Banking System Project Documentation

- **Project Name:** Unity Financial Bank System
- **Language:** Java

Description:

The Unity Financial Bank System is a Java-based application that provides basic banking functionalities for up to five clients. The system allows users to register as clients, log in, check account balances, deposit funds, withdraw funds, view daily transaction reports, and close their accounts. The project utilizes standard Java libraries, such as Scanner for input handling and DecimalFormat for formatting currency outputs.

Features:

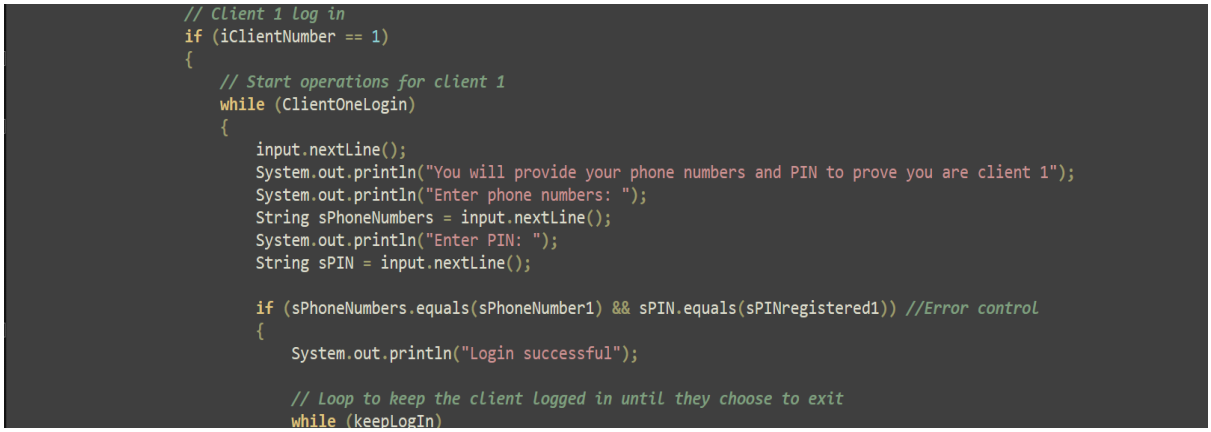
Account Registration

- Clients can register by providing their first name, last name, phone number, ID number, and setting up a PIN.
- Input validation is enforced to ensure phone numbers are 10 digits and ID numbers are 13 digits.

```
input.nextLine(); // Removing keyboard buffer
System.out.println("Enter your first name: ");
sFirstName1 = input.nextLine();
System.out.println("Enter your last name: ");
sLastName1 = input.nextLine();
System.out.println("Enter your phone number: ");
sPhoneNumber1 = input.nextLine();

while (sPhoneNumber1.length() != 10)
{
    System.out.println("Phone number must be 10 digits");
    System.out.println("Please re-enter your phone number");
    sPhoneNumber1 = input.nextLine();
}

System.out.println("Enter your ID number: ");
sIDnumber1 = input.nextLine();

while (sIDnumber1.length() != 13)
{
    System.out.println("ID number must be 13 digits");
    System.out.println("Please re-enter your ID number");
    sIDnumber1 = input.nextLine();
}

System.out.println("Create PIN: ");
sPINcreate = input.nextLine();
System.out.println("Confirm your PIN: ");
sPINregistered1 = input.nextLine();

// PIN validation
while (!sPINcreate.equals(sPINregistered1))
{
```

Login

- Clients log in using their phone number and PIN. The system verifies credentials before granting access.
- Keeps users logged in until they decide to log out or close their account.

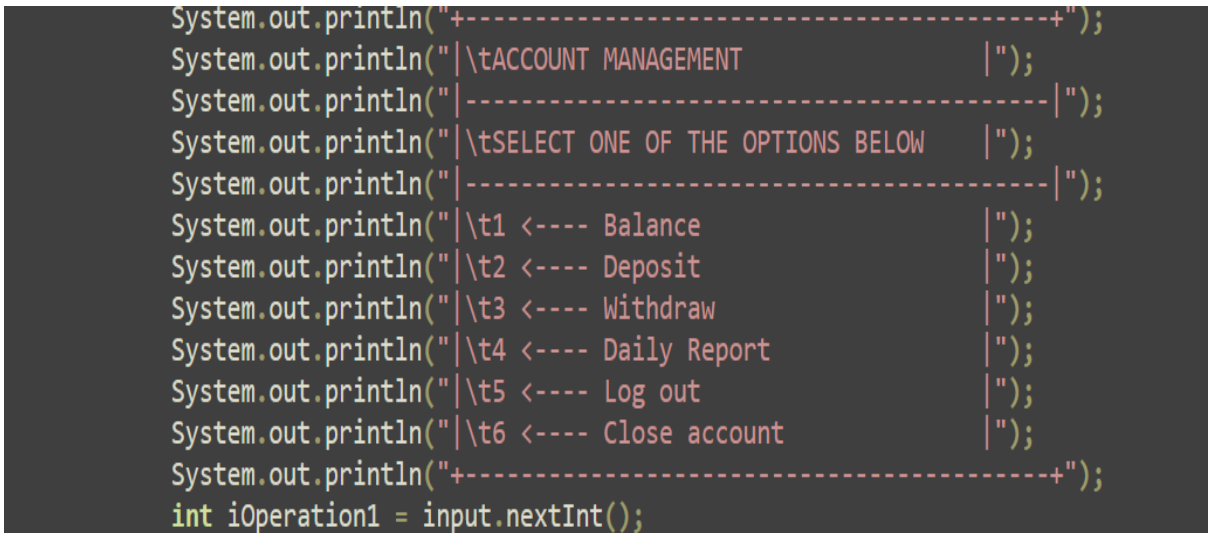


```
// Client 1 log in
if (iClientNumber == 1)
{
    // Start operations for client 1
    while (ClientOneLogin)
    {
        input.nextLine();
        System.out.println("You will provide your phone numbers and PIN to prove you are client 1");
        System.out.println("Enter phone numbers: ");
        String sPhoneNumbers = input.nextLine();
        System.out.println("Enter PIN: ");
        String sPIN = input.nextLine();

        if (sPhoneNumbers.equals(sPhoneNumber1) && sPIN.equals(sPINregistered1)) //Error control
        {
            System.out.println("Login successful");

            // Loop to keep the client logged in until they choose to exit
            while (keepLogIn)
```

Account Management

Clients can manage their accounts with the following options:

- Balance Inquiry: View the current balance.
- Deposit Funds: Add funds to the account.
- Withdraw Funds: Withdraw funds from the account with balance checks and error handling.
- Daily Reports: View the number of deposits, withdrawals, and available balance.
- Log Out: Securely log out of the system.
- Close Account: Option to close the account, which deletes all associated data.



```
System.out.println("+----------------------------------------+");
System.out.println("|\tACCOUNT MANAGEMENT                     |");
System.out.println("|----------------------------------------|");
System.out.println("|\tSELECT ONE OF THE OPTIONS BELOW        |");
System.out.println("|----------------------------------------|");
System.out.println("|\t1 <---- Balance                        |");
System.out.println("|\t2 <---- Deposit                        |");
System.out.println("|\t3 <---- Withdraw                       |");
System.out.println("|\t4 <---- Daily Report                   |");
System.out.println("|\t5 <---- Log out                        |");
System.out.println("|\t6 <---- Close account                  |");
System.out.println("+----------------------------------------+");
int iOperation1 = input.nextInt();
```

Error Handling

The system handles incorrect input scenarios, such as invalid login attempts, negative withdrawal amounts, and attempts to withdraw more than the available balance.
Prevents users from registering multiple clients under the same client number.

```
                    System.out.println("Invalid option");
                    System.out.println("Please try again");
                break;
            }

            // Exit the loop if the user chooses to log out
            if (iOperation2 == 5)
            {
                break;
            }
        }
        break; // Exit login loop after the session ends
    } else {
        System.out.println("|---------------------------------------------|");
        System.out.println("|\tINVALID LOGIN                               |");
        System.out.println("|---------------------------------------------|");
        ClientTwoLogin = false;
        break;
    }
```

```
        while ( rBalance1 == 0 )
        {
            System.out.println("You have no funds in your account. Withdrawal not possible.");
            System.out.println("PLEASE enter 0 to continue");
            rWithdrawAmount1 = input.nextDouble();
            break;
        }

        while (rWithdrawAmount1 < 0)
        {
            System.out.println("Invalid withdrawal amount! Please enter a positive value.");
            rWithdrawAmount1 = input.nextDouble();
        }

        while (rWithdrawAmount1 > rBalance1)
        {
            System.out.println("Insufficient balance! Unable to withdraw the requested amount.");
            System.out.println("Try again");
            rWithdrawAmount1 = input.nextDouble();
        }
```

```
{
    if (iClientNumberFill1 == 1)
    {
        System.out.println("Another client already registered under this client number");
        System.out.println("Please choose another client number between client number 2, 3, 4 or 5");
        iClientNumber = input.nextInt();
    }
```

System Components:

Classes Used:

Scanner from java.util: For user input handling.
DecimalFormat from java.text: For formatting currency.

```java
import java.util.Scanner; // Scanner class is imported from java.util package
import java.text.DecimalFormat; // DecimalFormat class is imported from java.text package
```

```java
// Instantiation of object
Scanner input = new Scanner(System.in);
DecimalFormat formatter = new DecimalFormat("R00.00");
```

Variables:

Strings for client names, phone numbers, ID numbers, and PINs.
Integers for tracking the number of deposits and withdrawals.
Double variables for managing account balances.
Boolean flags to control loops and user sessions.

```java
// Declare variables for login process
String sPINregistered1 = "", sPINregistered2 = "", sPINregistered3 = "", sPINregistered4 = "", sPINregistered5 = "";
String sPhoneNumbersRegistered1 = "", sPhoneNumbersRegistered2 = "", sPhoneNumbersRegistered3 = "", sPhoneNumbersRegistered4 = "", sPhoneNumbersRegistered5 = "";
int iClientNumber;

//Declare variables for daily reports
int iNumDeposit1 = 0, iNumDeposit2 = 0, iNumDeposit3 = 0, iNumDeposit4 = 0, iNumDeposit5 = 0;
int iNumWithdraw1 = 0, iNumWithdraw2 = 0, iNumWithdraw3 = 0,  iNumWithdraw4 = 0,  iNumWithdraw5 = 0;

// Declaring variables for client names registration
String sFirstName1 = "", sLastName1 = "";
String sFirstName2 = "", sLastName2 = "";
String sFirstName3 = "", sLastName3 = "";
String sFirstName4 = "", sLastName4 = "";
String sFirstName5 = "", sLastName5 = "";

//Declaring variables for client numbers
int iClientNumberFill1 = 0, iClientNumberFill2 = 0, iClientNumberFill3 = 0, iClientNumberFill4 = 0, iClientNumberFill5 = 0;

// Declaring variables for client Phone number registration
String sPhoneNumber1 = "", sPhoneNumber2 = "", sPhoneNumber3 = "", sPhoneNumber4 = "", sPhoneNumber5 = "";

// Declaring variables for IDnumberRegistration
String sIDnumber1 = "", sIDnumber2 = "", sIDnumber3 = "", sIDnumber4 = "", sIDnumber5 = "";

// Declaring variables for PIN registration
String sPINcreate;

// Client Balances
double rBalance1 = 0.0, rBalance2 = 0.0, rBalance3 = 0.0, rBalance4 = 0.0, rBalance5 = 0.0;

//This boolean variable keeps the user logged in until the user logs out and closes an account
boolean keepLogIn = true;

//The following declared boolean variables exit a loop for client login when the user enters an incorrect login credentials and when you do not have an account
boolean ClientOneLogin = true, ClientTwoLogin = true, ClientThreeLogin = true, ClientFourLogin = true, ClientFiveLogin = true;
```

Control Structures:

Switch Statements: Used for user operations and account management.
While Loops: Manage user sessions and enforce continuous operation until termination.
Conditional Statements: Validate inputs, process transactions, and handle account registration limits.

```
// Client 1 transactions
switch (iOperation1)
{
    case 1:
        System.out.println("Balance: " + formatter.format(rBalance1));
    break;

    case 2:
        System.out.println("Enter the deposit Amount: ");
        double rDeposit1 = input.nextDouble();
        rBalance1 = rBalance1 + rDeposit1;
        System.out.println("Amount deposited : " + formatter.format(rDeposit1));
        System.out.println("Available Balance: " + formatter.format(rBalance1));
        iNumDeposit1++;
    break;

    case 3:
```

```
// Start operations for client 1
while (ClientOneLogin)
{
    input.nextLine();
    System.out.println("You will provide your phone numbers and PIN to prove you are client 1");
    System.out.println("Enter phone numbers: ");
    String sPhoneNumbers = input.nextLine();
    System.out.println("Enter PIN: ");
    String sPIN = input.nextLine();
```

```
case 6:
  System.out.println("DO you really want to close your account?");
  System.out.println("Enter <YES> to confirm");
  System.out.println("Enter <NO> to return to Account Management");
  char cAnswer1 = Character.toUpperCase(input.next().charAt(0));

    if ( cAnswer1 == 'Y' )

    {
        System.out.println("Your account has been closed successfully");
        sFirstName1 = " ";
        sLastName1 = " ";
        rBalance1 = 0;
        rWithdrawAmount1 = 0;
        rDeposit1 = 0;
        sIDnumber1 = " ";
        iClientNumberFill1 --;
        sPhoneNumber1 = " ";
        sPINregistered1 = " ";
        iNumDeposit1 = 0;
        iNumWithdraw1 = 0;
        keepLogIn = false;
        break;

    }
```

```
    default:
        System.out.println("Invalid option");
        System.out.println("Please try again");
        break;
```

```
input.nextLine(); // Removing keyboard buffer
System.out.println("Enter your first name: ");
sFirstName1 = input.nextLine();
System.out.println("Enter your last name: ");
sLastName1 = input.nextLine();
System.out.println("Enter your phone number: ");
sPhoneNumber1 = input.nextLine();

while (sPhoneNumber1.length() != 10)
{
    System.out.println("Phone number must be 10 digits");
    System.out.println("Please re-enter your phone number");
    sPhoneNumber1 = input.nextLine();
}

System.out.println("Enter your ID number: ");
sIDnumber1 = input.nextLine();

while (sIDnumber1.length() != 13)
{
    System.out.println("ID number must be 13 digits");
    System.out.println("Please re-enter your ID number");
    sIDnumber1 = input.nextLine();
}

System.out.println("Create PIN: ");
sPINcreate = input.nextLine();
System.out.println("Confirm your PIN: ");
sPINregistered1 = input.nextLine();

// PIN validation
while (!sPINcreate.equals(sPINregistered1))
{
    System.out.println("PINs do not match. Please try again.")
    System.out.println("Create PIN: ");
    sPINcreate = input.nextLine();
    System.out.println("Confirm your PIN: ");
    sPINregistered1 = input.nextLine();
}

System.out.println("Client registration successful");
System.out.println("Please log in on your client number");
sPINcreate = "";
iClientNumberFill1++;
```

Future Improvements:

- **Increase Client Limit**: Extend the system to accommodate more clients.
- **Enhance Security**: Implement encryption for sensitive data like PINs.
- **Graphical User Interface (GUI)**: Create a user-friendly interface to enhance the user experience.
- **Transaction History**: Add a feature to view detailed transaction history over multiple sessions.

Author: Matsi Lethabo Frans
Date : 22 October 2024
Module: Introduction to Programming