Bank Account Application Report:

Name: Omar Elsisi
Student ID: 501218838
Section Number:

Use Case Diagram

The Use Case Diagram for the Bank Account Application illustrates the interactions between two main actors: the Manager and the Customer. The diagram highlights the primary functionalities each actor can perform within the system. The Manager can log in, log out, add customers, and delete customers. The Customer can log in, log out, deposit money, withdraw money, check their balance, and perform online purchases. Each of these functionalities is represented as a distinct use case, clearly demonstrating the relationship between the actors and their corresponding actions within the application.

Use Case: Add Customer

- Name: Add Customer
- Actor: Manager
- Description: The Manager adds a new customer to the bank system.
- Preconditions: The Manager is logged into the system.
- Postconditions: A new customer is added to the system with an initial balance of $100, and their information is stored in a file.

Main Flow:

1. The Manager navigates to the 'Add Customer' page.
2. The Manager enters the new customer's username and password.
3. The Manager clicks the 'Add' button.
4. The system validates the input.
5. The system creates a new customer with the provided information and an initial balance of $100.
6. The new customer is added to the list of customers displayed to the Manager.

Alternate Flow:

- If the input is invalid (e.g., missing username or password), the system displays an error message, and the customer is not added.

Class Diagram

The Class Diagram of the Bank Account Application provides a comprehensive view of the system's structure, detailing the system's classes, their attributes, methods, and the interrelationships between these classes. The primary classes in the system include:

- BankMain: The main entry point of the application.
- ManagerController: Handles manager-related actions and interactions.
- CustomerController: Manages customer-related functionalities.
- Manager: Responsible for managing customer accounts, including creating and deleting customers.
- Customer: Encapsulates customer details and actions.
- Level Classes (Silver, Gold, Platinum): Represent different account levels based on the customer's balance.

The Customer class encapsulates all details and actions pertinent to a customer, while the Manager class focuses on administrative tasks. The Level classes implement the State design pattern, representing the customer's account level and adjusting fees and functionalities accordingly.

Selected Class: Customer

The Customer class was selected to demonstrate detailed documentation, including the Overview clause, Abstraction Function, Representation Invariant, and method clauses. This class encapsulates customer information and actions, ensuring proper documentation and maintenance of the code.

- Overview Clause: The Customer class represents a bank customer with attributes such as username, password, balance, role, and level. This class is mutable as customers can perform actions that modify their balance and level.
- Abstraction Function: The abstraction function maps the internal representation of the Customer object to a string representation.
- Representation Invariant: The rep invariant ensures that the customer's username and password are non-null, the balance is non-negative, and the level is valid based on the balance.

State Design Pattern

The State design pattern is implemented in the UML class diagram through the Level class and its subclasses (Silver, Gold, Platinum). The Customer class has an association with the Level class, and its state (level) changes dynamically based on the account balance. This design pattern ensures that the system handles changes in the customer's level efficiently, applying appropriate fees and functionalities based on the customer's current level.

References

- Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley Professional.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

This report outlines the essential components and interactions within the Bank Account Application, highlighting the use case and class diagrams, and detailing the selected Customer class with appropriate documentation and design patterns.