Seminar 3, Implementation

Object-Oriented Design, IV1350

Intended Learning Outcomes

This seminar concerns the learning outcomes *develop an object-oriented program by applying established guidelines for object-oriented programming* and *discuss the quality of a program referring to established guidelines for object-oriented programming.*

Goal

- Practice translating design model to source code.
- Practice writing unit tests.

Literature

- Chapters six and seven in *A First Course in Object Oriented Development.*

Grading

There are three possible grades:

Fail (0 points) To pass the course you must pass all four seminars. If you fail this seminar you have to report it again at the end of the course, at the fifth seminar.

1 point Active participation in seminar discussions. Written solution submitted in Canvas proving that you can, with minor defects, apply at least two best practices of object-oriented programming. It shall also prove that you can write a basic unit test.

2 points Active participation in seminar discussions. Written solution submitted in Canvas proving that you can, without defects, apply several best practices of object-oriented programming. In the written solution you must also satisfactory explain and motivate their use. It shall also prove that you understand how to write and use unit tests.

Tasks

There are two tasks, task one is to write the program and task two is to write tests for the program. As can be seen in task two, to pass (1p) it's sufficient to write tests for two classes. Still, you're encouraged to try to write more tests, since tests are important.

Task 1

Write a program implementing the basic flow, the startup scenario, and the alternative flow 3-4b specified in the document with tasks for seminar one, which you designed in seminar two. You do not have to program any other alternative flows or add any other functionality. You are also not required to code the view, you may replace the user interface with one single class, `View`, which contains hard-coded calls to the controller. Neither is there any requirement on databases or external systems. Instead of a database, you can just store the data in the object in the integration layer, which should have been responsible for calling the database if there had been one. The external systems can simply be omitted, but there must be objects responsible for calling them.

The solution must meet the following requirements.

- The code shall be compilable and executable. You have to program in Java, since everyone must be able to understand your solution at the seminar. Also, don't use exceptions now, since that's a topic of seminar four, not understood by everyone now at seminar three.
- If the user interface is replaced with hard-coded method calls, **the class calling the controller must print everything that is returned by the controller**. Also the receipt must be printed (to `System.out`).
- Your code **must follow all guidelines presented in chapter six in the textbook**. Regarding comments this means there must be one comment for each public declaration.
- Try to follow the design from seminar two, but it is perfectly OK to change the design if you discover flaws. The solution must however have high cohesion, low coupling and good encapsulation with a well-defined public interface.
- In the `Method` chapter of your report, explain how you worked and how you reasoned when writing the program.
- In the `Result` chapter of your report, briefly explain the program. Include links to your git repository, and make sure the repository is public. **Also include a printout of a sample run**. Listing 1 below is an example result of a sample run, illustrating the expected output.
- In the `Discussion` chapter of your report, evaluate your program using applicable assessment criteria from the document `assessment-criteria-seminar3.pdf`, which is available on the *Seminar Tasks* page in Canvas.

```
Add 1 item with item id abc123:
Item ID: abc123
Item name: BigWheel Oatmeal
Item cost: 29:90 SEK
VAT: 6%
Item description: BigWheel Oatmeal 500g, whole grain oats,
                  high fiber, gluten free

Total cost (incl VAT): 29:90 SEK
Total VAT: 1:69 SEK

Add 1 item with item id abc123:
Item ID: abc123
Item name: BigWheel Oatmeal
Item cost: 29:90 SEK
VAT: 6%
Item description: BigWheel Oatmeal 500ml, whole grain oats,
                  high fiber, gluten free

Total cost (incl VAT): 59:80 SEK
Total VAT: 3:38 SEK

Add 1 item with item id def456:
Item ID: def456
Item name: YouGoGo Blueberry
Item cost: 14:90 SEK
VAT: 6%
Item description: YouGoGo Blueberry 240g, low sugar youghurt,
                  blueberry flavour

Total cost (incl VAT): 74:70 SEK
Total VAT: 4:23 SEK

End sale:
Total cost (incl VAT): 74:70 SEK

------------------ Begin receipt --------------------
Time of Sale: 2024-02-12 16:05

BigWheel Oatmeal            2 x 29:90    59:80 SEK
YouGoGo Blueberry           1 x 14:90    14:90 SEK

Total:                                   74:70 SEK
VAT: 4:23

Cash:                                    100 SEK
Change:                                  25:30 SEK
------------------ End receipt ----------------------

```

```
50 Change to give the customer: 25:30 SEK
```

Listing 1 Expected output of a program execution. This example is only meant to illustrate the expected data. It's perfectly fine to change text formatting, and also to change values.

Task 2

Write tests for your program.

- To pass (1 point) you must write unit tests for two classes. Try to find something more interesting to test than get/set methods. You have to write new test classes, you're not allowed to use the given tests, that is `AmountTest` from the textbook and `MainTest` from the lecture *Practice Programming and Unit Testing*.
- To pass with distinction (2 points) you must write unit tests for all classes in the layers `controller`, `model`, and `integration`, except classes that have just getters and constructors that only store values. It is also not required to test that output to `System.out` is correct, just ignore testing methods that only produce output to `System.out`.
- In the `Method` chapter of your report, explain how you worked and how you reasoned when writing the unit tests. Explain how you chose which tests to write.
- In the `Result` chapter of your report, briefly explain the tests. Include links to your git repository, and make sure the repository is public.
- In the `Discussion` chapter of your report, evaluate your unit tests using applicable assessment criteria from the document `assessment-criteria-seminar3.pdf`, which is available on the *Seminar Tasks* page in Canvas.