Seminar 3

Object-Oriented Design, IV1350

Mohamed Mahdi
Mmahdi@kth.se

1-4-19

Contents

1 Introduction

The task for seminar 3 was to write the code that is an implementation of the design from previews seminar task. The design was made for the basic and alternative flow for Point of Sale. The task also consisted also of a testing for each public implemented method. The seminar task was done in a group that consist of 7 people. Their names are Sadeq Gholami, Hassan Yaasin, Mohamed Mohamoud and Abdirahman Abdullahi, Zeineb Amri and Fariba Yazdani.

2 Method

To implement the code the group the examples that can be found in course literature, *A First Course in Object Oriented Development.* By loosely following the design all the methods and classes that needed to do was already decided. The group also had to fix any design flaw that was clear when trying to implement in to code so in the end the group would be able to create a program that worked together.
The group worked using GitHub which allowed people to work on different part of the program creating new methods and fixing others mistakes to later merge to the master branch that is shared between all 7 members.

3 Result

https://github.com/sadeq-gholami/Seminar3POS

Figure 3.1: Completed program POS, shows the whole program in git repository

Includes two folders POS and testPackage. POS is the whole program that is spilt into folder representing the package the classes are in. testPackage is test files which only serve to test all the public methods in POS folder.

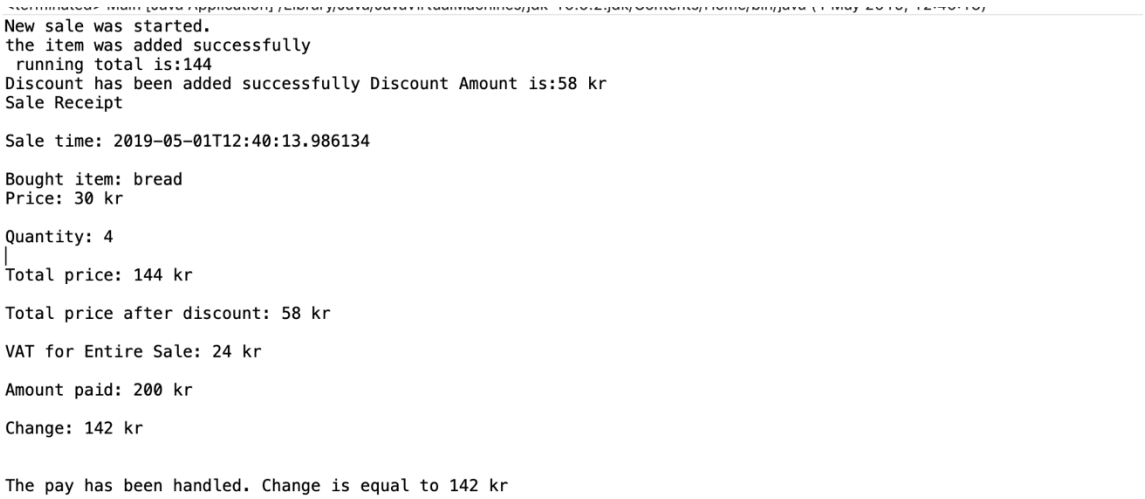

Figure 3.2: Main execute, shows the output when executing the Main in the program

What is shown is that a sale has been started and items has been added which shows the price of your purchase. The next step is checking if you are eligible for discount and how much percentage u get cut from your price, which is not shown in the out print but lets you know I was successful and your new price to pay. The last part is the Receipt that shows all the information of the purchase.

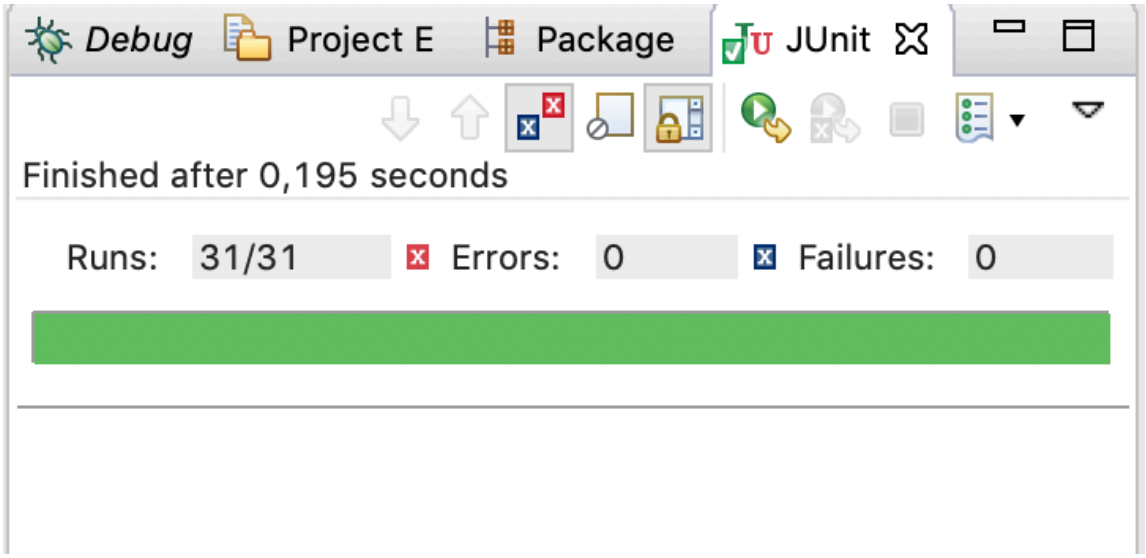

Figure 3.3: Test package, shows all the test that are in the Test package and the result when run, zero failures (works!)

The test package includes a test method that handles every public method in the program. When starting test, you can see if expected result is same as what the method returns. All the 31 tests returned the expected result which leads to zero failures.

4 Discussion

The majority of the code is written so that the names are directly correlated with the methods function. There are few instances where the choosing of the name followed the design that were little less clear, but we kept it to the final draft to not lose track of all the methods. Example of this is countDiscount() which is not clear that what it returns is the total price after discount but it haven't been changed.

There is barley any duplicated code in the project. An instance where it looks like duplicated code is discount rules that serves as the place that decides what discount that are going to be added to the sale. What makes it look like duplicated code is that there are two methods that accentually works the same way, but one is designed for the discounts that are available for members and the other not members. In the call they do not return same values but in their implementation, they work the same way. The reason for this is that the groups design for discount was not working so when coding discount there was short time to redesign. A solution to make it look clearer would have been creating another private method that returns percentage depending if you are member or not. This way there would only be need for one method that returns the sum of all the discount.

Majority of the classes and the methods are short and straight to point. The longest class is receipt, but the reason is when creating the string for receipt. The more you append to the string the longer the method is.

The group avoided using primitive data instead of objects are used as much as possible. An example is the use of amount which serves a placeholder for any time there would have been an int and string. There is not any method with unnecessary long parameter because when we send list arrays the parameter will naturally become long because of it, other than that the rest are kept short. We haven't written any comment inside a method. All the comments are above all the public methods and kept to a minimum explaining briefly about them.

We had high cohesion because of all the classes have only methods that are relevant to the class. You can always achieve higher cohesion but for the time we put in to the project this was as far we could get. The coupling was not so appealing since many classes are connected with sale. This was because of last minute changes with the design of discount that made the coupling worse. Encapsulation was achieved with making all the methods that needed to be called from the outside public. This is needed and can't be changed since our methods are depended on them. What is private is all the method that does not need to be seen from the rest of the program. We made as many methods as we could private to higher our encapsulation.

The package structure is the same as the structure shown in the book. Since we follow example in the book, we believe the structure is good.

Unit test was very useful to know that every method return is the same as what you believed would be returned. If a problem would have occurred in a method that calls another method the test would serve to see if that call returned what you expected.