Assignment 2

Version: March 17, 2024

- Create an abstract domain class model
- Practice using behavioral diagrams
- Use and understand different levels of detail in behavioral diagrams
- Implement your design in Java

Prerequisites

1. Watch the lecture videos on class diagrams (especially the Domain Class model)
2. Watch the lecture videos on use case models, sequence diagrams
3. Go through the slides (video) about the shop model

General

You are supposed to create 3 different diagrams in this assignment and implement a basic prototype.

Make sure to read through all the requirements in this document first before you attempt it. Remember, your diagrams need to be consistent.

All diagrams should be included in ONE Astah file!

PLEASE watch the "The model" videos from module 1, which shows some tips and tricks for Astah `https://youtu.be/fRMQKIG7O50`.

Create one PDF document in which you will include all of your diagrams and answers for this assignment.

Part 1: Domain Class model

Task 1: Create a Domain Class Model (DMC) (20 points)

Remember that there is not one correct version of how to design this system; there are many different solutions, and some might be better or worse than others. Therefore, your goal should be to try to find a good design that meets the requirements in the TutoringRequirements.pdf. This is what we will be looking for while grading: syntax, the right level of abstraction, all requirements are met, etc.

You are supposed to use specific levels of abstraction for these diagrams. These were all explained in the videos, and you have examples in the lecture slides (there is also a set of lecture slides that go over the consistency between diagrams and the most important points). Make sure your diagrams use the correct elements and level of abstraction.

The diagram should be created in Astah.

Use the notes given in the TutoringRequirements.pdf describing your system. Use these notes to create a **Domain Class Model** for this system in Astah. You should create a domain class model as it was discussed in the lecture in the week 1 module.

Export the diagram and include it in your document under Section Part 1 Task 1.

Part 2: Use-Case diagram and Sequence Diagram (20 points)

Task 1: Use Case Diagram (10 points)

Create one use-case diagram with 2-5 Use Cases and all of the system's actors. This diagram should describe the whole system, and one use case can have one or more scenarios.

List these scenarios and who invokes them with your use-case diagram.

Export your diagram and the listed scenario(s) as an image into your document under Part 2 Task 1.

Task 2: Sequence diagram (10 points)

Create a Sequence diagram for a Student canceling an appointment. Check the TutoringRequirements.pdf for details. Assume the email address and appointment id provided in this system operation does exist in the system. The other alternatives described in the requirements need to be modeled.

The sequence diagram you are supposed to create should be of a **high level** of abstraction (Actor-System communication only).

This diagram should be done in Astah. Remember that the Astah syntax is a bit different than the one in UML.

Export the model as an image and place the image in your document under Part 2 Task 2.

Part 3: Implementation based on your Design(20 points)

Implement the structure (10 points)

To get a better understanding of what you have done, you are supposed to create a simple Java implementation based on your design (Remember Reverse Engineering: UML CLASS DIAGRAM TO CODE AND BACK). We want to keep it very simple; remember, we are at a very high level of abstraction, and this is supposed to help you understand the multiplicities in your Domain Class model and what the sequence diagram means. It is only a **partial** implementation of the system on a very, very high level of abstraction. Usually, you would not implement it at this stage, but after knowing more details.

IMPORTANT: Do NOT export your code using Astah. You have to manually implement the code for this task; I want you to know exactly what you are doing. I also included a short video linked with this assignment on Canvas for this part of the assignment; please watch it.

All classes from your Domain Class Model (DMC) should be included exactly as they are in your DMC (you will have to add types to your attributes, of course). Only include the structure at this point (no methods are needed, only exactly what is in your diagram

at this point). For now, you can have all attributes public (I know it is not good OO, but we will make it better in time).

Create a Main class in which you instantiate some of your objects. Create at least some Students, Tutors, Appointments, and Subjects and link them statically together. You still do not need methods for this. You only create objects and add them to the correct attributes. E.g., in your DMC, your Student and Appointments are hopefully connected through an Association. Now, in your implementation, you can have a student *david: Student* with the attribute *appointments: Appoinments[]*. Now you just do *david.appointments = [app1, app2, app3]* where david would now be registered for the appointments app1, app2, app3.

Now, you should have a static system that has a specific system state (specific objects exist).

Create an object view of this system state (check the slides if you do not remember what the object view is. HINT: Basic UML for Reverse Engineering: class diagram intro: min 2:14 and min 6:30 and the video provided with this assignment). Add this object view diagram to your PDF.

Implement behavior (5 points)

Now, implement the method "Student cancels an appointment" in your Main class.

Implement it the exact same way you modeled it in your Sequence Diagram (SD). Use the exact same arguments from your SD (you will have to specify types for your arguments in your programâ€"not in your diagram) and the alternatives. You should be returning string messages.

This method should work on the current system state and check the alternatives, and in these alternatives, return the correct message based on your SD. You DO NOT implement that the Student really cancels the appointment (e.g., you do not delete the reference between the two).

Test if your method returns the correct values for every case (returns the correct string). To do so, you need to use different system states, e.g., a student who is 24 hours late or a student who has no appointment. When we run your main, we want to see the different strings printed in the console.

IMPORTANT: This task will only be graded if it is based on your design; thus, creating a perfect implementation of the system independent of your design will lead to 0 points. Only implement what is stated here and is in your diagrams, you will not get more points for doing more. Make sure your implementation does not have compile time errors and that we can run your Main. You do not have to handle exceptions and special cases. Just base it on the high-level models you have here. **I want to see that you understand what your diagrams mean!**

Run your program (5 points)

For your submission make sure that when you go into your src folder with a command prompt (command line, cmd on Windows, terminal on Mac) and call *javac Main.java* your class files should compile (eg. .class files are created) and when calling *java Main* then it should print the output of your Main method in your command line.

Submission Part:

You will need to submit three files: one Astah file (diagrams_asurite.asta), one PDF file (assignment2_asurite.pdf) and a zip file (tutor_asurite.zip).

- The Astah file should include all your diagrams
- The PDF document needs to include the following
  - domain class model
  - use-case diagram with scenarios
  - sequence diagram
  - object view from your implementation
- The zip file should include your implementation. Make sure we can import and run it.