SENG 271 - Assignment 1

Goal: Deepen your understanding of using UML structural and behavioral models during software engineering.

The following assignment is to be performed individually.

Problem description

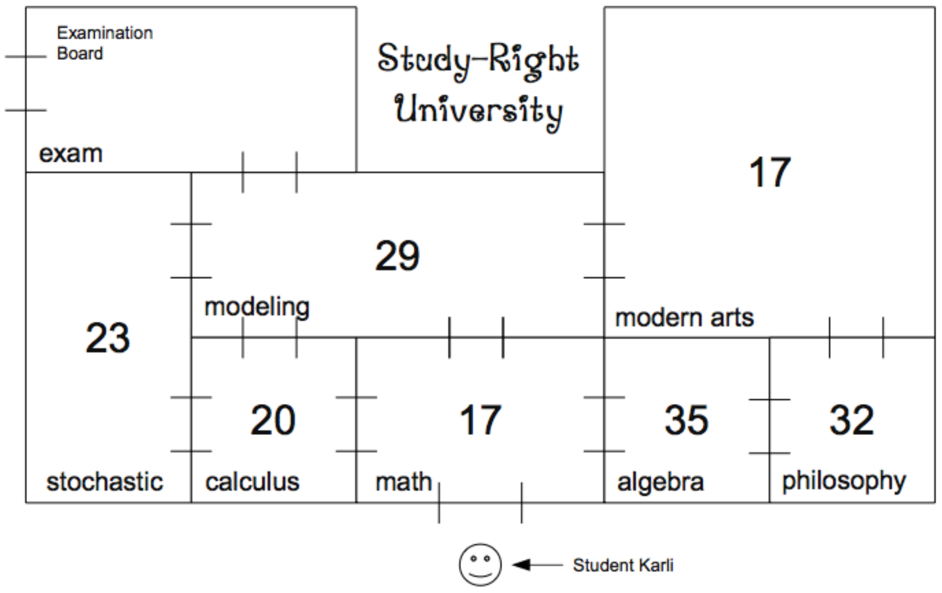

Figure 1. Room layout at "Study Right University"

We imagine a big university called "Study-Right University" with plenty of interconnected lecture halls as shown in the floor plan in Figure 1. The entrance to the university is from the lower edge through the math room. The exit is via the examination room in the upper left corner. The examination room hosts the examination board. To pass the exam for a degree, students need a specific amount of credit points, not more (who would do more anyway?) and not less.

In every lecture hall a professor teaches students for the amount of credit points shown in the floor plan. Being taught costs the attending students the same amount of motivation points that they can earn as credit points in a class. The examination board only hands out a degree to the student if they have all their respective credit points and no motivation left to stay at the university and block other students. If students enter a lecture hall they will be automatically taught as long as necessary until their knowledge is worth the respective credit points of this class. Then they can move to the next room.

If students have no motivation points left, they have to go on vacation and start over. All credit points earned so far will be lost. The students can visit lecture rooms multiple times. As the courses in Study-Right University are ongoing they can earn more credit points when they visit a room another time. However, after they have earned the credit points of a room, they first have to leave the room and be taught something different before they can come back and learn more.

The student Karli wants to achieve a degree at the Study-Right University. For a degree we need at the moment 214 credit points and Karli starts with 214 motivation points. The university is depicted in Figure 1. The amount of credit points (and hence the number of motivation points to be spent), which will be earned in each room are written as numbers in each room. The examination board is in the upper left corner. Each room has a topic, which is taught in it, and its name is written in the lower left corner of each room.

The overall goal is to develop a program that allows to find a way for a student (such as Karli) to pass the exam in the Study-Right University.
Well, do we actually need a program? Wouldn't it be faster to find a solution for Karli, manually? OK. Spend some 30 seconds and try it.
Not so easy? Yep. Still, finding a single solution is probably faster than programming a solver. Thus, programming the solver makes sense only, if you expect to use it over and over again. For example, the floor plan of the Study-Right University might change. Or the number of credits required for an exam changes. Or the credits per topic change. Or you have to identify reasonably small yet interesting initial credits for new students that result in nontrivial paths through the University.

Tasks

A. Structural modelling

Design a UML Class Model and a UML Object Model for the above program.

Note:

- Use the "noun underline" approach as a rule of thumb, identifying your classes / objects. (You don't have to submit any intermediate work results on using that approach, though.)
- The Object Diagram should display the example situation in Figure 1.
- You can use any tool for designing your models for this assignment (even Visio, or a pencil and paper if you choose to). However, an advanced UML tool such as MagicDraw UML (or ArgoUML) will provide you with code generation.

B. Behavioural modelling

Design two UML Sequence Diagrams that show different example scenarios on how the objects in your model will interact in order to solve the problem task.

C. Implementation

Implement the program in Java.

Note:

- You may generate you Java classes from the UML model - which may save you some time.
- The lab has Netbeans as well as Eclipse installed, if you would like to use an IDE.
- No GUI is needed for the implementation. A commandline user interface is sufficient.
- As part of your implementation, provide a test driver that solves the situation in Figure 1. (If you are familiar with JUnit, you may use that framework. Otherwise, just implement your test driver as part of the "Main" class.

Submission

Submit your assignment in form of a zip archive to the Moodle Web site by May 31. (No other archiver please.) The zip archive should contain:

- A single document (in MS Word or PDF format - no other format please) containing your models, including a textual description of these models.
- Your Java program code, along with a README text file on how to compile and run the program from the command line. (Please do not provide instructions on how to do this with an IDE.)

Evaluation

Each task (A,B,C) has equal weight in grading this assignment.

Grading of modelling tasks (A and B)

marginal (C and lower)	**solid, but could be improved (B)**	**meets expectations (A)**	**exceeds expectations (A+)**
major parts of models are incorrect, either with respect to the UML or with respect to the requirements	•some incorrect use of UML *or* •some requirements are not faithfully modelled	•UML used correctly •Models adequately implement the requirements	• Models follow "good design" rationale, which is laid out in the paper

Grading of implementation task (C)

marginal (C and lower)	**solid, but could be improved (B)**	**meets expectations (A)**	**exceeds expectations (A+)**
program incomplete	•program compiles and runs but has flaws, e.g., it does not faithfully implement the model	•program faithfully implements the model and is correct	• Program is well documented. In particular, comments in the program clearly link the code to the model •Test driver uses JUnit framework