Seminar 1

Object-Oriented Design, IV1350

Leo Karsikas leoka@kth.se

27 mars 2024

Contents

1 Introduction

This seminar will cover the two tasks described in the first assignment. This report aims to implement a Domain Model and a Sequential Sequence diagram.

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 also used. Analyzing the text of the assignment for all the nous that are the class candidates. '

Category list

For not missing out on potential essential classes without nouns, a category list was used. In this step, the group tries to think through different scenarios which are described in the assignment and the potential solution for each of them.

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

Task 2

For the Sequential sequence diagram the method for implementing a diagram was to start with going through the scenarios described in the task and after that also focusing on the alternative scenarios which could result in optional statements or else statements 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 are all of the requirements presented in the assignment task and considering 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*.

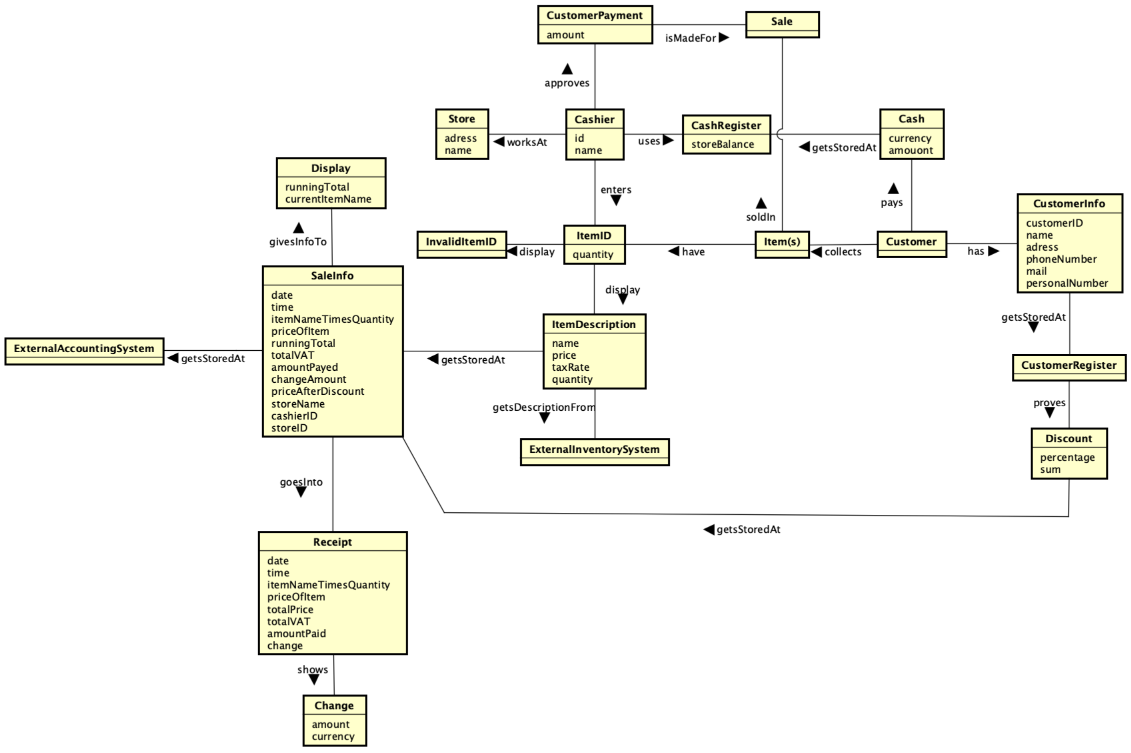

Figure 3.1: Diagram for the DM

Task 2

The second task was about implementing an SSD (Sequential System 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.

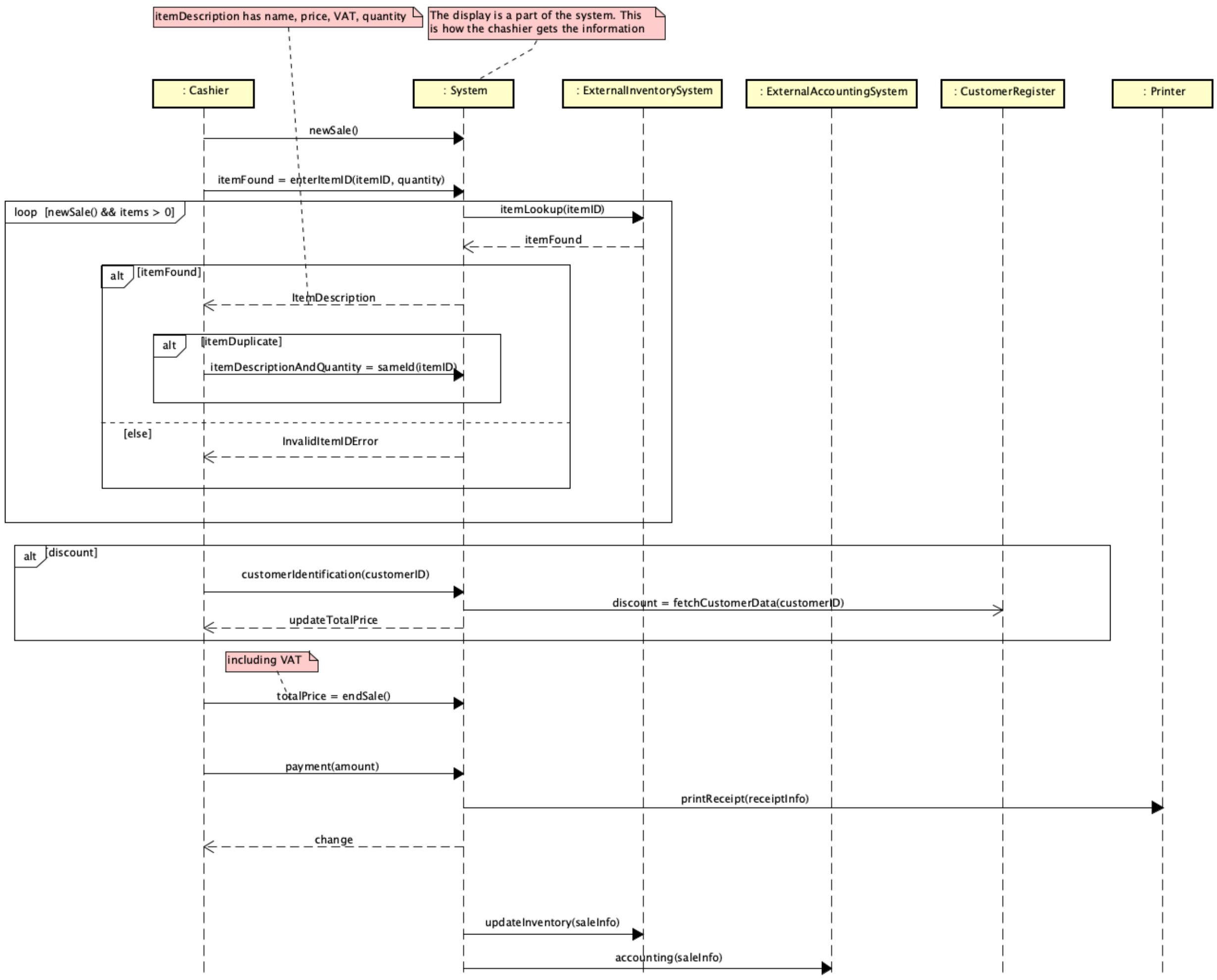
itemDescription has name, price, VAT, quantity
The display is a part of the system. This is how the chashier gets the information
: Cashier
: System
: ExternalInventorySystem
: ExternalAccountingSystem
: CustomerRegister
: Printer
newSale()
itemFound = enterItemID(itemID, quantity)
loop [newSale() && items > 0]
itemLookup(itemID)
itemFound
alt [itemFound]
ItemDescription
alt [itemDuplicate]
itemDescriptionAndQuantity = sameId(itemID)
[else]
InvalidItemIDError
alt [discount]
customerIdentification(customerID)
discount = fetchCustomerData(customerID)
updateTotalPrice
including VAT
totalPrice = endSale()
payment(amount)
printReceipt(receiptInfo)
change
updateInventory(saleInfo)
accounting(saleInfo)

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 ”getsDescriptionFrom”

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.