Software Modelling, fall 2009, exercise 4

This week's exercises 1, 2, 4 and 5 are heavily based on chapter 9 of Craig Larman's book Applying UML and patterns.

1. In 2nd week's exercises we created a use case model of a system that supports the usage hour reporting of programming projects.

   Find the conceptual classes for the system. Before doing this you should have read chapters 9.1-9.7 of Larman's book. The chapter 9.5 of the book describes how the classes can be found from a textual description of a problem.

2. Do a domain model for the hour reporting system. Read the rest of chapter 9 of Larman for advice.

3. In the next page you'll find a small Java program. Draw class diagram that corresponds the program. Draw also sequence diagram which describes what happens when the main-method of class Room is executed.

   Note that main is a static method of Room, so it is executed in context of the class. In sequence diagram you should have separate "box" for Room class (that executes the main) and the Room object that is created by main.

4. Let us get back to the LAL system from 2nd weeks exercises.

   Find the conceptual classes for the system.

5. Consider 4 first paragraphs of the LAL system description (the paragraph starting with *We intend to hire people for a variety of jobs.* is now left out of considerations).

   Do a domain model for the limited part of LAL system. So leave out all issues that are mentioned only in the last paragraph.

```
public class Room {
      private int temperature = 0;
      int getTemperature() { return temperature; }
      void addTemperature(int t) { temperature = temperature + t; }
      public static void main(String[] args) {
            Room r = new Room();
            Thermostat t = new Thermostat(r);
            t.stabilizeAt(20);
      }
}

class Thermostat {
      private Sensor sensor;
      private Heater heaterA;
      private Heater heaterB;
      Thermostat(Room r) {
            sensor = new Sensor(r);
            heaterA = new Heater(r);
            heaterB = new Heater(r);
      }
      void stabilizeAt(int degrees) {
            while (sensor.getReading() < degrees) {
                  heaterA.heatTheRoom();
                  heaterB.heatTheRoom();
            }
      }
}

class Sensor {
      private Room room;
      Sensor(Room r) { room = r; }
      int getReading() { return room.getTemperature(); }
}

class Heater {
      private Room room;
      Heater(Room r) { room = r; }
      void heatTheRoom() { room.addTemperature(5); }
}
```