Seminar 1

Object-Oriented Design, IV1350

Casper Kristiansson, CasperKr@kth.se

2021-03-29

Contents

1 Introduction

The first seminar contains two different tasks. The first task involves creating a domain module for a retail store where we need to implement both a basic flow and an alternative flow for describing Process Sale. While in the second task the objective is to create a system sequence diagram (SSD) illustrating both basic and alternative flows of Process Sale. This assignment was completed alone by the author, but the exercises was discussed with Michell Dib and Fredrik Lundström.

2 Method

To solve the first task which involved creating a domain model of a retail store the author first of used noun identification to identify different classes of the program. With this method you can easily identify all the different classes that needs to be created. The next step is to identify more by simply categorizing the different classes. The categories can be found in course book “Object Oriented Development” in chapter 4. After categorizing them we can easily identifying more of them that would be useful to represent the program.

After we have identified all the classes, we can separate them if they can be used as an attribute. This is because some classes should not remain as classes because they would better as attributes. Attributes consists either of a string, number, time, or a Boolean number. In this domain model we can really benefit from attributes for example for the classes amount, itemPrice, itemDescription, registerAmount.

The next step of the domain model is to create association between the different classes to help clarify the model even more. This can be completed by simply creating a map and drawing lines between the different classes. We also name the association between the classes for even more clarification.

The second part of the seminar involved in creating an SSD (system sequence diagram). When solving the second task of the seminar the author started by outlining the different interactions between system and actors. The next step was to go through both the basic and alternative flow and implementing each step and creating the diagram to represent the program relative to time.

3 Result

Seminar 5, Improve Your Score

There was a total of 5 mistakes in seminar 1 that needed to be fixed.

- The first mistake was that the SSD model was missing a “startSale” operation. This was fixed by adding the method before the “for loop” for entering items.
- The second mistake was that the error message was sent as a message from system to the cashier. A correct solution to this is by changing the messages to a return value when calling a method.
- The third mistake was that the alternative flow 3-4b and 3-4c was not modeled the right way. This could be fixed by first adding an if statement in the for loop to check if it exists multiple of the same items with “quantity > 1”. The second thing that was needed to be fixed was that if an item already has been entered, rather than adding it to the list we increase the sold quantity by the product. This is something that the system does internally. Therefore the author added a note describing it over “:System”.
- The fourth mistake was that the SSD was missing an “endSale” operation. This could be fixed by adding the method after the for loop for entering items.
- The last mistake was that the display was not really required and could be removed completely. We instead focused on sending the information as return messages to the cashier.

The author also updated the SSD image below, see figure 3.2.

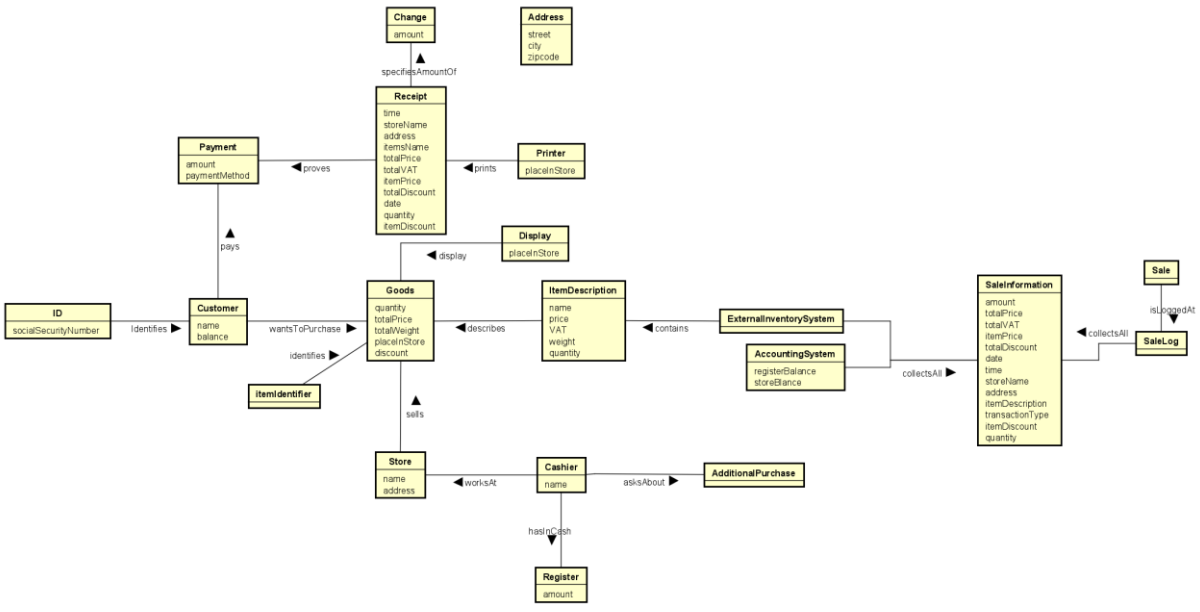

Figure 3.1: The figure above describes a domain model that models the process of a sale at a retail store. The model represents the association between different classes and how the model incorporates behavior and data. In this model we can see that it centralizes around Goods which is all the products that the customer wants to buy.

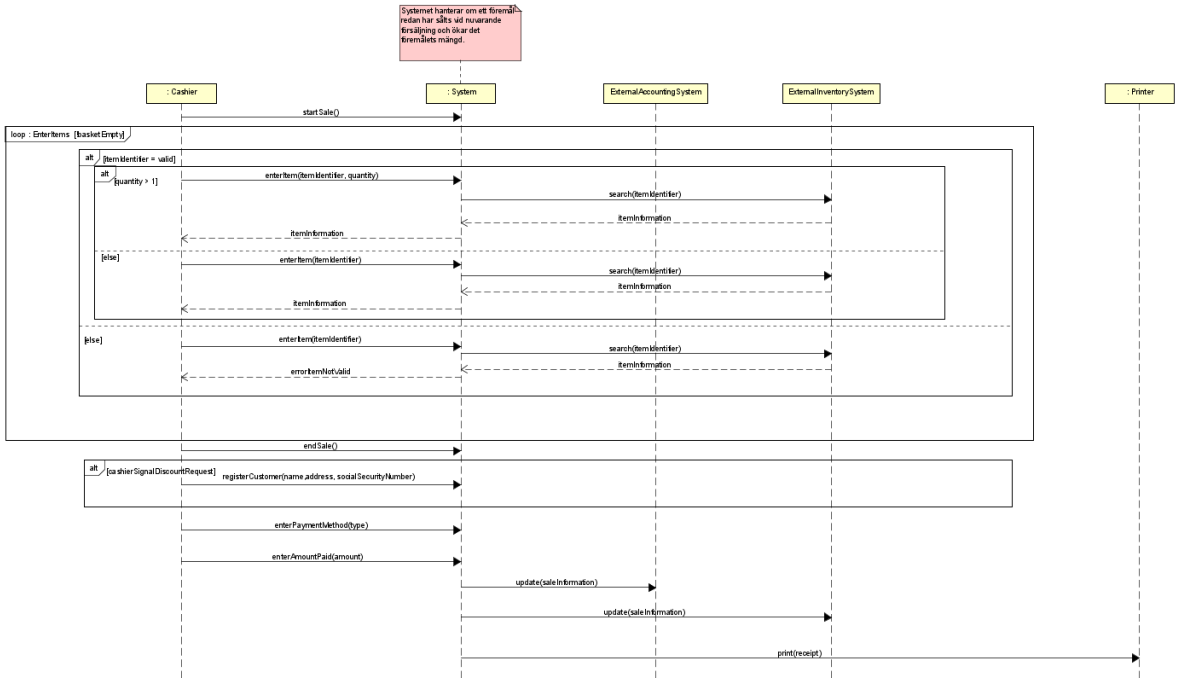

Figure 3.2: The figure above describes an SSD (System Sequence Diagram) which illustrates an interaction between the actors and the system. The diagram is constructed so that it will represent how the program executes relative to time.

4 Discussion

This section of the report the author will discuss the results of the seminar tasks. In the first task where the objective was to create a DM (domain model) which is a conceptual model that incorporates behavior and data. The author created a model that follows all the guidelines including not creating a programmatic or naïve DM. In this case the programmatic DM illustrates a model that becomes too complex to understand for a non-programmer. This is a big mistake because the main objective of a domain model is to clarify a concept. A second mistake would be creating a naïve DM where the actors (customer or cashier) become the central classes of the model. The authors domain model is centralized around objects which in this case is goods (the desired products).

The domain model does not have a spider-in-the-web classes. This is because the classes that are associated to the centralized class “goods” are also associated with other important classes. Example we can see that there are a lot of important classes that exist around the outer part of the model and that some of the classes does not have an association with the centralized class. Meaning that the centralized class “goods” does not become a spider-in-the-web class.

The author correctly constructed the DM where it is easily to understand the process of a sale both representing a basic and alternative flow. In the DM there are reasonable number of classes which represents all the different parts of the behavior and data of a sale. The classes where carefully selected to make sure that all the important classes exist and that it does not contain any unimportant classes.

When forming the DM, the author selected which classes would be better fitted as attributes. In this selection process the author only had one main objective which was that the attributes should only contribute with more clarifying to DM. The author also carefully planned out and mapped which classes would be associated with each other. It was also important to remove all the classes that did not contribute anything to the DM. This could be done by categorizing the different classes, where we can easily see which classes are needed and which were not important which led to that the remaining classes and associations are sufficient for the situation.

The author correctly named all the different classes and objects that follows all the different guidelines in both the DM and SSD. The UML has been correctly used by the author where both the DM and SSD has been correctly modelled to visually represent the process of the sale for both the basic and alternative flow. When the author modelled the SSD, he carefully considers the layout relative to time and that all the different objects include the correct operations and return values. This includes that the different objects point to the right actor or system.