Department of Computer Science & Engineering
UNIVERSITY OF LIBERAL ARTS BANGLADESH

University of Liberal Arts Bangladesh

Course Code: CSE 412

Course Title: Programming with JAVA

Open Ended Project Report

"Bank Management Software"

Submitted To

Dr. Mohammed Ashikur Rahman

Assistant Professor

Department of Computer Science and Engineering

University of Liberal Arts Bangladesh

Submitted By

Name	ID
Iqbal Hoshen Saif	**201014040**
Farjahan Akter Boby	**201014007**
Nahian Raisa	**201014076**

Submission Date: December 23, 2022

Contents

Contribution Table

Contributor	**Task**
Farjahan Akter Boby	1. Coding 2. Bug fixing 3. Report Writing 4. Slide Creating
Iqbal Hoshen Saif	1. Coding
Nahian Raisa	1. Report Writing 2. Slide Creating

Department of
Computer Science & Engineering
University of Liberal Arts Bangladesh

Assessment Rubric for Solution of Complex Engineering Project

Criteria	No / wrong answer (0)	Poor (1)	Developing (2)	Accomplished (3)
Understanding the requirements	No / wrong answer	Defined requirements and specifications poorly.	Defined requirements and specifications moderately.	Defined requirements and specifications clearly & properly.
Design & develop the solutions	No / wrong answer	Poorly designed & developed the solutions using appropriate tools, techniques, and skills	Partly/ moderately designed & developed the solutions using appropriate tools, techniques, and skills	Properly designed & developed the solutions using appropriate tools, techniques, and skills
Result & Analysis	No / wrong answer	Weak demonstration of the ability to draw correct conclusion of results obtained from the tool.	Partly demonstrate the ability to draw correct conclusion of results obtained from the tool.	Fully demonstrate the ability to draw correct conclusion of results obtained from the tool.
Communication (Organization, Formatting, Errors and Effectively Comprehend and Write Report)	No / wrong answer	**Poor** organization and formatting; and contains errors. **Cannot** Communicate key concepts and solution.	Partly organized and formatted with relatively less error. Communicate the key concepts and solution up to **some extent**.	**Well** organized and properly formatted without any error. Communicate the key concepts and solution **effectively**.

“Bank Management Software”

Problem Statement: Demonstrate a bank management system software using object-oriented programming concepts of the JAVA programming language.

Objective: The aims to build the bank management software will be:

1. To create two systems for employee and customer
2. Employee system will allow employees to operate operations such as add accounts, delete accounts and show account information
3. The account type can be current or savings as per customer’s choice
4. Customer system will allow customers to deposit money, transfer money or withdraw money
5. All these operations need to be fulfilled using OOP concepts of JAVA

Modern Tools: Apache NetBeans IDE 15

Requirement Analysis

A. Methodology: It refers to the methods that we will be needing to develop our designed system

OOP concepts: As our prime requirements are to use JAVA OOP concepts We will use inheritance, abstraction, polymorphism, interface, and encapsulation concepts of JAVA OPP here. Now let us know about these concepts.

1. Inheritance: Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming systems). The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

To increase coding reusability inheritance is used. Through it, we can inherit a class’s object, method, and attributes to other class/classes. Inheritance can be of 5 types. Such as:

1. Single-level inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Multiple Level inheritance
5. Hybrid inheritance

Amongst all these types we will be using single-level inheritance. Single inheritance is the simplest type of inheritance in java. In this, a class inherits the properties from a single class. The class which inherits is called the derived class or child class or subclass, while the class from which the derived class inherits is called the base class or superclass, or parent class.

2. Abstraction: Data abstraction is the process of hiding certain details and showing only essential information to the user. When we want to hide the implementation but want its functionality to be presented to the user, in that case, abstraction is used. Abstraction can be achieved with either abstract classes or interfaces.

Generally, an abstract class in Java is a template that stores the data members and methods that we use in a program. We cannot instantiate the abstract class in Java directly. Instead, we can subclass the abstract class. When we use an abstract class as a subclass, the abstract class method implementation becomes available to all of its parent classes. In our program, we have used the abstraction class and inherited the abstract methods in a normal class.

3. Polymorphism: The word "polymorphism" means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. This polymorphic behavior can be done by method overloading, method overriding, constructor overloading, or by the polymorphic behaviour of the object. There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism.

In our system we will be using method overriding to apply polymorphism.

If a subclass provides the specific implementation of the method declared by one of its parent classes, it is known as method overriding. The conditions of method overriding implementation are:

1. Methods MUST be in two different classes.
2. There MUST be an inheritance between the two classes.
3. Method Name MUST be same.
4. Method Parameter MUST be the same.
5. Method Return Type MUST be same.

4. Interface: An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritances in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. At the same time, it achieves loose coupling. Which means a function is not dependent on another function. In our system, we have applied an interface to ensure code reusability.

5. Encapsulation: Encapsulation in Java refers to integrating data (variables) and code (methods) into a single unit. In encapsulation, a class's variables are hidden from other classes and can only be accessed by the methods of the class in which they are found. In our program, we will achieve encapsulation through private and protected variables which can only be accessed by the inherited classes.

B. Library Function:

To build this project some prebuilt function needs to be utilized here. Such that

Function	Used for
substring ()	To get a specific part of the string this function was used.
equals ()	To check whether two strings are equal or the same this function was used.
add ()	This one is used to add a specific element to a Set collection.
size ()	To get the length of the string this function was utilized.
super ()	To call the superclass constructor to a child class this function was used.
random ()	This function is used to choose random numbers. The range of it is 0.0 to 1.0. Thus, by default its return type is double.
remove ()	This method removes the element at the specified index and returns it
pow ()	This is used to return the value of the first argument raised to the power of the second argument.

Design of the Developed Solution:

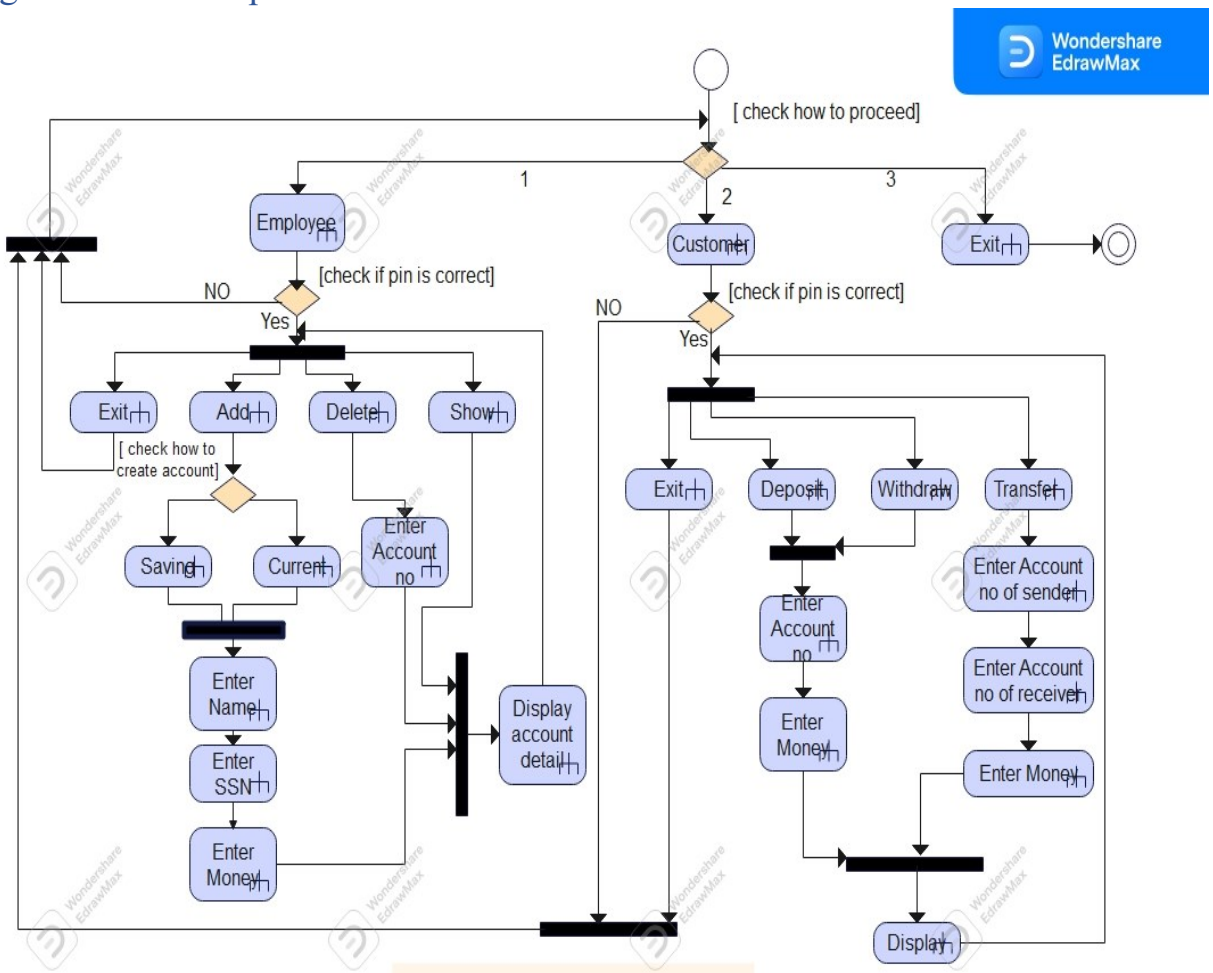


Figure: UML Activity Diagram of Bank Management Software

Explanation of the Developed Solution

Employee:

In this project, we are about to build Bank Management System. Through this system, at first, there will be two interfaces. One for the employee and another for the customer.

When the system user enters here with the first option, system will ask for the password to access the employee section. The wrong password will not allow employee to enter in the section. The right password will open 4 choices for the employee. These are showing account information, adding an account, deleting, and exiting from the employee interface.

Suppose that the employee chooses to add an account, then they can do it in two ways. One is Creating a saving account and another is creating a current account. If the user specifies a saving account or current account, then they have to push the account holder's name, SSN, and initial deposit money for both. Account for the users will be created successfully with that provided information.

After the successful creation of the current account the account information such as Account No, Account Holder Name, Total Balance, Rate and Accrued Interest, Debit Card Number, and Pin number will be shown to the employee who has created the account for the customer.

After the successful creation of the savings account the account information such as Account No, Account Holder Name, Total Balance, Rate and Accrued Interest, Safety Deposit Box ID, and Safety Deposit Box Key will be shown to the employee who has created the account for the customer.

If the employee chooses to delete an account, then they just have to input the account holder's account number. If it is provided correctly that means the account number matches with one of the existing account numbers, then the deletion of the account will be successful. All the information of account holders can be shown using the show information option.

Customer:

When the system user enters here with the second option, the system will ask for the password to access the customer section. The wrong password will not allow the customer to enter the section. The right password will open 3 choices for the customer. These are deposit money, withdraw money, and transfer money. The exit menu will terminate the program.

If the customer wants to deposit money, then they have to provide the account number and amount of money. If the user already has an account with that account number, then depositing will be successful.

On the other hand, to withdraw money similarly they have to push the account number and amount of money. If the user already has an account with that account number, then the withdrawal will be successful.

To transfer money the customer has to push the account numbers from which account they want to transfer money to which account. If the user already has an account with that account number, then transferring will be successful.

Result and Analysis

A. Class Description: There are in total of 6 classes in our program. All the class descriptions which include a method, attributes, and constructors' description are described below:

1. IBaseRate Class:

This class was created using the Interface which includes a method getBaseRate. The return type of this method is double. Later on, this class was inherited by the abstract class Account.

2. Account Class:

This is an abstract class that inherited the IBaseRate class. To inherit that 'implements' keyword was used as the IBaseRate is an interface class.

Attributes:

Here the 'name', 'SSN', 'balance', 'and accrued Interest' were kept private and the 'account number' and 'rate' were protected. This was done because 'name', 'SSN', 'balance', 'and accrued Interest' will be accessed only within the Account class while 'account number' and 'rate' will be used on the other classes within the same package.

Constructor:

This class has a constructor name Account which has three parameters that name, SSN, and initial deposit. This was called by the Current and Saving class using the super keyword. Within that, encapsulation was used.

At the same time, the index was increased whenever a new account was created. To maintain batch or serial numbers of account numbers, an index was used. Again, a setAccNo was called which was assigned to the accNumber. Then the setRate, the compound method was called within this constructor.

Methods:

setAccNo:

This was built to generate an account number for each user. With it using some condition unique account number will be generated. When the user will put in SSN while creating an account, the last two digits of it will be retrieved which was assigned to the lastTwoOfSSN attribute.

Then a random number was set. Lastly will return the account number in such a way that the last two digits of SNN will be added followed by the index number that was discussed before. And at the end of the number that random number will be added.

setRate:

This is an abstract method that was overridden by the Current class and Saving class.

compound:

While creating an account the total balance and accrued Interest need to be added. Which was done by this method.

showInfo:

This method was overridden by both the Current and Saving class.

Lastly, using the getter and setter method the values of the account class were updated and returned.

3. Current Class: This is the child class of the account class.

Attributes:

This child class has its own attributes called debitCardNo and debitCardPIN which are private. In the initial phase, the debit card number was initialized with the null value.

Constructor:

The name of the constructor of the class is Current which has three parameters name, SSN, and initial deposit. In this constructor, the constructor of the parent class was called using the super keyword. So here is the user's name, initially deposited money, account number, rate, and accrued Interest all the calculations were done.
Again, next to it the account number was updated adding the character 2 in front of the accNumber that was generated in the Account class before. This 2 indicates that this is a Current account. Again, within the constructor, another method "setDebitCard" was called.

Methods:

setDebitCard:

In this method, the debit card number and debit card pin were generated. Here to generate the card number, at first, a loop of 12 digits was created so that it can create the number of length 12 where digits will be randomly selected from 0 to 9.
After that to generate the PIN, from (0 to 9999) some digits will be randomly chosen. Then this will be converted to the integer type.

showInfo:

This method was overridden here to print the Debit card number and pin along with the account holder information. To do so, from the parent class showInfo method was called using the super keyword.

setRate:

This was overridden from the Account class. Again, here specific amount was multiplied with getBaseRate which was retrieved from the interface type class IBaseRate.

4. Savings Class: This is the child class of the account class.

Attributes:

This child class has its own attributes called safetyBoxID and safetyBoxKey which are private.

Constructor:

The name of the constructor of the class is Current which has three parameters name, SSN, and initial deposit. In this constructor, the constructor of the parent class was called using the super keyword. So here is the user's name, initially deposited money, account number, rate, and accrued Interest all the calculations were done.
Again, next to it the account number was updated adding the character 1 in front of the accNumber that was generated in the Account class before. This 1 indicates that this is a savings account. Again, within the constructor, another method "setSafetyBox" was called.

Methods:

setSafetyBox:

Here to generate ID, from (0 to 999) some digits will be randomly chosen and will be assigned to safetyBoxID. And to generate the key, from (0 to 9999) some digits will be randomly chosen and will be assigned to safetyBoxKey.

showInfo:

This method was overridden here to print the Depositbox ID and key along with the account holder information. To do so, from the parent class showInfo method was called using the super keyword.

setRate:

This was overridden from the Account class. Again, here specific amount was subtracted from the getBaseRate which was retrieved from the interface type class IBaseRate.

5. Bank Class:

Here a LinkedList was created and referred by the account class whose object is accList. This was done so that we can easily add an account to a list and easily can retrieve it from a specific portion. We have imported java.util package in this class to perform the related tasks to the package.

Methods:

addAccount():

In this function we will be adding the account details pf account holders. At first using scanner we have taken the input for which type of account the user wants to open. Then name, SSN number, and initial deposited money amount has been taken as input from users in order to open the bank account for users. The account name and SSN variables are of string data type and the initial deposit amount is of double data type.

Then using if else condition we have checked which type of account list should store the account holder's information. If the user wants to have a savings account, then we have added the taken information in the savings account list. If user wants to have a current account, then we have added the taken information in the current account list. In both cases, we have called the showInfo() function using the account class's object after inserting the account information in list in order to show it from system to system user.

deleteAccount():

The delete account will successfully delete the account and related information to the account. For this function, we will ask for the unique account number which is a string-type variable from the user as input.

If the account number matches the previously listed account numbers than the account and account details will be successfully deleted otherwise no change will occur. This matching process will take place in a loop where the loop will iterate the whole list until it founds the

desired account number. After the deletion the loop will break and a successful account deletion message will be shown to the system user.

deposit(String accNo, double amount):

This function will help to deposit money in accounts by account holders. For this we will ask for the unique account number which is a string type variable, the deposited amount which is a double type variable from the user as input in main function and will pass it to the deposit function as parameter.

If the account number matches with the previously listed account numbers then the new deposited money will be added to the previous balance. This matching process will take place in a loop where the loop will iterate the whole list using a object of account class until it founds the desired account number. After the successful deposition of money the new total balance will be shown to the user by calling the printBalance() function by the account class object.

withdraw(String accNo, double amount):

This function will help to withdraw money in accounts by account holders. For this we will ask for the unique account number which is a string type variable , the withdrawal amount which is a double type variable from the user as input in main function and will pass it to the withdraw function as parameter.

If the account number matches with the previously listed account numbers then the withdraw money will be subtracted from the current balance. This matching process will take place in a loop where the loop will iterate the whole list using a object of account class until it founds the desired account number. After the successful withdrawal of money the new total balance will be shown to the user by calling the printBalance() function by the account class object.

transfer(String fromWhere, String toWhere, double amount):

This function will help to transfer money of accounts by account holders. For this we will ask for the unique account number which is a string type variable, the account number which is a double type variable from which the amount will be withdrawn, the account number which is a double type variable to which the amount will be deposited from the user as input in main function and will pass it to the transfer function as parameter.

After that inside the function we will take a nested loop to perform the transfer process. The outer loop will iterate the whole list using a object of account class until it founds the desired account number from where the money will be transferred. Inside of the outer loop if the account number matches with the previously listed account numbers then the program will enter to inner loop.

The inner loop will also iterate till the last of the list using another object of account class. Inside it using set balance(), getBalance(), and account class objects the transferable money will be deducted from the account it has been withdrawn and will be added to the account where it needs to be deposited. After the successful transfer process a message will be shown

and the remaining balance of the withdrawn money account will be displayed using printBalance() function.

6. Main Class:

This is the main class of our program where we have taken user input for all of our system functionalities. As we are going to take user inputs so we have imported java.util package here. Inside the main function we have printed a welcome panel.

The whole system will run inside a while loop until the user wants to exit. Inside the while we have taken three scanner objects which will be used to take inputs for different functions.

At first the user will be shown three choices using println() function. If user's choice is equal to 1 then he/she will be taken to the Employee panel and will be asked for the pin number. If the pin number matches with the required pin number then he/she can access the employee panel's functionalities. The employee panel offers several options to access which can be accessed through pressing the right choice.

If user presses 1 then user will be shown the previously registered account information using println() function. Similarly if user chooses option 2 which is add account then inside else if the addAccount() will be called using Bank class object.

Similarly if user chooses option 3 which is delete account then inside else if the deleteAccount() will be called using Bank class object. If the user presses option 4 which is exit then the user will be taken to the welcome panel where user can chose again that which system he/she wants to access.

If user chooses option 2 which is customer panel then then he/she will be taken to the customer panel and will be asked for the pin number. If the pin number matches with the required pin number then he/she can access the customer panel's functionalities.

The customer panel offers several options to access which can be accessed through pressing the right choice. If user presses 1 then system will ask for the necessary details from the user which are require to perform deposit operation. Then taken inputs will be passed to the deposit() function.

If user presses 2 then system will ask for the necessary details from the user which are require to perform withdraw operation. Then taken inputs will be passed to the withdraw() function. Similarly If user presses 3 then system will ask for the necessary details from the user which are require to perform transfer operation. The taken inputs will be passed to the transfer() function.

If the user presses option 4 which is exit then the user will be taken to the welcome panel where user can chose again that which system he/she wants to access. If the user presses 3 then the system will be terminated. We have used println() function to print invalid value message shown upon when user will provide invalid values which goes against of the program system.

B: Programming Code

Account Class

```java
package practice.main1;

public abstract class Account implements IBaseRate {

        private String name;
        private String ssn;
        private double balance;
        protected String accNumber;
        protected double rate;
        private static int index = 10000;
        private double accruedInterest;
        public Account(String name, String ssn, double initialDeposit) {
                this.name = name;
                this.ssn = ssn;
                this.balance = initialDeposit;
                index++;
                // set account# method called
                this.accNumber = setAccNo();
                // set base rate
                setRate();
                // find out the accrue interest
                compound();
        }
        private String setAccNo() {
                String lastTwoOfSSN = ssn.substring(ssn.length() - 2, endIndex:ssn.length());
                int randNo = 257;
                return lastTwoOfSSN + index + randNo;
        }
        public abstract void setRate();
        public void compound() {
                accruedInterest = balance* (rate/100);
                balance += accruedInterest;
        }
```

```java
        public void showInfo() {
                        System.out.println("Name: " + name +
                                        "\nAcc#: " + accNumber +
                                        "\nBalance: " + balance +
                                        "\nRate: " + rate + "%" +
                                        "\nAccrued Interest: $" + accruedInterest);
                }

        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public double getBalance() {
                return balance;
        }
        public void setBalance(double balance) {
                this.balance = balance;
        }
        public String getSsn() {
                return ssn;
        }
        public String getAccNumber() {
                return accNumber;
        }
        public double getRate() {
                return rate;
        }
        public double getAccruedInterest() {
                return accruedInterest;
        }
        public void printBalance() {
                System.out.println("Your balance is: $"+ balance);
        }
}
```

Current Class:

```
package practice.main1;
import java.util.Random;

public class Current extends Account {

        private String debitCardNo = "";
        private int debitCardPIN;

        public Current (String name, String ssn, double initialDeposit) {
                super(name,ssn,initialDeposit);
                accNumber = "2"+accNumber;
                // set debit card method
                setDebitCard();
        }
        private void setDebitCard() {
                for(int i=0;i<12;i++) {
                        Random rand = new Random();
                        String value = String.valueOf( i:rand.nextInt( bound: 9));
                        debitCardNo = debitCardNo + value;
                }
                debitCardPIN = (int) (Math.random() * Math.pow( a:10,  b:4)); //th
                                                                              // a
        }
        @Override
        public void showInfo() {
                super.showInfo();
                System.out.println(
                                "Your Current Account has been created :"+
                                "\nDebit Card#: "+ debitCardNo+
                                "\nCard PIN: "+ debitCardPIN +
                                "\n**************************");
        }
        @Override
        public void setRate() {
                rate = getBaseRate() * .15;
        }
}

```

IBaseRate Class:

```
package practice.main1;

public interface IBaseRate {
        // method that returns the base rate
        default double getBaseRate() {
                return 2.5;
        }
}

```

Savings Class:

```java
package practice.main1;

public class Saving extends Account {

	private int safetyBoxID;
	private int safetyBoxKey;

	public Saving (String name,String ssn, double initialDeposit) {
		super(name,ssn,initialDeposit);
		accNumber = "1"+accNumber;
		// set the deposit box
		setSafetyBox();
	}

	private void setSafetyBox() {
		safetyBoxID = (int)(Math.random()*Math.pow( a:10,  b:3));
		safetyBoxKey = (int)(Math.random()*Math.pow( a:10,  b:4));
	}

	@Override
	public void showInfo() {
		super.showInfo();
		System.out.println(
				"Your Savings Account has been created :"+
				"\nSafety Depositbox ID: "+ safetyBoxID+
				"\nSafety Depositbox Key: "+ safetyBoxKey +
				"\n**************************");
	}

	@Override
	public void setRate() {
		rate = getBaseRate() - .25;

	}
}
```

Bank Class:

```java
package practice.main1;
import java.util.*;

public class Bank {

	List<Account> accList = new LinkedList<>();

// add account method
	public void addAccount() {
		Scanner in = new Scanner( source:System.in);
		System.out.println( x:"Please specify the type of account that you want to open (Savings/ Current):");
		String type = in.nextLine();

		System.out.println( x:"Please insert the name of the account holder:");
		String name = in.nextLine();
		System.out.println( x:"Please insert the SSN:");
		String ssn = in.nextLine();
		System.out.println( x:"Please insert the initial deposit:");
		double initDeposit = in.nextDouble();

		if(type.equals( anObject:"Savings")) {
			accList.add(new Saving(name,ssn, initialDeposit:initDeposit));
			Account acc = accList.get(accList.size() - 1);
			acc.showInfo();
		}
		else if(type.equals( anObject:"Current")) {
			accList.add(new Current(name,ssn, initialDeposit:initDeposit));
			Account acc = accList.get(accList.size() - 1);
			acc.showInfo();
		}
		else System.out.println( x:"Incorrect selection");
	}

```

```
34      // delete account method using the account #
35          public void deleteAccount() {
36                  Scanner in = new Scanner( source:System.in);
37                  System.out.println( x:"Please insert the account number to delete:");
38                  String accNo = in.nextLine();
39
40                  for(Account a: accList) {
41                          if(a.getAccNumber().equals( anObject:accNo)) {
42                              accList.remove( o:a);
43                              break;
44                          }
45                  }
46                  System.out.println("Account: "+ accNo + " has been deleted");
47          }
48
49      // deposit funds methods
50          public void deposit(String accNo,double amount) {
51                  for(Account a: accList) {
52                          if(a.getAccNumber().equals( anObject:accNo)) {
53                              a.setBalance(a.getBalance() + amount);
54                              System.out.println("Depositing: $" + amount);
55                              a.printBalance();
56                          }
57                  }
58          }
59
60      // withdraw funds method
61          public void withdraw(String accNo,double amount) {
62                  for(Account a: accList) {
63                          if(a.getAccNumber().equals( anObject:accNo)) {
64                              a.setBalance(a.getBalance() - amount);
65                              System.out.println("Withdrawing: $" + amount);
66                              a.printBalance();
67                          }
68                  }
69          }
```

```
60      // withdraw funds method
61          public void withdraw(String accNo,double amount) {
62                  for(Account a: accList) {
63                          if(a.getAccNumber().equals( anObject:accNo)) {
64                              a.setBalance(a.getBalance() - amount);
65                              System.out.println("Withdrawing: $" + amount);
66                              a.printBalance();
67                          }
68                  }
69          }
70      // transfer funds method
71          public void transfer(String fromWhere, String toWhere, double amount) {
72                  for (Account a : accList) {
73                          if (a.getAccNumber().equals( anObject:fromWhere)) {
74                              for (Account b : accList) {
75                                  a.setBalance(a.getBalance() - amount);
76                                  b.setBalance(b.getBalance() + amount);
77                              }
78                          }
79                          System.out.println("Transferring $" + amount + " to " + toWhere);
80                          a.printBalance();
81                          break;
82                  }
83          }
84
85
86      }
87
```

Main1 Class:

```
package com.mycompany.bankapp;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        System.out.println("    ************** Welcome To Our Bank Management Software **************\n");
        Bank bank = new Bank();
        while (true) {
            Scanner s = new Scanner(System.in);
            Scanner s1 = new Scanner(System.in);
            Scanner s2 = new Scanner(System.in);
            /**
             * ********************************************************************************************
             */
            System.out.println("\t\t ****************************");
            System.out.println("\t\t\t 1) Employee");
            System.out.println("\t\t\t 2) Customer ");
            System.out.println("\t\t\t 3) Exit ");
            System.out.println("\t\t ****************************");
            System.out.println("Enter a Number :");
            int inpt6 = s.nextInt();
            /**
             * ******************************************************
             */
            if (inpt6 == 1) {

                System.out.println("Enter Pin :");
                int inpt5 = s.nextInt();

                if (inpt5 == 123) {
                    boolean t = true;
                    while (t) {
                        System.out.println("\t ------------------------------");
                        System.out.println("\t\t 1) Show Information");
                        System.out.println("\t\t 2) Add Account ");
```

```
                        System.out.println("\t\t 3) Delete Account");
                        System.out.println("\t\t 4) Exit");
                        System.out.println("\t ------------------------------");
                        System.out.println("Enter your choice : ");
                        int inpt = s.nextInt();
                        if (inpt == 1) {
                            for (Account a : bank.accList) {
                                // a.showInfo();
                                System.out.println("\t *********************************");
                                System.out.println("\t\t Name: " + a.getName()
                                        + "\n\t\t Acc#: " + a.getAccNumber()
                                        + "\n\t\t Balance: " + a.getBalance()
                                        + "\n\t\t Rate: " + a.getRate() + "%"
                                        + "\n\t\t Accrued Interest: $" + a.getAccruedInterest() + "\n");
                                System.out.println("\t *********************************");
                            }
                        } else if (inpt == 2) {
                            bank.addAccount();
                        } else if (inpt == 3) {
                            bank.deleteAccount();
                        } else if (inpt == 4) {

                            break;

                        } else {
                            System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!");
                        }
                    }
                } else {
                    System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!");

                }
            } /**
             * ********************************************************************************
             */
            else if (inpt6 == 2) {
                System.out.println("Enter Pin :");
```

```
Source  History
75                  int inpt7 = s.nextInt();
76
77                  if (inpt7 == 201014) {
78                      while (true) {
79                          System.out.println("\t ------------------------------");
80                          System.out.println("\t\t 1.Deposit\n\t\t 2.Withdraw\n\t\t 3.Transfer\n\t\t 4.Exit");
81                          System.out.println("\t ------------------------------");
82                          int inpt3 = s.nextInt();
83                          if (inpt3 == 1) {
84                              System.out.println("Account number");
85                              String inpt1 = s1.nextLine();
86                              System.out.println("Amount to deposite");
87                              double inpt2 = s2.nextDouble();
88                              bank.deposit(inpt1, inpt2);
89                          } else if (inpt3 == 2) {
90                              System.out.println("Account number");
91                              String inpt1 = s1.nextLine();
92                              System.out.println("Amount to withdraw");
93                              double inpt2 = s2.nextDouble();
94                              bank.withdraw(inpt1, inpt2);
95                          } else if (inpt3 == 3) {
96                              System.out.println("From which account number");
97                              String inpt1 = s1.nextLine();
98                              System.out.println("To which account number");
99                              String inpt2 = s1.nextLine();
100                             System.out.println("Amount to transfer");
101                             double inpt4 = s2.nextDouble();
102                             bank.transfer(inpt1, inpt2, inpt4);
103                         } else if (inpt3 == 4) {
104                             break;
105
106                         } else {
107                             System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!!");
108                         }
109                     }
110                 } else {
111                     System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!!");
```

```
Source  History
84                              System.out.println("Account number");
85                              String inpt1 = s1.nextLine();
86                              System.out.println("Amount to deposite");
87                              double inpt2 = s2.nextDouble();
88                              bank.deposit(inpt1, inpt2);
89                          } else if (inpt3 == 2) {
90                              System.out.println("Account number");
91                              String inpt1 = s1.nextLine();
92                              System.out.println("Amount to withdraw");
93                              double inpt2 = s2.nextDouble();
94                              bank.withdraw(inpt1, inpt2);
95                          } else if (inpt3 == 3) {
96                              System.out.println("From which account number");
97                              String inpt1 = s1.nextLine();
98                              System.out.println("To which account number");
99                              String inpt2 = s1.nextLine();
100                             System.out.println("Amount to transfer");
101                             double inpt4 = s2.nextDouble();
102                             bank.transfer(inpt1, inpt2, inpt4);
103                         } else if (inpt3 == 4) {
104                             break;
105
106                         } else {
107                             System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!!");
108                         }
109                     }
110                 } else {
111                     System.out.println("!!!!!!!!!!Invalid value!!!!!!!!!!!");
112
113                 }
114             } else {
115                 System.exit(0);
116             }
117         }
118     }
119
120 }//end of code
```

C. Outcome:

This is our welcome panel where we can see the options available for system user. As soon as user presses 1 user will be asked for pin number which is 123. When the pin number matched user was taken to the employee panel where below showed options were available to access.

```
     ************** Welcome To Our Bank Management Software **************


                  ****************************
                          1) Employee
                          2) Customer
                          3) Exit
                  ****************************
Enter a Number :
1
Enter Pin :
123
          ------------------------------
                  1) Show Information
                  2) Add Account
                  3) Delete Account
                  4) Exit
          ------------------------------
```

After that when user chose option 2 then user was asked for which type of account user wants to open. When user typed Savings then system asked for username, SSN, and initial deposited money amount.

After successful insertion user can see the below showed account information along with the safety deposit box Key and ID.

```
          ------------------------------
                  1) Show Information
                  2) Add Account
                  3) Delete Account
                  4) Exit
          ------------------------------
Enter your choice :
2
Please specify the type of account that you want to open (Savings/ Current):
Savings
Please insert the name of the account holder:
iqbal
Please insert the SSN:
55
Please insert the initial deposit:
50000
                  Name: iqbal
                  Acc#: 15510001257
                  Balance: 51125.0
                  Rate: 2.25%
                  Accrued Interest: $1125.0
Your Savings Account has been created :
Safety Depositbox ID: 89
Safety Depositbox Key: 603
*************************
```

After that when user again chose option 2 then user was asked for which type of account user wants to open. When user typed Current then system asked for username, SSN, and initial deposited money amount.

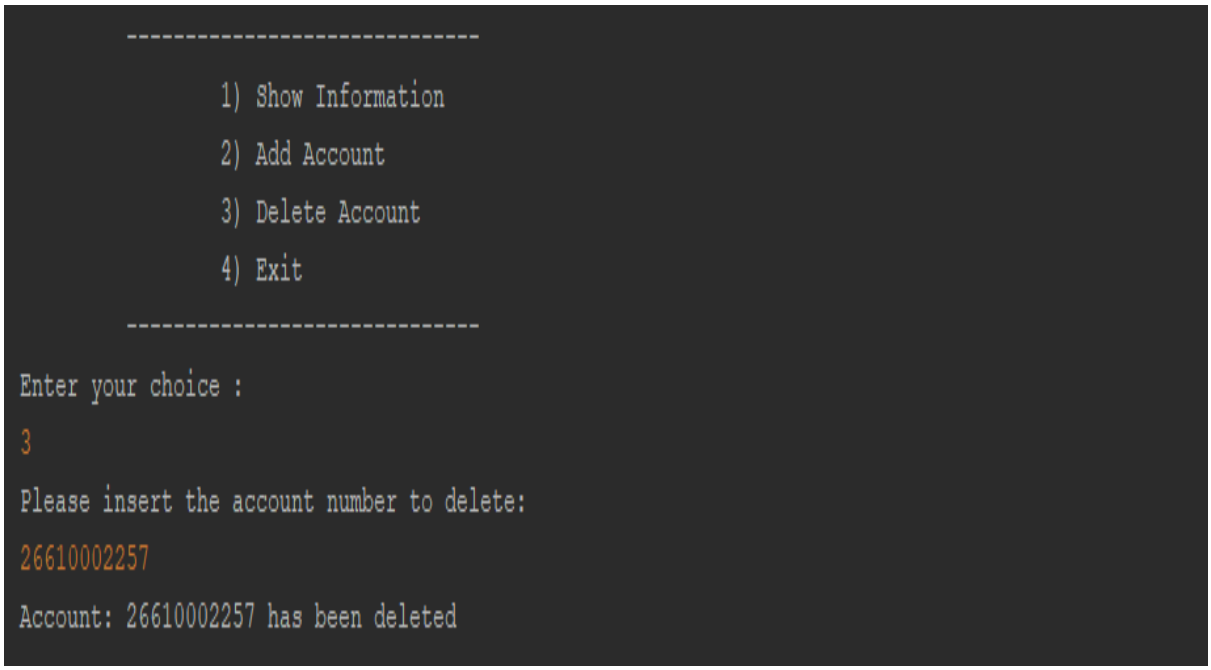
After successful insertion user can see the below showed account information along with the debit card number and card pin number.

```
          ------------------------------
                  1) Show Information
                  2) Add Account
                  3) Delete Account
                  4) Exit
          ------------------------------
Enter your choice :
2
Please specify the type of account that you want to open (Savings/ Current):
Current
Please insert the name of the account holder:
Saif
Please insert the SSN:
66
Please insert the initial deposit:
60000
                  Name: Saif
                  Acc#: 26610002257
                  Balance: 60225.0
                  Rate: 0.375%
                  Accrued Interest: $225.0
Your Current Account has been created :
Debit Card#: 643567552802
Card PIN: 6916
****************************
          ------------------------------
```

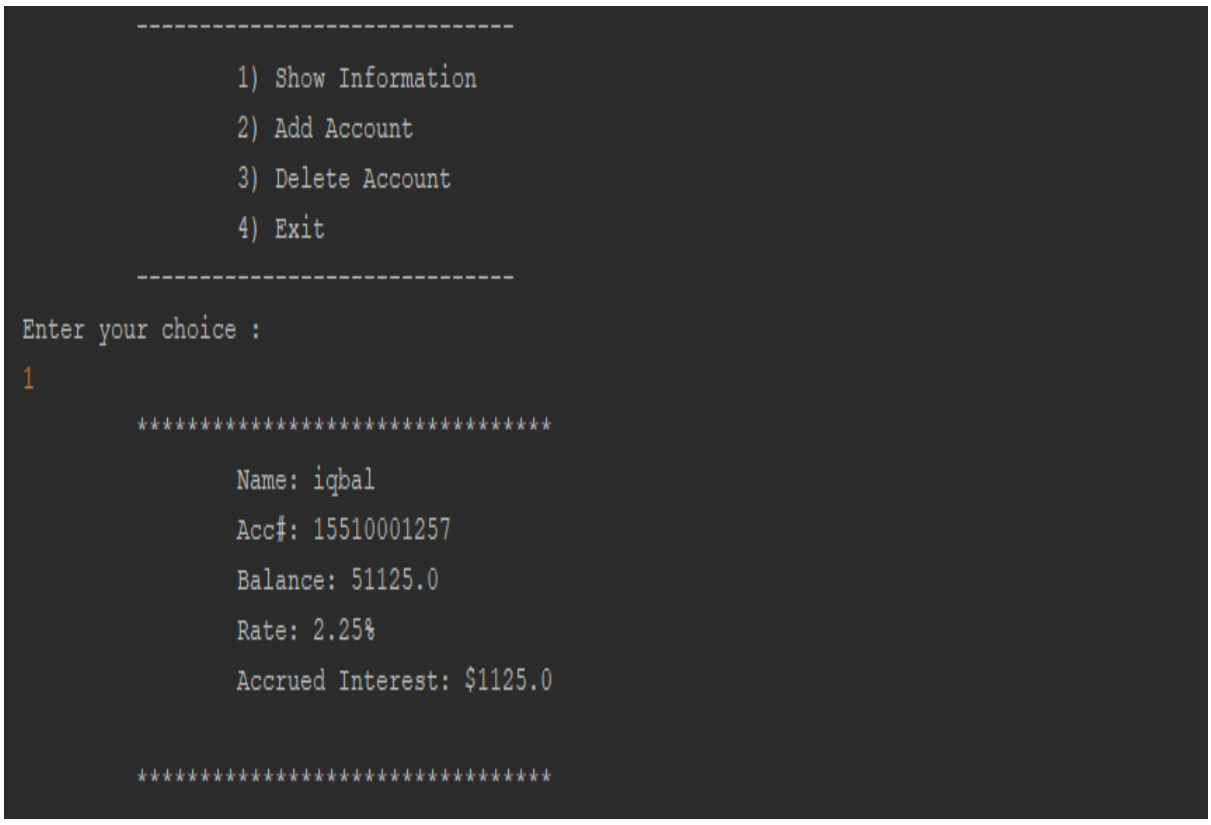
When user pressed the option 1 then user could see the savings and current account information which were inserted just before.

```
         ------------------------------
                  1) Show Information
                  2) Add Account
                  3) Delete Account
                  4) Exit
         ------------------------------
Enter your choice :
1
         *********************************
                  Name: iqbal
                  Acc#: 15510001257
                  Balance: 51125.0
                  Rate: 2.25%
                  Accrued Interest: $1125.0

         *********************************
         *********************************
                  Name: Saif
                  Acc#: 26610002257
                  Balance: 60225.0
                  Rate: 0.375%
                  Accrued Interest: $225.0

         *********************************
```

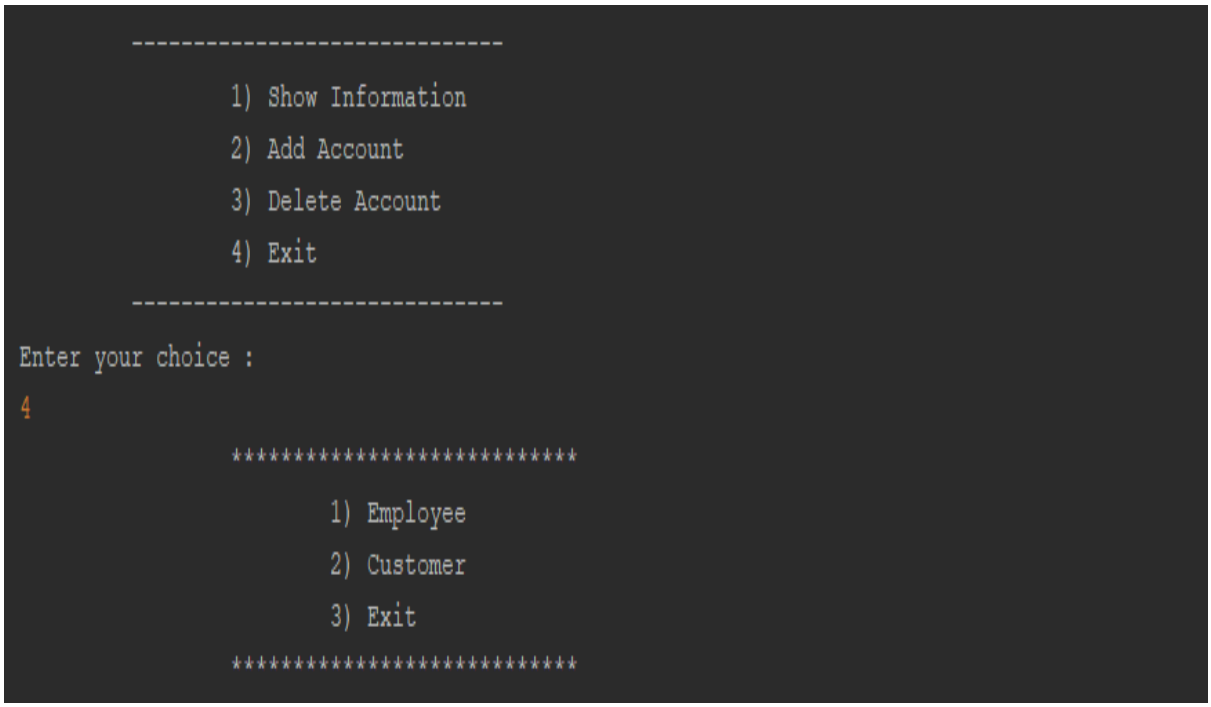
When user chose the option 3 which is delete account and user asked for the account number which user wants to delete. Upon matching the account number the account was deleted by the system.



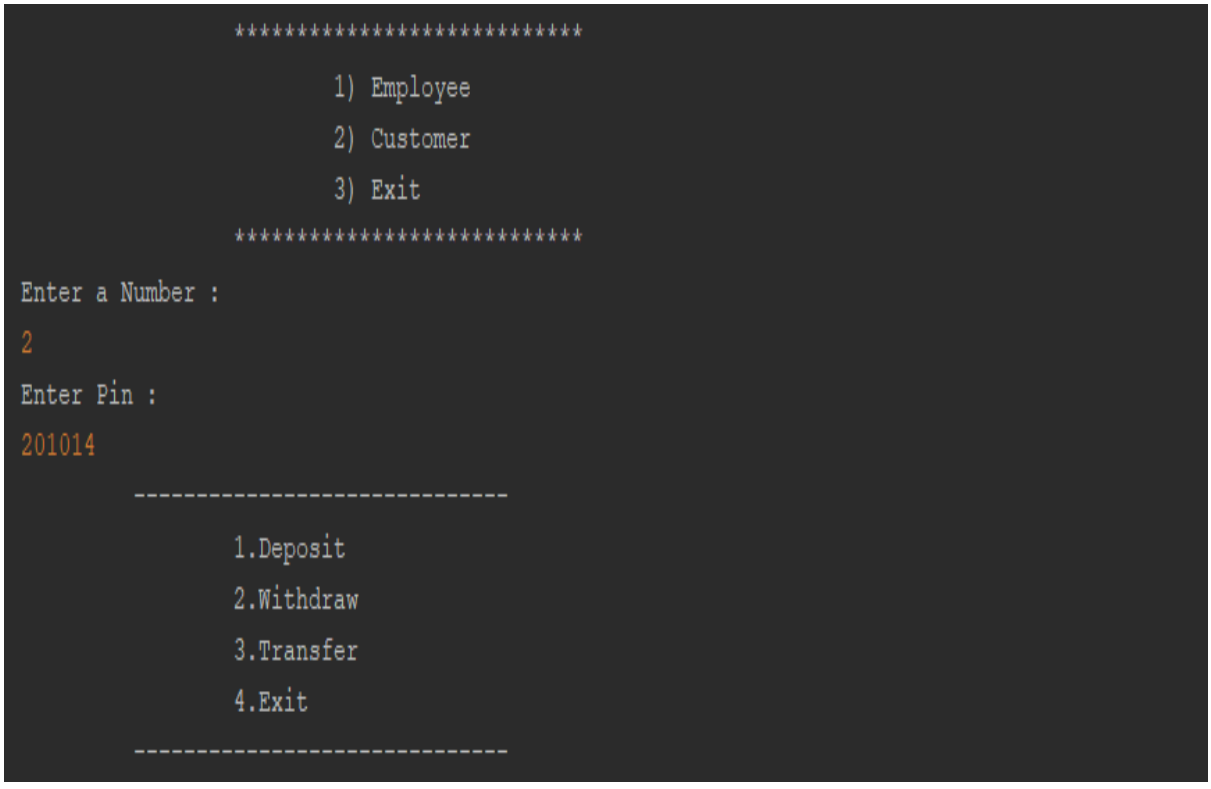
Here we can see the remaining account details by pressing 1 which is show information that the deletion process has been successful and only savings account information is still there in the list.



Here when user will press the option 4 then system will take user out from the employee panel and will led to the welcome panel section as shown below.



As soon as user presses 2 user will be asked for pin number which is 201014. When the pin number matched user was taken to the customer panel where below showed options were available to access.



When the user will choose option 1 from the customer section system will ask for account number and amount to deposit money from the user.

If the account number already exists then deposit action will be performed and the updated balanced will be shown to the user.

```
        ----------------------------
                1.Deposit
                2.Withdraw
                3.Transfer
                4.Exit
        ----------------------------
1
Account number
15510001257
Amount to deposite
10000
Depositing: $10000.0
                Your balance is: $61125.0
```

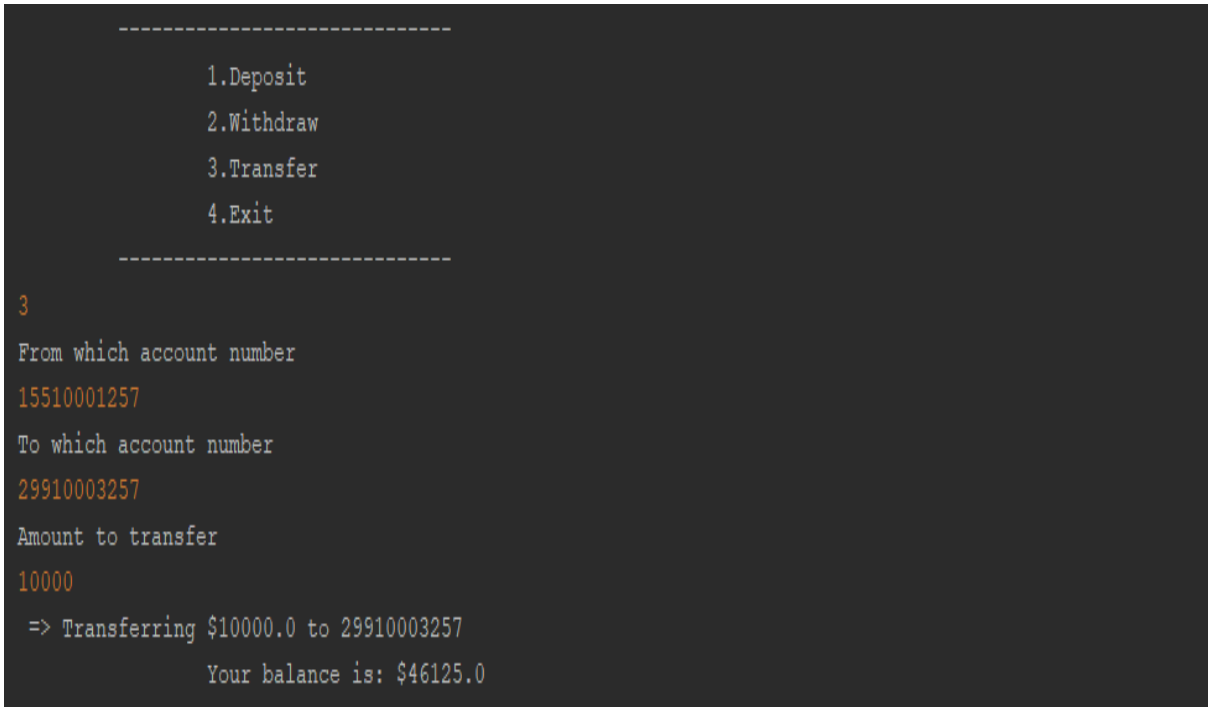
When the user will choose option 2 from the customer section system will ask for account number and amount for withdraw money from the user.

If the account number already exists then withdraw action will be performed and the updated balanced will be shown to the user.

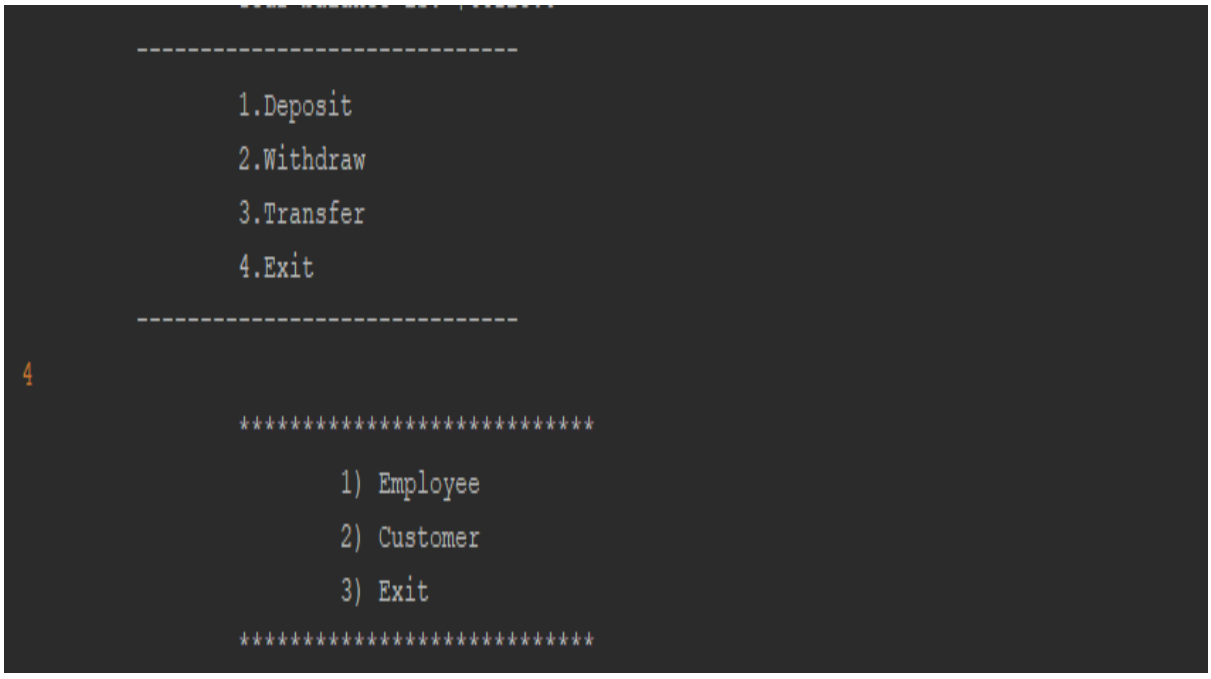
```
        ----------------------------
                1.Deposit
                2.Withdraw
                3.Transfer
                4.Exit
        ----------------------------
2
Account number
15510001257
Amount to withdraw
5000
Withdrawing: $5000.0
                Your balance is: $56125.0
```

When the user will choose option 3 from the customer section system will ask for account two numbers. One is from where the money will be withdrawn and another is to where the withdrawn money will be deposited.

Also the amount for transfer money from the user will be taken by the system. If the account number already exists then withdraw action will be performed and the updated balanced after withdrawal will be shown to the user.

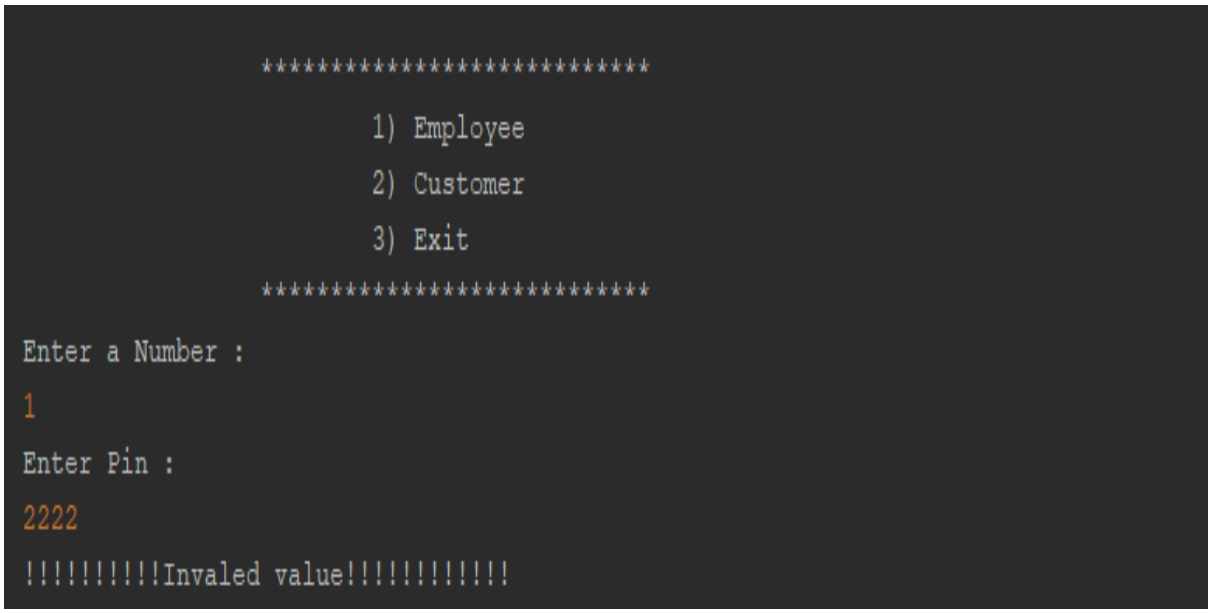


Here when user will press the option 4 which is Exit option then system will take user out from the customer panel and will led to the welcome panel section as shown below.

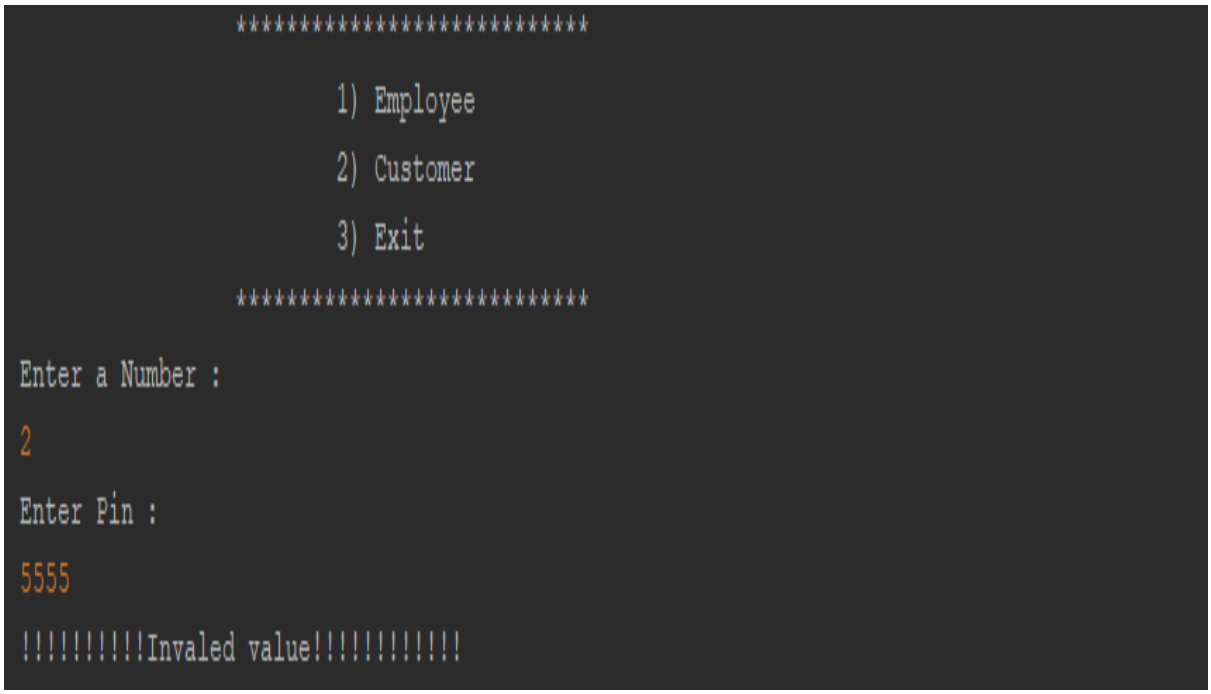


Below shown pictures are to show some invalid inputs given by the user which goes against of the system functionalities. For which the user has shown invalid value to the user. Now let us look at the invalid inputs which can stop the program to run.

1. When user will provide wrong password in employee panel instead of the right password which is 123 then a invalid value message will be shown by the system.

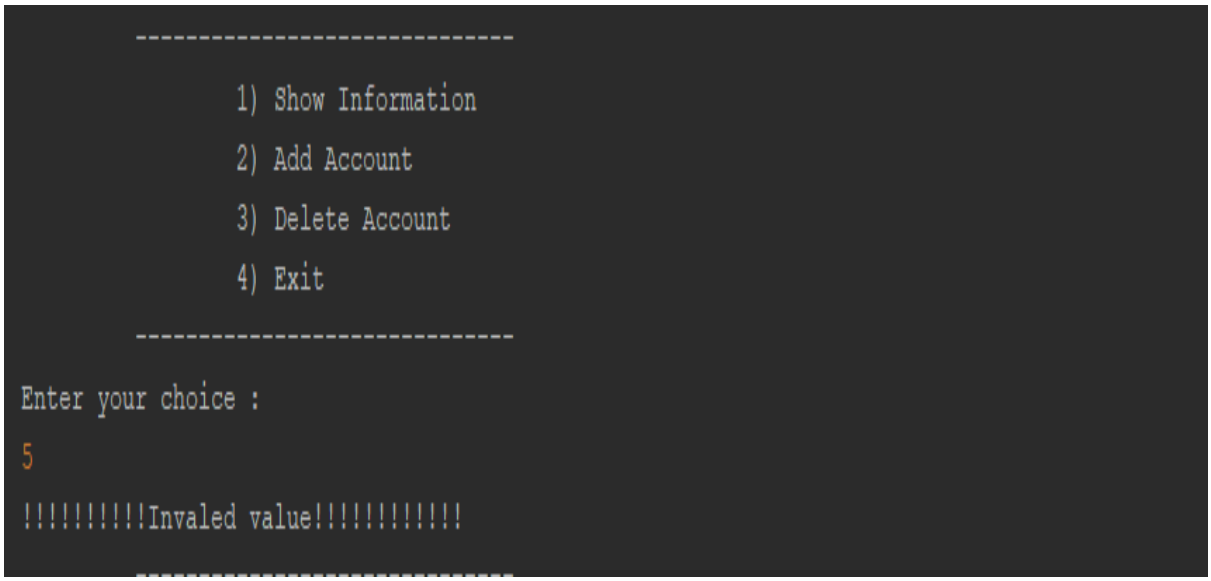


2. When user will provide wrong password in customer panel instead of the right password which is 201014076 then a invalid value message will be shown by the system.



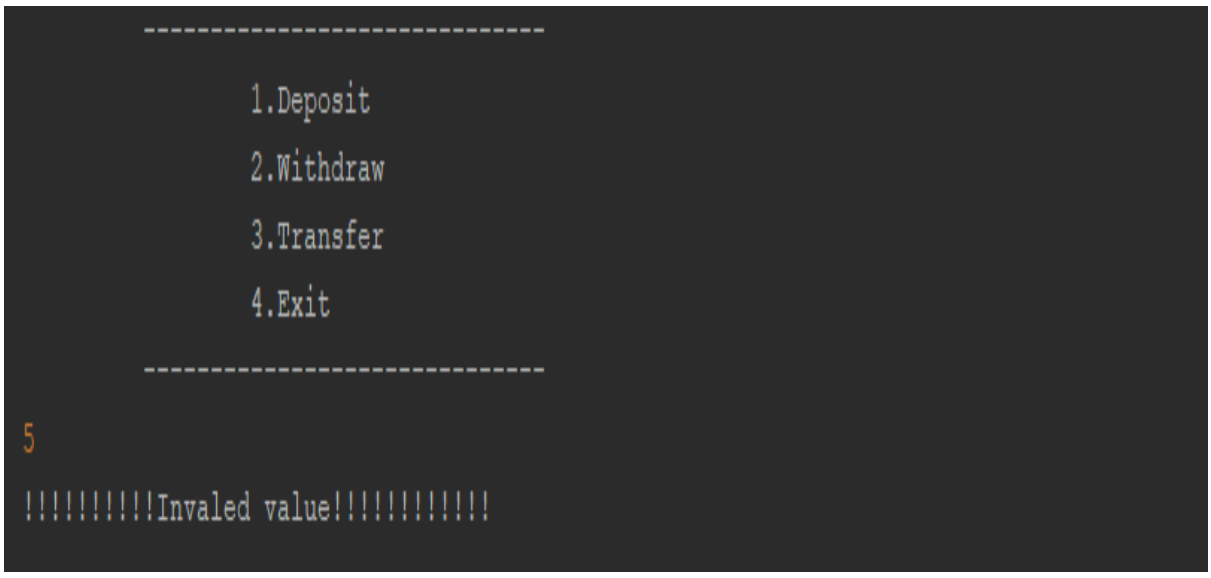
3. When user will press a option which does not exist in the employee menu then another invalid message will be shown to the user by the system.

Such as below we can see in the employee section there are 4 options available to access but the user has pressed option number 5 which actually does not there in the menu and so is not accessible.



4. Similarly, when user will press a option which does not exist in the customer menu then another invalid message will be shown to the user by the system.

Such as below we can see in the customer section there are 4 options available to access but the user has pressed option number 5 which actually does not there in the menu and so is not accessible.



Conclusion:

To sum up we can state that we have worked on the bank management system where we tried to apply the basic and main concepts of object-oriented programming using JAVA coding language. We built the system software with some basic features for both customer and employee. Both the ends can choose their desired options to perform different actions. The basic operations such as deposit money, withdraw money, transfer money by a bank account user.

Basic operations done by the bank employee such as crate bank account, delete bank account can be easily performed using this software. For employee and customer security we have kept password system which will ensure the safety and security concerns of both the end users. We have tried to keep the interface of software user friendly so that user can easily access it without any hazel.

There are still places to improve our program. We will fix the bugs of our system in near future to make the system more efficient for users. Besides this, we will try to add more features to our program and will update the system to make the user experience advanced for the end user. For this, we will try new concepts of OOP in an enriched way.

JAVA OOP concepts have been helpful for us to create the bank management software as it has many concepts which can be easily used to build this type of management software. Such encapsulation helped to hide the data from other classes of the code as per need. Abstraction and interface also helped to limit the use of specific functionalities in the entire program.

Inheritance has played a role to keep the code size limited as we could reuse the same code again and again in subclasses from the child classes. Polymorphism has been there to make sure the use of method overriding from one class to another class so that we can modify the same function in different classes for a different type of use. However, all the OOP concepts have been implemented together to build the structure of the software successfully.