ITECH7201 - Software Engineering: Analysis and Design

Assignment 1

Design and develop a Payment system in Java

Student Name: Melvin M. Flores

Student ID: 30352985

Lecturer: Mustafa Hashmi

Date Submitted: 09 September 2019

Table of Contents

[1. Requirements Analysis and Design 2](#_Toc18884929)

[1.1 Use Case Diagram 2](#_Toc18884930)

[1.2 Use Cases 2](#_Toc18884931)

[1.2.1 Use Case: Cash Payment 2](#_Toc18884932)

[1.2.1.1 Scope 2](#_Toc18884933)

[1.2.1.2 Level 2](#_Toc18884934)

[1.2.1.3 Primary Actors 2](#_Toc18884935)

[1.2.1.4 Pre-Conditions 3](#_Toc18884936)

[1.2.1.5 Post-Conditions 3](#_Toc18884937)

[1.2.1.6 Main Success Scenario 3](#_Toc18884938)

[1.2.1.7 Extensions 4](#_Toc18884939)

[1.2.2 Use Case: Card Payment 5](#_Toc18884940)

[1.2.2.1 Scope 5](#_Toc18884941)

[1.2.2.2 Level 5](#_Toc18884942)

[1.2.2.3 Primary Actors 5](#_Toc18884943)

[1.2.2.4 Pre-Conditions 5](#_Toc18884944)

[1.2.2.5 Post-Conditions 6](#_Toc18884945)

[1.2.2.6 Main Success Scenario 6](#_Toc18884946)

[1.2.2.7 Extensions 6](#_Toc18884947)

[1.3 Class Diagram 7](#_Toc18884948)

[1.4 Sequence Diagram 7](#_Toc18884949)

[2 Development of Code (Pseudo Code) 8](#_Toc18884950)

[2.1 Functionality to process cash payments 8](#_Toc18884951)

[2.2 Functionality to process card payments 8](#_Toc18884952)

[2.3 Functionality of payments received 8](#_Toc18884953)

[2.4 Functionality of interface and polymorphism 9](#_Toc18884954)

[2.4.1 Interface 9](#_Toc18884955)

[2.4.2 Polymorphism 9](#_Toc18884956)

[3 Reflection on Learning. 10](#_Toc18884957)

# Requirements Analysis and Design

## 1.1 Use Case Diagram

A use case describes use of a software system, in our case, the payment system. It describes how it is used by its users. The use case diagram was brainstormed and transcribed in paper describing the stakeholders involved called actors, as well as the processes involved in the payment system, using the assignment specification as input. Lecture slides and laboratory exercises were studied and applied. Then eventually transferred it into Enterprise architecture. See attached eap file.

## Use Cases

## Use Case: Cash Payment

## Scope

Cash register application

## Level

User goal

## Primary Actors

1. Customer
   * 1. Wants a purchase
     2. Fast service
     3. Minimum effort
     4. Accurate price
     5. Proof of purchase
2. Cashier / Cash Register
   * 1. Wants an accurate entry
     2. Fast entry
     3. No cash payment errors

## Pre-Conditions

Cashier is authorized to access the cash register system

## Post-Conditions

1. Sale is saved
2. Tax is correctly calculated
3. General ledger is updated
4. Receipt is generated

## Main Success Scenario

1. Customer arrives at Cash register
2. Cashier starts a new sale
3. Customer presented the item/s to purchase
4. Cashier enters the item/s
5. Cash register calculates the total price and tax
6. Cash register displays the total price with tax
7. Cashier tells the customer the total price with tax, and asks for payment method
8. Customer selected cash payment option
9. Customer gives cash
10. Cashier gets cash
11. Cashier enters payment at Cash register
12. Cash register processes payment
13. Cash register updates general ledger (Accounting system, Inventory system)
14. Cash register printed a receipt
15. Cashier gives the receipt to the customer and change (if any)
16. Customer accepted the receipt

## Extensions

1. Cash register fails, at any point in the sale process
2. Cashier entered an incorrect entry
3. Cash register cannot find the item to purchase in the system
4. Printer fails, cannot produce a receipt
5. Paper runs out in the printer, cannot produce a receipt
6. Customer pays using a counterfeit bill
7. Power went off, at any point in the sale process

## Use Case: Card Payment

## Scope

EFT POS application

## Level

User goal

## Primary Actors

1. Customer
2. Wants a purchase
3. Fast service
4. Minimum effort
5. Accurate price
6. Proof of purchase

2. Cashier / EFT POS application

1. Wants an accurate entry
2. Fast entry
3. No credit payment errors

## Pre-Conditions

Cashier is authorized to access the EFT POS system and Cash register

## Post-Conditions

1. Sale is saved

2. Tax is correctly calculated

3. General ledger is updated

4. Credit Card Balance is updated

5. Receipt is generated

## Main Success Scenario

1. Customer arrives at Cash register with the EFT POS system

2. Cashier starts a new sale

3. Customer presented the item/s to purchase

4.Cashier enters the item/s

1. Cash register calculates the total price and tax
2. Cash register displays the total price with tax
3. Cashier tells the customer the total price with tax, and asks for payment method
4. Customer selected card payment option
5. Customer gives credit card
6. Cashier gets credit card
7. Cashier enters payment at Cash register
8. Cash register communicated to EFT POS system
9. Cashier taps card at the EFT POS system
10. EFT POS system verified the card
11. EFT POS system requested for payment transaction in the bank of

credit card owner

1. Bank of credit card owner approves the payment transaction
2. Bank of credit card owner updates the cards credit balance
3. EFT POS system received successful transaction
4. EFT POS system displayed transaction complete
5. EFT POS system communicated to Cash register
6. Cash register processes card payment
7. Cash register updates general ledger (Accounting system, Inventory system)
8. EFT POS system printed a receipt
9. Cashier gives the receipt to the customer
10. Customer accepted the receipt

## Extensions

1. EFT POS system fails, at any point in the sale process

2. EFT POS system request for card payment transaction was declined

## 1.3 Class Diagram

The class diagram was also thought of deeply, and generated in paper, identifying the classes need in the payment system, using the assignment specification as input. I have identified the classes involved as well as the interface needed. Classes I used were Payment class. Then I created a CashPayment class and a CardPayment class with card attributes, which extends Payment class. These demonstrated the polymorphism and inheritance features of Java. As for the interface, I used the Payable interface, with pay () method. Then I created a BitcoinPayment class, which implements Payable interface. This makes the Payment system flexible because we can include the BitcoinPayment functionality later without having to have the entire program rewritten. Lecture slides and laboratory exercises were also consulted. It was eventually transferred it into Enterprise architecture. See attached eap file.

## 1.4 Sequence Diagram

A sequence diagram models how responsibilities are spread between objects. Since the actors were identified in the use case diagram, I have displayed it in the sequence diagram and plotted the responsibilities over time, in sequential order. In the assignment specification, only the processing for a credit card purchase was requested for the students to accomplish. I then transferred it into Enterprise architecture. See attached eap file.

# Development of Code (Pseudo Code)

## 2.1 Functionality to process cash payments

Process Cash Payment until user selects display option

Ask user to enter cash details

Store Cash Payment in an Array

Compute for total cash payment

Display total cash payment

## 2.2 Functionality to process card payments

Process Card Payment until user selects display option

Ask user card type

Per card type, compute for surcharge

Ask user to enter card details

Store Card Payment details in an Array

Compute for total card payment and total surcharge

Display total card payment and total surcharge

## 2.3 Functionality of payments received

Ask user to select display option

Display total cash sales in the cash array

Return total cash sales by calling the cash array

Display total card sales and total surcharges in the card array

Return total card sales with surcharges by calling the card array

Compute total product sales

Display total product sales

## Functionality of interface and polymorphism

## Interface

Create Payable interface, with pay () method

Create BitcoinPayment class, which implements Payable

Create BitcoinPaymentDriver class, which tests the pay () method

## Polymorphism

Create Payment class, with attributes amount and paymentWithSurcharge

Create CashPayment class, which extends Payment class

Create CardPayment class with card attributes, which extends Payment class

# Reflection on Learning.

Starting the design right from the start is important, as this will serve as the foundation of the program that we will build. If we choose a wrong design, we can end up building an application that is hard to code from the start, as well as to test and maintain. For the stakeholders involved, it can be costly.

In the Java programming language, Java is simple because it does not use scope terminators, and just ends with a simple curly brace '{'. This is my first time to program in Java, and I have used some keywords like 'public' and 'static'. I also used my own programmer-defined data type, like CashPayment data type and CardPayment data type. I have learned how to use in Java the different loop statements like while statement and for statement, as well as using the switch statement.

I have learned and applied the concept of setters, getters, toString methods and constructors. I have learned and applied the concept of polymorphism and inheritance, as well as the interface.

There were challenges encountered during the coding part, and I initially thought that I will not be using arrays. I attempted coding twice not to use arrays, but my needed variable and value cannot exist outside the method. Thus, I eventually learned that array processing is needed to accomplish my program.

Overall, I have learned a lot in this assignment, which I will use when I am employed by a software company in the future.