Ryerson University

Department of Electrical & Computer Engineering
Faculty of Engineering & Architectural Science

Course Title:	Object Oriented Design and Analysis
Course Number:	COE528
Semester/Year (e.g.F2016)	F2018

Instructor:	Olivia Das

Assignment/Lab Number:	
Assignment/Lab Title:	Project

Submission Date:	24/11/2018
Due Date:	25/11/2018

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
LE	TRAN	500783715	9	

Reset Form

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

COE528 PROJECT REPORT

1. Use Case Diagram

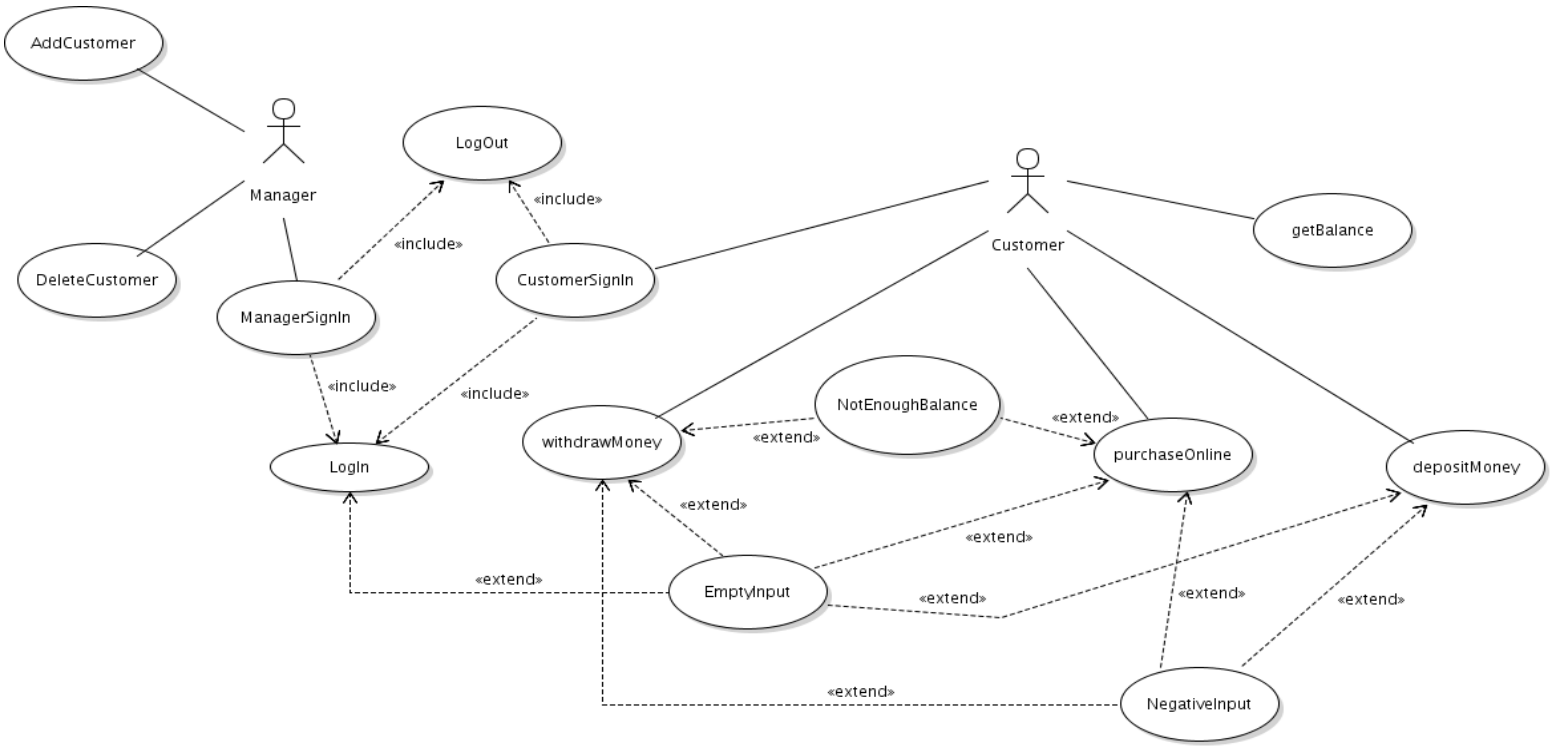

Figure 1. UML Use Case Diagram for the project.

The diagram maps the use cases for two actors: Manager and Customer. There are in total 13 use cases: Manager has 3 use cases (AddCustomer, DeleteCustomer and ManagerSignIn) and Customer has 5 use cases (CustomerSignIn, GetBalance, WithdrawMoney, DepositMoney and PurchaseOnline.) There are 5 use cases that are included in or extend another use case: LogIn and LogOut are both included in ManagerSignIn and CustomerSignIn. EmptyInput extends LogIn, WithdrawMoney, DepositMoney and PurchaseOnline. NegativeInput extends WithdrawMoney, DepositMoney and PurchaseOnline. NotEnoughBalance extends WithdrawMoney and PurchaseOnline.

AddCustomer Use Case Description:

1.Name: AddCustomer

2.Participating Actor: Manager

3.Entry Condition: Manager logged in with username “admin” and password “admin”

4.Exit Condition: Manager has created a new bank account

5.Flow of Events:

1. Manager inputs the username of the new Customer to be added

2. Manager inputs the password of the Customer

3. Manager presses the Add button to add the Customer

4. If the username is taken by another Customer, the application notifies Manager to input a new username; otherwise the application displays the Customer has been added

2. Class Diagram

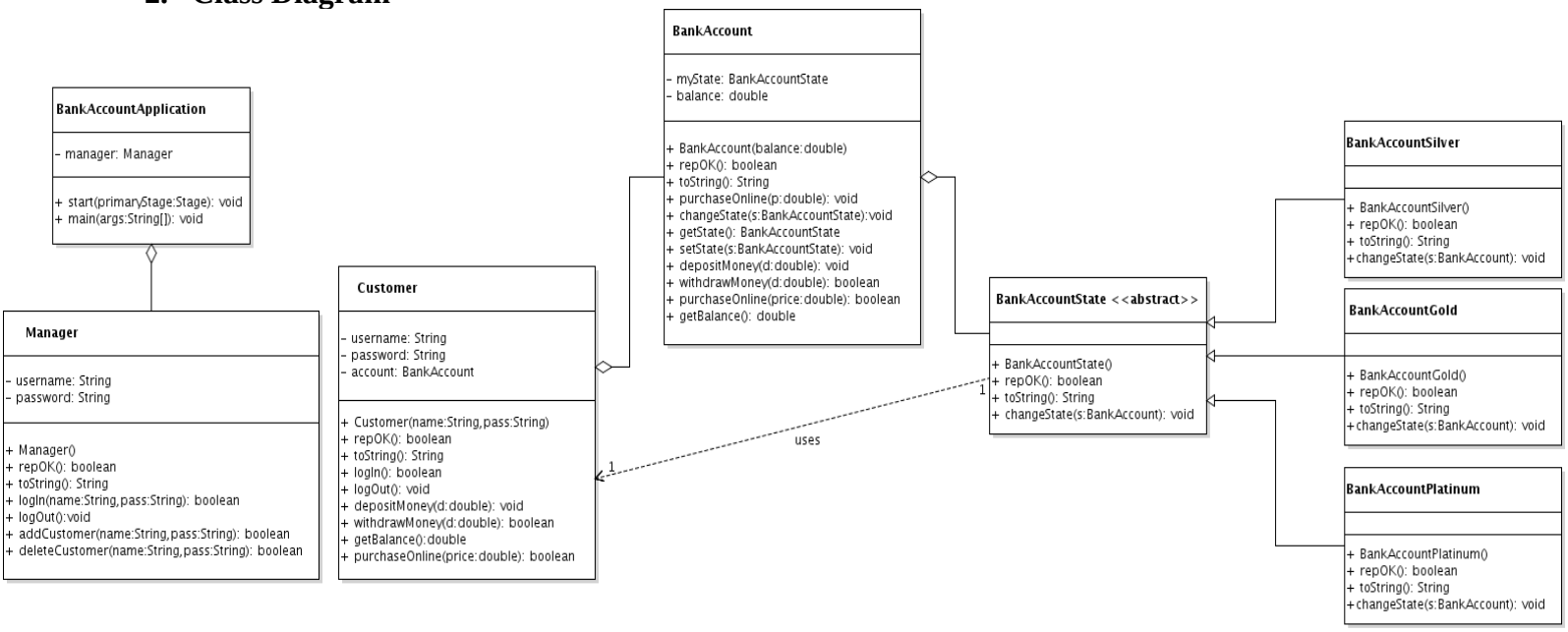

Figure 2. UML Class Diagram for the project.

There are 8 classes in the diagram: Manager class is an aggregate of BankAccountApplication class (JavaFX Class.) Customer class has a BankAccount class, and BankAccount class has a BankAccountState class. BankAccountState class uses Customer class. There are three classes which inherit from BankAccountState class (BankAccountSilver, BankAccountGold, and BankAccountPlatinum.)

3. State Design Pattern in Class Diagram

The BankAccount class is the Context class which includes an abstract State class (BankAccountState). In the BankAccount class, there are methods that uses the BankAccountState class such as getState, setState and changeState. The concrete states which represent the ‘level’ of an account are BankAccountSilver, BankAccountGold, and BankAccountPlatinum. The changeState method is inherited from BankAccountState class by those three classes for polymorphism. These state classes use the Context class which is the BankAccount class in the changeState method.