Ryerson University

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

Course Title:	Object Oriented Eng Analysis and Design
Course Number:	COE 528
Semester/Year (e.g.F2016)	W2024

Instructor:	Boujemaa Guermazi

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

Submission Date:	Tuesday, March 26, 2024
Due Date:	Tuesday, March 26, 2024

Student LAST Name	**Student FIRST Name**	**Student Number**	**Section**	**Signature***
Sundaresan	Ashwin	501159998	14	Ashwin.S

*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

Ashwin Sundaresan
501159998
Section 14

Introduction

The Simple Bank App is an easy-to-use computer program made with JavaFX and Java on NetBeans 20. The program's main purpose is to allow the manager and customer to perform tasks unique to their role. The manager can add or remove customers from the bank while the customer can check balance, withdraw/deposit money and do online shopping. Customers are split into three levels based on their account balance: silver, gold, and platinum. Each level has its own benefits and purchase fees that the customer must pay when using the online store. This app is a simplified version of a large-scale bank application that can be used as a base to further improve on.

Class Diagram

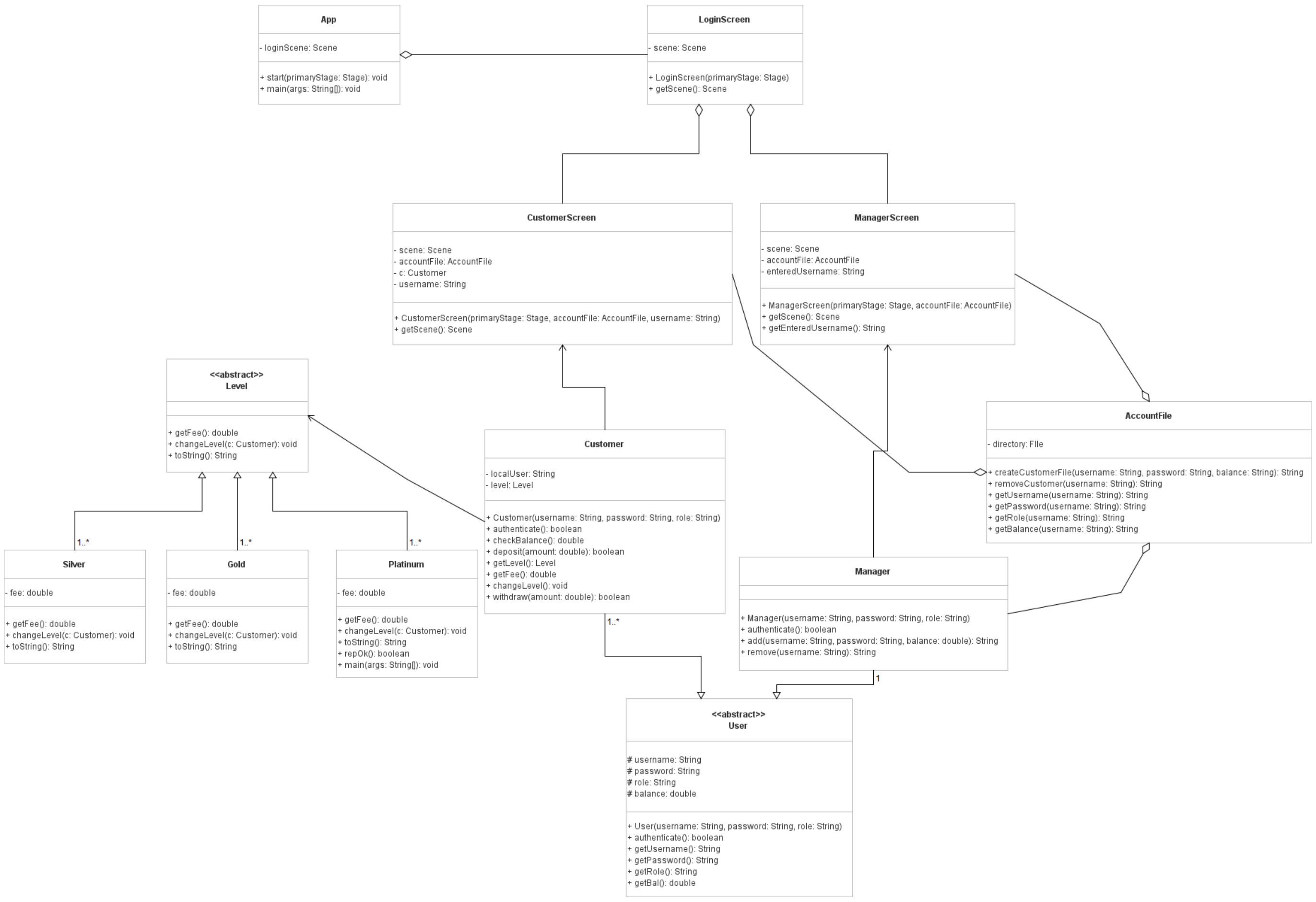
App
- loginScene: Scene
+ start(primaryStage: Stage): void
+ main(args: String[]): void
LoginScreen
- scene: Scene
+ LoginScreen(primaryStage: Stage)
+ getScene(): Scene
CustomerScreen
- scene: Scene
- accountFile: AccountFile
- c: Customer
- username: String
+ CustomerScreen(primaryStage: Stage, accountFile: AccountFile, username: String)
+ getScene(): Scene
ManagerScreen
- scene: Scene
- accountFile: AccountFile
- enteredUsername: String
+ ManagerScreen(primaryStage: Stage, accountFile: AccountFile)
+ getScene(): Scene
+ getEnteredUsername(): String
<<abstract>>
Level
+ getFee(): double
+ changeLevel(c: Customer): void
+ toString(): String
Customer
- localUser: String
- level: Level
+ Customer(username: String, password: String, role: String)
+ authenticate(): boolean
+ checkBalance(): double
+ deposit(amount: double): boolean
+ getLevel(): Level
+ getFee(): double
+ changeLevel(): void
+ withdraw(amount: double): boolean
AccountFile
- directory: FIle
+ createCustomerFile(username: String, password: String, balance: String): String
+ removeCustomer(username: String): String
+ getUsername(username: String): String
+ getPassword(username: String): String
+ getRole(username: String): String
+ getBalance(username: String): String
1..*
1..*
1..*
Silver
- fee: double
+ getFee(): double
+ changeLevel(c: Customer): void
+ toString(): String
Gold
- fee: double
+ getFee(): double
+ changeLevel(c: Customer): void
+ toString(): String
Platinum
- fee: double
+ getFee(): double
+ changeLevel(c: Customer): void
+ toString(): String
+ repOk(): boolean
+ main(args: String[]): void
Manager
+ Manager(username: String, password: String, role: String)
+ authenticate(): boolean
+ add(username: String, password: String, balance: double): String
+ remove(username: String): String
1..*
1
<<abstract>>
User
# username: String
# password: String
# role: String
# balance: double
+ User(username: String, password: String, role: String)
+ authenticate(): boolean
+ getUsername(): String
+ getPassword(): String
+ getRole(): String
+ getBal(): double

Figure 1. Class Diagram for the Simple Bank Application

As seen above in Figure 1, the Simple Bank App utilized various classes to perform the necessary tasks. The specific screen classes are used to aid and minimize the lines of code in the LoginScreen class. The LoginScreen class is called in the App class to start the GUI and the program. The user can now enter their username and password on the popup to login to their account. When the manager has logged in, they are redirected to the MangerScreen GUI where they can either add customers to the bank, remove customers from the bank or log out. Once the customer has logged in, they are redirected to the CustomerScreen GUI where the customer can perform various tasks like checking their balance, withdrawing/depositing money, do online shopping or simply log out. The CustomerScreen, ManagerScreen and Manager classes all use necessary functions in the AccountFile class that are used to get the corresponding username and password for the customer or are used to create and remove customers from the bank. The Customer class and the Manger class both extend the User class, they both contain their own authentication system and methods based on the type of user. The customer class also splits into 3 levels silver, gold, and platinum, the highest. Each of these levels is associated with the Level class and inherits methods from it. The classes come into play when the user withdraws, deposits, or does online shopping as it updates according to their new account balance.

<u>Use Case Diagram</u>

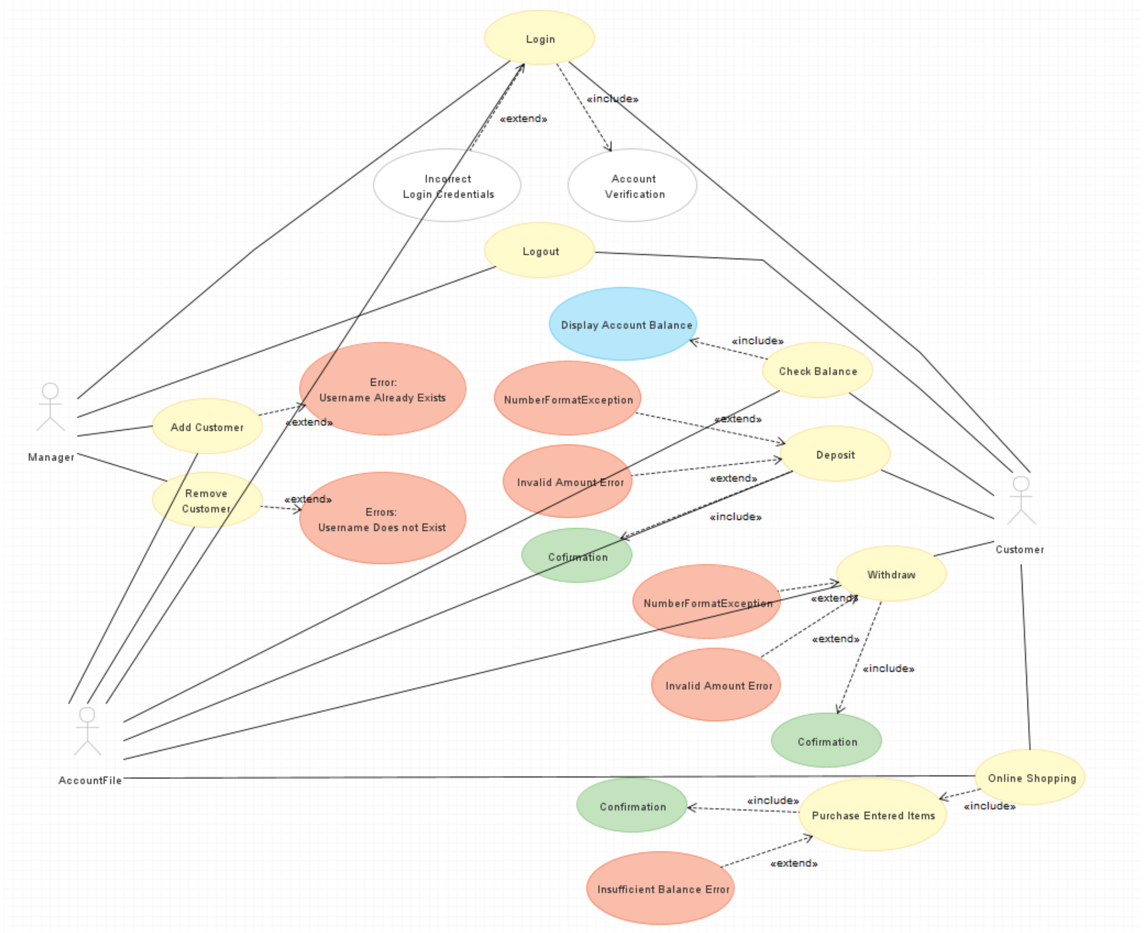

Figure 2. Use Case Diagram for the Simple Bank Application

In the Use Case diagram seen above in Figure 2, the Customer and the Manager are the primary actors that interact with the program. Both the Customer and Manager can log in and log out when the correct credentials are entered. The Manager can add or remove customers from the bank, an error message will pop up if the customer already exists when trying to add, or if the customer does not exist when trying to remove. The Customer can deposit money, withdraw money, check their balance, and do online shopping. When withdrawing money or doing online shopping the amount being withdrawn or spent is checked to see if it is a real amount and if the Customer's account balance goes negative or not. When depositing money, the amount being deposited is checked to see if it is a real amount. All actions display a confirmation when succeeded. All actions, except the log out scenario, done by the primary actors, Manager and Customer, are relayed to a secondary Actor, AccountFile, Updates to the AccountFile are made if methods tell it to do so.

Description of the Online Shopping Use Case

Use Case Name:	Online Shopping
Participating Actors:	Initiated by Customer Information communicated to AccountFile
Flow of Events:	1. Customer enters purchase amount and product name. 2. Customer presses "Confirm" button to make purchase. - CustomerScreen validates if entered amount can be purchased or not. - The fee associated with the Level of the customer and the price of the item is withdrawn from the account. - If the Level of the customer needs to be changed it is changed. 3. AccountFile changes to the Customer's file. E.g. Changes the balance and level. - CustomerScreen displays the updated balance and level.
Entry Condition:	Customer must be logged into the application.
Exit Condition:	CustomerScreen displays the updated balance, or the Customer is notified that the transaction cannot be made and the reason why.
Quality Requirements:	Balance and Level is immediately updated if purchase is made.