BANKING SYSTEM

Group Members

1. Jackline Wavinya I163/5446/2022
2. Viola Kambuni I163/6929/2022
3. Ngige Mwaniki I163/4777/2022
4. Joey Shiundu I163/4776/2022
5. James Njange I163/4747/2022
6. Veronica Ndemo I163/1503/2021
7. Daglas Machafu I163/6830/2022
8. Kevin Strong I163/6829/2022

**Simple Banking System Project Documentation**

**Table of Contents**

1. [Introduction](#introduction)
2. [Features](#features)
3. [Class Descriptions](#class-descriptions)
4. [GUI Design](#gui-design)
5. [Usage Instructions](#usage-instructions)
6. [Code Examples](#code-examples)
7. [Error Handling](#error-handling)
8. [Future Enhancements](#future-enhancements)

**Introduction**

The Simple Banking System project is a Java-based application with a graphical user interface (GUI) designed to simulate essential banking operations. It provides users with functionalities such as account creation, deposit, withdrawal, and PIN management. This project was created to demonstrate fundamental Java programming skills, including OOP (Object-Oriented Programming) concepts, ArrayLists for data storage, and event-driven programming with a little GUI functionality.

**Features**

* **Account Creation**: Allows users to create a new account with an initial balance and a PIN.
* **Deposit**: Enables users to deposit money into an existing account.
* **Withdrawal**: Allows users to withdraw money from an account.
* **PIN Management**: Provides a function to change the account's PIN.
* **Data Storage**: Uses an ArrayList to store accounts within the BankingSystem class.

**Class Descriptions**

**1. Account.java**

* **Fields**:
  + name: Stores the account holder's name.
  + accountNumber: Stores a unique account number.
  + accountBalance: Stores the account’s balance.
  + pin: Stores the PIN for account security.
* **Methods**:
  + getAccountBalance(): Returns the account balance.
  + getAccountNumber(): Returns the account number.
  + withdraw(double amount): Deducts a specified amount from the balance.
  + deposit(double amount): Adds a specified amount to the balance.
  + changePin(String newPin): Updates the PIN.
  + checkPin(String inputPin): Verifies the input PIN.

**2. BankingSystem.java**

* **Fields**:
  + accounts: An ArrayList of Account objects.
* **Methods**:
  + createAccount(String name, double initialBalance, String pin): Creates a new account.
  + deposit(String accountNumber, double amount): Deposits a specified amount.
  + withdraw(String accountNumber, String pin, double amount): Withdraws a specified amount.
  + changePin(String accountNumber, String oldPin, String newPin): Changes the account's PIN.
  + findAccount(String accountNumber): Finds and returns an account by its account number.

**3. BankingAppGUI.java**

* **Purpose**: Provides the GUI interface for interacting with the BankingSystem.
* **Components**:
  + **Text Fields**: For account details (name, balance, PIN, etc.).
  + **Buttons**: For operations (create account, deposit, withdraw, change PIN).
* **Layout**: Vertically arranged fields and buttons, with a GUI size of 320x240 pixels.

**GUI Design**

The BankingAppGUI class is designed for ease of use. The GUI flows vertically, with each field labeled and aligned for user readability. Users enter required information in text fields, then click a button for the respective operation (e.g., Create Account, Deposit, Withdraw, Change PIN).

* **Frame Size**: 320x240 pixels.
* **Error Handling**: Displays error messages via JOptionPane dialogs for invalid input or unsuccessful operations.

**Usage Instructions**

1. **Running the Project**:
   * Compile Account.java, BankingSystem.java, and BankingAppGUI.java.
   * Run BankingAppGUI to launch the application.
2. **Creating an Account**:
   * Enter name, initial balance, and PIN, then click Create Account.
3. **Making a Deposit**:
   * Enter the account number and deposit amount, then click Deposit.
4. **Withdrawing Funds**:
   * Enter the account number, PIN, and amount to withdraw, then click Withdraw.
5. **Changing PIN**:
   * Enter the account number, old PIN, and new PIN, then click Change PIN.

**Error Handling**

The application provides basic error handling to ensure data integrity:

* **Invalid Balance**: Ensures entered balance is a valid double.
* **Empty PIN**: Checks for non-empty PIN input.
* **Invalid Account Number or PIN**: Prompts user if account number or PIN is incorrect.
* **Insufficient Balance**: Displays a message if the user tries to withdraw more than the account balance.