**INTERNSHIP PROJECT REPORT**

**DONE BY**

NAME : AJAY.S

COMPANY NAME : DATASOFTIXS

INTERNSHIP ID: JP100021

DOMAIN : JAVA PROGRAMMING

COLLEGE : PANIMALAR INSTITUTE OF

TECHONOLOGY

**Project 1: Basic Banking Application**

1. **Objective:**

The project is a console-based banking application designed to simulate basic banking operations such as checking balance, depositing money, withdrawing money, and transferring funds between accounts. It demonstrates the usage of core object-oriented programming (OOP) principles, input validation, and user interaction.

**2. Key Components:**

A. Account Class

* Purpose: Represents a bank account.
* Attributes:
  + account Number: Unique identifier for the account.
  + accountholder Name: Name of the account holder.
  + balance: Current balance in the account.
* Methods:
  + deposit: Adds money to the balance.
  + withdraw: Deducts money from the balance after validation.
  + transferTo: Transfers funds between two accounts if conditions are met.

B. BasicBankingApplication Class

* Purpose: Acts as the main driver for the application, providing the user interface and handling operations.
* Attributes:
  + accounts: A HashMap storing account details, using the account number as the key.
  + scanner: For reading user input.
* Functionalities:
  + Menu-driven interface to interact with the user.
  + Methods for checking balance, depositing money, withdrawing money, and transferring funds.

**3. Features:**

A. Check Balance

* Prompts the user to enter an account number.
* Retrieves the corresponding account from the HashMap and displays:
  + Account holder’s name.
  + Current balance.

B. Deposit Money

* Asks the user to enter the account number and deposit amount.
* Validates that the deposit amount is positive.
* Updates the account balance and displays the new balance.

C. Withdraw Money

* Prompts the user to enter the account number and withdrawal amount.
* Ensures the withdrawal amount is valid and does not exceed the current balance.
* Deducts the amount from the account balance and displays the updated balance.

D. Transfer Money (Optional Feature)

* Prompts the user to enter:
  + Sender’s account number.
  + Recipient’s account number.
  + Transfer amount.
* Validates both accounts exist and that the sender has sufficient balance.
* Transfers funds by deducting the amount from the sender and adding it to the recipient.

**4. Core Concepts:**

A. Object-Oriented Programming (OOP)

* Encapsulation: The Account class encapsulates account details and operations, ensuring that only valid changes are made to an account's state.
* Reusability: The Account class methods (deposit, withdraw, transferTo) are reusable across multiple accounts.

B. Input Validation

* Prevents invalid operations like:
  + Depositing or withdrawing negative amounts.
  + Withdrawing or transferring more money than available.

C. Data Structures

* HashMap: Used to store and retrieve accounts efficiently using account numbers as keys.

D. User Interaction

* Console-based menu system for simple and intuitive interaction.

**5. Example Usage Flow:**

1. A user selects an operation from the menu.
2. The program prompts the user for required inputs (e.g., account number, amount).
3. It performs the operation if inputs are valid and updates account details accordingly.
4. The program loops back to the main menu until the user decides to exit.

**6. Benefits:**

* Educational Value:
  + Teaches core Java concepts like classes, objects, data structures, and input handling.
* Practical Application:
  + Simulates real-world banking functionalities.
* Extensibility:
  + Additional features like account creation, account deletion, or transaction history can be added.

**7. Possible Extensions:**

* Account Creation and Deletion:
  + Allow users to create new accounts or delete existing ones.
* Authentication:
  + Add a password or PIN for account security.
* Transaction History:
  + Maintain a log of all deposits, withdrawals, and transfers.
* Graphical Interface:
  + Replace the console-based UI with a graphical UI using JavaFX or Swing.

**Code:**

import java.util.HashMap;

import java.util.Scanner;

class Account {

private String accountNumber;

private String accountHolderName;

private double balance;

public Account(String accountNumber, String accountHolderName, double initialBalance) {

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.balance = initialBalance;

}

public String getAccountNumber() {

return accountNumber;

}

public String getAccountHolderName() {

return accountHolderName;

}

public double getBalance() {

return balance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

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

} else {

System.out.println("Invalid deposit amount.");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

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

} else {

System.out.println("Invalid withdrawal amount or insufficient balance.");

}

}

public boolean transferTo(Account recipient, double amount) {

if (amount > 0 && amount <= balance) {

this.balance -= amount;

recipient.balance += amount;

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

return true;

} else {

System.out.println("Invalid transfer amount or insufficient balance.");

return false;

}

}

}

public class BasicBankingApplication {

private static HashMap<String, Account> accounts = new HashMap<>();

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

public static void main(String[] args) {

System.out.println("Welcome to the Basic Banking Application!");

// Create some sample accounts

accounts.put("12345", new Account("12345", "John Doe", 1000));

accounts.put("67890", new Account("67890", "Jane Smith", 1500));

boolean running = true;

while (running) {

System.out.println("\nMain Menu:");

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

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

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

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

System.out.println("5. Exit");

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

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

checkBalance();

break;

case 2:

depositMoney();

break;

case 3:

withdrawMoney();

break;

case 4:

transferMoney();

break;

case 5:

running = false;

System.out.println("Thank you for using the Basic Banking Application. Goodbye!");

break;

default:

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

}

}

scanner.close();

}

private static void checkBalance() {

Account account = getAccountByNumber();

if (account != null) {

System.out.println("Account Holder: " + account.getAccountHolderName());

System.out.println("Current Balance: " + account.getBalance());

}

}

private static void depositMoney() {

Account account = getAccountByNumber();

if (account != null) {

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

double amount = scanner.nextDouble();

account.deposit(amount);

}

}

private static void withdrawMoney() {

Account account = getAccountByNumber();

if (account != null) {

System.out.print("Enter withdrawal amount: ");

double amount = scanner.nextDouble();

account.withdraw(amount);

}

}

private static void transferMoney() {

System.out.print("Enter your account number: ");

String senderAccountNumber = scanner.nextLine();

Account senderAccount = accounts.get(senderAccountNumber);

if (senderAccount == null) {

System.out.println("Account not found.");

return;

}

System.out.print("Enter recipient account number: ");

String recipientAccountNumber = scanner.nextLine();

Account recipientAccount = accounts.get(recipientAccountNumber);

if (recipientAccount == null) {

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

return;

}

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

double amount = scanner.nextDouble();

senderAccount.transferTo(recipientAccount, amount);

}

private static Account getAccountByNumber() {

System.out.print("Enter your account number: ");

String accountNumber = scanner.nextLine();

Account account = accounts.get(accountNumber);

if (account == null) {

System.out.println("Account not found.");

}

return account;

}

}

**Output:**

Welcome to the Basic Banking Application!

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 1

Enter your account number: 12345

Account Holder: John Doe

Current Balance: 1000.0

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 2

Enter your account number: 12345

Enter deposit amount: 500

Deposit successful. New balance: 1500.0

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 3

Enter your account number: 12345

Enter withdrawal amount: 200

Withdrawal successful. New balance: 1300.0

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 4

Enter your account number: 12345

Enter recipient account number: 67890

Enter transfer amount: 300

Transfer successful. New balance: 1000.0

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 1

Enter your account number: 67890

Account Holder: Jane Smith

Current Balance: 1800.0

Main Menu:

1. Check Balance

2. Deposit Money

3. Withdraw Money

4. Transfer Money

5. Exit

Enter your choice: 5

Thank you for using the Basic Banking Application. Goodbye!