Super Shine Car Wash Management System

Designed by **K O M - P L E T**

Krishna

Omar

Mahfuzur

Outline

Car wash system is everywhere, and it is usually associated with fuel filling. However, this project is all about Self Car Wash System.

This document provides the documentation for the entire project which will be frequently used by mainly the developers, and others when needed. Developers use various working tools and techniques, depending on the chosen model, to design and implement a System in close relation to the requirement. Requirement is a thing that is needed or wanted. Therefore, our focus in this project is what the User (I.e. Customer) need or want and how our System react to it in response, full User interaction and satisfaction. In short, the functional requirements of our System.

In our case, our main tools will be jGRASP for java implementation, Gantt Chart for planning and drawing diagrams in Visual Paradigm (VP) for visual representation.

Appendix

1.Vision

- 1.1 Introduction
- 1.2 Purpose
- 1.3 Scope
- 1.4 Definitions, Acronyms, and Abbreviations
- 1.5 References

2.Phase Plan

- 2.1 Inception
- 2.2 Elaboration

3. Software Analysis & Design

- Domain Model
- Class Diagram
- Use Case Diagrams
  - Car Wash System Use case
  - Check Balance SSD case
  - Recharge Balance SSD case

2.Vision

1. Introduction

This introduction provides an overview of the Sunshine Carwash Management system. It includes the purpose, scope, definition, acronyms, abbreviations, references and overview.

1.1 Purpose

This section provides information of the technical software design for the self-car wash management system. The purpose is to describe the technical evaluation for the realization of the business requirement, which describes the various technical parts of the system.

1.2 **Scope** This document also will be a major reference for the system developers.

1.3 Definitions, Acronyms, and Abbreviations

In this project these are the following terms used

· *Customer*: - The client refers to a person who interacts with our system interface.

· *Owner*: - The owner refers to a person or any institution who has invested money to build this system.

· *Washing Card*: - The washing card refers to the special card which connects the system and client.

· *System*: - The system refers to the whole environment which is designed by developers where the client interacts with it using a washing card.

· *UI*: - The UI refers to the system user interface which allows the client to interact with the system and use its functionality.

. *Statistic*: - The System should be able to record all payments and transactions details so that the Owner can get it easily.

. *Payment:* - The System should handle payment calculation and deducts correctly regardless of the Customer choice, including taking account the discount eligibility.

. *Wash Type*: - The Customer ability to freely choose between the options, and the System capability to handle no matter what the choice be.

. *Recharge Card*: - Customer to be able to recharge or deposit his Card in order to use it for washing car using his Bank Credit Card, and the System to allow smoothly & save everything.

1.4 **References**

Project references: -

1. Applying UML by Larman.

2. Pdfs provided by teachers in class.

3.Thesis provided by Douglas as an inspiration.

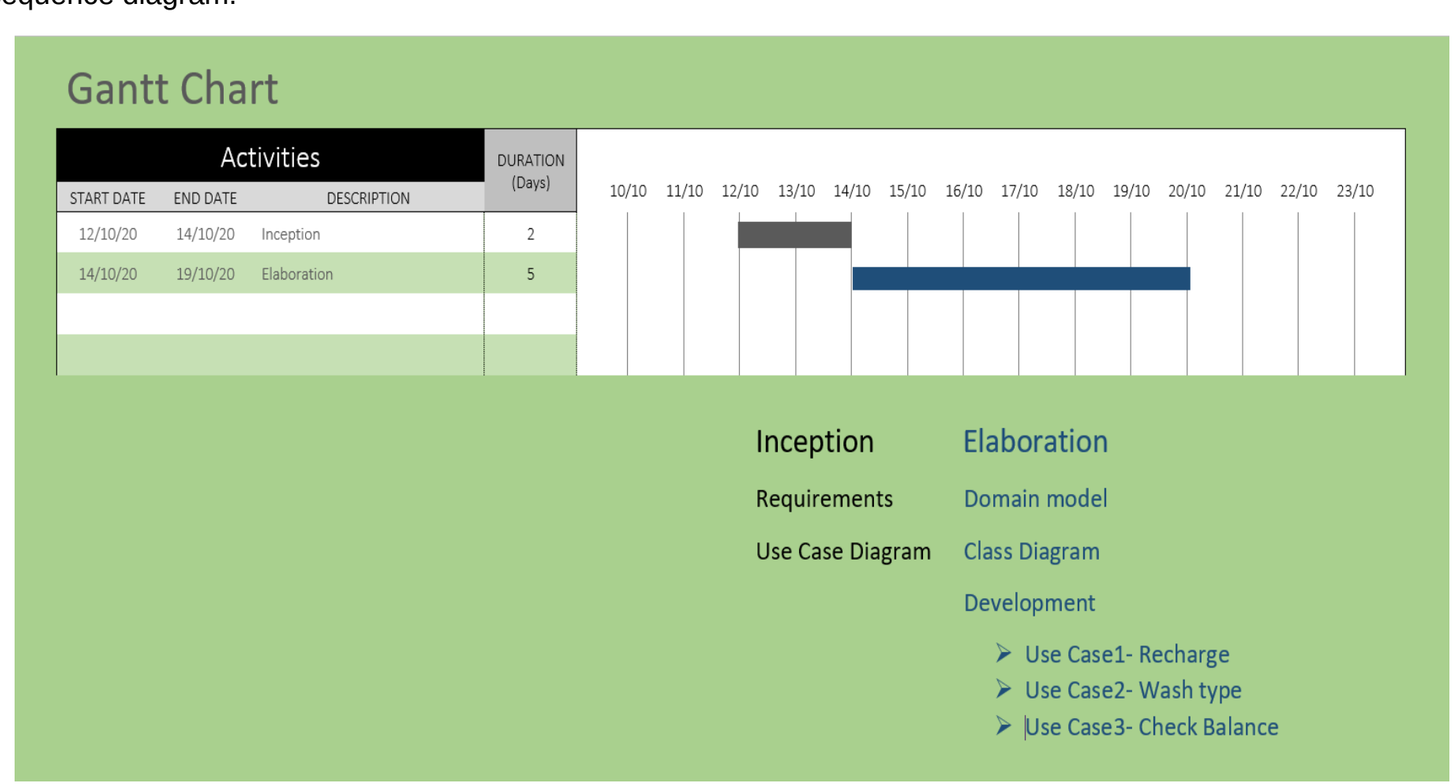
2. Phase plan:

2.1 Inception

Inception phase is the initial phase of the entire software development process. It is shorter than other phases. In this phase, the project vision and scope are defined. The requirements are identified. All the requirements are collected and listed.

2.2 Elaboration: -

This is the second phase of the process. In this phase, the system requirements are highly important. It consists of designing the use case diagram, its descriptions, class diagram, domain model diagram and system sequence diagram.



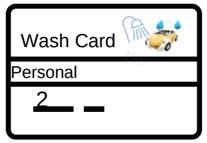
3. Software Analysis & Design

Problem Statement:

The requirements of Car Wash Management System are description of features and functionalities of the system. The System requirements are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the developers. The latter, though, are expressed in structural language (i.e. used inside the company).

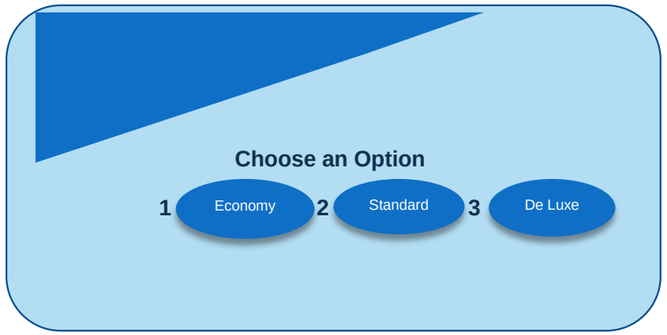
Now we need to design a self-service car wash system, In relation to our requirements, which:

- Accepts only Wash Card (- can be ordered online)
- can choose two options (personal –i.e. registered or common card)
- Allow user to select a specific wash service type (only 3 option available –see below)
- Allow users to recharge their wash card (max 1000 DKK)
- Allow users to check their wash card balance and/or print for receipt if needed.



Here are the 3 Wash type options:

- Economy DKK 50
- Standard DKK 80
- De Luxe DKK 120



The primary actor of our system is our Customer who is the main character interacting with our system.

However, in terms of our Stakeholders, we have the Owner, Car wash system machine maintainers, Suppliers, Bank and so on so forth.

Now, let us start by illustrating a high-level model also knows as Domain Model (I.e.30000-foot view) in UML diagram. but first, let's describe – what is a Domain Model?

Domain Model is an organized and structured knowledge of the problem. It should represent the vocabulary and key concepts of the problem domain. Now, let's take a look as an example our Car Wash Management System Domain model and analyze its components. We are dealing with a simplified Car Wash System Domain.

A Car Wash System is a machine that a Customer use through a Card only in order to buy or recharge its Card or check its balance in its Card. And the Owner would like to get Statistics from time to time.

So, what are the key concepts:

_ *Customers, Wash Card,* Car Wash System*,* Wash Types*,* Owner*,* Car*,* Statistics*,* Credit card*...etc.*

Now, our job is to transform the above list of names into their corresponding 'Class-names'.Domain Model is a visual representation of the conceptual classes or real-world objects in a domain interest. And as such, we have tried to recruit all the nouns and decided those that worth ending up in our Domain model, in accordance with our requirements.

In short, we just need the Domain objects and their associations as shown perfectly below

UML Design Feature:

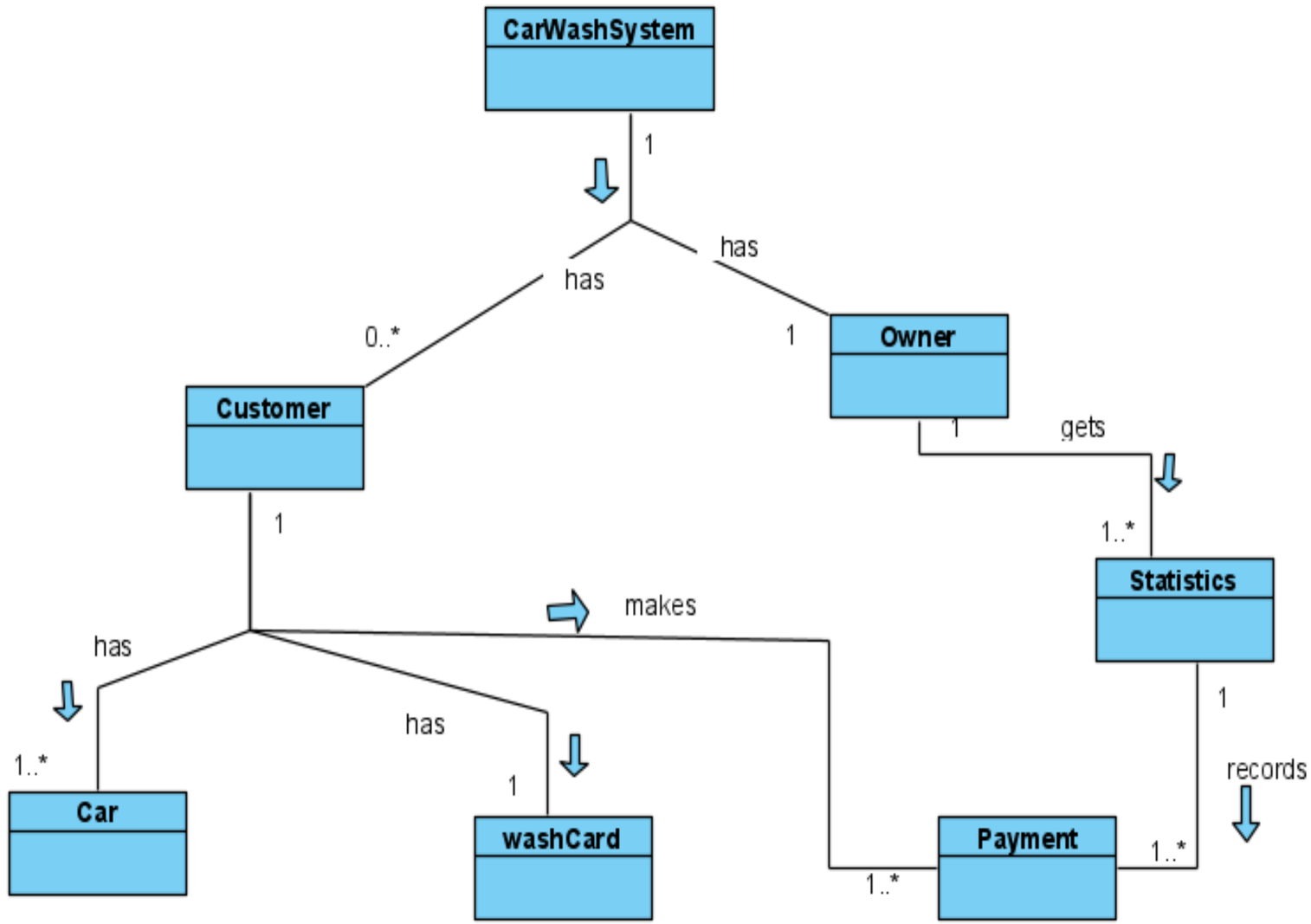
- Create Domain Model
- Create Class Diagram
- Create Use Case Diagram
- Create System Sequence Diagrams and their corresponding Use Case Description side-by-side.

System passes the operation to wash the car

System operates according to the user input using the program

User uses his wash card and gives input to the

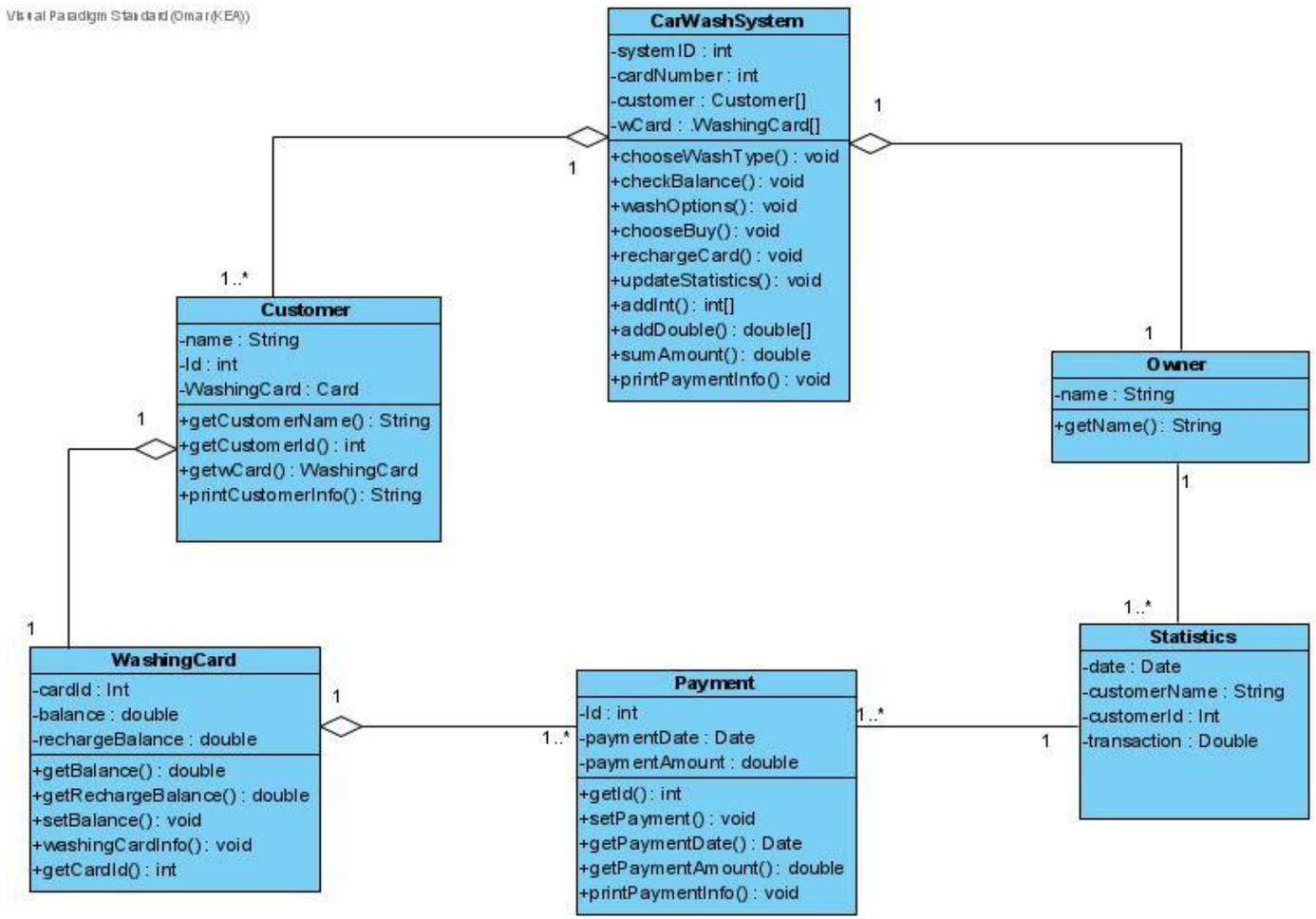
Domain model



Class Diagram:

It defines the structures of the entire System by identifying the Static structure of the objects in the System. It also defines the Attributes & Operations for the objects of each Class, plus the relationship between the objects in the System. In short, it is the Graphical representation of the entire System and helps in analyze while interacting with the Customers. Below, we will illustrate Class Model along with all its elements like the objects, classes, links and associations between all the classes. An object is an abstracted thing that comprises Data structure and Behavior. Each object can be identified distinctly from the application system perspective. And a Class it's a group of objects that share or possess the same Data and Operations. Finally, to define the relationship between objects and classes two terms are used, namely, link and association. A link defines the connection between two objects. In comparison, an association is a group of links, and a link is an instance of association.

For example, a Car Wash System has an Owner, and has Customer. In the same token, Customer has Wash Card to use in the System, and Wash card has Payment. But there is normal association between Owner & Statistics, as the Owner can get Statistics if needed. Same goes between Statistics and Payment, because there are payments detail in the Statistics.

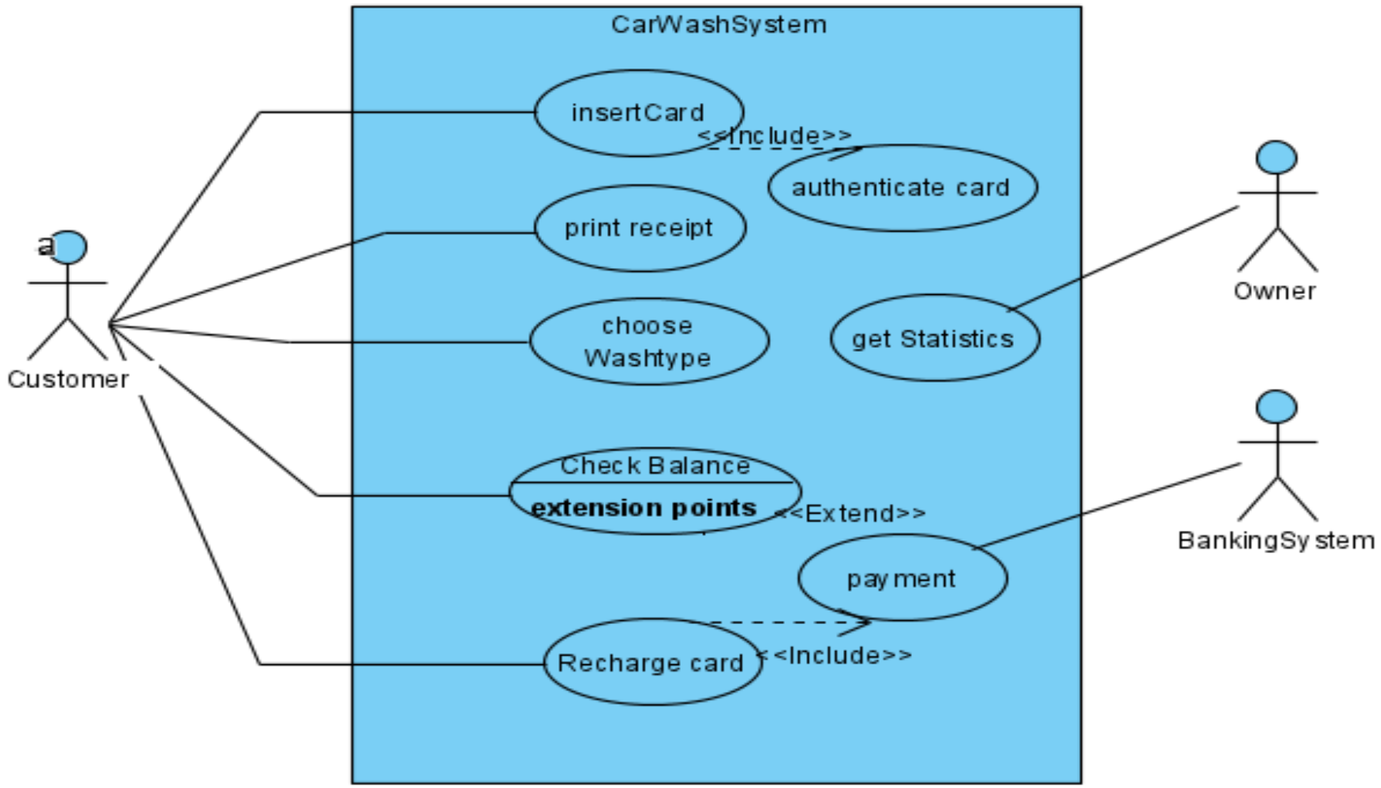


Use case Diagram and Descriptions:

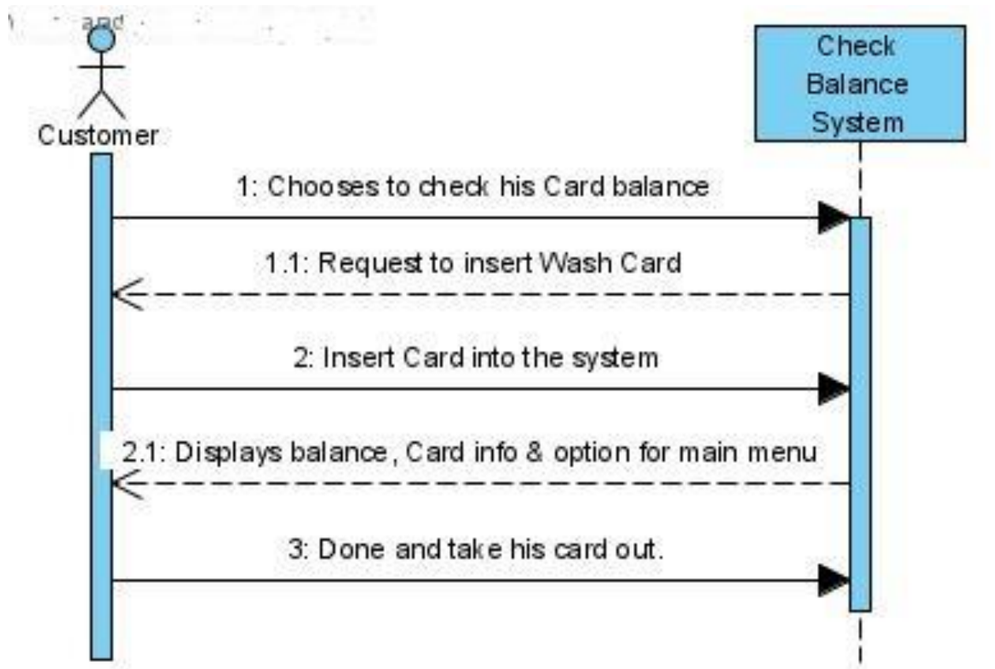
We will try to illustrate first and foremost, our main Use Case. And then, we decided to better showcase each Use Case description and its corresponding System Sequence Diagram (SSD). We choose side-by-side for better illustration and design purposes.

Car Wash System Use Case:

As seen below, we have only one Primary Actor being the Customer and two Supporting Actors namely Bank system and Owner. The reason being is that our two supporting actors are not really interacting per se with our System. However, after deep thought, one can say we may have a Technician for maintenance, and Administrator for overall supervision...so on so forth. But for better design and implementation we choose for simplified version.



Check balance SSD case:



Use Case Description

Check balance

1. Customer: Chooses to check his card balance.

2. System: Request Customer to insert his Wash Card in the system.

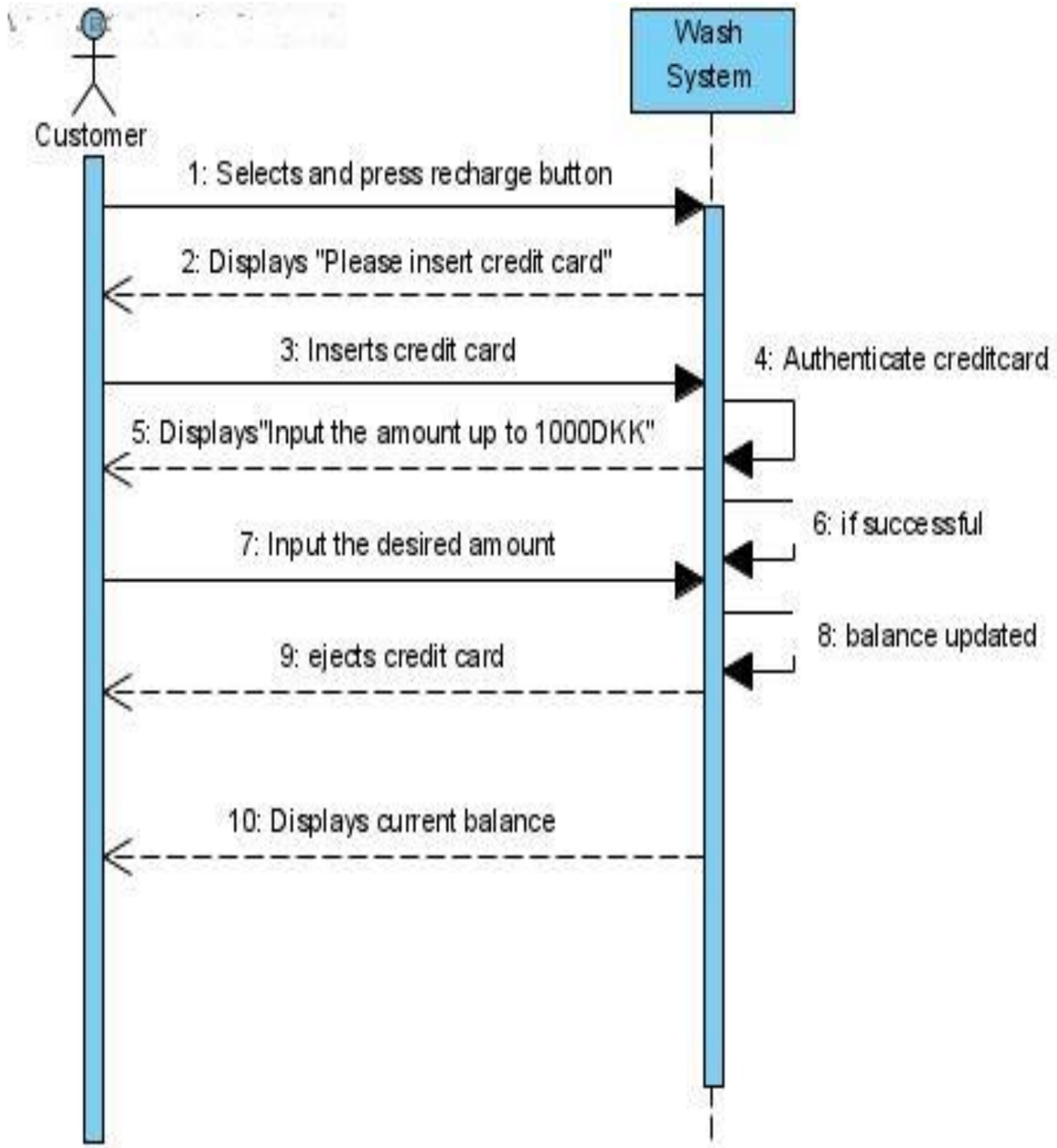
3. Customer: Insert his card in.

4. System: Displays balance, card info & option for main menu.

5. Customer: Finished and take his card out.

```
public static void checkBalance(){
try{
   Scanner scanner = new Scanner(System.in);
   System.out.println("Enter the customer Id");
   int i = scanner.nextInt();
   while(i>customer.length){
   System.out.println("please enter a valid number from 1 upto "+(customer.length));
   i = scanner.nextInt();
   }
   System.out.println("Your balance  is " + wCard[i-1].getBalance()+" DKK.");}
catch(InputMismatchException exception){
   System.out.println("Input error, please enter valid options");}
catch(ArrayIndexOutOfBoundsException e){
   System.out.println("Array out of bound , please enter valid options.");}}
```

Recharge Card SSD case:



Recharge card use case description:

1. Customer select and press recharge button in the carwash system.

2.System displays "Please insert credit card"

3. Customer insert his card

4. System - Authentication validation process and then displays "Input amount to deposit"(max 1000 DKK)

5. Customer input desired amount

6.System updates Customer Card and output "OK"

7. System displays current balance

8. Customer take his card out or proceed otherwise.

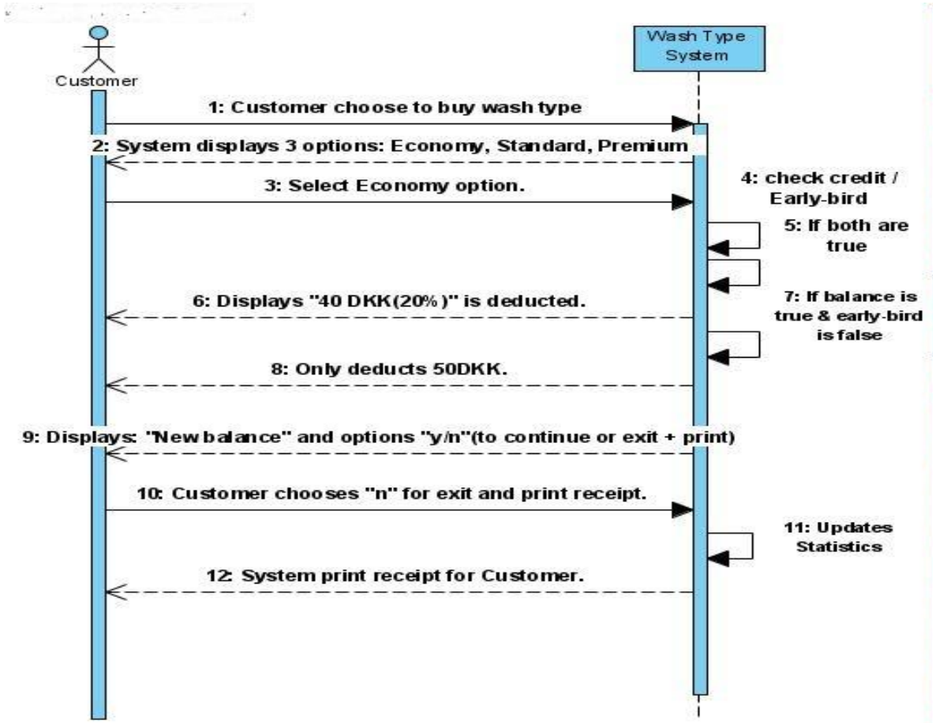
```
public static void rechargeCard(){
  try{

    Scanner scanner = new Scanner(System.in);
    System.out.println("Please enter the customer Id");

    int j = scanner.nextInt();
    while (j>customer.length){
       System.out.println("Please enter valid number");
       j = scanner.nextInt();}
       System.out.println("How much money would you like to deposit on your wash card?");
       double rechargeAmount = scanner.nextDouble();
       while(rechargeAmount>1000){
          System.out.println("You can only recharge upto 1000 DKK.");
          rechargeAmount = scanner.nextDouble();}
          paymentInfo[j-1]= paymentInfo[j-1]+rechargeAmount;
          wCard[j-1].setBalance(wCard[j-1].getBalance() + rechargeAmount);
          System.out.println("Your new balance is " +wCard[j-1].getBalance() + "DKK");}
     catch(InputMismatchException exception){
          System.out.println("Input error, please enter valid options");}
    catch(ArrayIndexOutOfBoundsException e){
                       System.out.println("Array out of bound , please enter valid options.");}}
```

Here is SSD and Use case description for 3 Wash Type options

However, in the SSD we opted at choosing one of the three available option of Wash Type, namely, the Economy one.



FULLY DRESSED
USE CASE DESCRIPTION

CHOOSE WASH TYPE

Primary Actor:
_ Customer

Stakeholders & Interest
Customer: wants the ability to choose & buy wash options and ultimately wash his car.
Owner: wants smooth, easy to use System, profitable & able to get statistics of all transactions.

Precondition:
Customer already has Washing Card.

Post condition:
After choosing option System updates the Washing Card balance & records it.

Main Success Scenario:
1. Customer wants to buy wash type option.
2. System displays different options (Economy, Standard & Premium)
3. Customer selects Economy option
4. System check first if there is enough balance, and, second discount eligibility.
_ if the balance is true and early-bird discount is true then System deducts 40DKK (minus 20%).
_ If balance is true and early-bird discount is false then System deducts 50 DKK.
7. System then displays "y" for continue and "n" for exit & print receipt".
8. Customer selects "n".
9. System print the receipt and thank the Customer for visiting & purchasing.

Alternative Scenario:
1. Customer insert card into the system
2. System authenticate washing card
3. Customer choose the option Economy
4. System displays warning that the card does not have enough credit.
5. Customer clicks the exit button.

Special Requirements:
1. Simple and easy UI for the Customer
2. Multi language options.

```
switch(washType){
case 1:
   if((wCard[i-1].getBalance()>=50)&& (currentTime>=6) &&(currentTime<=13)){
      double firstBalance = wCard[i-1].getBalance();
      //double discount = 50*20/100 = 10
      wCard[i-1].setBalance(firstBalance-40);
      double secondBalance = wCard[i-1].getBalance();
      double totalSpent = firstBalance - secondBalance;
      totalAmountSpent = addDouble(totalAmountSpent,totalSpent);
      eachTotalSpent[i-1]= eachTotalSpent[i-1]+totalSpent;
      System.out.println("Thank you for choosing economy option\n You got 20% discount(From 6am t
      }
      else if((wCard[i-1].getBalance()>=50)&& (currentTime<6) ||(currentTime>13)){
         double firstBalance = wCard[i-1].getBalance();
         wCard[i-1].setBalance(wCard[i-1].getBalance()-50);
         double secondBalance = wCard[i-1].getBalance();
         double totalSpent = firstBalance - secondBalance;
         totalAmountSpent = addDouble(totalAmountSpent,totalSpent);
         eachTotalSpent[i-1]= eachTotalSpent[i-1]+totalSpent;
         System.out.println("Your current balance is"+ wCard[i-1].getBalance());}
         else {
            System.out.println("You have insufficient balance please recharge first.");
            rechargeCard();}
            break;
```