




**Department of Electrical,
Computer, & Biomedical Engineering**
Faculty of Engineering
& Architectural Science

Course Title:	COE528
Semester/Year	W2024

Instructor	Boujemaa Guermazi
Teaching Assistant (TA)	Aman Kumar

Assignment:	Final Project: Banking Account Application
--------------------	--

Submission Date:	26/02/2024
Due Date:	26/03/3024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Saini	Jannis	501168150	14	

*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.rverson.ca/senate/current/pol60.pdf>

Bank Account Application Report

Introduction

This project required the creation of a simple Bank Account Application. The application created holds one manager and zero to infinite amounts of customers. Each customer holds one bank account with a level associated with the balance they currently have deposited; Silver, Gold, and or Platinum. Managers have the power to add and delete customers. Customers once created by the manager are able to deposit, withdraw and see their balance and level in their account. Customers are also able to use their account for online purchases that are \$50 and above. Depending on the customer's level they will also be charged an additional tax fee when completing an online transaction.

The Overview Clause for this project has been written in the abstract BankAccount class. Below is an explanation for the use case and class diagram created to represent the code. In addition, while writing the code for my GUI I tried to add a few additional features where the manager can search for a customer and a table view where customers are listed is visible. Unfortunately, I was not able to get that part of the code working in time so there are a few things commented out and a few extra methods and variables in some classes (mainly in the managerInterfaceController and manager class).

Use Case Diagram Description

The Use Case Diagram can be found under the appendix as *Figure 1.0*. The Use Case Diagram helps to visualize the interactions between methods and their actors that initiate them. For this project we have two actors; manager and customer. The manager and customer are each able to initiate their methods through button interactions on the GUI. The diagram helps see the flow of events after a method is initiated by an actor. Both manager and customer can login and logout of the system. Managers are able to add and delete customers as well. The customer however is able to do more through button interactions. A customer can deposit, withdraw, get their balance, and complete an online purchase. Each of these procedures have their own flow of events. Below is a description of the withdrawMoney case.

Use case name: withdrawMoney

Participating actors: Initiated by customer

Flow of events:

1. Customer enters the amount of money they wish to withdraw
2. Customer clicks the Withdraw button.
3. getMoney is called to see the current balance
4. getMoney confirms the withdrawal

5. The customer clicks the update button and their balance and account level is checked and updated on screen

Entry condition:

- Customer has logged in
- Customer opens the withdraw money window

Exit condition:

- Money was successfully withdrawn
- Customer receives an error message about the withdrawal failure

Quality Requirements:

- Withdrawal money can not exceed current balance in the account

Class Diagram Description

The Class Diagram can be found in the appendix below as *Figure 2.0*. A class diagram is used to help visually see the relationships between classes in a program. The main class in my program that has the most responsibility and operations would be the BankAccount class. This class holds the customer's balance and controls the level assigned to the customer. Any operations that involve changing the balance and level of the customer is done through this class. The Silver, Gold, and Platinum class extend BankAccount. A customer has-a bank account so the relationship between BankAccount and Customer is an aggregation. Customer extends BankAccount and uses its methods and variables to hold and create its balance and level. Through this relationship Customer has the power to receive its balance, account level, tax associated with its account level, and perform operations involving online purchasing. The userInterfaceController and customerInterfaceController control the GUI. It takes advantage of the Customer class and completes button operations using Customer and its relationship with BankAccount. The final class is Manager. The manager has-a customer so its relationship can also be described as an aggregation. Using this relationship the Manager class adds and deletes its customers using a username and password. The mangerInterfaceController along with the userInterfaceController take advantage of this relationship to complete their button operations.

State Design Pattern

The State Design Pattern in the Class Diagram can be seen between BankAccount and the level classes; Silver, Gold, and Platinum. Inside the Customer class a variable was made using... BankAccount account; By default when a customer is created by the manager the account variable is set to equal new Silver(). While a customer is logged in and depositing, withdrawing, or completing an online purchase their balance is continuously checked and changed. At the end setAccountLevel() is called which goes through three if statements and checks the level requirements. The customer's account is then changed to equal the new level. This creates a state pattern by assigning levels to a customer through their bank account depending on the state their balance is in.

Appendix

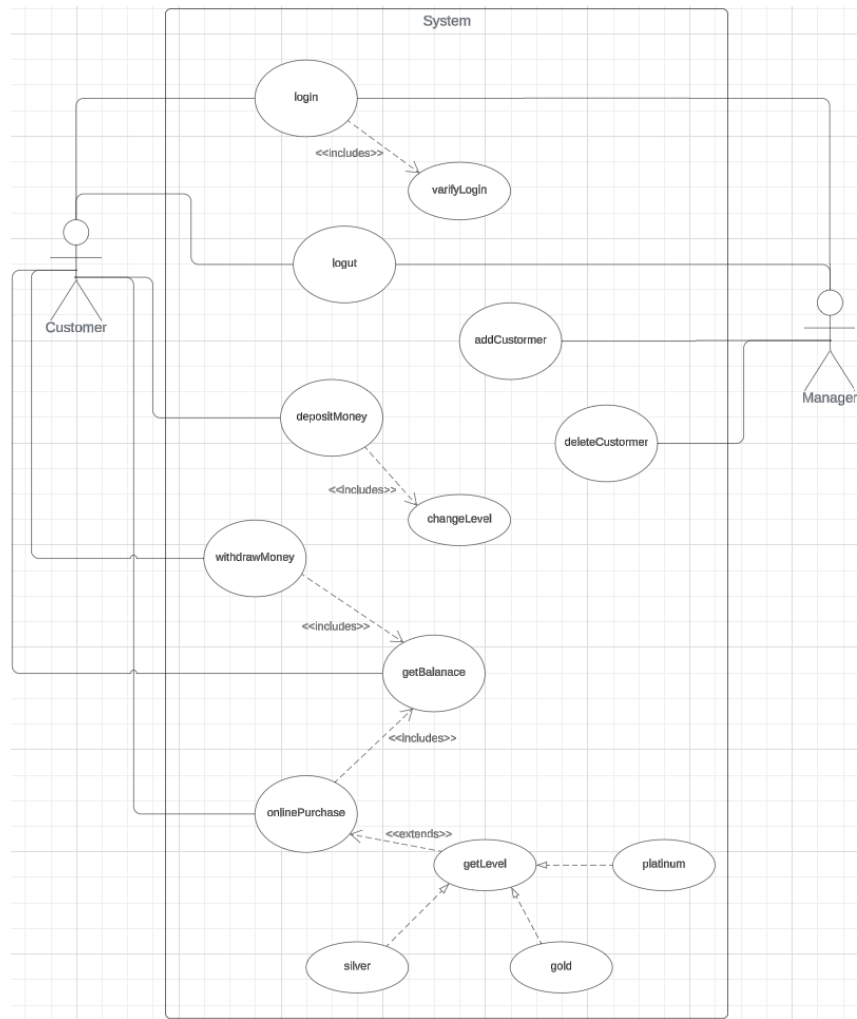


Figure 1.0: Use Case Diagram

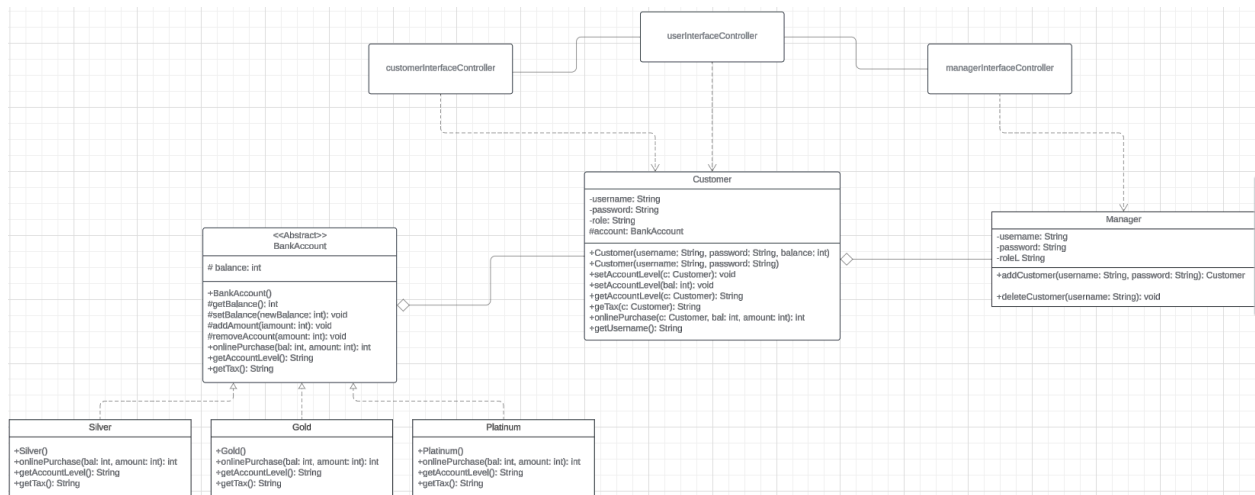


Figure 2.0: Class Diagram