

Seminar 1

Object-Oriented Design, IV1350

Leo Karsikas leoka@kth.se

27 mars 2024

Contents

1	Introduction	3
2	Method	4
3	Result	5
4	Discussion	7

1 Introduction

This seminar will cover the two tasks described in the first assignment. This report aims to implement a Domain Model and a System Sequence diagram. The resources for implementing this assignment were taken entirely from the course material from the video lectures and the chapter in the book under sections 4.1-4.3.

2 Method

Task 1

Noun-identification

Before proceeding with the implementation of the Domain model. To avoid missing any important classes, the noun identification method was used. Examine the text of the task to identify all the nouns that serve as potential candidates for the class.

Category list

To make sure we don't miss any important classes, we made a list of categories. Then, we thought about different situations described in the assignment and came up with possible solutions for each.

After having all of the potential classes, we proceeded with implementing the DM in Astah.

Task 2

For the SSD (System Sequence Diagram) we started by reading through all of the scenarios described in the task and focusing on the alternative scenarios which could result in optional statements or else statements, which was implemented for a bigger loop.

For this part of the assignment, the main method was to think more from a programmer's perspective and how to implement the flow according to the lecture.

3 Result

Task 1

The figure below presents a Domain model that consists of classes, attributes, and relations. The DM is describing the POS, which was given in the assignment. The following DM fulfills all of the requirements presented in the assignment task and considers the assessment criteria.

The DM considers both the Basic Flow and the Alternative flows. It can be seen as handling the case where an item is "not found" in the system.

We decided to connect the *ItemDescription* class with the *SalesInfo*, which could simplify the retrieval of the storage inventory from the *ExternalInventorySystem*.

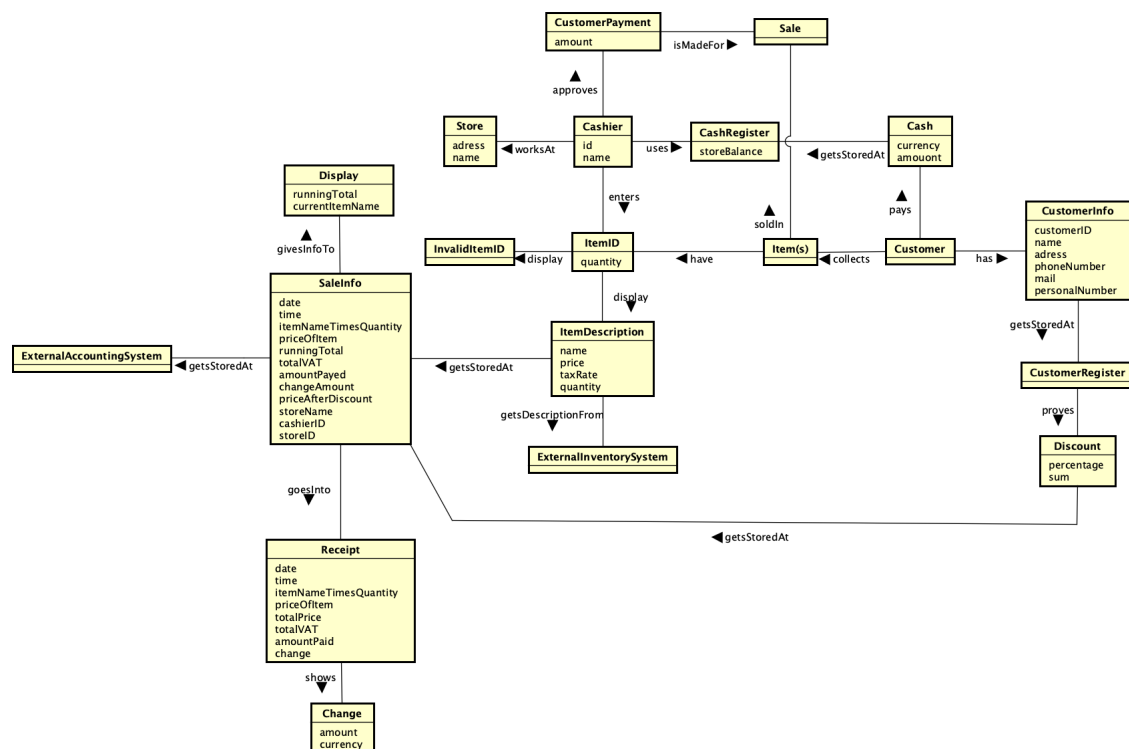


Figure 3.1: Diagram for the DM

Task 2

The second task was about implementing an SSD (System Sequence Diagram). This part of the assignment aims to implement an SSD that will reflect both the basic flow and the alternative flows for the POS.

As seen in the figure below, the loop and the return variables were used. A good example of that is the handling of the invalid items (3-4a.), where there is only a return variable from the System.

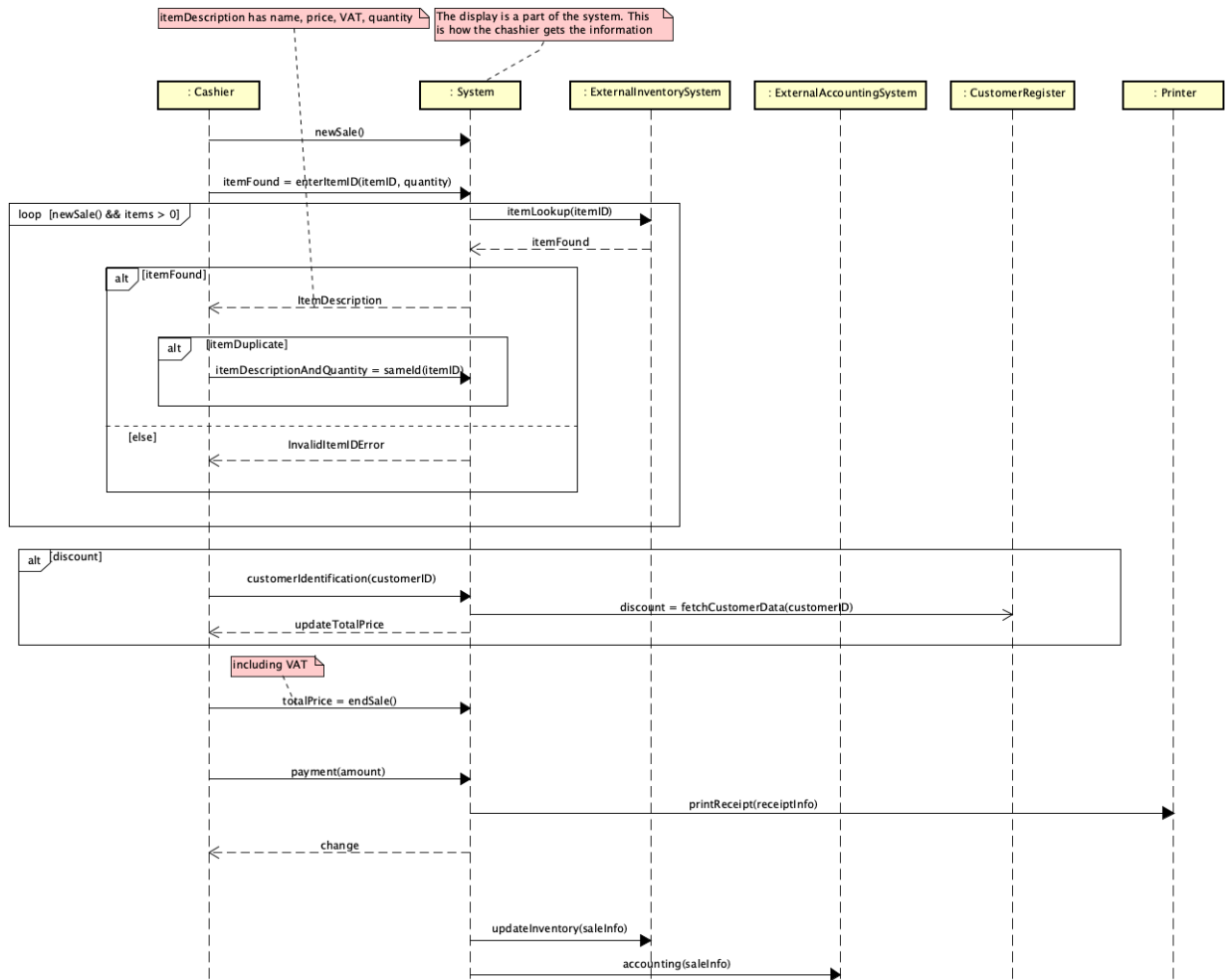


Figure 3.2: A sample code extract.

4 Discussion

After carefully reading the definitions of "programmatic DM" and "naive DM", we concluded our model does not fit any of those descriptions.

The DM is simple and easy to read. We can implement both the basic flow and the alternative flow without using extra notes to explain some of the attributes or associations between classes.

I would argue that the DM has a reasonable amount of classes. However, one could argue that the display is unnecessary because is not mentioned anywhere in the task description. To our defense, I would argue a display represents a real implementation of a PoS and in that way benefits the DM for a better implementation of the model in real life.

There are no "spider-in-the-web" classes. One could argue that the cashier could be classified as a "spider-in-the-web" class.

The number of classes and attributes was reduced after implementing both the noun-identification method and the category list, to attributes essential for this DM.

The associations between the classes are both reasonable and easy to understand.

In both of the diagrams, the naming conventions are followed. In the format of "getDescriptionFrom"

The UML was based on the YouTube video provided by the teacher.

For the SSD it was decided to use return variables as arrows for a simpler understanding of the flow, of the diagram.