**Seminar 1**

Object-Oriented Design, IV1350

Ossian Fjordefalk

[Ossian.f@gmail.com](mailto:Ossian.f@gmail.com)

2/5-2017

**Contents**

1 Introduction 3

2 Method 4

3 Result 5

4 Discussion 7

1. **Introduction**

The task of seminar 3 was to create a program from the design models made in previous seminars. We then created tests for the program using JUnit. The purpose was to learn how to create a program from a model and then test it. In addition we learned how to use Git. I did all the work on my own and then discussed my solutions with class mate Anders Pettersson.

1. **Method**

**Program**

I started of by creating all the classes and packages included in the class diagram created in seminar 2. I then looked at communication diagrams from seminar 2 and implemented the necessary methods into the classes. Some translations from model to actual program was very straightforward, I just created the methods mentioned in the communication diagrams and used the same parameters. But some things had to be changed to work in the real program. A simple view program was written to test run the methods.

**Tests**

I started with doing the whole program and then do the tests. I choose two classes to write tests. I choose the classes with the methods that needed testing the most. Most of the methods just called another method or created an instance of some class. The methods a I choose to test made some form of calculation which could be tested by controlling that the right value was calculated from known variables. I choose not to test getters, setters, constructors or methods simply calling other methods.

1. **Result**

**Program**

Github link: https://github.com/KrokusMorning/Seminarie3/tree/master/src/main/java

The program simulates a vehicle inspection. Data is inputted through the view class which then calls methods in the controller class. The view then visualises the data.

The method newInspection is located in the controller and calls a method in Garage which in turn opens the garage door and increases the que number on the display.

The method closeDoor is located in the controller and calls a method in Garage which in turn closes the garage door.

The method costForInspection is located in the controller. It takes vehicle as a parameter and calls a method in database manager which creates and returns an instance of InspectionChecklist which is then used to create an inspection instance. Inspection then calculates the cost based on the information about which inspections to be done in the inspection checklist. costForInspection then returns the inspection checklist. The cost for the inspection can then accessed through a getter. In the model the int value cost was returned from the method. In the program I made the function update information in an object instead as I found it more convenient.

In the model this method was called enterRegistartionNumber as a primitive value containing a registration number was used in the method call. In the program I choose to create an instance of vehicle and use that in the method call instead, and the name was changed to costForInspection

The payWithCard method is located in the controller. It takes an instance of CreditCard and two primitive values as parameters and calls the cardPayment method in Payment. The payment authorization class is then called to verify the payment and returns a Boolean value. The method in payment then creates an instance of CardReceipt resembling a receipt which is then passed on in a method call to the printCardReceipt method in the Printer class. The printer writes out all the necessary information about the payment.

In the model I had the payment class create the credit card, but that included passing multiple primitive values as parameter through multiple method calls. I solved it by creating the credit card from the view and then pass the object instead.

The whatInspectRequest method located in the controller updates the checklist with the result for a specified inspection. It takes the result vehicle and checklist to update as parameters. The data base manager checks if it’s the right vehicle and updates the first inspection that is not passed and returns. One method call updates one inspection result.

The inspectionComplete method in controller calls a method in database manager which creates an instance of inspectionResult which copys all the checklist results and generates a result. The database manager then calls the printer which prints out the result.

**Tests**

I chose to test the class DatabaseManager and Inspection.

Inspection:

<https://github.com/KrokusMorning/Seminarie3/blob/master/test/main/java/model/InspectionTest.java>

DatabaseManager

<https://github.com/KrokusMorning/Seminarie3/blob/master/test/main/java/integration/DatabaseManagerTest.java>

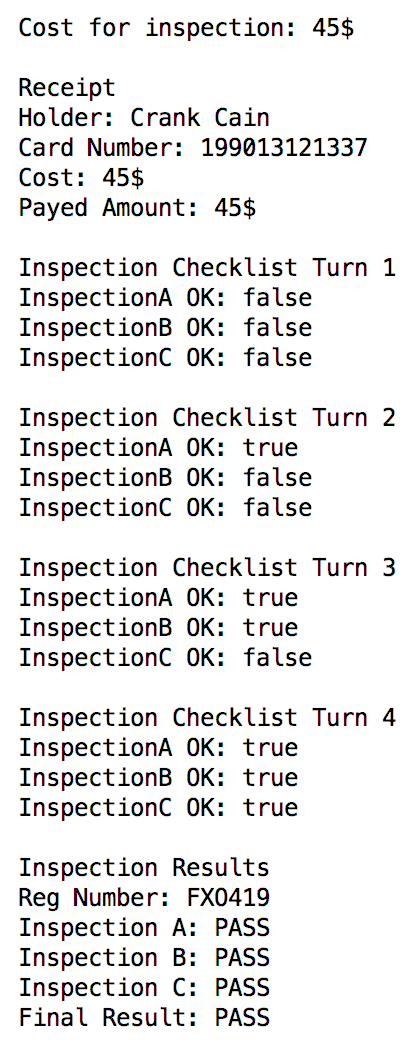


Figure Sample run

1. **Discussion**

**Program**

I strived for making the program easy to understand and follow. I made comments explaining the methods/classes and used logic meaningful names for all methods and variables.

There are some code duplication when dealing with the inspections in the inspection checklist. I could have avoided that by using a list and loop through it instead. But there are no major code duplication.

I tried to use as few primitive data parameters as possible and use objects instead. Maybe I could have made the program even better by creating a new object containing multiple objects to avoid using multiple parameters in some method calls.

**Test**

I started by doing the program and then do the tests. The optimal would have been to do the tests simultaneously as I did the program as I could then use the test to see that the program was functioning properly.

I tried to only make meaningful tests that could really be of use, and I think I succeeded. I did not come up with that many tests to do, but the ones I did was at least useful. As mentioned earlier most of the methods just created an object or called another method, it was redundant to test them. The only way I know of to test them would be to create an object in the test and then see if the method did the same, but I figured it would be similar to test getters and setters so I decided not to.