Introduction
This module discusses issues related to Object-Oriented design.

Objectives
- To learn about the software life cycle
- To learn how to discover new classes and methods
- To understand the use of CRC cards for class discovery
- To be able to identify inheritance, aggregation, and dependency relationships between classes
- To master the use of UML class diagrams to describe class relationships
- To learn how to use object-oriented design to build complex programs

Reading
Text book:
   Chapter 12: Object-Oriented Design (whole chapter)

Review questions
Review exercises:

Programming exercises:
   Implement above R12.12

References
Cay Horstmann, CH 2007, Big Java, 3rd Edition
Lab session

1) Discuss the review exercises

R12.1
The software life cycle comprises all activities between the initial conception of a software product and its final retirement.

R12.2
This book recommends the following object-oriented development process:

1. Discover classes
2. Determine the responsibilities of each class
3. Determine the relationships between the classes

R12.3
Classes correspond to *nouns* in the task description.

R12.4
Methods correspond to *verbs* in the task description.

R12.5
In an object-oriented programming language such as Java, a method must belong to a class, namely the class that is responsible for the method.

R12.6
- University-Student: aggregation; a university *has* students
- Student-TeachingAssistant: inheritance; a teaching assistant *is* a student
- Student-Freshman: inheritance; a freshman *is* a student
- Student-Professor: no relationship
- Car-Door: aggregation; a car *has* doors
- Truck-Vehicle: inheritance; a truck *is* a vehicle
- Traffic-TrafficSign: no relationship
- TrafficSign-Color: no relationship. (Color is an *attribute*)
The class BMW can inherit from the class Vehicle—every BMW is a vehicle. However, the BMW company (which is different from the class of all BMWs) is an object of the class VehicleManufacturer. Inheritance is not appropriate for describing the relationship between an object and a class.
2) Discovering Classes

a. You have been given an assignment to write a program to schedule classrooms in your college's building. Classrooms can be reserved for classes, seminars, guest speakers, and for meetings for clubs (such as the Java Users Group). Clubs can meet only after 5 p.m. and only if the room is not needed by any other group. Each group needing a classroom is given a priority. Classes in your college have the highest priority, with clubs having the lowest priority. Some classes in your college have higher priority over others, especially if the instructor has tenure. Classrooms are characterized by size, amenities (overhead projectors, etc.) and comfortable seats. Groups can specifically request a room by room number or simply give a description of what they need. Groups also may request specific time slots or give a range of acceptable time slots. Rooms are not available after 10 p.m. and before 5 a.m.

Make a list of classes that you may need for your program.

b. From your list of potential classes in your classroom-scheduling program, pick three potential classes and identify their main responsibilities.

c. A CRC card describes a class, its responsibilities, and its collaborating classes. You already identified the classes and responsibilities in the classroom-scheduling program. Now you will identify the collaborating classes.

From your list of potential classes in your classroom-scheduling program, pick three potential classes and identify their collaborators.

3) Relationships Between Classes

d. Using the list of classes for your classroom-scheduling program, identify any objects that may use inheritance. List the superclasses (if any) followed by their subclasses. If you see no need for inheritance, explain.

e. Inheritance can be described as an "is-a" relationship between classes. Aggregation can be described as a "has-a" relationship. Identify any aggregation relations in the classroom scheduling problem.
Identify the relationships between classes in the following code:

class GroceryBag extends BasicContainer {
    private Fruit[] fruits;
    private Vegetable[] veggies;
    
    public void addItem(Food item) {
        
    }
    public Money computeTotal() {
        
    }

    
}

**Inheritance**
From which classes does GroceryBag inherit?

**Aggregation**
Identify any aggregation relationships (has-a relationship).

**Dependency**
On which classes does GroceryBag depend (uses relationship)?
Solution to a.

BuildingSchedule
ScheduleItem
Classroom
Amenity
Seat
Schedulable
Class
Seminar
Club
GuestSpeakerPresentation

Solution to b.

BuildingSchedule
  print schedule
  schedule by room
  schedule by time slot
  schedule by amenities
  get room schedule
  get class schedule

ScheduleItem
  set room
  set time slot
  set schedulable
  get room
  get time slot
  get schedulable

Schedulable
  set priority
  get priority

Classroom
  add amenity
  add seat
  set number
  set description
  get number
  get description

Amenity
  set description
  get description

Seat
  set comfort level
  get comfort level
Class
  set name
  set professor
  set credits
  set hour units
  get name
  get professor
  get credits
  get hour units

Seminar (subclass of Class or a special case of Class)

Club
  set name
  set description
  get name
  get description

GuestSpeakerPresentation
  set speaker name
  set length
  set description
  get speaker name
  get length
  get description

Solution to c.
  BuildingSchedule: ScheduleItem, Classroom, Amenity, Schedulable (Class, Seminar, Club and GuestSpeakerPresentation)
  ScheduleItem: Classroom, Schedulable (Class, Seminar, Club and GuestSpeakerPresentation)
  Classroom: Amenity and Seat

Solution to d.
  Seminar can be a subclass of Class
  Class, Seminar and GuestSpeakerPresentation should implement Schedulable interface or, they can be a subclass of Schedulable, if Schedulable is not an interface

Solution to e.
  BuildingSchedule has (several) ScheduleItem
  ScheduleItem has a Classroom and a Schedulable
  Classroom has (zero or more) amenities and (one or more) seats

Solution to f.
  GroceryBag is a subclass of BasicContainer (inheritance relationship).
  GroceryBag has objects of Fruit and Vegetable (aggregation relationship).
  GroceryBag uses objects of Food and Money (dependency relationship).