### Banking System

#### **Project Objective :**

The objective of this project is to develop a simple banking system application using Java Swing for the graphical user interface. The application allows users to perform basic banking operations such as creating accounts, logging in, depositing money, withdrawing money, checking balances, viewing transaction history, and deleting accounts. Additionally, it continuously displays the total balance of all accounts in real-time and lists accounts with balances above a specified threshold.

#### Functional Requirements

1. **Account Operations:**
   1. Create Account: Users can create new accounts by providing necessary details such as account number, account holder name, initial balance, and password.
   2. Login: Existing users can log in using their account number and password.
   3. Delete Account: Users can delete their accounts, which also deletes associated transaction history.
2. **Banking Operations:**
   1. Deposit: Logged-in users can deposit money into their accounts.
   2. Withdraw: Logged-in users can withdraw money from their accounts, ensuring they have sufficient balance.
   3. Balance Inquiry: Users can check their current account balance.
   4. Transaction History: Users can view a list of transactions performed on their account.
3. **Real-time Display:**
   1. Total Balance Display: The application continuously displays the total balance of all accounts, updating every 10 seconds.
4. **Additional Features:**
   1. List Accounts Above Threshold: The application lists accounts with balances above a specified threshold (default threshold is $100).

#### Explanation of the Project

The project utilizes Java Swing for the graphical interface to interact with users. It employs a simple MVC (Model-View-Controller) architecture where:

* **Model:** Classes like AccountDetails and Transaction represent the data and business logic of the application.
* **View:** The GUI components (JFrame, JPanel, JButton, etc.) are used to create the user interface and handle user interactions.
* **Controller:** Action listeners manage events triggered by user interactions, invoking appropriate methods for account management and banking operations.

Key functionalities are synchronized where necessary (e.g., deposit and withdrawal operations) to ensure thread safety using ReentrantLock from java.util.concurrent.locks.

#### Execution of the Project in Steps

1. **Run the Application:**
   * Execute the main method in BankingSystem class.
   * The main menu of the banking system will appear.
2. **Main Menu Options:**
   * **Create Account:** Click to create a new account with required details.
   * **Login:** Login to an existing account.
   * **Delete Account:** Delete an existing account after confirmation.
   * **Exit:** Close the application.
3. **Banking Operations Menu:**
   * Upon successful login, the user will be directed to the banking operations menu.
   * Options include Deposit, Withdraw, Balance Inquiry, Transaction History, and Logout.
4. **Real-time Display and Threshold Listing:**
   * The application continuously displays the total balance of all accounts and lists accounts with balances above the specified threshold.
5. **Logout and Exit:**
   * Users can logout from their account, which returns them to the main menu.
   * Exiting the application terminates the program.

#### Outcomes

The project aims to provide a functional banking system simulation with a user-friendly GUI. Users can perform essential banking operations securely and efficiently. The real-time display and threshold listing feature enhance usability by providing timely account information.