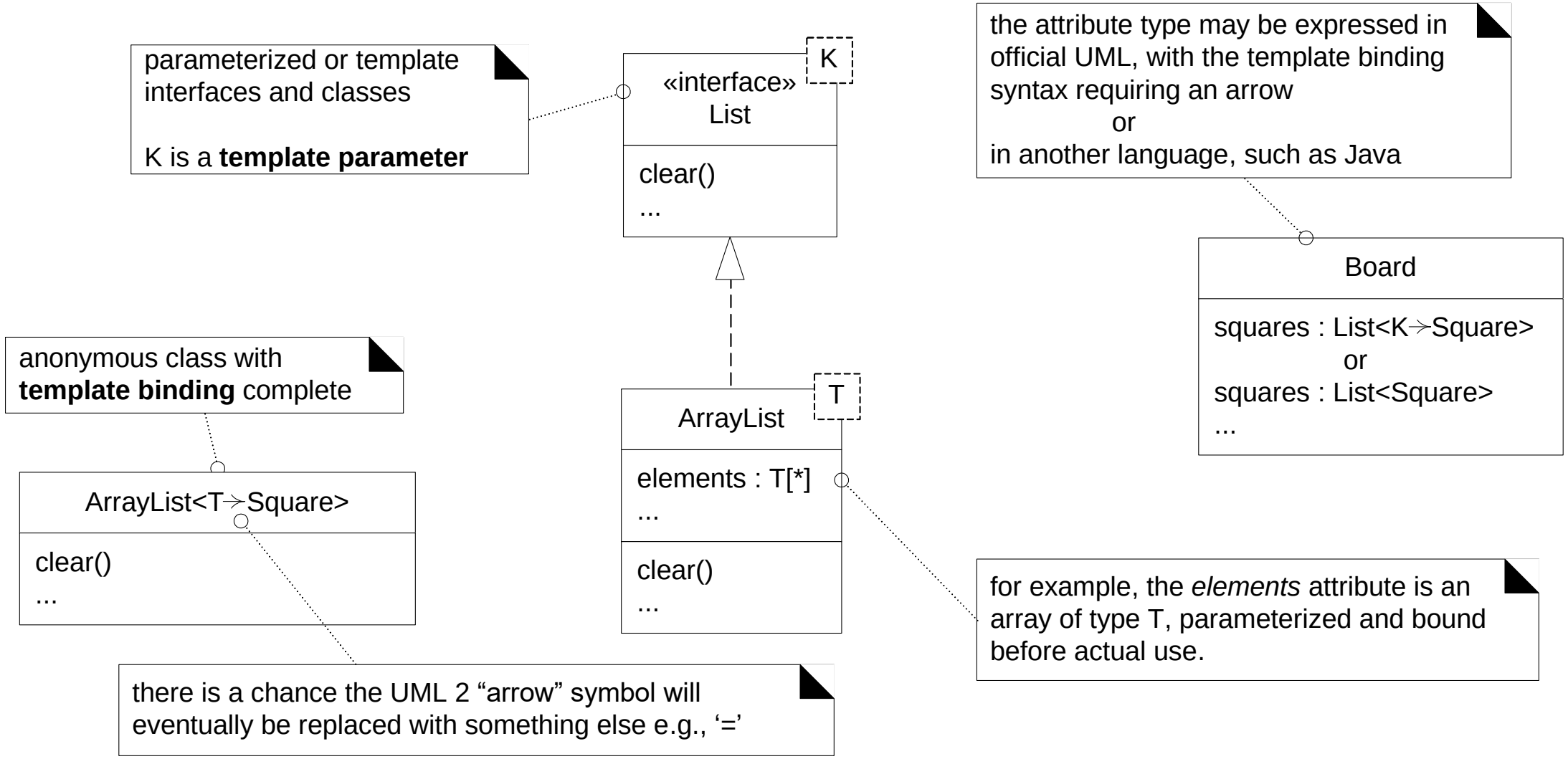
UML-Class Diagram

[Larman, 2005] Fig. 16.17



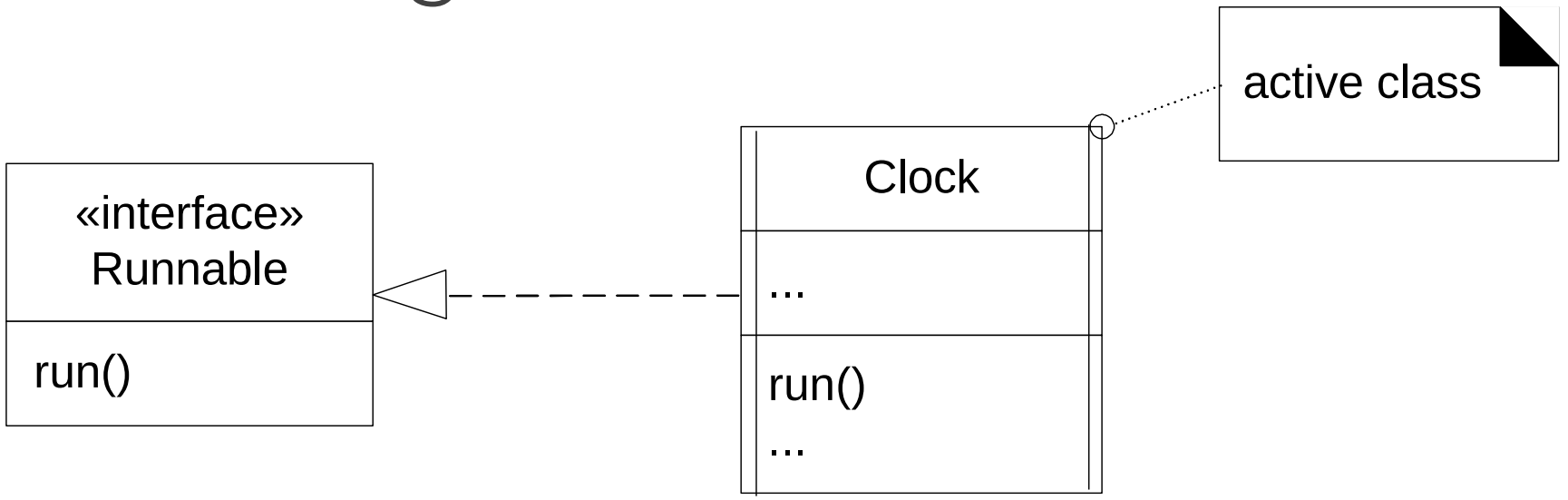
UML-Class Diagram

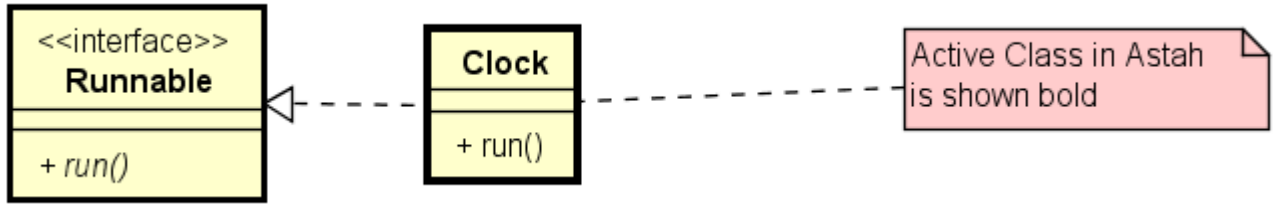
[Larman, 2005] Fig. 16.18



UML-Class Diagram

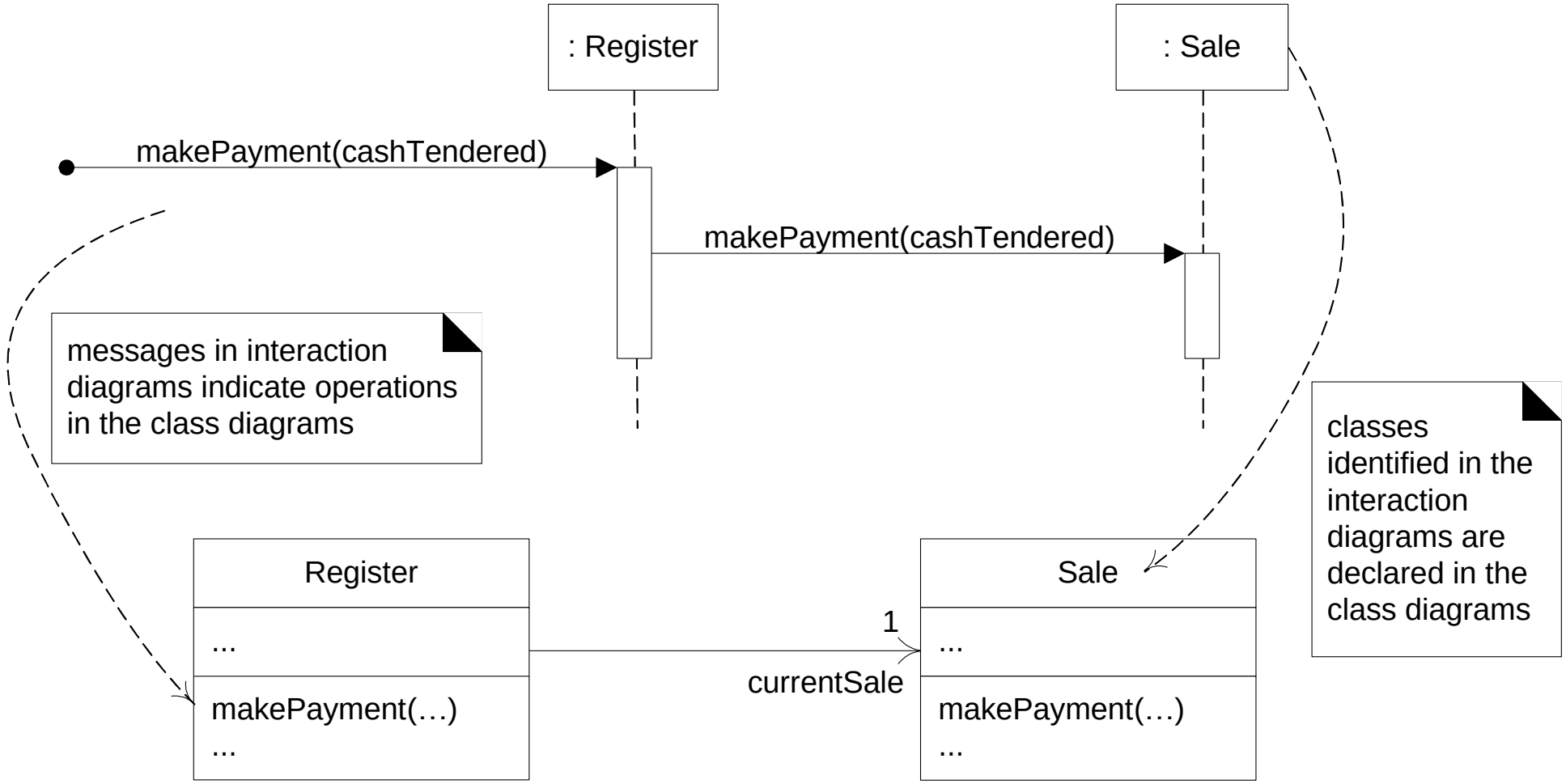
[Larman, 2005] Fig. 16.20





UML-Class Diagram

[Larman, 2005] Fig. 16.21



Exercise

1. Implement the diagrams shown in [Larman, 2005] figure 16.9, 16.13 and 16.17 in Java - just with empty methods

2. Implement the class diagram to the right – just with empty methods except what can be seen in the diagram