Department of Statistics & Computer Science

Faculty of Science, University of Kelaniya.

PROJECT ON
"Bank Management System"

Project Report

COSC 12043 project

Group Members

Register No.	Name	Task
PS/2019/127	Y.M.S.N.R YAPA	BankingApplication ,Bank, Account
PS/2019/164	J.A.K.N JAYAKODY	Loan
PS/2019/170	A.A.S.S ADIKARI	deposit
PS/2019/012	P.D.I.G RATHNASIRI	withdraw
PS/2019/244	J.M.N.C JAYAMANNE	getprev_transaction

Submitted to:

Dr. B.M. Thosini Kumarika
Name of the lecturer

Abstract

- The Bank Account Management System is an application for maintaining a person's account in a bank. In this project, we tried to show the working of a banking account system and cover the basic functionality of a Bank Account Management System. To develop a project for solving financial applications of a customer in a banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks. Also to enable the user’s workspace to have additional functionalities which are not provided under a conventional banking project.
- The Bank Account Management System undertaken as a project is based on relevant technologies. The main aim of this project is to develop software for Bank Account Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manual systems, which are overcome by this software.
- Thus, this project will save transaction time and therefore increase the efficiency of the system.

Table of Contents

1.Introduction

- The "Bank Account Administration System" venture could be a demonstrate Web Keeping money Site. This site enables the clients to perform the fundamental managing an account exchanges by sitting at their office or at homes through PC or tablet. The system gives the get to to the client to make an account, deposit/withdraw money from his account, too to see reports of all accounts present. The clients can get to the banks site for seeing their Account points of interest and perform the exchanges on account as per their prerequisites. With Web Managing an account, the brick and mortar structure of the conventional managing an account gets changed over into a press and portal model, subsequently giving a concept of virtual managing an account a genuine shape. Hence today's keeping money is no longer limited to branches. E-banking encourages managing an account exchanges by clients round the clock universally.

  The essential point of this "Bank Account Administration System" is to supply an improved design methodology, which conceives long haul extension, and alteration, which is necessary for a center segment like managing an account. This requires the plan to be expandable and modifiable and so a measured approach is utilized in creating the application software. Anybody who is an Account holder in this bank can gotten to be a part of Bank Account Management Framework. He must fill a frame with his individual points of interest and Account Number.

  Bank is the put where clients feel the sense of security for their property. Within the bank, customers store and pull back their cash. Transaction of money too could be a portion where customer takes shield of the bank. Presently to keep the conviction and believe of clients, there's the positive require for administration of the bank, which can handle all this with consolation and ease. Smooth and proficient administration influences the fulfillment of the clients and staff members, by implication. And of course, it energizes administration committee in taking some needed choice for future improvement of the bank.

1.1Main Purpose

- The conventional way of keeping up the subtle elements of a client in a bank was to enter the points of interest and record them. Each time the client ought to perform a few exchanges he must go to the bank and perform the fundamental activities, which may not be so doable all the time. It may be a hard-hitting assignment for the clients and the financiers as well. The extend gives a real-life understanding of Online Managing an account Framework and activities performed by different parts within the supply chain. Here, we offer mechanization for managing an account framework through the Web. Online Banking System extends captures exercises performed by distinctive parts in genuine life keeping money which gives upgraded strategies for keeping up the desired data up-to-date, which comes about in proficiency. The extend gives a genuine life understanding of Online Managing an account Framework and exercises performed by different parts within the supply chain.

1.2Goals and Objectives

1. Main Goals:

- our main objective is the customer's satisfaction considering today's faster in the world.

2. Customer Satisfaction:

- Client can do his operations comfortably without any risk or loss of his privacy.
- Our software will perform and fulfill all the tasks that any customer would desire.

3. Saving Customer Time:

- Client doesn't need to go to the bank to do the small operations.

4. Protecting The Customer:

- It helps the customer to be satisfied and comfortable in his choices, this protection contains the customer's account, money, and his privacy.

5. Transferring Money:

- Help client transfer money to/or another bank or country

2. Classes, their attributes and methods in each

2.1functionalities of the system

class (Bank)

Attributes/Variables of the class (Bank)

Private Attributes/Variables:

We can assign private attributes/ values to this class, but suppose that currently we are not willing to make the attributes private.

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +account_no : int
- +moNo :int
- +S :boolean
- +name : String
- +birth : String
- +email :String
- +acctype :String
- +password :String
- +address : int
- +name : String
- +salary :int

Functions of the class (bank)

There are following functions in the mentioned class diagram.

- +openAccount( )
- +showAccount( )

class (Account)

Attributes/Variables of the class (Account)

Private Attributes/Variables:

We can assign private attributes/ values to this class, but suppose that currently, we are not willing to make the attributes private.

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +account_no : int
- +moNo :int
- +S :boolean
- +name : String
- +birth : String
- +email :String
- +acctype :String
- +password :String
- +address : int
- +name : String
- +salary :int

Class (Loan)

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +amount : int
- +time :int
- +t2 :double
- +rate :double
- +total : double
- +interest : double
- +t : double

Functions of the class (Loan)

There are following functions in the mentioned class diagram.

- +loan1( )

Class (deposit)

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +balance :int
- +dep : int
- +prev_transaction : int
- +amount:int

Functions of the class (deposit)

There are following functions in the mentioned class diagram.

- +depositFunc( )

Class (withdraw)

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +withdrawn : int
- +balance :int
- +prev_transaction : int

Functions of the class (withdraw)

There are following functions in the mentioned class diagram.

- +withdrawFunc( )

Class (getprev_transaction)

Public Attributes/Variables:

There are following public attributes in the mentioned class diagram.

- +prev_transaction : int

Functions of the class (getprev_transaction)

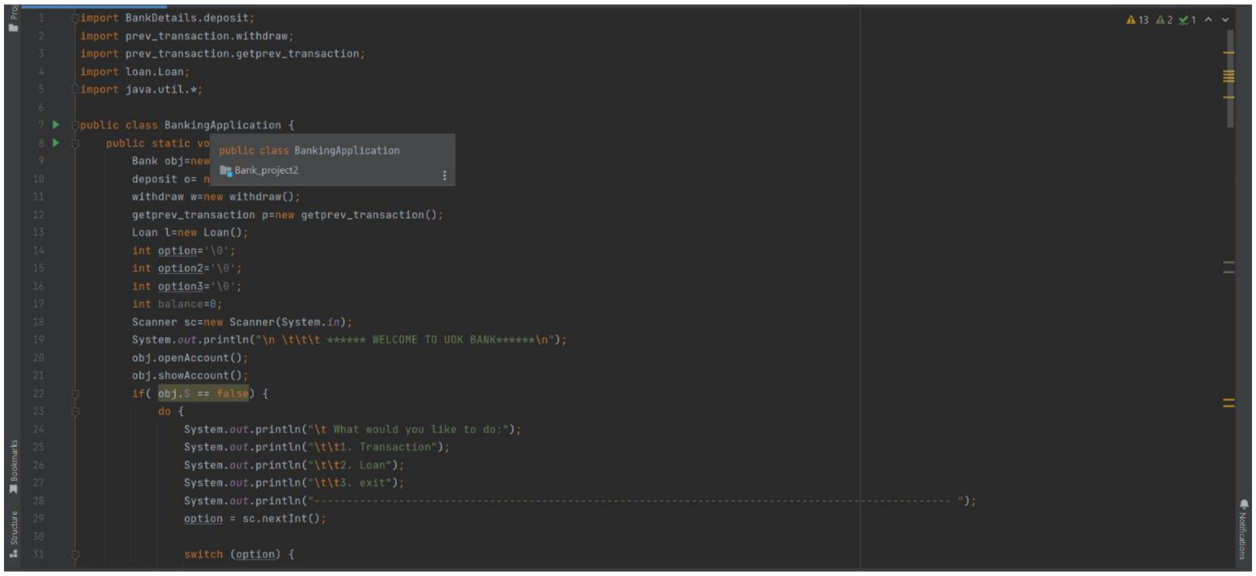
There are following functions in the mentioned class diagram.

- +prevFunc( )

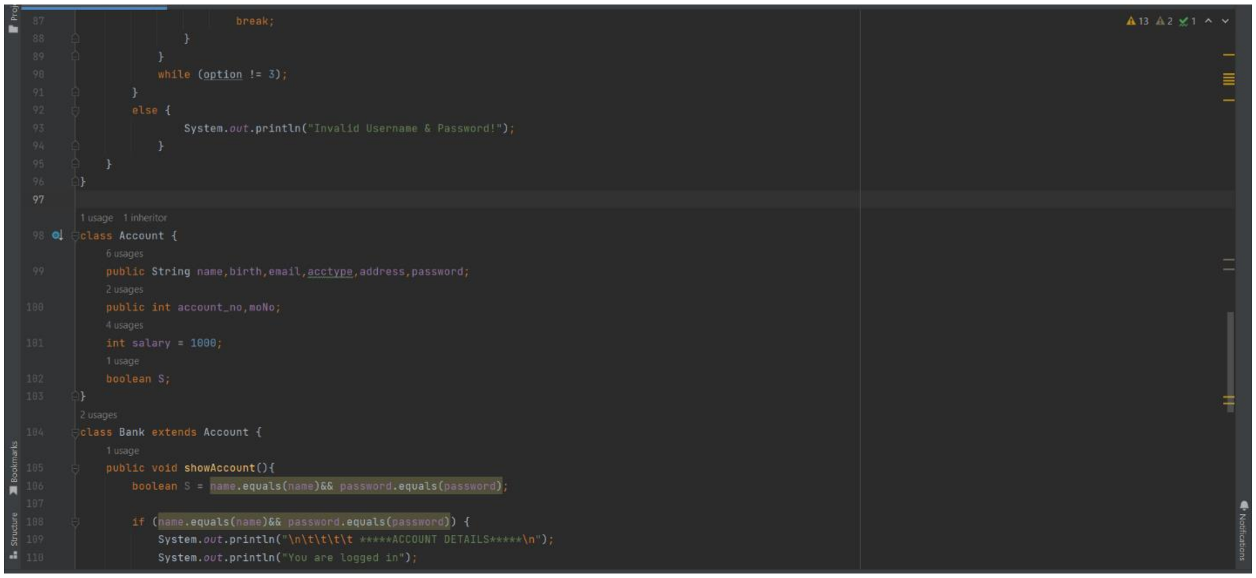
2.1Source codes and outputs

Y.M.S.N.R YAPA – PS/2019/127

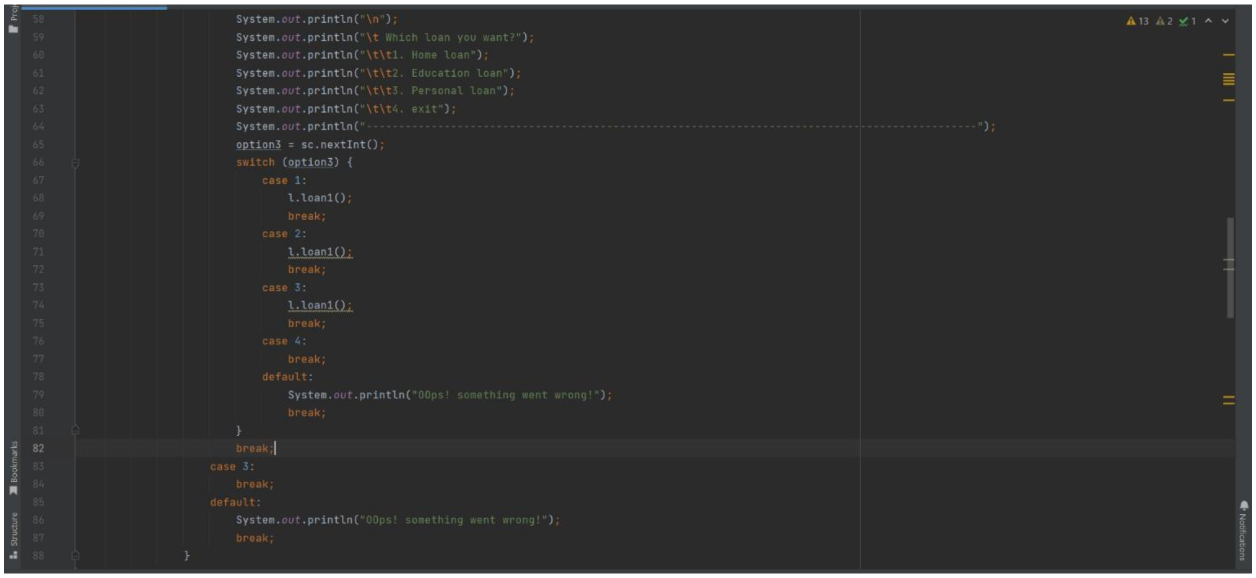
Source code



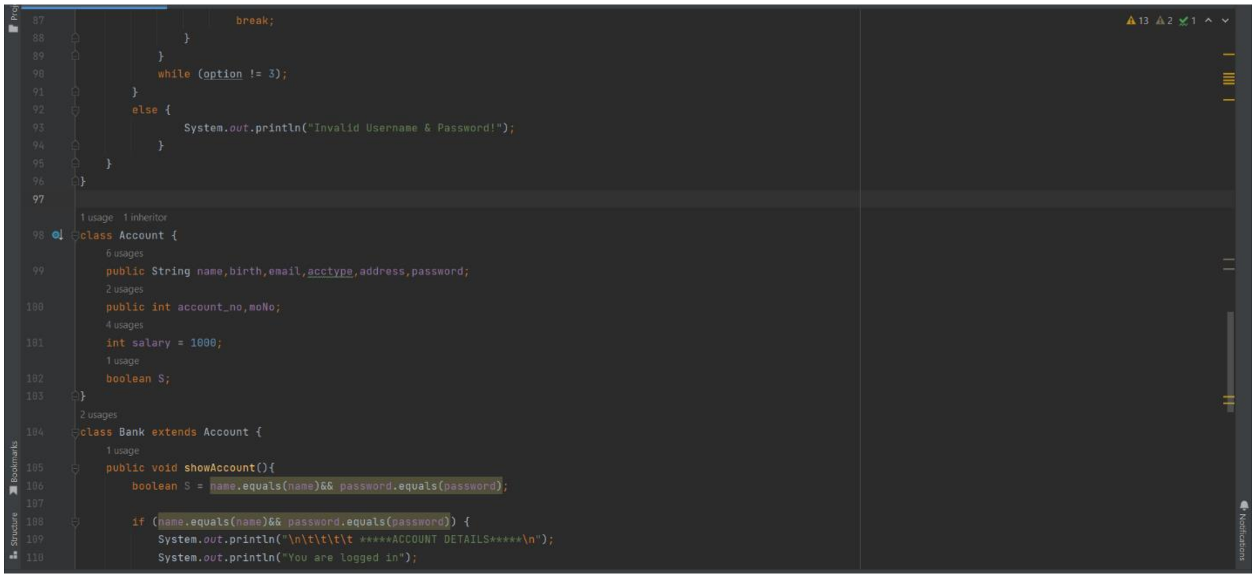
```java
import BankDetails.deposit;
import prev_transaction.withdraw;
import prev_transaction.getprev_transaction;
import loan.Loan;
import java.util.*;

public class BankingApplication {
    public static vo
        Bank obj=new
        deposit o= n
        withdraw w=new withdraw();
        getprev_transaction p=new getprev_transaction();
        Loan l=new Loan();
        int option='\0';
        int option2='\0';
        int option3='\0';
        int balance=0;
        Scanner sc=new Scanner(System.in);
        System.out.println("\n \t\t\t ****** WELCOME TO UOK BANK******\n");
        obj.openAccount();
        obj.showAccount();
        if( obj.S == false) {
            do {
                System.out.println("\t What would you like to do:");
                System.out.println("\t\t1. Transaction");
                System.out.println("\t\t2. Loan");
                System.out.println("\t\t3. exit");
                System.out.println("------------------------------------------------------------------------------------------------- ");
                option = sc.nextInt();

                switch (option) {
```



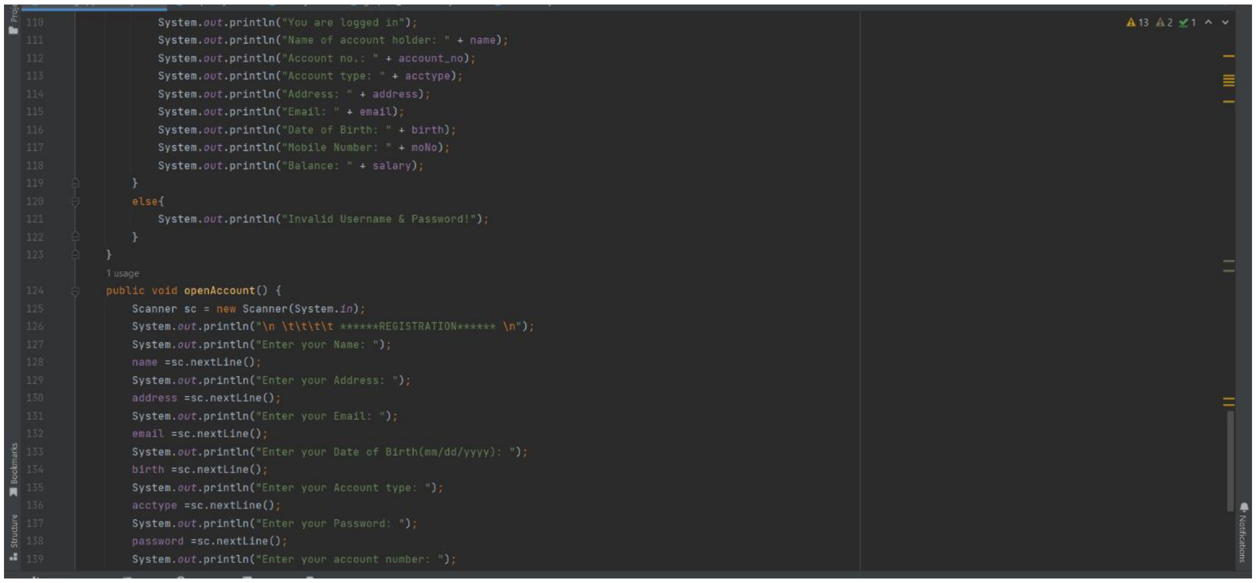
```java
                        break;
                }
            }
            while (option != 3);
        }
        else {
                System.out.println("Invalid Username & Password!");
            }
    }
}

1 usage   1 inheritor
class Account {
    6 usages
    public String name,birth,email,acctype,address,password;
    2 usages
    public int account_no,moNo;
    4 usages
    int salary = 1000;
    1 usage
    boolean S;
}
2 usages
class Bank extends Account {
    1 usage
    public void showAccount(){
        boolean S = name.equals(name)&& password.equals(password);

        if (name.equals(name)&& password.equals(password)) {
            System.out.println("\n\t\t\t\t *****ACCOUNT DETAILS*****\n");
            System.out.println("You are logged in");
```



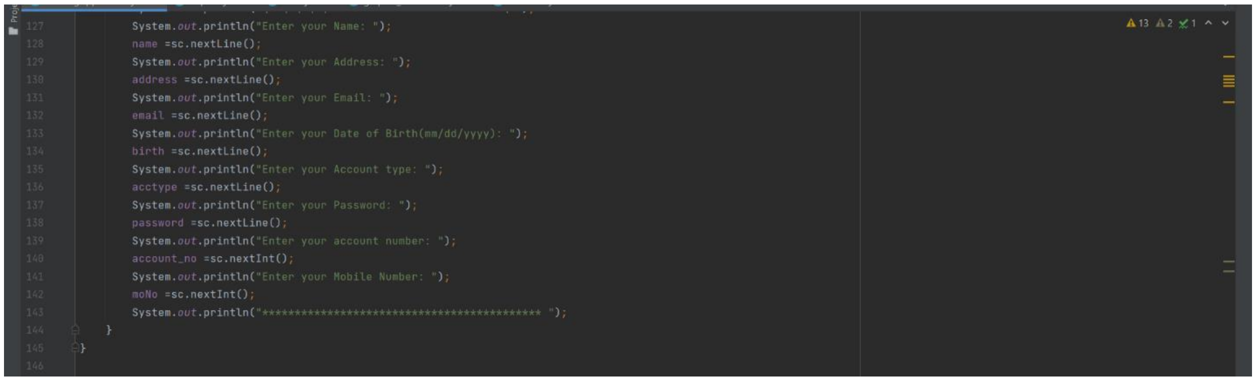
```java
                        System.out.println("\n");
                        System.out.println("\t Which loan you want?");
                        System.out.println("\t\t1. Home loan");
                        System.out.println("\t\t2. Education loan");
                        System.out.println("\t\t3. Personal loan");
                        System.out.println("\t\t4. exit");
                        System.out.println("---------------------------------------------------------------------------------------------");
                        option3 = sc.nextInt();
                        switch (option3) {
                            case 1:
                                l.loan1();
                                break;
                            case 2:
                                l.loan1();
                                break;
                            case 3:
                                l.loan1();
                                break;
                            case 4:
                                break;
                            default:
                                System.out.println("OOps! something went wrong!");
                                break;
                        }
                        break;
                    case 3:
                        break;
                    default:
                        System.out.println("OOps! something went wrong!");
                        break;
                }
```



```java
                        break;
                }
            }
            while (option != 3);
        }
        else {
                System.out.println("Invalid Username & Password!");
            }
    }
}

1 usage   1 inheritor
class Account {
    6 usages
    public String name,birth,email,acctype,address,password;
    2 usages
    public int account_no,moNo;
    4 usages
    int salary = 1000;
    1 usage
    boolean S;
}
2 usages
class Bank extends Account {
    1 usage
    public void showAccount(){
        boolean S = name.equals(name)&& password.equals(password);

        if (name.equals(name)&& password.equals(password)) {
            System.out.println("\n\t\t\t\t *****ACCOUNT DETAILS*****\n");
            System.out.println("You are logged in");
```

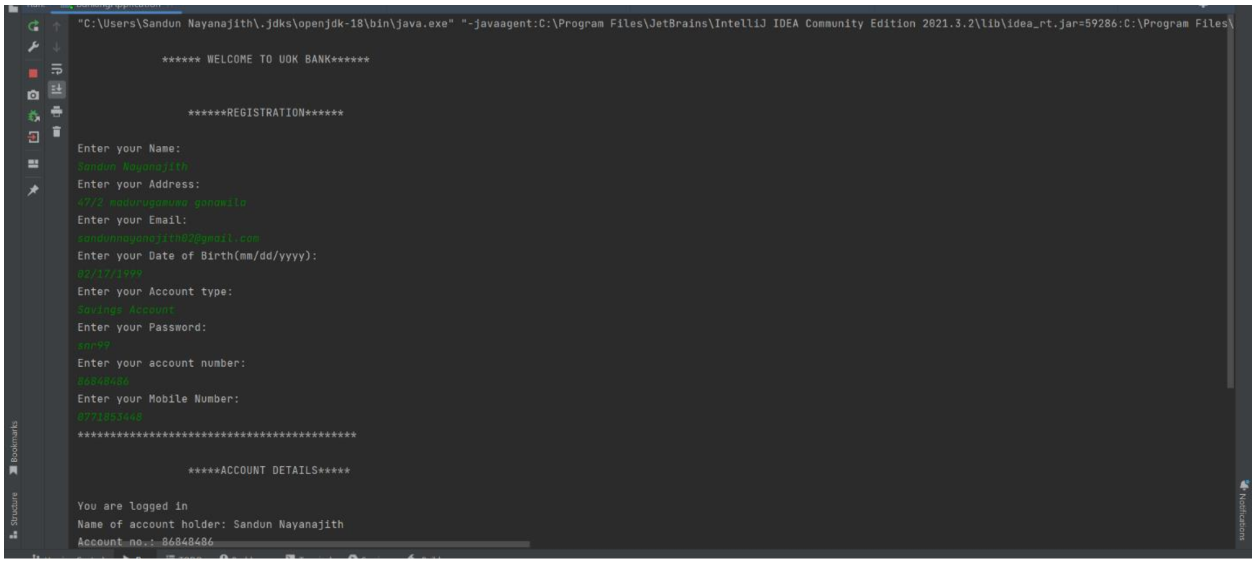


```java
            System.out.println("You are logged in");
            System.out.println("Name of account holder: " + name);
            System.out.println("Account no.: " + account_no);
            System.out.println("Account type: " + acctype);
            System.out.println("Address: " + address);
            System.out.println("Email: " + email);
            System.out.println("Date of Birth: " + birth);
            System.out.println("Mobile Number: " + moNo);
            System.out.println("Balance: " + salary);
        }
        else{
            System.out.println("Invalid Username & Password!");
        }
    }
    1 usage
    public void openAccount() {
        Scanner sc = new Scanner(System.in);
        System.out.println("\n \t\t\t\t ******REGISTRATION****** \n");
        System.out.println("Enter your Name: ");
        name =sc.nextLine();
        System.out.println("Enter your Address: ");
        address =sc.nextLine();
        System.out.println("Enter your Email: ");
        email =sc.nextLine();
        System.out.println("Enter your Date of Birth(mm/dd/yyyy): ");
        birth =sc.nextLine();
        System.out.println("Enter your Account type: ");
        acctype =sc.nextLine();
        System.out.println("Enter your Password: ");
        password =sc.nextLine();
        System.out.println("Enter your account number: ");
```

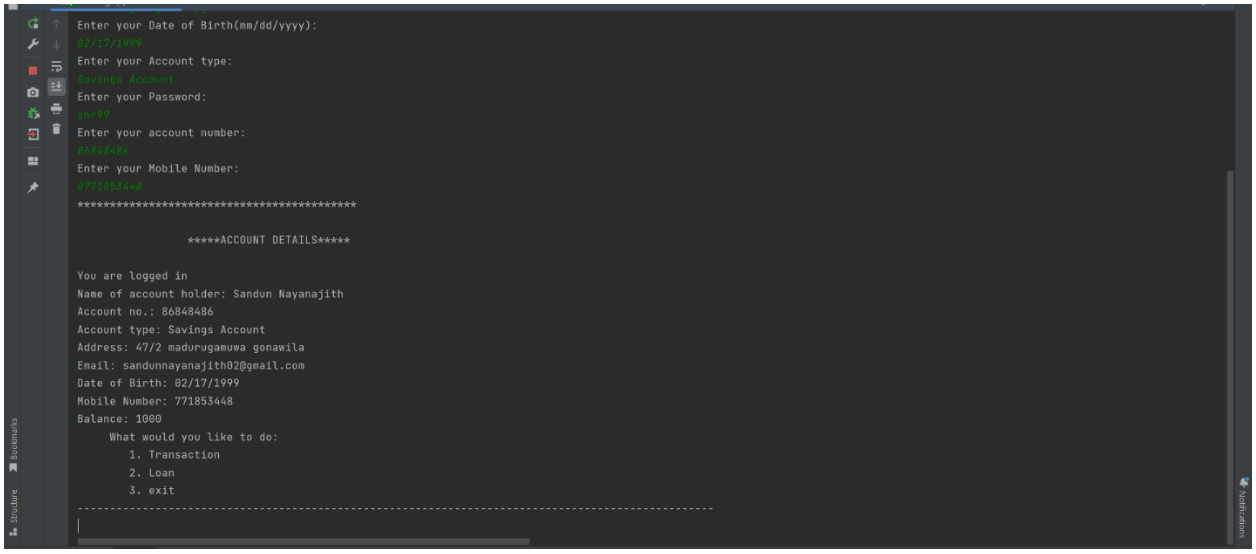


```java
            System.out.println("Enter your Name: ");
            name =sc.nextLine();
            System.out.println("Enter your Address: ");
            address =sc.nextLine();
            System.out.println("Enter your Email: ");
            email =sc.nextLine();
            System.out.println("Enter your Date of Birth(mm/dd/yyyy): ");
            birth =sc.nextLine();
            System.out.println("Enter your Account type: ");
            acctype =sc.nextLine();
            System.out.println("Enter your Password: ");
            password =sc.nextLine();
            System.out.println("Enter your account number: ");
            account_no =sc.nextInt();
            System.out.println("Enter your Mobile Number: ");
            moNo =sc.nextInt();
            System.out.println("******************************************* ");
        }
}
```

Output



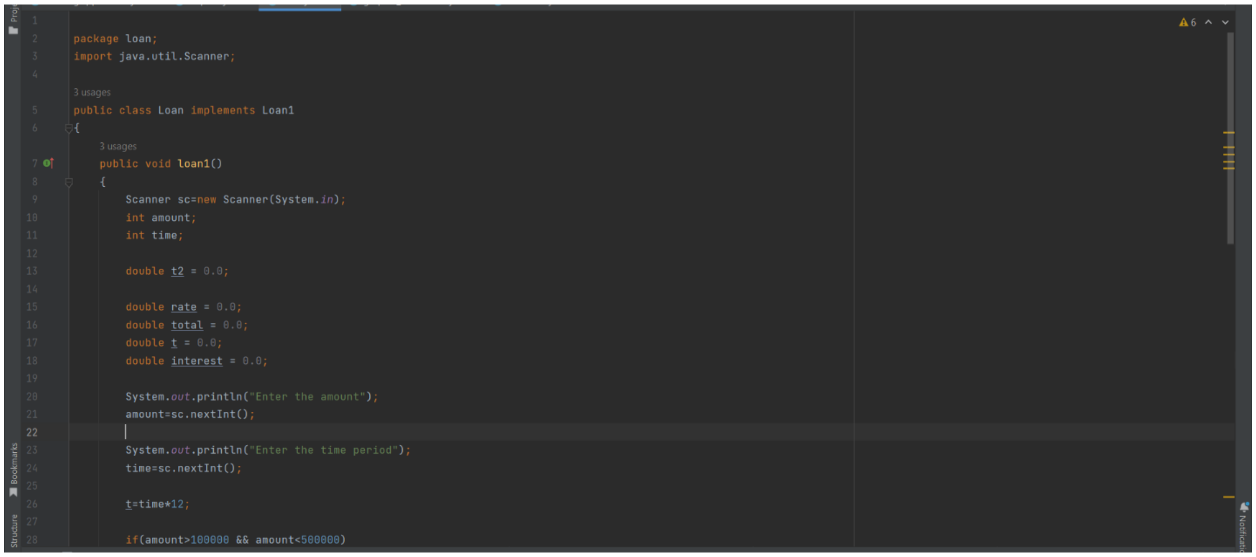
```
"C:\Users\Sandun Nayanajith\.jdks\openjdk-18\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3.2\lib\idea_rt.jar=59286:C:\Program Files\

             ****** WELCOME TO UOK BANK******


                 ******REGISTRATION******

Enter your Name:
Sandun Nayanajith
Enter your Address:
47/2 madurugamuwa gonawila
Enter your Email:
sandunnayanajith02@gmail.com
Enter your Date of Birth(mm/dd/yyyy):
02/17/1999
Enter your Account type:
Savings Account
Enter your Password:
snr99
Enter your account number:
86848486
Enter your Mobile Number:
0771853448
*******************************************

                 *****ACCOUNT DETAILS*****

You are logged in
Name of account holder: Sandun Nayanajith
Account no.: 86848486
```



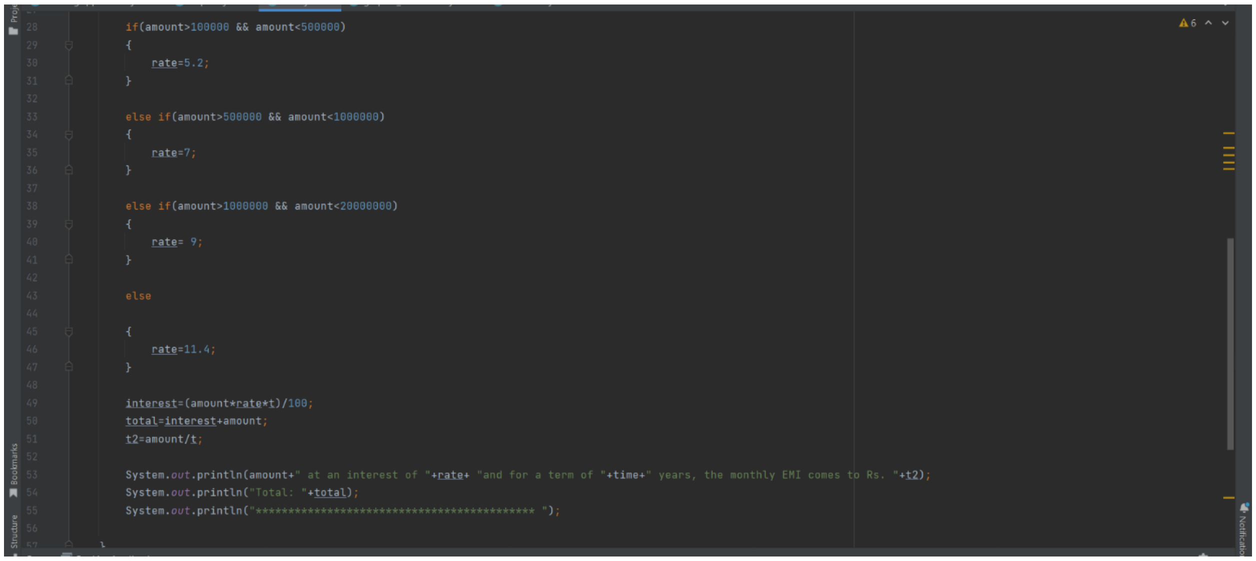
```
Enter your Date of Birth(mm/dd/yyyy):
02/17/1999
Enter your Account type:
Savings Account
Enter your Password:
snr99
Enter your account number:
86848486
Enter your Mobile Number:
0771853448
*******************************************

                 *****ACCOUNT DETAILS*****

You are logged in
Name of account holder: Sandun Nayanajith
Account no.: 86848486
Account type: Savings Account
Address: 47/2 madurugamuwa gonawila
Email: sandunnayanajith02@gmail.com
Date of Birth: 02/17/1999
Mobile Number: 771853448
Balance: 1000
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
-------------------------------------------------------------------------------------------------
|
```

J.A.K.N JAYAKODY – PS/2019/164

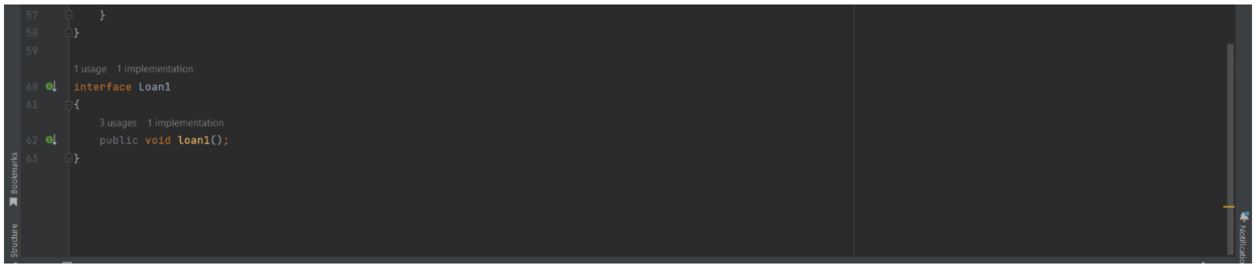
Source code



```java

package loan;
import java.util.Scanner;

3 usages
public class Loan implements Loan1
{
    3 usages
    public void loan1()
    {
        Scanner sc=new Scanner(System.in);
        int amount;
        int time;

        double t2 = 0.0;

        double rate = 0.0;
        double total = 0.0;
        double t = 0.0;
        double interest = 0.0;

        System.out.println("Enter the amount");
        amount=sc.nextInt();

        System.out.println("Enter the time period");
        time=sc.nextInt();

        t=time*12;

        if(amount>100000 && amount<500000)
```

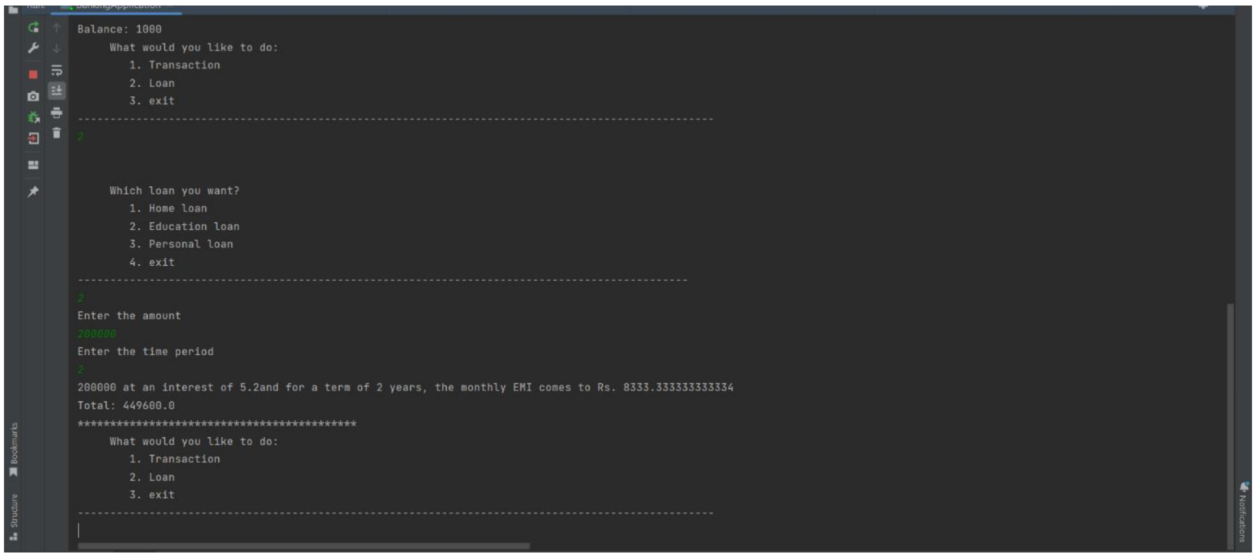


```java
        if(amount>100000 && amount<500000)
        {
            rate=5.2;
        }

        else if(amount>500000 && amount<1000000)
        {
            rate=7;
        }

        else if(amount>1000000 && amount<20000000)
        {
            rate= 9;
        }

        else

        {
            rate=11.4;
        }

        interest=(amount*rate*t)/100;
        total=interest+amount;
        t2=amount/t;

        System.out.println(amount+" at an interest of "+rate+ "and for a term of "+time+" years, the monthly EMI comes to Rs. "+t2);
        System.out.println("Total: "+total);
        System.out.println("******************************************* ");

    }
```



```java
    }
}

1 usage   1 implementation
interface Loan1
{
    3 usages   1 implementation
    public void loan1();
}
```

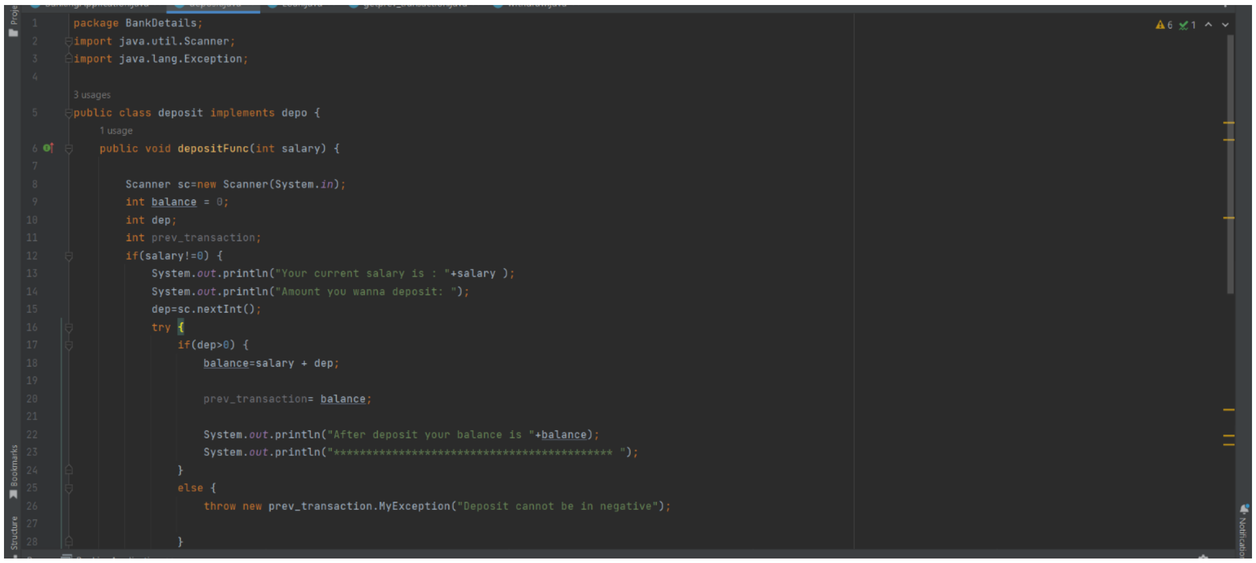
OUTPUT



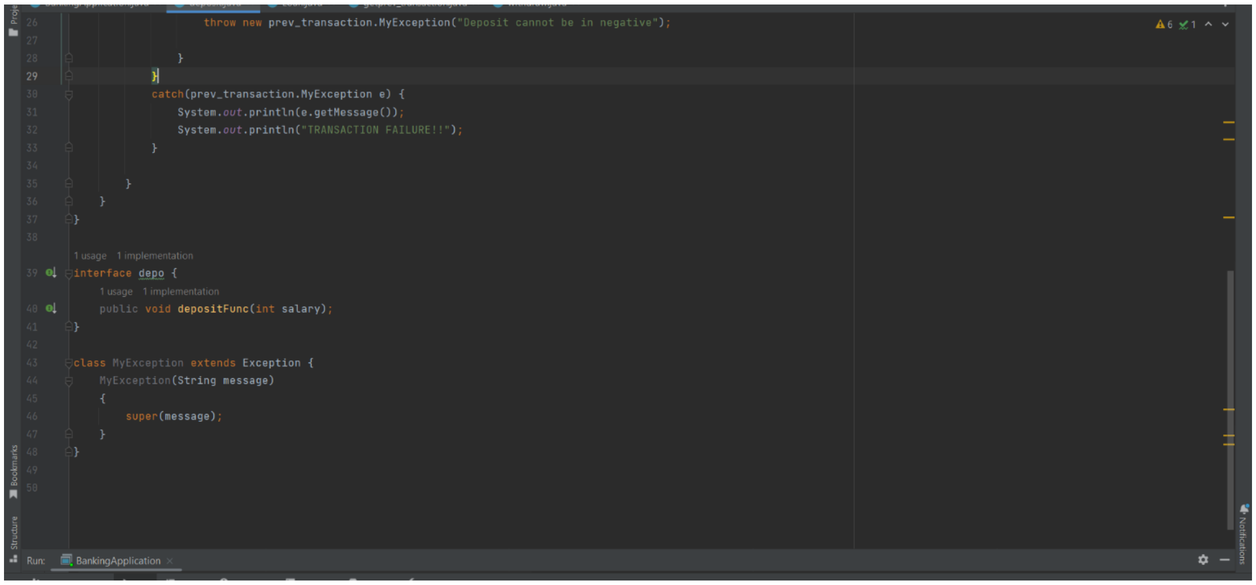
```
Balance: 1000
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
-----------------------------------------------------------------------------------------------
2



     Which loan you want?
        1. Home loan
        2. Education loan
        3. Personal loan
        4. exit
-------------------------------------------------------------------------------------------
2
Enter the amount
200000
Enter the time period
2
200000 at an interest of 5.2and for a term of 2 years, the monthly EMI comes to Rs. 8333.333333333334
Total: 449600.0
*******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
-----------------------------------------------------------------------------------------------
```

A.A.S.S ADIKARI – PS/2019/170

Source code

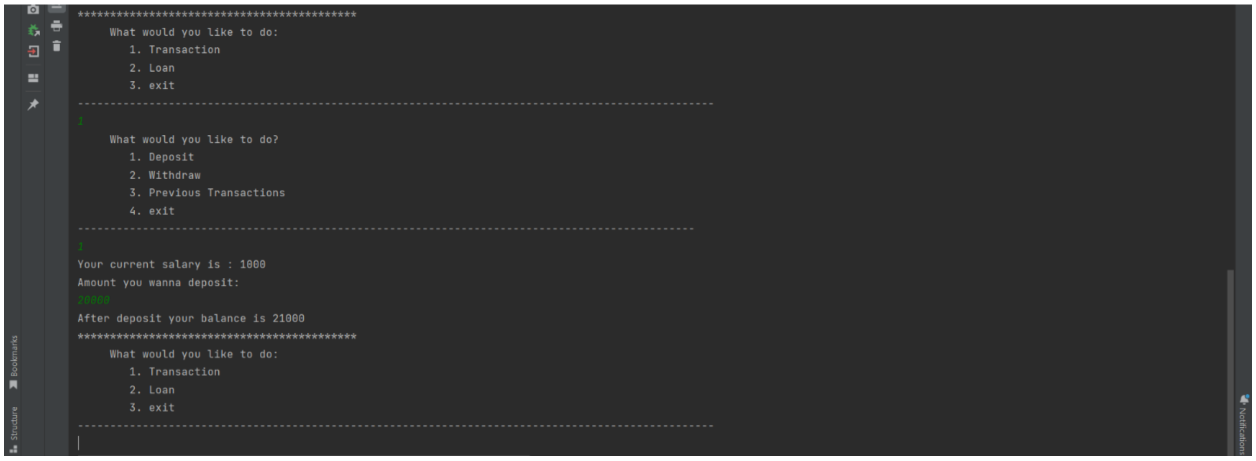


```java
package BankDetails;
import java.util.Scanner;
import java.lang.Exception;

public class deposit implements depo {
    public void depositFunc(int salary) {

        Scanner sc=new Scanner(System.in);
        int balance = 0;
        int dep;
        int prev_transaction;
        if(salary!=0) {
            System.out.println("Your current salary is : "+salary );
            System.out.println("Amount you wanna deposit: ");
            dep=sc.nextInt();
            try {
                if(dep>0) {
                    balance=salary + dep;

                    prev_transaction= balance;

                    System.out.println("After deposit your balance is "+balance);
                    System.out.println("******************************************* ");
                }
                else {
                    throw new prev_transaction.MyException("Deposit cannot be in negative");

                }
```



```java
                    throw new prev_transaction.MyException("Deposit cannot be in negative");

                }
            }
            catch(prev_transaction.MyException e) {
                System.out.println(e.getMessage());
                System.out.println("TRANSACTION FAILURE!!");
            }

        }
    }
}

interface depo {
    public void depositFunc(int salary);
}

class MyException extends Exception {
    MyException(String message)
    {
        super(message);
    }
}
```

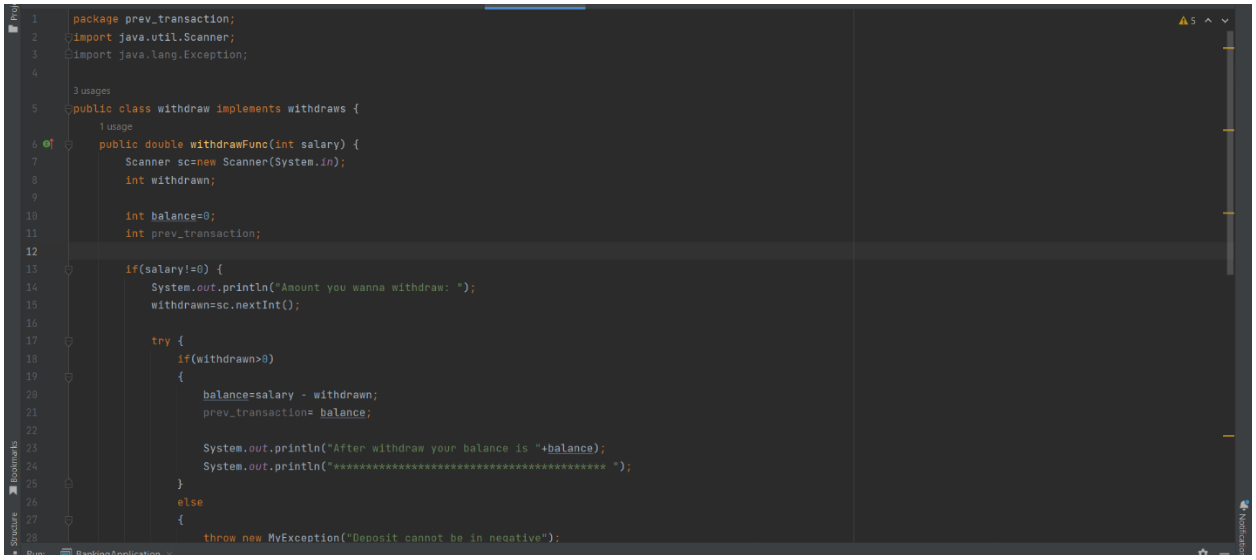
OUTPUT



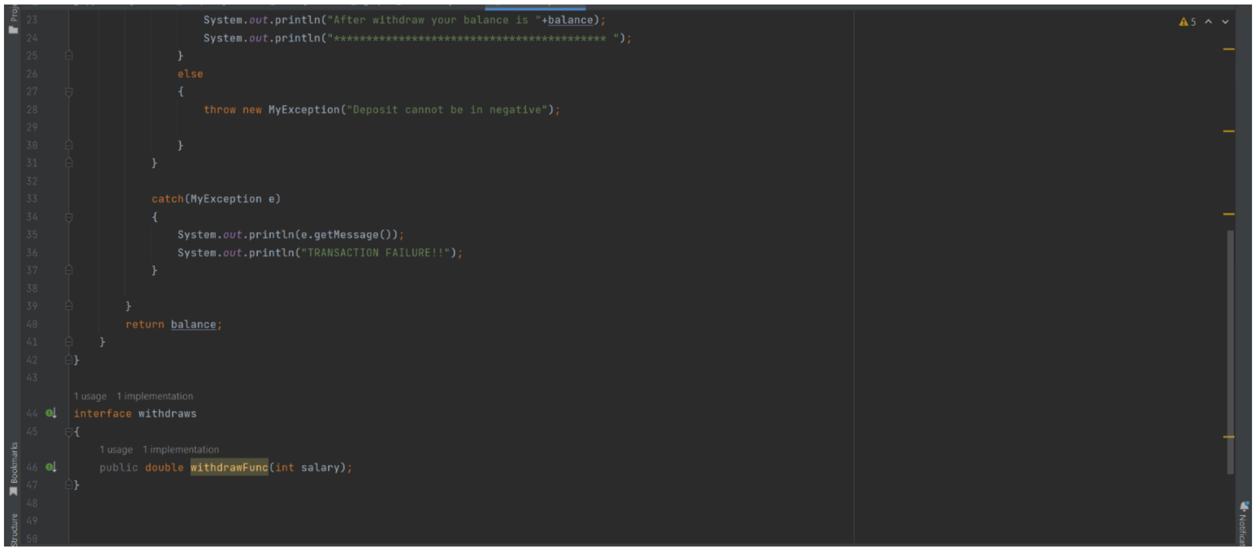
```
*******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
---------------------------------------------------------------------------------------------------
1
     What would you like to do?
        1. Deposit
        2. Withdraw
        3. Previous Transactions
        4. exit
------------------------------------------------------------------------------------------------
1
Your current salary is : 1000
Amount you wanna deposit:
20000
After deposit your balance is 21000
*******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
---------------------------------------------------------------------------------------------------
```

P.D.I.G RATHNASIRI – PS/2019/012

Source code

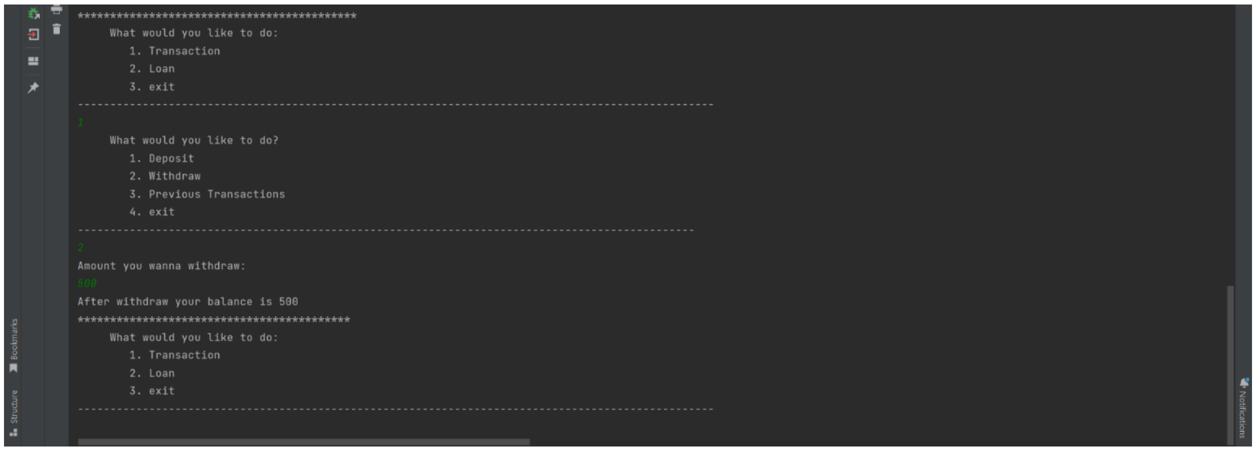


```
package prev_transaction;
import java.util.Scanner;
import java.lang.Exception;

3 usages
public class withdraw implements withdraws {
    1 usage
    public double withdrawFunc(int salary) {
        Scanner sc=new Scanner(System.in);
        int withdrawn;

        int balance=0;
        int prev_transaction;

        if(salary!=0) {
            System.out.println("Amount you wanna withdraw: ");
            withdrawn=sc.nextInt();

            try {
                if(withdrawn>0)
                {
                    balance=salary - withdrawn;
                    prev_transaction= balance;

                    System.out.println("After withdraw your balance is "+balance);
                    System.out.println("****************************************** ");
                }
                else
                {
                    throw new MyException("Deposit cannot be in negative");
```



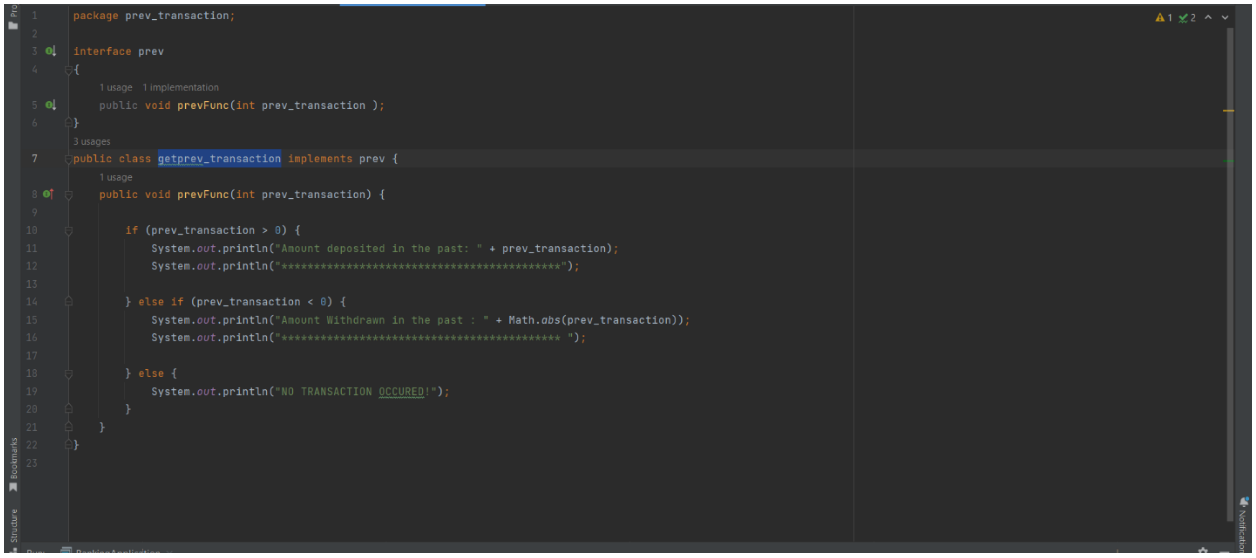
```
                    System.out.println("After withdraw your balance is "+balance);
                    System.out.println("****************************************** ");
                }
                else
                {
                    throw new MyException("Deposit cannot be in negative");

                }
            }

            catch(MyException e)
            {
                System.out.println(e.getMessage());
                System.out.println("TRANSACTION FAILURE!!");
            }

        }
        return balance;
    }
}

1 usage  1 implementation
interface withdraws
{
    1 usage  1 implementation
    public double withdrawFunc(int salary);
}
```

OUTPUT



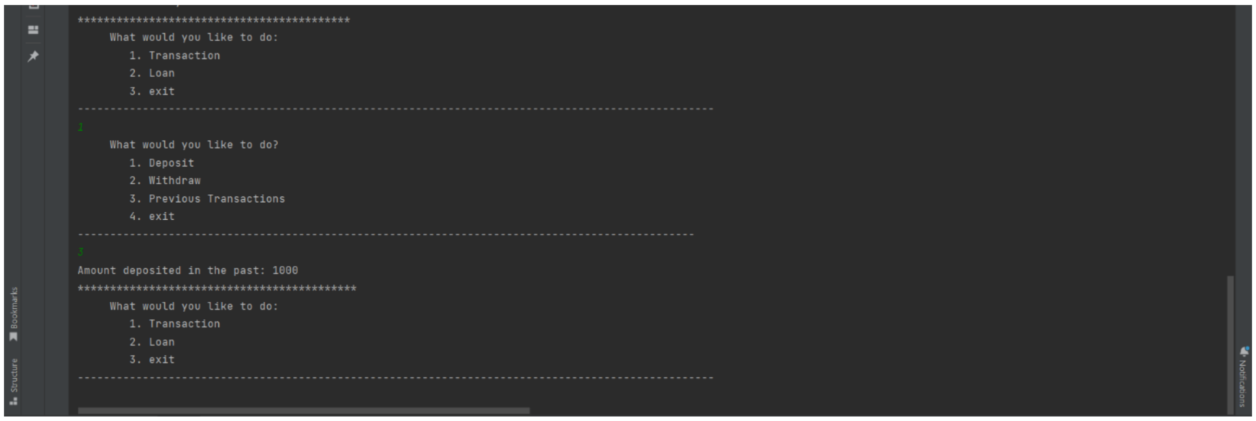
```
*******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
--------------------------------------------------------------------------------------------------
1
     What would you like to do?
        1. Deposit
        2. Withdraw
        3. Previous Transactions
        4. exit
-----------------------------------------------------------------------------------------------
2
Amount you wanna withdraw:
500
After withdraw your balance is 500
******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
--------------------------------------------------------------------------------------------------
```

Source code



```java
package prev_transaction;

interface prev
{
    public void prevFunc(int prev_transaction );
}
public class getprev_transaction implements prev {
    public void prevFunc(int prev_transaction) {

        if (prev_transaction > 0) {
            System.out.println("Amount deposited in the past: " + prev_transaction);
            System.out.println("******************************************");

        } else if (prev_transaction < 0) {
            System.out.println("Amount Withdrawn in the past : " + Math.abs(prev_transaction));
            System.out.println("****************************************** ");

        } else {
            System.out.println("NO TRANSACTION OCCURED!");
        }
    }
}
```

OUTPUT



```
******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
------------------------------------------------------------------------------------------------
1
     What would you like to do?
        1. Deposit
        2. Withdraw
        3. Previous Transactions
        4. exit
---------------------------------------------------------------------------------------------
3
Amount deposited in the past: 1000
*******************************************
     What would you like to do:
        1. Transaction
        2. Loan
        3. exit
------------------------------------------------------------------------------------------------
```

3.Use cases of OOP concepts in Bank Management system

Abstraction

Abstraction lets programmers create useful and reusable tools. For example, a programmer can create several different types of objects, which can be variables, functions, or data structures. Programmers can also create different classes of objects as ways to define the objects.

For instance, a class of variable might be an address. The class might specify that each address object shall have a name, street, city and zip code. The objects, in this case, might be bank addresses, customer addresses.

Encapsulation

In the example below, encapsulation is demonstrated as an OOP concept in Java. Here, the variable "name" is kept private or "encapsulated."

```
Public class Account{
        Private String name;
        Private double balance;
        public double getBalance(){
                return balance;
                }
        }
Public class BankingApp{
        Public static void main(String []args){
                Account bal = new Account();
                bal.getBalance();
                }
}
```

Inheritance

It's quite simple to achieve inheritance as an OOP concept in Java. Inheritance can be as easy as using the extends keyword:

```
class Account{

String name;

int account_no;

int salary;

}

class Bank extends Account{

}
```

Polymorphism

Polymorphism in Java works by using a reference to a parent class to affect an object in the child class.

Two more examples of polymorphism in Java are method overriding and method overloading.

In method overriding, the child class can use the OOP polymorphism concept to override a method of its parent class. That allows a programmer to use one method in different ways depending on whether it's invoked by an object of the parent class or an object of the child class.

In method overloading, a single method may perform different functions depending on the context in which it's called. This means a single method name might work in different ways depending on what arguments are passed to it

4.Demonstration

- This project is developed to nurture the needs of a user in a banking sector by embedding all the tasks of transactions taking place in a bank. Future version of this project will still be much enhanced than the current version. Writing and depositing checks are perhaps the most fundamental ways to move money in and out of a checking account, but advancements in technology have added ATM and debit card transactions. All banks have rules about how long it takes to access your deposits, how many debit card transactions you're allowed in aday, and how much cash you can withdraw from an ATM. Access to the balance in your checking account can also be limited by businesses that place holds on your funds. Banks are providing internet banking services also so that the customers can be attracted. By asking the bank employs we came to know that maximum numbers of internet bank account holders are youth and business man. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable in a volatile and competitive marketplace of today. If proper training should be given to customer by the bank employs to open an account will be beneficial secondly the website should be made friendlier from where the first time customers can directly make and access their accounts. Thus the Bank Management System it is developed and executed successfully.

5.Teamwork

- Teamwork has great importance in life. It is the key to success in any field of life. It builds trust and confidence among team members and also enhances their individual capabilities. When people work together, they can achieve more than what they could have achieved working individually.
- Working in a team requires good communication, coordination, and collaboration skills. It is important to be able to work well with others in order to achieve common goals. Teamwork can help people to better utilize their skills and talents, and it can also create a sense of camaraderie and bonding among team members.

- Our Team Members are:
  - Y.M.S.N.R YAPA – PS/2019/127
  - J.A.K.N JAYAKODY – PS/2019/164
  - A.A.S.S ADIKARI – PS/2019/170
  - P.D.I.G RATHNASIRI – PS/2019/012
  - J.M.N.C JAYAMANNE – PS/2019/244