Seminar 3

Object-Oriented Design, IV1350

Ossian Dillner – ossiand@kth.se

2017-05-02

Contents

1 Introduction

In preparation for this seminar the student was tasked with performing a basic implementation of the previously designed program as well as writing unit tests for classes in aforementioned program.

While implementing their design the student most adhere to all guidelines for Java development present in the course literature. The student must also attempt to follow the design from the previous seminar but is free to make changes if needed.

The student is required to write unit tests for at least two of the implemented classes and if they want to pass the seminar with distinction they are required to write tests for all (!) classes in the program, excluding classes in either the view or startup package.

2 Method

Task 1

Writing this program proved very simply due to the rigorous design and analysis process, the process consisting of simply following the earlier outlined design, making very small adjustments along the way.

Unfortunatly, this rigorous design process combined with my earlier programming experience allowed me to write this program very quickly and therefore I find it hard to write much of anything about the process.

Task 2

When implementing the unit tests I picked two classes that I found interesting to test and tried to create tests for edge case scenarios, not that many of those really exist in a program with such basic flow.

3 Result

Task 1

Link to GitHub repository containing source code and unit tests.

I made the decision not to include code excerpts in the report as the code in it's enterity is much easier to view on the GitHub repository in the link above, including huge screenshots of the code would only serve to convolute the report.

Since the purpose of this task is to properly implement the previously created design into code, not to make any kind decisions regarding the design of that code I find that there is not much to explain about the result of this task. I simply followed the general guidelines for Java development, outlined in the course literature among other places.

Task 2

I choose to create unit tests for the InspectionRegistry class and the Inspection class. Due to the simple nature of these classes the test ended up not being very comprehensive, simply checking for expected behavior and some edge cases where applicable.

Printout of sample run:

```
VIEW> Calling Controller.nextCustomer()
GARAGE> Door is: open
GARAGE> Current queue number displayed: 1
VIEW> Calling Controller.closeGarageDoor()
GARAGE> Door is: closed
VIEW> Calling Controller.enterRegNumber('ABC123')
VIEW> Total cost of inspection is: 2250
VIEW> Creating dummy credit card
VIEW> Dummy credit card contains following data: owner: Sven Svensson, number: 1111222233334444, expiryDate: 12/12, CVC: 123, pin: 1234
VIEW> Creating dummy payment
VIEW> Dummy payment contains following data: amount: 2250, card: owner: Sven Svensson, number: 1111222233334444, expiryDate: 12/12, CVC: 123, pin: 1234
VIEW> Calling Controller.performCCPayment
PRINTER> Printing receipt...

### Receipt of payment ###
20170601_173421
Customer name: Sven Svensson
Payment amount: 2250
############################
VIEW> Starting inspection loop
VIEW> Calling Controller.getNextInspection()
VIEW> Inspection object received
VIEW> Inspection contains following data: descr: wheel, cost: 100
VIEW> Calling Controller.performCurrInspection()
VIEW> Current inspection passed
VIEW> Calling Controller.getNextInspection()
VIEW> Inspection object received
VIEW> Inspection contains following data: descr: engine, cost: 1000
VIEW> Calling Controller.performCurrInspection()
VIEW> Current inspection passed
VIEW> Calling Controller.getNextInspection()
VIEW> Inspection object received
VIEW> Inspection contains following data: descr: oil, cost: 150
VIEW> Calling Controller.performCurrInspection()
VIEW> Current inspection passed
VIEW> Calling Controller.getNextInspection()
VIEW> Inspection object received
VIEW> Inspection contains following data: descr: lights, cost: 250
VIEW> Calling Controller.performCurrInspection()
VIEW> Current inspection passed
VIEW> Calling Controller.getNextInspection()
VIEW> Inspection object received
VIEW> Inspection contains following data: descr: transmission, cost: 750
VIEW> Calling Controller.performCurrInspection()
```

VIEW> Current inspection passed
VIEW> Calling Controller.getNextInspection()
VIEW> No inspections left to perform, exiting loop
VIEW> All inspections passed, calling Controller.finishInspection()
PRINTER> Printing result...

Result of inspection
Inspected vehicle RegNo:ABC123
List of performed inspections:
Inspection #1: descr: wheel, cost: 100
Inspection #2: descr: engine, cost: 1000
Inspection #3: descr: oil, cost: 150
Inspection #4: descr: lights, cost: 250
Inspection #5: descr: transmission, cost: 750
##############################

4 Discussion

While the presentation of the implementation might be a bit lacking, you'll most likely find that the design has been properly implemented, that the code is well written as well as the unit tests.