## Source Codes

import java.util.\*;

class UserAuthentication {

    private static Map<String, String> registeredUsers = new HashMap<>();

    private static String loggedInUser = null;

    public static boolean register(String username, String password) {

        if (!registeredUsers.containsKey(username)) {

            registeredUsers.put(username, password);

            System.out.println("Registration successful!");

            return true;

        } else {

            System.out.println("Username already exists. Please choose a different username.");

            return false;

        }

    }

    public static boolean login(String username, String password) {

        if (registeredUsers.containsKey(username) && registeredUsers.get(username).equals(password)) {

            loggedInUser = username;

            System.out.println("Login successful!");

            return true;

        } else {

            System.out.println("Invalid username or password. Please try again.");

            return false;

        }

    }

    public static void logout() {

        loggedInUser = null;

        System.out.println("Logged out successfully!");

    }

    public static boolean isLoggedIn() {

        return loggedInUser != null;

    }

}

class InsufficientFundsException extends Exception {

    public InsufficientFundsException(String message) {

        super(message);

    }

}

class DatabaseConnector {

    public static void connect() {

        System.out.println("Connecting to the database...");

    }

    public static void disconnect() {

        System.out.println("Disconnecting from the database...");

    }

    public static void executeQuery(String query) {

        System.out.println("Executing query: " + query);

    }

}

class BankUtilities {

    public static double transactionFee = 0.5;

    public static double interestRate = 0.02;

}

// BankingOperations

interface BankingOperations {

    void checkBalance();

    void deposit(double amount);

    void withdraw(double amount) throws InsufficientFundsException;

    void transferFunds(Account destinationAccount, double amount) throws InsufficientFundsException;

}

// Account class

abstract class Account implements BankingOperations {

    protected String firstName;

    protected String lastName;

    protected int accountID;

    protected double balance;

    protected int pin;

    public Account(String lastName, String firstName, int accountID, double balance, int pin) {

        this.lastName = lastName;

        this.firstName = firstName;

        this.accountID = accountID;

        this.balance = balance;

        this.pin = pin;

    }

    // Implementing BankingOperations

    public void checkBalance() {

        System.out.println("Current balance: R" + balance);

    }

    public void deposit(double amount) {

        balance += amount;

        System.out.println("Deposit successful. New balance: R" + balance);

    }

    public abstract void withdraw(double amount) throws InsufficientFundsException;

    public abstract void transferFunds(Account destinationAccount, double amount) throws InsufficientFundsException;

    public String getAccountDetails() {

        return "Name: " + lastName + " " + firstName + ", Account Number: " + accountID + ", Balance: R" + balance;

    }

    public int getAccountID() {

        throw new UnsupportedOperationException("Unimplemented method 'getAccountID'");

    }

    public int getPin() {

        throw new UnsupportedOperationException("Unimplemented method 'getPin'");

    }

}

// SavingsAccount class

class SavingsAccount extends Account {

    public SavingsAccount(String lastName, String firstName, int accountID, double balance, int pin) {

        super(lastName, firstName, accountID, balance, pin);

    }

    public void withdraw(double amount) throws InsufficientFundsException {

        if (balance >= amount) {

            balance -= amount;

            System.out.println("Withdrawal successful. New balance: R" + balance);

        } else {

            throw new InsufficientFundsException("Insufficient funds.");

        }

    }

    public void transferFunds(Account destinationAccount, double amount) throws InsufficientFundsException {

        if (balance >= amount) {

            balance -= amount;

            destinationAccount.deposit(amount);

            System.out.println("Transfer successful. New balance: R" + balance);

        } else {

            throw new InsufficientFundsException("Insufficient funds for transfer.");

        }

    }

}

// CheckingAccount class

class CheckingAccount extends Account {

    public CheckingAccount(String lastName, String firstName, int accountID, double balance, int pin) {

        super(lastName, firstName, accountID, balance, pin);

    }

    public void withdraw(double amount) throws InsufficientFundsException {

        if (balance - amount >= -1000) {

            balance -= amount;

            System.out.println("Withdrawal successful. New balance: R" + balance);

        } else {

            throw new InsufficientFundsException("Insufficient funds.");

        }

    }

    public void transferFunds(Account destinationAccount, double amount) throws InsufficientFundsException {

        if (balance - amount >= -1000) {

            balance -= amount;

            destinationAccount.deposit(amount);

            System.out.println("Transfer successful. New balance: R" + balance);

        } else {

            throw new InsufficientFundsException("Insufficient funds for transfer.");

        }

    }

}

public class MzanziBank {

    private static Scanner scanner = new Scanner(System.in);

    private static ArrayList<Account> accounts = new ArrayList<>();

    private static int lastAccountId = 10000000;

    public static void main(String[] args) {

        int choice;

        do {

            System.out.println("\nWELCOME TO MZANZI BANK SYSTEM");

            System.out.println("1. Register");

            System.out.println("2. Login");

            System.out.println("3. Exit");

            System.out.print("Enter your choice: ");

            choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    register();

                    break;

                case 2:

                    login();

                    break;

                case 3:

                    System.out.println("Thank you for using Mzanzi Bank system. Goodbye!");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        } while (choice != 3);

    }

    private static void register() {

        System.out.println("\nREGISTRATION");

        System.out.print("Enter your last name: ");

        String lastName = scanner.next();

        System.out.print("Enter your first name: ");

        String firstName = scanner.next();

        System.out.print("Enter your initial balance: R");

        double balance = scanner.nextDouble();

        System.out.print("Enter a PIN: ");

        int pin = scanner.nextInt();

        System.out.print("Confirm PIN: ");

        int confirmPin = scanner.nextInt();

        if (pin == confirmPin) {

            int accountID = lastAccountId++;

            Account account = new SavingsAccount(lastName, firstName, accountID, balance, pin);

            accounts.add(account);

            System.out.println("Account created successfully!\nYour Account ID is: " + accountID);

        } else {

            System.out.println("PINs do not match. Registration failed.");

        }

    }

    private static void login() {

        if (accounts.isEmpty()) {

            System.out.println("No accounts registered yet. Please register first.");

            return;

        }

        System.out.println("\nLOGIN");

        System.out.print("Enter your Account ID: ");

        int accountID = scanner.nextInt();

        System.out.print("Enter your PIN: ");

        int pin = scanner.nextInt();

        boolean found = false;

        for (Account account : accounts) {

            if (account.getAccountID() == accountID && account.getPin() == pin) {

                found = true;

                displayAccountMenu(account);

                break;

            }

        }

        if (!found) {

            System.out.println("Invalid Account ID or PIN. Login failed.");

        }

    }

    private static void displayAccountMenu(Account account) {

        int choice;

        do {

            System.out.println("\nMain MENU");

            System.out.println("1. Check Balance");

            System.out.println("2. Deposit");

            System.out.println("3. Withdraw");

            System.out.println("4. Transfer Funds");

            System.out.println("5. Create a New Account");

            System.out.println("6. Logout");

            System.out.print("Enter your choice: ");

            choice = scanner.nextInt();

            switch (choice) {

                case 1:

                    account.checkBalance();

                    break;

                case 2:

                    System.out.print("Enter amount to deposit: R");

                    double depositAmount = scanner.nextDouble();

                    account.deposit(depositAmount);

                    break;

                case 3:

                    System.out.print("Enter amount to withdraw: R");

                    double withdrawAmount = scanner.nextDouble();

                    try {

                        account.withdraw(withdrawAmount);

                    } catch (InsufficientFundsException e) {

                        System.out.println(e.getMessage());

                    }

                    break;

                case 4:

                    System.out.print("Enter recipient's Account ID: ");

                    int recipientID = scanner.nextInt();

                    System.out.print("Enter amount to transfer: R");

                    double transferAmount = scanner.nextDouble();

                    Account recipientAccount = findAccountByID(recipientID);

                    if (recipientAccount != null) {

                        try {

                            account.transferFunds(recipientAccount, transferAmount);

                        } catch (InsufficientFundsException e) {

                            System.out.println(e.getMessage());

                        }

                    } else {

                        System.out.println("Recipient account not found.");

                    }

                    break;

                case 5:

                    register(); // Call the register method to create a new account

                    break;

                case 6:

                    System.out.println("Logging out...");

                    break;

                default:

                    System.out.println("Invalid choice. Please try again.");

            }

        } while (choice != 6);

    }

    private static Account findAccountByID(int accountID) {

        for (Account account : accounts) {

            if (account.getAccountID() == accountID) {

                return account;

            }

        }

        return null;

    }

}

## Program Screenshot

## 1.Welcome menu

