CSE291 - Introduction To Software Engineering
(Fall 2018)

Lecture 13

Class Diagrams

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.

Class diagrams

- Class diagrams in the UML can be expressed at different levels of detail.
- When you are developing a model, the first stage is usually to look at your proposed system, identify the essential objects, and represent these as classes.
- The simplest way of writing these is to write the class name in a box.
  - Show the existence of an association by drawing a line between classes.

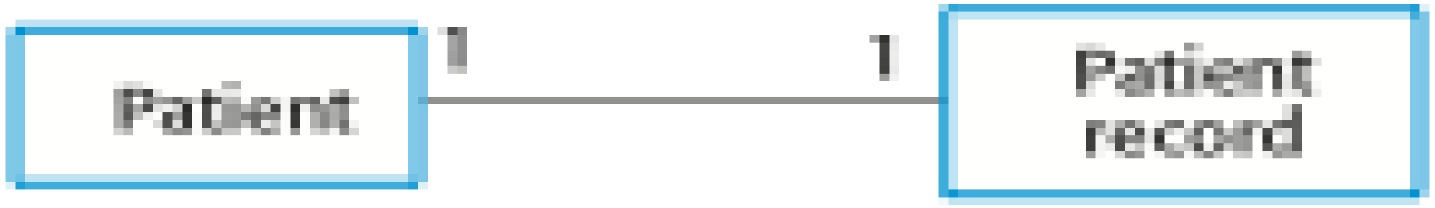
UML Classes and association

- A simple class diagram showing two classes: Patient and Patient Record with an association between them

- In this example, each end of the association is annotated with a 1, meaning that there is a 1:1 relationship between objects of these classes

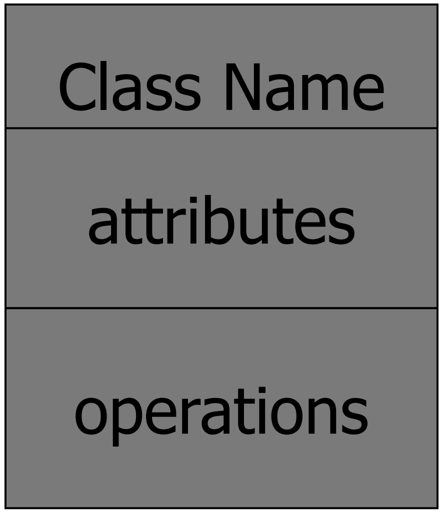
.

- That is, each patient has exactly one record and each record maintains information about exactly one patient.



Classes

The class diagram also describes/show the attributes and operations of a class.



A *class* is a description of a set of objects that share the same attributes, Operations and relationships.

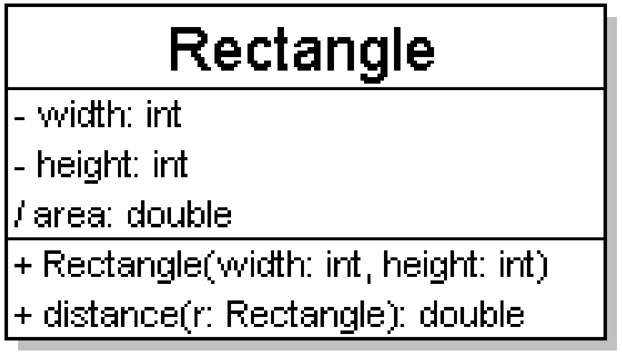
In UML-

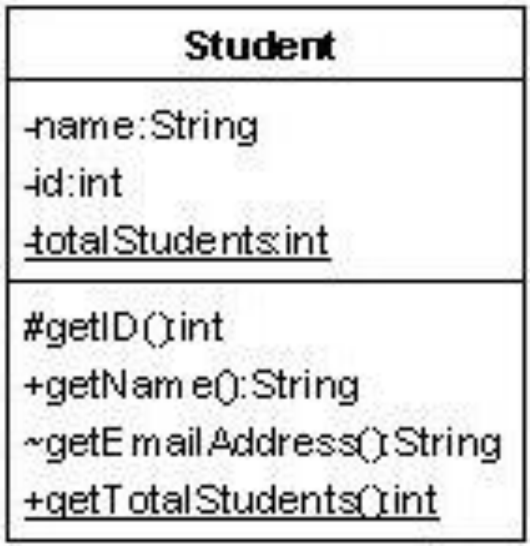
An object class is represented as rectangle, usually including its name, attributes, and operations in separate, designated compartments.

Diagram of One Class

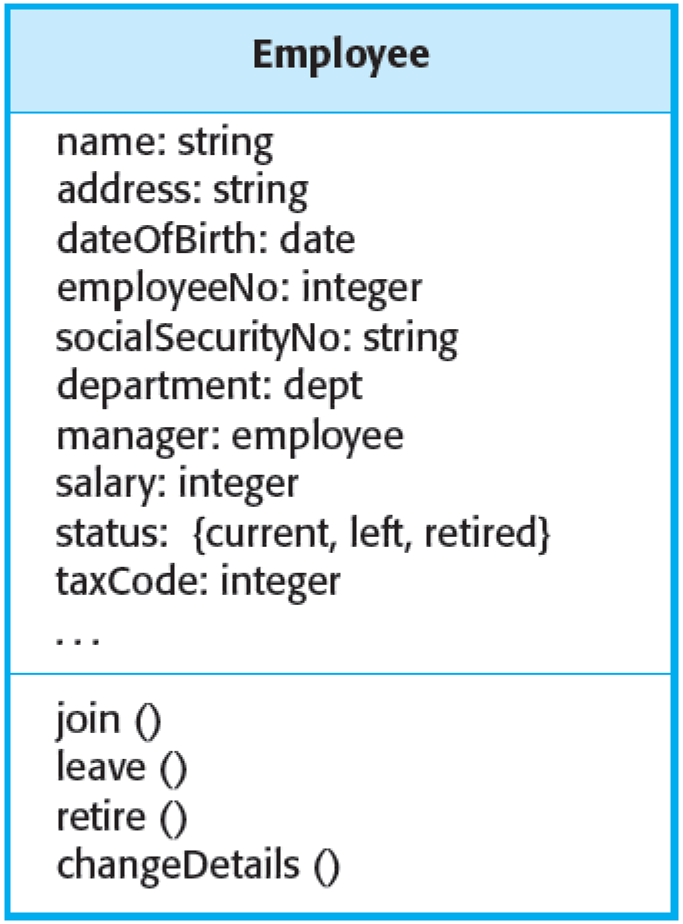
- Class name in top of box
  - write <<interface>> on top of interfaces' names
  - use *italics* for an *abstract class* name

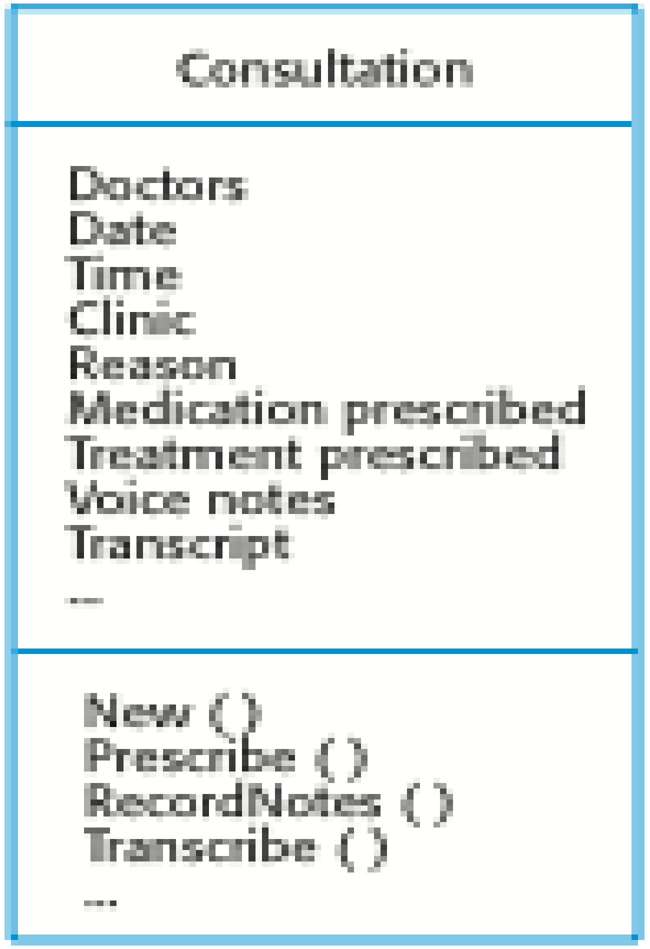
- attributes (optional)
- operations / methods (optional)


Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double



Examples

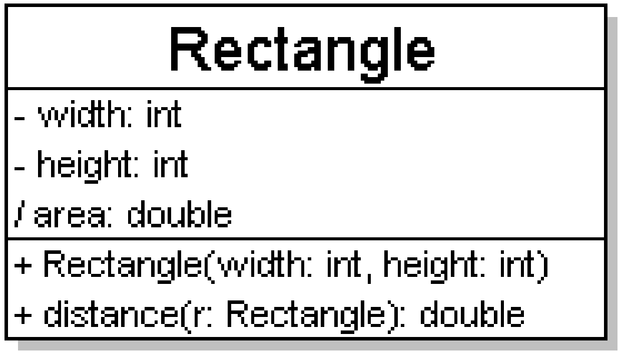

Employee
name: string
address: string
dateOfBirth: date
employeeNo: integer
socialSecurityNo: string
department: dept
manager: employee
salary: integer
status: {current, left, retired}
taxCode: integer
. . .
join ()
leave ()
retire ()
changeDetails ()

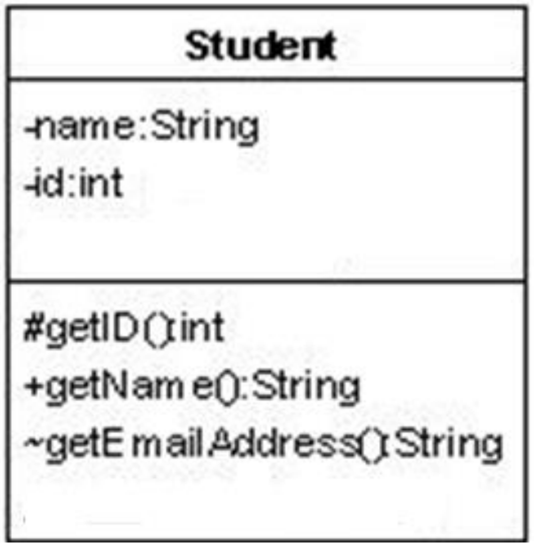

Consultation
Doctors
Date
Time
Clinic
Reason
Medication prescribed
Treatment prescribed
Voice notes
Transcript
...
New ( )
Prescribe ( )
RecordNotes ( )
Transcribe ( )
...

Class Attributes

- attributes
  - *name* : *type*
  - visibility:

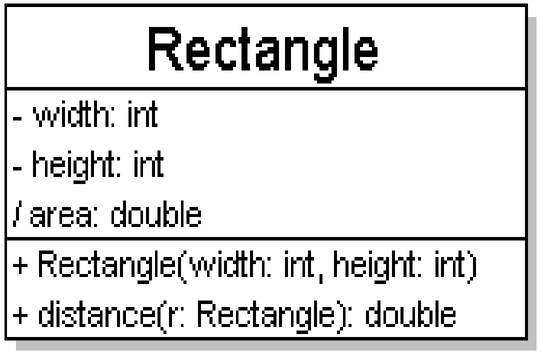
+	public
#	protected
-	private
/	derived

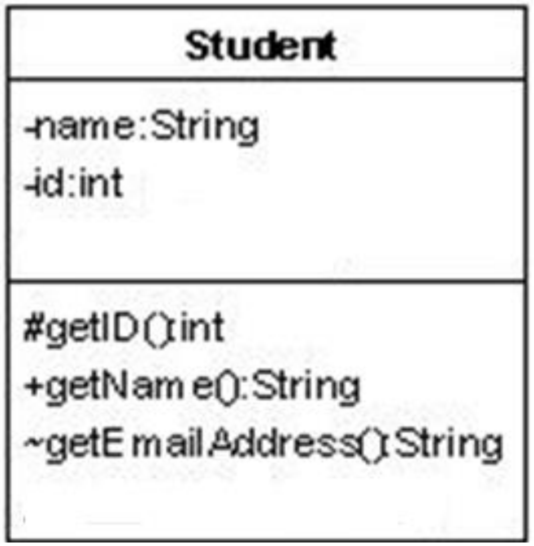

Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double



Class Operations / Methods

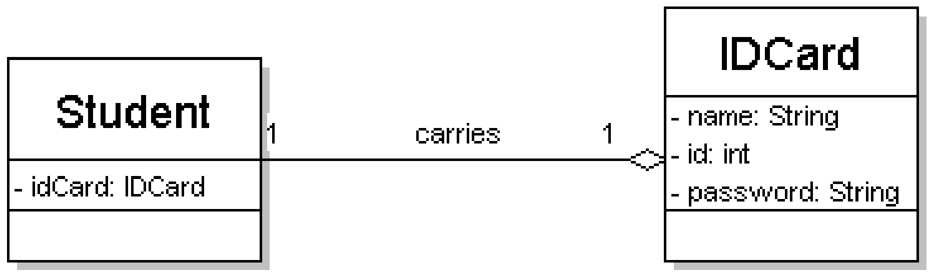
- operations / methods
  - *name* (*parameters*) : *return type*
  - visibility: + public
    # protected
    - private
  - parameter types listed as (name: type)


Rectangle
- width: int
- height: int
/ area: double
+ Rectangle(width: int, height: int)
+ distance(r: Rectangle): double

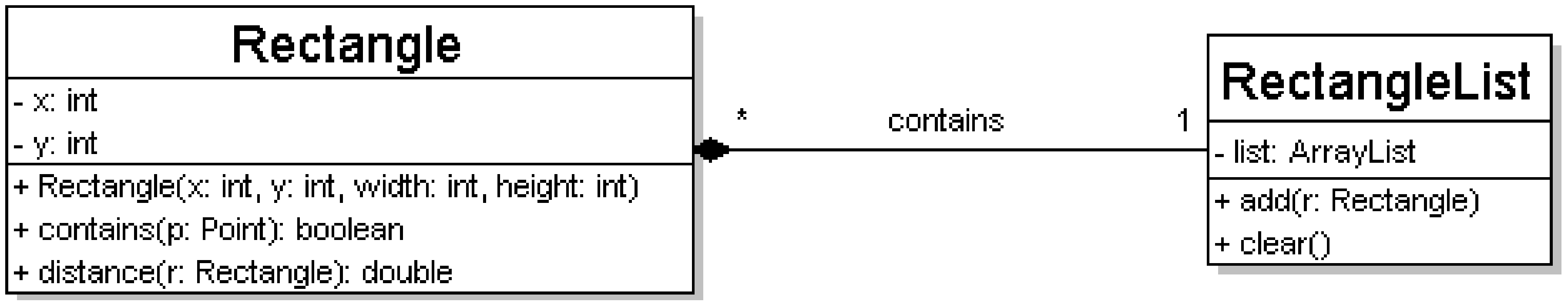


Multiplicity of Associations

- one-to-one
  - each student must carry exactly one ID card

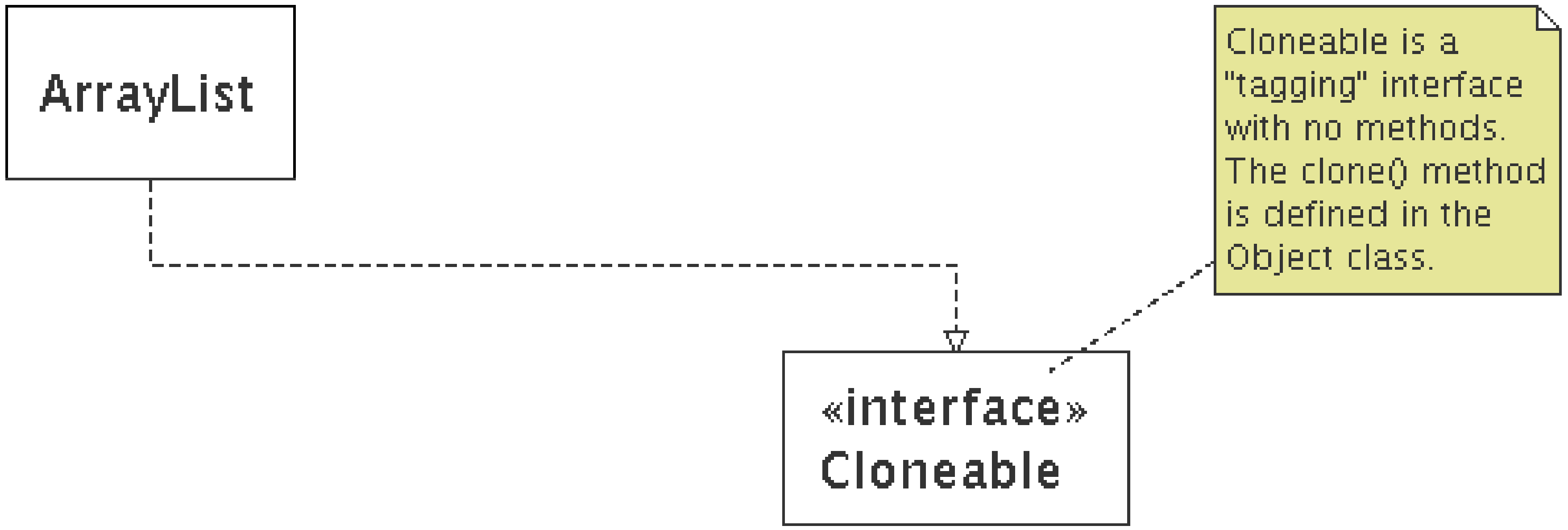


- one-to-many
  - one rectangle list can contain many rectangles


Rectangle
- x: int
- y: int
+ Rectangle(x: int, y: int, width: int, height: int)
+ contains(p: Point): boolean
+ distance(r: Rectangle): double
*
contains
1
RectangleList
- list: ArrayList
+ add(r: Rectangle)
+ clear()

Comments

- Represented as a folded note, attached to the appropriate class/method/etc by a dashed line


ArrayList
«interface» Cloneable
Cloneable is a "tagging" interface with no methods. The clone() method is defined in the Object class.

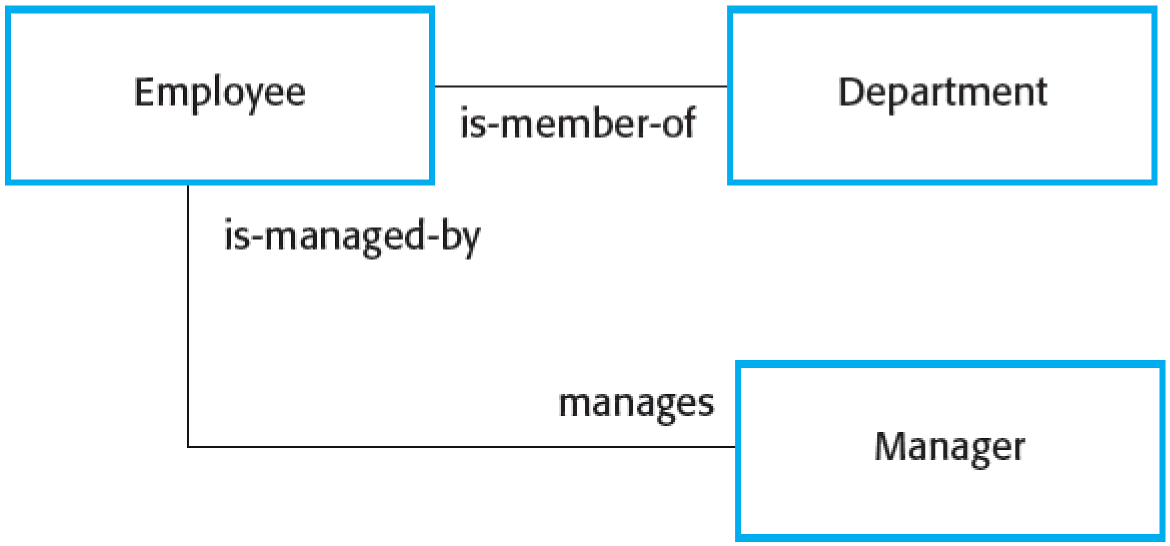
Relationships

Association

- Objects that are members of an object class participate in relationships with other objects.
- These relationships modeled by describing the associations between the object classes.
- In the UML, associations are denoted by a line between

  the object classes that may optionally be annotated with information about the association.

Association Model



Generalization

- Generalization is an everyday technique that we use to manage complexity.
- Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, etc.) and learn the characteristics of these classes.
- This allows us to infer that different members of these classes have some common characteristics

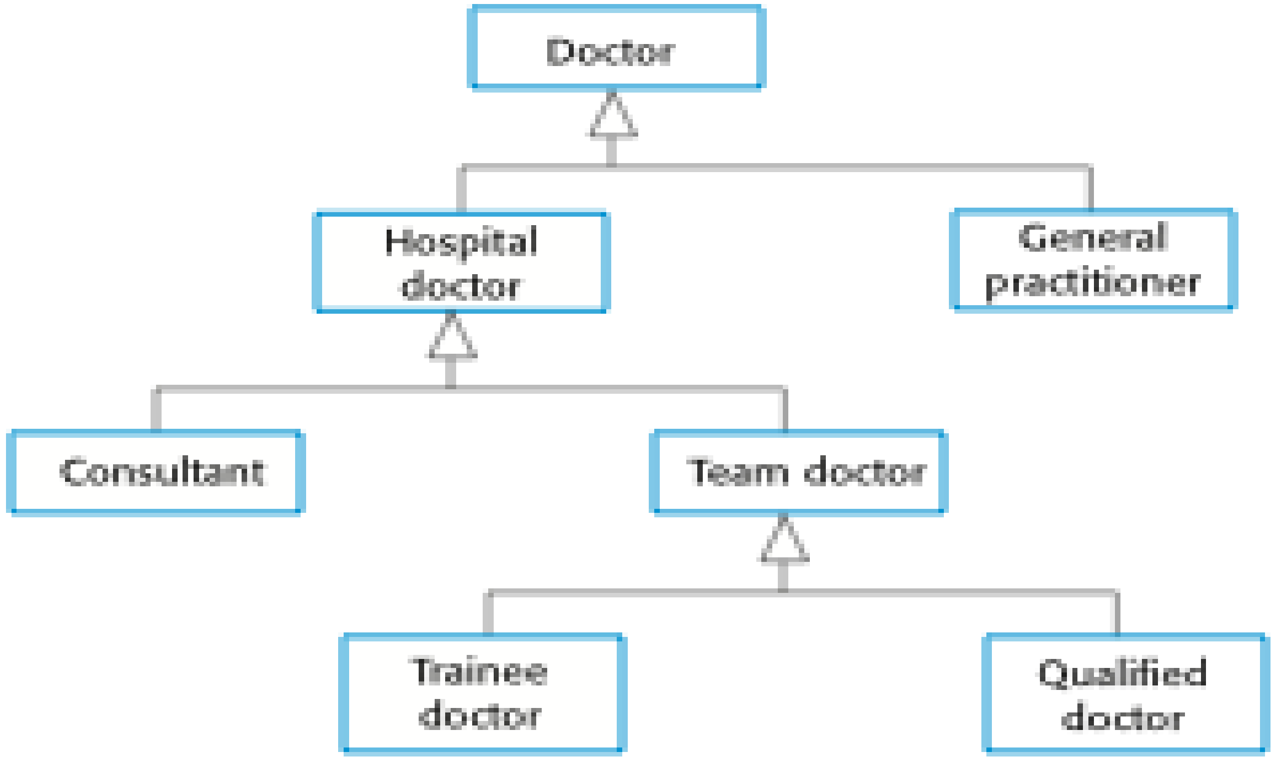
Generalization

- In object-oriented languages, such as Java, generalization is implemented using the class inheritance mechanisms built into the language.

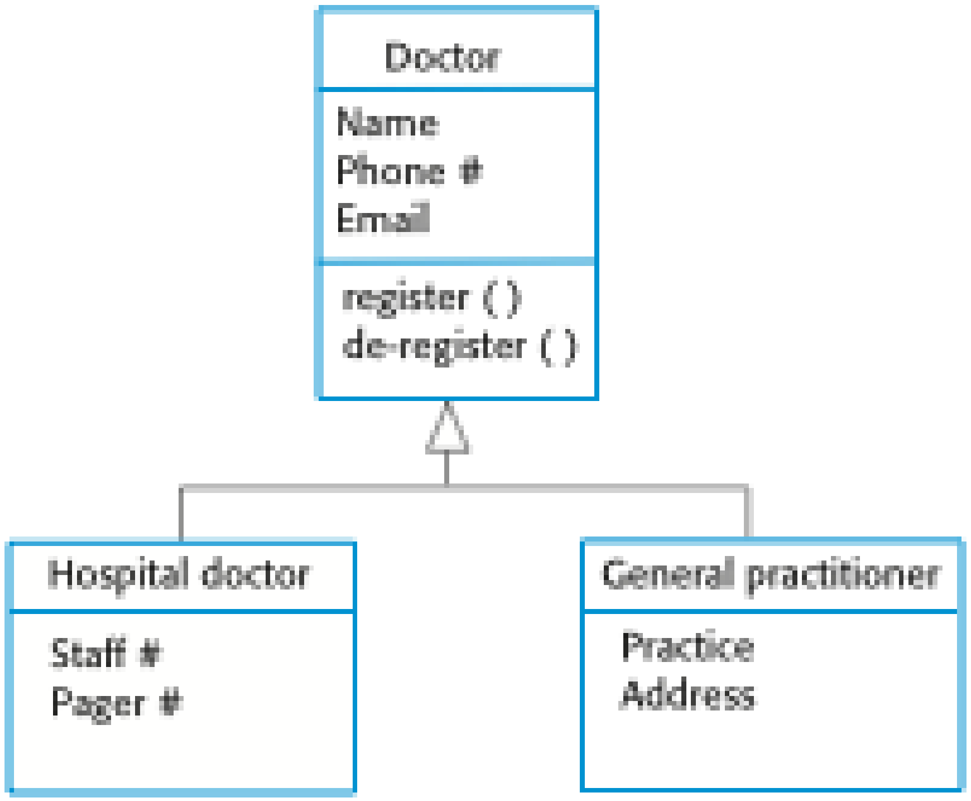
- In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level classes.

- The lower-level classes are subclasses inherit the attributes and operations from their super classes. These lower-level classes then add more specific attributes and operations.

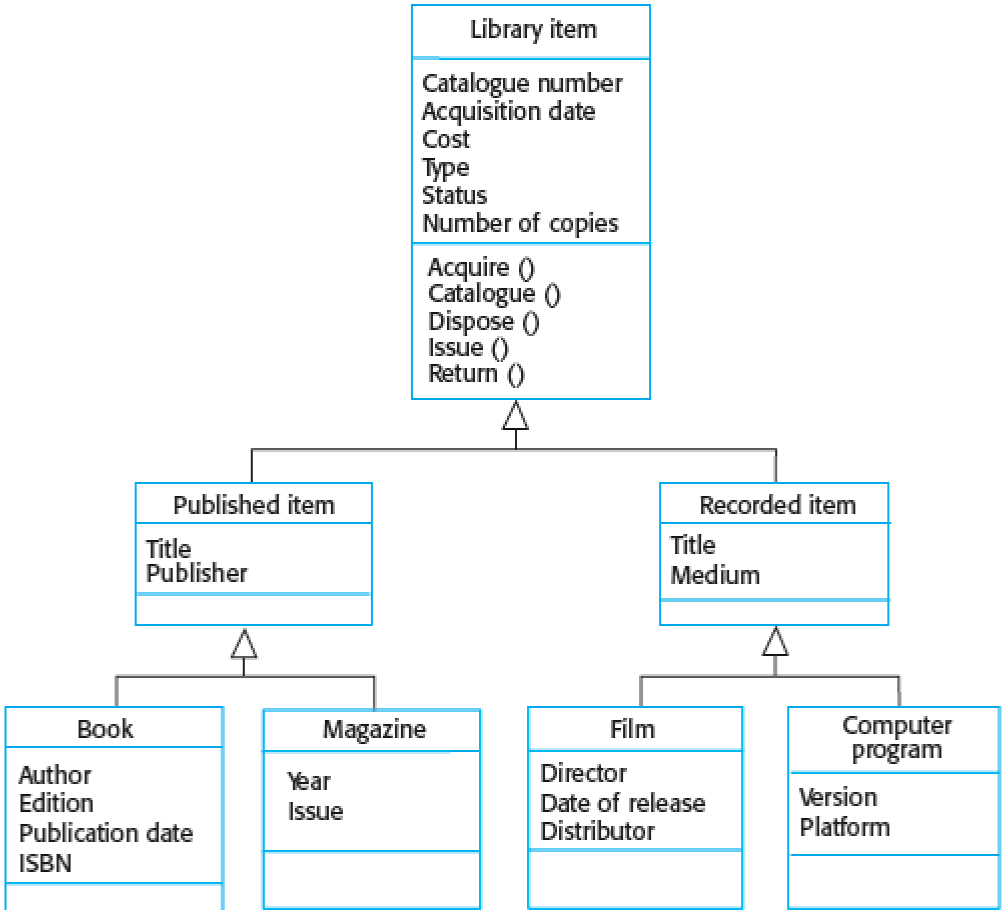
A generalization hierarchy



A Generalization hierarchy with added detail



Library Class Hierarchy

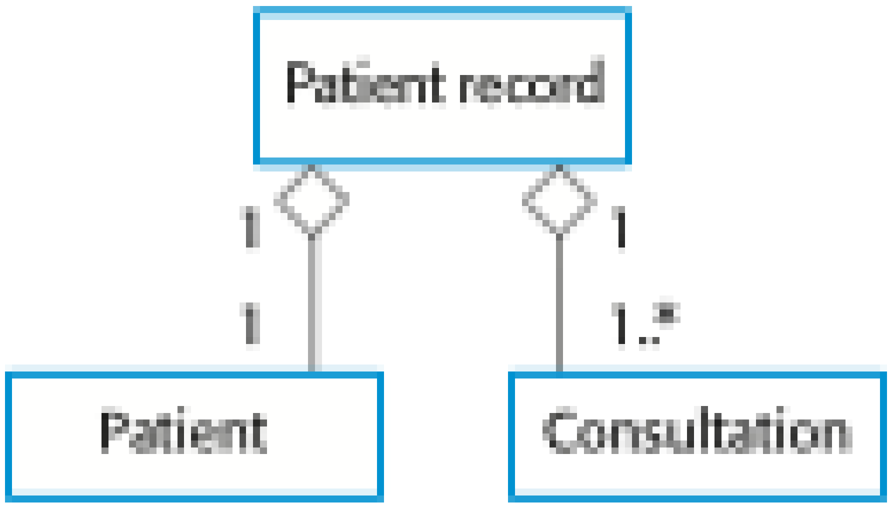


Object class aggregation models

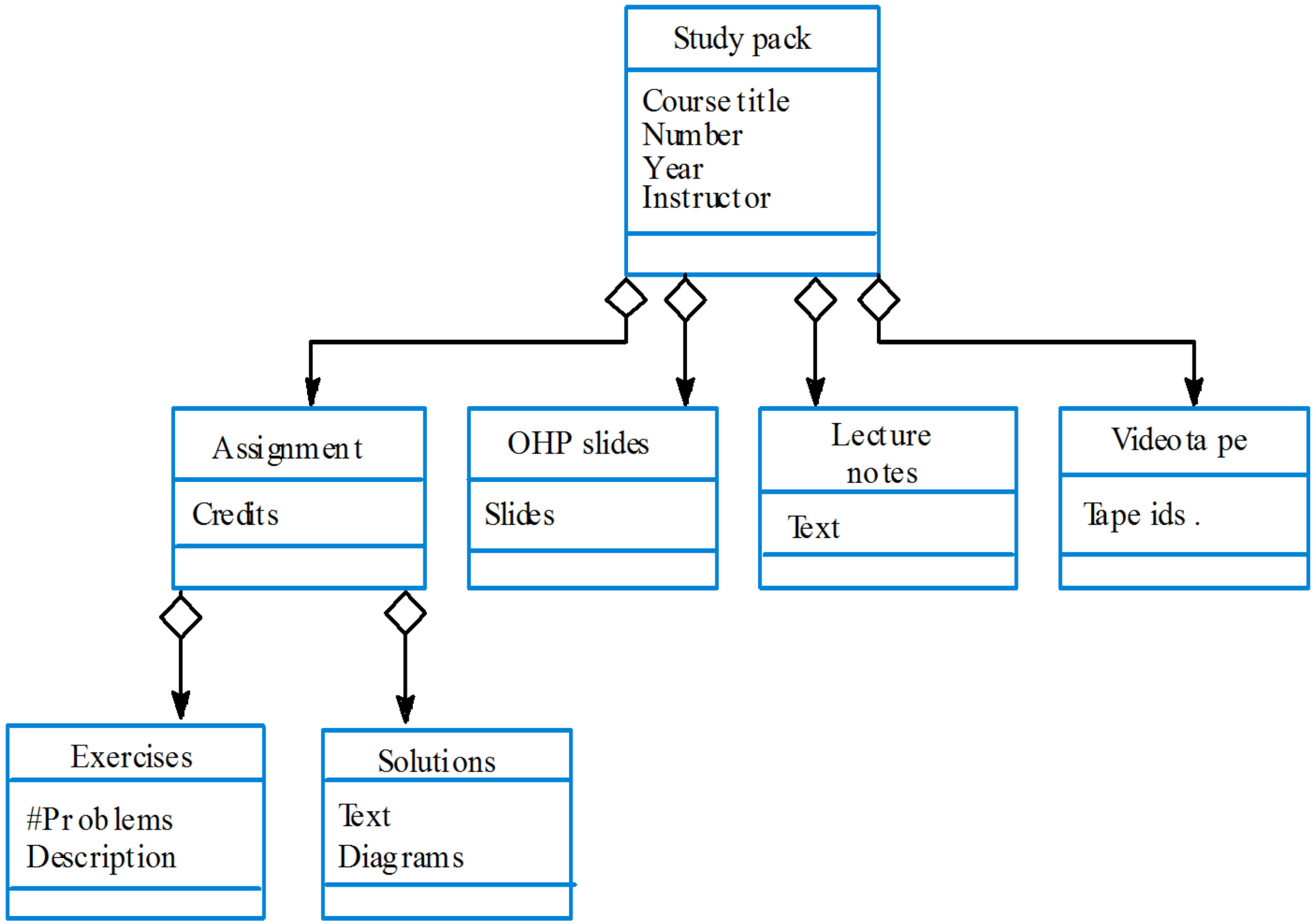
- Objects in the real world are often composed of different parts.
- For example, a study pack for a course may be composed of a book, PowerPoint slides, quizzes, and recommendations for further reading. Sometimes in a system model, you need to illustrate this.
- The UML provides a special type of association between classes called aggregation that means that one object (the whole) is composed of other objects (the parts).
- To show this, we use a diamond shape next to the class that represents the whole.

The aggregation association

This is shown in Figure, which shows that a patient record is a composition of Patient and an indefinite number of Consultations.



Object Aggregation

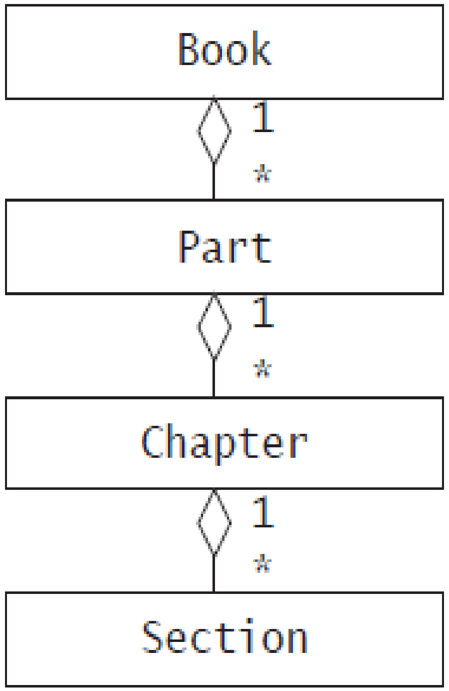


Exercise 1

Draw a class diagram representing a book defined by the following statement:

“A book is composed of a number of parts, which in turn are composed of a number of chapters. Chapters are composed of sections.”

Solution

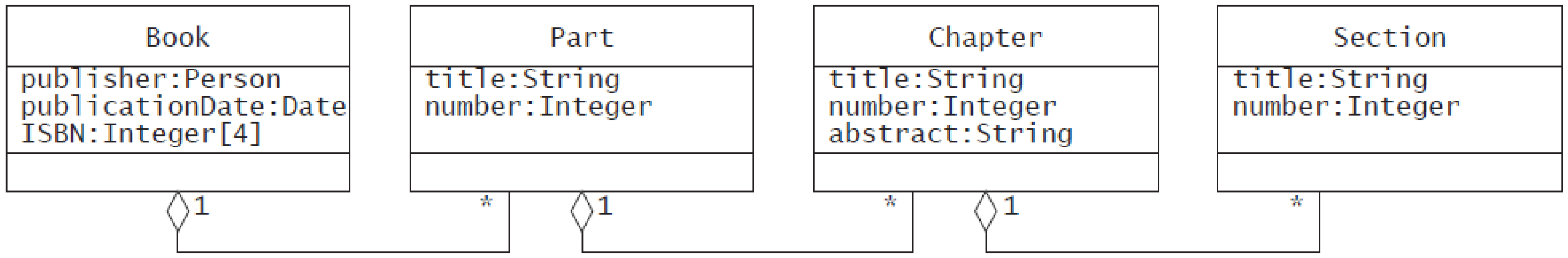


Exercise 2

Apply following activities on the class diagram that you drew in A part.

- **Book** includes a publisher, publication date, and an ISBN
- **Part** includes a title and a number
- **Chapter** includes a title, a number and an abstract

Solution


Book
publisher:Person
publicationDate:Date
ISBN:Integer[4]
1
*
Part
title:String
number:Integer
1
*
Chapter
title:String
number:Integer
abstract:String
1
*
Section
title:String
number:Integer

UML Class Diagram Exercise

In a university there are different classrooms, offices and departments. A department has a name and it contains many offices. A person working at the university has a unique ID and can be a professor or an employee.

- A professor can be a full, associate or assistant professor and he/she is enrolled in one department.
- Offices and classrooms have a number ID, and a classroom has a number of seats.
- Every Employee works in an office