Toronto Metropolitan University

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

Course Title:	COE
Course Number:	528
Semester/Year (e.g.F2016)	2024

Instructor:	Boujemaa Guermazi

Assignment/Lab Number:	5
Assignment/Lab Title:	Term Project: Bank Account Application

Submission Date:	3/23/2024
Due Date:	3/23/2024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Rathod	Ricky	501104462	092	R.R

*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: https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf

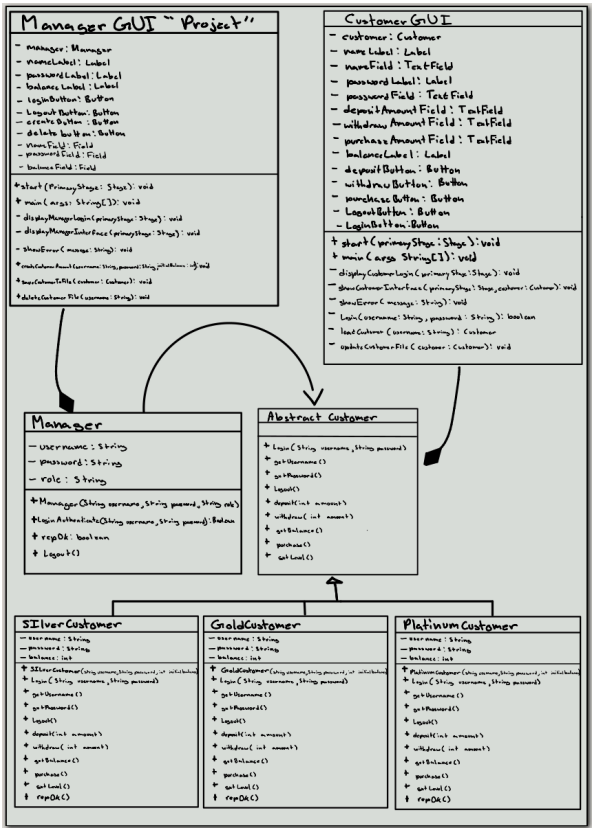
To begin with, I would like to explain my logic on how I thought this program should've been implemented. As mentioned in the project manual, there are two types of users in the Bank application. Specifically, there can only be one Manager, and zero or more customers. Moreover, it is noted that there are three distinct types of customers with their level of status based on their initial balance.

Customers can log in, deposit or withdraw their money, check their balance, make online purchases, and log out. Based on this information, my logic was to create an abstract customer class that has abstract behaviours that each level of customer will have access to—specifically, silver, gold, and platinum customers. These features will be implemented in the CustomerGUI.

As for the manager, this is where I will implement the add, delete, login, and logout features. Moreover, the manager would be the one to assign the level for each customer, so I shall implement a feature where when the manager creates the account for the customer, the manager can automatically assign a level to that customer based on their starting balance (user input would apply here for username, pass and balance). This will be completed in the ManagersGUI.

Therefore, I only see a requirement for 5 classes. That being, the Manager class, Abstract Customer, GoldCustomer, SIlverCustomer and PlatinumCustomer. Excluding the two GUI's.

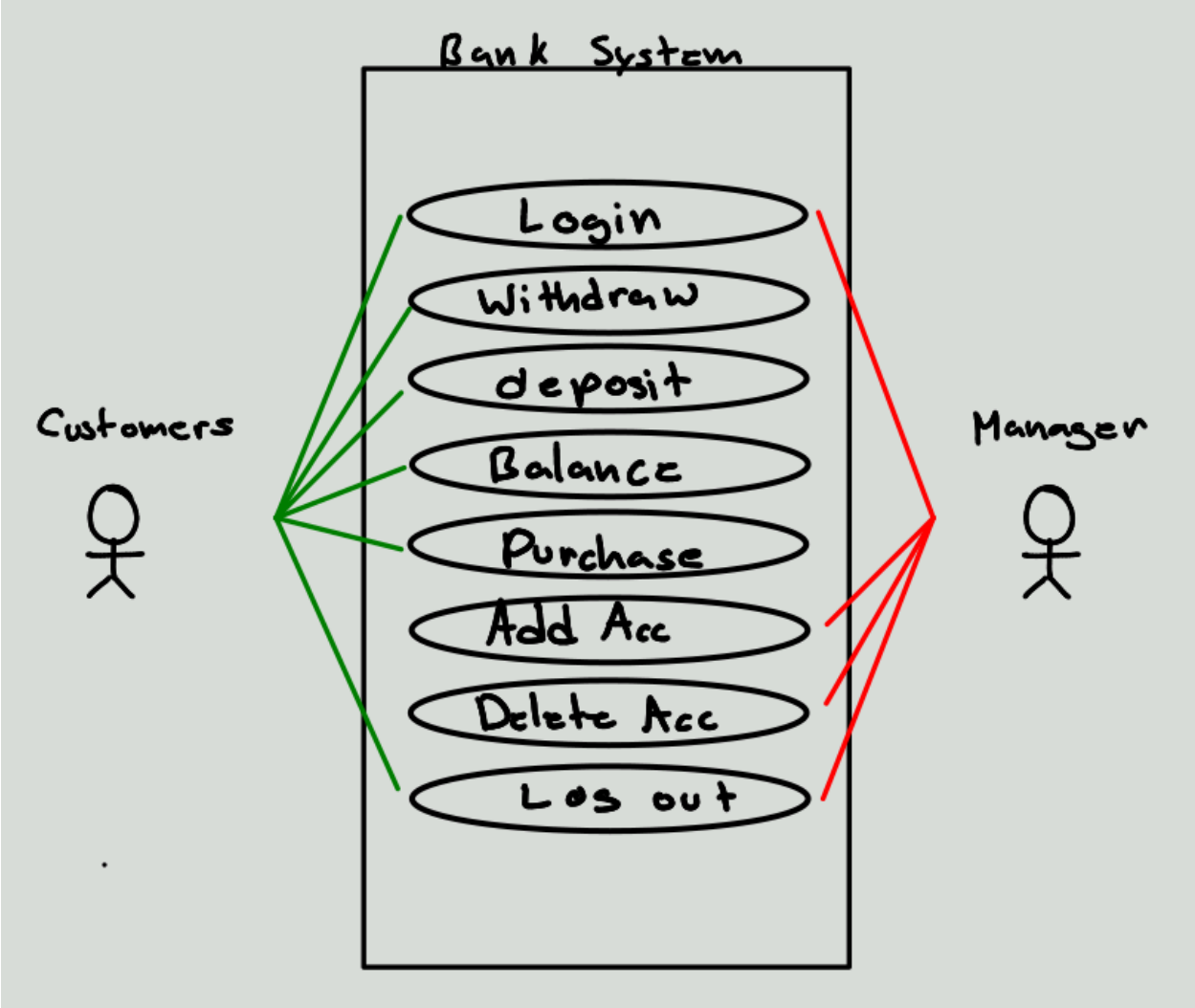
UML Class Diagram



"To view said diagram in more detail, refer to the added PDF Tittled Project Diagrams"

Taking A look at the Class Diagram, it can be seen that I've created three subclasses named "SilverCustomer, GoldCustomer, and PlatinumCustomer". These subclasses inherit the behaviour of the parent/SuperClass named "Abstract Customer". As mentioned before, my logic was that since there are multiple types of customers accessing this application, it would be best if there was an abstract class where the children of said class would simply inherit the features. Moving Forward, it can also be seen that the Manager Class although has similar functionality to the abstract Customer class, is not directly related and does not inherit the same features but is its own entity. With both the Manager and Customer classes and their subclasses completed, I then moved on to implementing the features of both classes into two separate GUIs. It can also be noted that these GUIs would not function without the information from their respective classes, meaning that the managerGUI would not work without the information from the manager.java files/class. And likewise with the customer GUI. This is in turn the reason for the composition relationship which is denoted as a solid diamond pointing to the two original classes.

UML Case Diagram



"Diagram depicting the Use Case Diagram based on the Project"

Moving onto the Use Case Diagram, it can be seen that two people are interacting with this banking system. The Customer and the Manager. It can be seen that the Manager can log in, create an account, Delete an account and log out. The customers can log in, see their balance, withdraw or deport money and also make online purchases including the ability to log out.

Specifically, the login Use case is similar for both the manager and the customer. However, the difference is that for the customer side, the system reads the account information created by the manager. What I mean by that is that, when asked to log in on the customer side, the file readers and authentication process occur to ensure the right "Account" is accessed. The process looks something like this.

Customers Use Case.

1. Login.
2. DisplayCustomerLogin.
    a. Username, Password entry [Must not be null]
    b. if (login(username, password)) {

```
// If login successful, transition to customer interface (i.e "Account" is found)
primaryStage.close();
Customer customer = loadCustomer(username);
if (customer != null) {
    showCustomerInterface(primaryStage, customer);
} else {
    showError("Failed to load customer data.");
}
} else {
// If login failed, display error message
showError("Invalid login credentials. Please try again.");
}
```

3. showCustomerInterface.

References

[1]

Login - Toronto Metropolitan University Central Authentication Service. (n.d.). https://courses.torontomu.ca/d2l/le/content/839005/viewContent/5644040/View