Name: Ruben Plaza

Student ID: 500915545

Section: 01

COE528 Project: Easy Financial

1. Describe your UML Use Case Diagram

The UML Use Case Diagram for this banking application has two actors, a customer, and a manager. Both actors have two use cases in common, ‘login’ and ‘logout’. The ‘login’ use case includes a ‘verify log in credentials’ use case which verifies the user’s credentials depending on who initiated the base use case. The customer actor has four of its own use cases. They include: ‘get balance’, ‘make an online purchase’, ‘withdraw money’, and ‘deposit money’. The ‘make an online purchase’ use case includes a ‘verify minimum purchase amount’ use case and an ‘able to withdraw’ use case to see if the user has entered a valid amount. This use case also has an extends relationship with the ‘update state’ use case, to change the state if the account balance falls between a certain range. Like the ‘make an online purchase’ use case, the ‘withdraw money’ use case has an extends relationship with the ‘update state’ user case and an includes relationship with the ‘able to withdraw’ use case. The ‘deposit money’ use case has an includes relationship with the ‘valid amount’ use case which checks if the amount to deposit is greater than zero. The manager actor has two additional use cases unique to itself. It has ‘delete customer’ and ‘add customer’ use cases. The ‘delete customer’ use case has an includes relationship with the ‘check if the user exists’ use case which will see if the user entered a valid customer. The ‘check if the user exists’ use case checks if the user exists and depending on if it does or not it will have an extends relationship with the ‘delete associated account file’ use case. The second use case for the manager is ‘add customer’. This use case has an includes relationship with the ‘verify unique customer’ which in turn has extends relationships with the use cases, ‘create bank account for the customer’ and ‘record customer info’. This sums up all the use cases for the banking application and its actors.

2. Describe your UML Class Diagram

The UML Class Diagram has a top-down structure, with nine classes. There is an abstract User class that contains the info about the user’s username, password, and role. This class describes the common functionality of customers and managers to log in and logout. The Manager class extends from the User class and uses the singleton design pattern. The instance of the Manager class has a list of customer usernames for its reference. The manager is able to add a customer and delete a customer. The Customer class also extends from the User class and has a reference to an instance of the BankAccount class and an instance of the Level Class. The customer is able to deposit, withdraw, get balance, and make an online purchase. The customer's account level is updated dynamically when the balance falls within the different levels' dollar ranges. The customer's bank account is responsible for handling the customer's balance. The BankAccount class has a single instance variable which is balance. The BankAccount class implements the ATM interface and overrides the functions for the deposit, withdraw, get balance, and make online purchase methods. There is also an abstract Level class that is responsible for describing the customer's account level. The abstract Level class has three subclasses, SilverLevel, GoldLevel, PlatinumLevel. These subclasses are responsible for updating the customer's account level dynamically as the balance changes.

3. Mention the class you have selected to document

The class that I selected to write the clauses for is BankAccount.java. This file can be found within the source folder under the 'classes' package.

4. Indicate the parts in the UML Class Diagram that form the state design pattern

The part in the UML Class Diagram that forms the state design pattern is the abstract Level class and its three subclasses, SilverLevel, GoldLevel, and PlatinumLevel. These subclasses methods, increaseLevel, and decreaseLevel are responsible for changing the current state.

5. References

https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm
https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm
https://www.callicoder.com/javafx-fxml-form-gui-tutorial/
https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

6. Notes

In case the program does not compile, check the proof of completion file, it proves that I used everything specified within the 'Programming environment for the project' announcement. The JDK 8 and the NetBeans version is 8.2 were used for this project. To begin using the project, add customers by signing into the manager interface with username: admin and password: admin. The JavaDoc for the project is located in the JavaDoc folder in the project folder. If it cannot be opened, go to the project in NetBeans, under 'run' click 'generate JavaDoc'. The project does not have the package declaration coe528.project because it was causing errors with opening and writing to files, I hope this does not cause too much confusion.