**K L E F deemed to be University**

**Green Fields, Vaddeswaram, Guntur (Dt) :: 522502**

**Department of Computer Science Engineering**

**Project Based Lab Report**

**OBJECT ORIENTED PROGRAMMING**

**17 CS 2002**

****

**Submitted by**

VINISHA SHUKLA-170031397

K.SAI TARUN-170030656

SHAIK SHAHID AFRIDI-170031202

A.PRANATHI-170030069

**Under Guidance of**

Ms D.Madhavi

**II Year Engineering Course Work**

**2018-19**



**DECLARATION**

We declare that the project work entitled “**BANKING SYSTEM** “was carried out by us during 2018-19, and this work is not the same as that of any other and has not been submitted for award of any other degree/diploma

Signature of the Student

VINISHA SHUKLA-170031397

K.SAI TARUN-170030656

SHAIK SHAHID AFRIDI-170031202

A.PRANATHI-170030069

****

**K L E F deemed to be UNIVERSITY**

**Green fields,Vaddeswaram,Guntur Dist**.

**CERTIFICATE**

This is to certify that this project work entitled “**BANKING SYSTEM**” by is a bonafide work carried out by them in Department of Computer Science and Engineering.

Project supervisor Head of the Department

Ms D.Madhavi Mr Hari Kiran Vege

(Assistant professor) (Associate professor)

****

**ACKNOWLEDGEMENT**

We express my sincere gratitude to Ms D.Madhavi madam for encouraging and guiding us to undertake this project work. We express my deep sense of gratitude to **Hari Kiran Vege** sir and our beloved course lecturers of department for their encouragement.

Place:Vaddeswaram

VINISHA SHUKLA-170031397

K.SAI TARUN-170030656

SHAIK SHAHID AFRIDI-170031202

A.PRANATHI-170030069

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Page No** |
| 1 | ABSTRACT | 6 |
| 2 | INTRODUCTION | 7 |
| 3 | SOFTWARE AND HARDWARE REQUIREMENTS | 8 |
| 4 | FUNCTIONAL REQUIREMENTS | 9 |
| 5 | NON-FUNCTIONAL REQUIREMENTS | 10 |
| 6 | MODULE DESCRIPTION | 11 |
| 7 | SOURCE CODE | 12 |
| 8 | OUTPUT SCREENS | 26 |
| 9 | CONCLUSION | 30 |

**ABSTRACT**

Now-a-days, time is treated as money. No one is willing to spend their half a day’s time at

bank for transactions like balance enquiry, money transfer etc.  Complete banking system which is implemented on Java provides complete solution for banking related transactions and you can perform your most important banking chores from the comfort of your home.

The existing system all the transactions are carried out at manually. This is a time consuming process with lots of paper work involved. Moreover, security is also a major concern.

**Proposed System**

Complete Banking system enables bank users to access their accounts and general information about bank’s products and services from the comfort of your PC.This system keeps the record of the employees and their respective involvement with the customers.

**INTRODUCTION**

**About Banking in India:**

The existing banking structure in India, evolved over several decades, is elaborate and has been serving the credit and banking services needs of the economy. There are multiple layers in today's banking structure to cater to the specific and varied requirements of different customers and borrowers. The banking structure played a major role in the mobilisation of savings and promoting economic development Banking in India in the modern sense originated in the last decades of the 18th century. The among the first banks were Bank of Hindustan, which established in 1770 and liquidated in 1829-32; and General Bank of India, established 1786 but failed in 1791.The largest bank, and the oldest still in existence, is the

[State Bank of India](http://en.wikipedia.org/wiki/State_Bank_of_India" \o "State Bank of India). It originated as the [Bank of Calcutta](http://en.wikipedia.org/wiki/Bank_of_Calcutta)in June 1806.

The Indian banking sector is broadly classified into

[Scheduled banks](http://en.wikipedia.org/wiki/Scheduled_bank" \o "Scheduled bank) and non-scheduled banks. The scheduled banks are those which are included under the 2nd Schedule of the Reserve Bank of India Act, 1934. The scheduled banks are further classified into: nationalised banks;

[State Bank of India](http://en.wikipedia.org/wiki/State_Bank_of_India" \o "State Bank of India) and its associates;

[Regional Rural Banks](http://en.wikipedia.org/wiki/Regional_Rural_Bank" \o "Regional Rural Bank) (RRBs); foreign banks; and other Indian private sector banks. The term commercial banks refers to both scheduled and non-scheduled commercial banks which are regulated under the

[Banking Regulation Act, 1949](http://en.wikipedia.org/wiki/Banking_Regulation_Act,_1949" \o "Banking Regulation Act, 1949).

OUR PROJECT: We are going to implement an application which deals with maintaining banking activities like account creations, withdraws, loans and Administration activities. Initially, all the information about users and passbook details will be entered and maintained. This system will reduce manual work for maintaining records in files.

This system provides search facility to search for a particular user. Regular transactions which include book account creation, passbook issue, withdrawals, etc. and exceptional transactions that are related to loss of passbooks, damage of passbooks, etc. also will have to be handled by the system.

**SOFTWARE AND HARDWARE REQUIREMENTS**

**Definition of Hardware:**

Computer hardware is the collection of physical elements that constitutes a computer system. Computer hardware refers to the physical parts or components of a computer such as the monitor, mouse, keyboard, computer data storage, hard drive disk (HDD), system unit (graphic cards, sound cards, memory, motherboardand chips), etc. all of which are physical objects that can be touched.[1] In contrast, software is instructions that can be stored and run by hardware.

**Minimum Hardware Requirements for our Program:**

* PROCESSOR : Pentium IV processor or Greater
* RAM : 1 Giga Byte (GB) or Greater
* HARDDISK  : 100 Giga Byte (GB) or Greater
* Keyboard & Mouse
* MONITOR : Colour (For Best Result)
* Printer

**Definition of software:**

Computer software, or just software, is any set of machine-readable instructions that directs a computer's processor to perform specific operations. The term is used to contrast with computer hardware, the physical objects (processor and related devices) that carry out the instructions. Computer hardware and software require each other and neither can be realistically used without the other.

**Minimum Software Requirements:**

* Operating System : Windows 2000/ xp /7
* Java and NetBeans

**FUNCTIONAL REQUIREMENTS**

**Purpose:**

The objective of banking system is to give structural design to banking system. The project provides Functionality and flexibility to banking system such that one can operate that system easily and efficiently. This project also provides a complete set of solutions for some common and specific are as of work in the banks.

**Searching of the account is so easy:**

It is easy to search for the wished account by just typing the account number. Then it just shows all the details about that account like name and current balance in that account.

**Reduce the possibility to make mistake:**

Due to excessive amount of work the employers tend to do mistakes by manual form. Here the chance of mistake is minimum

**Deposit and Withdraw can be done very easily:**

The overall method is very easy and based on few steps. No huge amount of knowledge is needed to complete the task.

**ACCOUNT INFORMATION FILE:**

This file is used maintain customers account details like account number, name and their current balance in their account.

**SEARCH FILE:**

This file is used to search a particular account details when user gives a particular search name.

**DISPLAY RECORD:**

This record shows all the created account details of the customers like name account number and current balance in that account.

**DEPOSIT AND WITHDRAWAL :**

This can make the user to deposit amount to their account or withdraw amount from their account based On their requirement and depending on the current balance in their account.

**NON-FUNCTIONAL REQUIREMENTS**

INTERFACE:

An interface is a reference type in Java. It is similar to class. It is a collection of abstract

Methods. A class implements an interface, thereby inheriting the abstract methods of

The  interface. Along with abstract methods, an interface may also contain constants, default

methods, static methods, and nested types.

INHERITANCE:

Inheritance in java is a mechanism in which one object acquires all the properties and

behavior of parent object. The idea behind inheritance in java is that you can create new

class that are built upon existing classes.

STATIC KEYWORD:

All instances share the same copy of the variable. A class variable can be accessed directly with the class, without the need to create a instance.

ARRAYS:

Java provides a data structure, the array, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type

EXCEPTION HANDLING:

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occur the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

**MODULE DESCRIPTION**

The basic functions being performed by our system are:

* 1. MANAGEMENT OF AN ACCOUNT
  2. MANAGEMANT OF LOANS
  3. DEPOSIT ,WITHDRWAL AND CHECKING BALANCE
  4. EMPLOYEE DETAILS

These functions will be handled with the help of following sub functions: -

* Creating a file containing customer account details
* Searching and displaying an account
* Create loan types and searching Them
* Implements deposit, withdrawal and checking balance
* Giving the employee details

**SOURCE CODE**

import java.util.Scanner;

import java.text.NumberFormat;

//import java.util.Scanner;

import java.io.\*;

interface EMI{

double calculateMonthlyPayment(int loanAmount, int termInYears, double interestRate);

}

class payment implements EMI{

public double calculateMonthlyPayment(int loanAmount, int termInYears, double interestRate)

{

interestRate /= 100.0;

double monthlyRate = interestRate / 12.0;

int termInMonths = termInYears \* 12;

double monthlyPayment =(loanAmount\*monthlyRate) /(1-Math.pow(1+monthlyRate, -termInMonths));

return monthlyPayment;

}

}

class Menu{

int ab;

void display(){

System.out.println("\t\t========================================");

System.out.println("\t\t\tWelcome to SBI loan");

System.out.println("\t\t=======================================");

System.out.println("\t\t 1.Education Loan");

System.out.println("\t\t2.Home Loan");

System.out.println("\t\t3.Search");

System.out.println("\t\t4. Exit");

System.out.println("\n\n Enter your choice:");

Scanner in=new Scanner(System.in);

ab=in.nextInt();

}

}

class loan extends Menu{

payment p=new payment();

void Eduloan() {

try{

Scanner in=new Scanner(System.in);

File file=new File("Eduloan.txt");

// if file doesnt exists, then create it

if (!file.exists()) {

file.createNewFile();

}

FileWriter fw = new FileWriter("Eduloan.txt",true);

BufferedWriter bw = new BufferedWriter(fw);

System.out.println("Enter father name:");

String fname=in.next();

System.out.println("Enter father bank account nuumber:");

String facc=in.next();

System.out.println("Enter father income per annum:");

double fin=in.nextDouble();

if(fin>=100000)

{

System.out.println("Enter name:");

String sname=in.next();

System.out.println("Enter aadhar card no:");

String saad=in.next();

System.out.println("Enter college details:");

String sc=in.next();

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

int loanAmount =in.nextInt();

System.out.print("Enter loan term (in years): ");

int termInYears = in.nextInt();

double interestRate =20;

double monthlyPayment =p.calculateMonthlyPayment(loanAmount, termInYears, interestRate);

NumberFormat currencyFormat =NumberFormat.getCurrencyInstance();

NumberFormat interestFormat =NumberFormat.getPercentInstance();

System.out.println("Loan Amount: "+currencyFormat.format(loanAmount));

System.out.println("Loan Term: "+termInYears+" years");

System.out.println("Interest Rate: "+interestFormat.format(interestRate));

System.out.println("Monthly Payment: "+currencyFormat.format(monthlyPayment));

bw.write("Father name:"+fname+" ");

bw.write("Father acc No."+facc+" ");

bw.write("Student name:"+sname+" "+"Student aadhaar:"+saad+" ");

bw.write("Term in years:"+termInYears+" ");

bw.write("Loan amount:"+currencyFormat.format(loanAmount)+" ");

bw.write("Interest rate:"+interestFormat.format(interestRate)+" ");

bw.write("EMI:"+currencyFormat.format(monthlyPayment)+" ");

bw.newLine();

}

else

{System.out.println("You are not eligible for loan");}

bw.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

void Homeloan(){

try{

Scanner in=new Scanner(System.in);

File file1=new File("Home Loan.txt");

// if file doesnt exists, then create it

if (!file1.exists()) {

file1.createNewFile();

}

FileWriter fwh = new FileWriter("Home Loan.txt",true);

BufferedWriter bwh = new BufferedWriter(fwh);

System.out.println("Enter Applicants name:");

String bs=in.next();

System.out.println("Enter bank account nuumber:");

String br=in.next();

System.out.println("Enter aadhar card no:");

String bc=in.next();

System.out.println("Enter security for loan(eg;land documents):");

String o=in.next();

System.out.println("Enter your income per annum:") ;

double fd=in.nextDouble();

if(fd>=100000)

{

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

int loanAmount =in.nextInt();

System.out.print("Enter loan term (in years): ");

int termInYears = in.nextInt();

double interestRate =15;

double monthlyPayment = p.calculateMonthlyPayment(loanAmount, termInYears, interestRate);

NumberFormat currencyFormat =NumberFormat.getCurrencyInstance();

NumberFormat interestFormat =NumberFormat.getPercentInstance();

System.out.println("Loan Amount: "+ currencyFormat.format(loanAmount));

System.out.println("Loan Term: "+ termInYears+" years");

System.out.println("Interest Rate: "+interestFormat.format(interestRate));

System.out.println("Monthly Payment: "+ currencyFormat.format(monthlyPayment));

bwh.write("Applicant name:"+bs+" "+"Bank acc no:"+br+" "+"income:"+fd+" ");

bwh.write("Loan amount:"+currencyFormat.format(loanAmount)+" ");

bwh.write("Interest rate:"+interestFormat.format(interestRate)+" ");

bwh.write("EMI:"+currencyFormat.format(monthlyPayment)+" ");

bwh.newLine();

}

else

{System.out.println("You are not eligible for loan");}

bwh.close();}

catch (IOException e) {

e.printStackTrace();

}

}

void searchedu()throws Exception{

int i=0;

Scanner in =new Scanner(System.in);

System.out.println("enter the name you want to search:");

String name= in.next();

File file=new File("Eduloan.txt");

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) {

String lineFromFile = sc.nextLine();

if(lineFromFile.contains(name)) {

System.out.println("I found " +name+ " in Data base");

System.out.println(lineFromFile);

i=1;

break;

}

}

if(i==0){

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

}

}

void searchhome()throws Exception{

int i=0;

Scanner in =new Scanner(System.in);

System.out.println("enter the name you want to search:");

String name= in.next();

File file=new File("Home Loan.txt");

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) {

String lineFromFile = sc.nextLine();

if(lineFromFile.contains(name)) {

System.out.println("I found " +name+ " in Data base");

System.out.println(lineFromFile);

i=1;

break;

}

}

if(i==0){

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

}

}

}

public class BankApp {

public static void main(String[] args)throws Exception {

Scanner s = new Scanner(System.in);

Bank myBank = new Bank();

int user\_choice = 2;

do {

//display menu to user

//ask user for his choice and validate it (make sure it is between 1 and 6)

System.out.println();

System.out.println("1) Open a new bank account");

System.out.println("2) Deposit to a bank account");

System.out.println("3) Withdraw to bank account");

System.out.println("4) Print short account information");

System.out.println("5) Print the detailed account information including last transactions");

System.out.println("6) To manage loans of account");

System.out.println("7) Quit");

System.out.println();

System.out.print("Enter choice [1-7]: ");

user\_choice = s.nextInt();

switch (user\_choice) {

case 1: System.out.println("Enter a customer name");

String cn = s.next();

System.out.println("Enter a opening balance");

double d = s.nextDouble();

System.out.println("Account was created and it has the following number: " + myBank.openNewAccount(cn, d));

break;

case 2: System.out.println("Enter a account number");

int an = s.nextInt();

System.out.println("Enter a deposit amount");

double da = s.nextDouble();

myBank.depositTo(an, da);

break;

case 3: System.out.println("Enter a account number");

int acn = s.nextInt();

System.out.println("Enter a withdraw amount");

double wa = s.nextDouble();

myBank.withdrawFrom(acn, wa);

break;

case 4: System.out.println("Enter a account number");

int anum = s.nextInt();

myBank.printAccountInfo(anum);

break;

case 5: System.out.println("Enter a account number");

anum = s.nextInt();

myBank.printTransactionInfo(anum);

break;

case 6: Scanner sc = new Scanner(System.in);

loan l = new loan();

while(true){

l.display();

int ch = l.ab;

switch(ch){

case 1: l.Eduloan();

break;

case 2: l.Homeloan();

break;

case 4: break;

case 3: System.out.println("GET THE STATUS \n 1.Education loan \n2.Home loan");

int ch1=sc.nextInt();

if(ch1==1)

l.searchedu();

else

l.searchhome();

break;

default: System.out.println("Wrong choice");

}

if(ch==4)

break;

}

case 7:break;

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

}

}

while (user\_choice != 7);

}

static class Bank {

private BankAccount[] accounts; // all the bank accounts at this bank

private int numOfAccounts; // the number of bank accounts at this bank

//Constructor: A new Bank object initially doesn’t contain any accounts.

public Bank() {

accounts = new BankAccount[100];

numOfAccounts = 0;

}

// Creates a new bank account using the customer name and the opening balance given as parameters

// and returns the account number of this new account. It also adds this account into the account list

// of the Bank calling object.

public int openNewAccount(String customerName, double openingBalance) {

BankAccount b = new BankAccount(customerName, openingBalance);

accounts[numOfAccounts] = b;

numOfAccounts++;

return b.getAccountNum();

}

// Withdraws the given amount from the account whose account number is given. If the account is

// not available at the bank, it should print a message.

public void withdrawFrom(int accountNum, double amount) {

for (int i =0; i<numOfAccounts; i++) {

if (accountNum == accounts[i].getAccountNum() ) {

accounts[i].withdraw(amount);

System.out.println("Amount withdrawn successfully");

return;

}

}

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

}

// Deposits the given amount to the account whose account number is given. If the account is not

// available at the bank, it should print a message.

public void depositTo(int accountNum, double amount) {

for (int i =0; i<numOfAccounts; i++) {

if (accountNum == accounts[i].getAccountNum() ) {

accounts[i].deposit(amount);

System.out.println("Amount deposited successfully");

return;

}

}

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

}

// Prints the account number, the customer name and the balance of the bank account whose

// account number is given. If the account is not available at the bank, it should print a message.

public void printAccountInfo(int accountNum) {

for (int i =0; i<numOfAccounts; i++) {

if (accountNum == accounts[i].getAccountNum() ) {

System.out.println(accounts[i].getAccountInfo());

return;

}

}

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

}

public void printTransactionInfo(int accountNum) {

for (int i =0; i<numOfAccounts; i++) {

if (accountNum == accounts[i].getAccountNum() ) {

System.out.println(accounts[i].getAccountInfo());

System.out.println("Last transaction: " + accounts[i].getTransactionInfo(accounts[i].getNumberOfTransactions()-1));

return;

}

}

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

}

// Prints the account number, the customer number and the balance of the bank account whose

// account number is given, together with last n transactions on that account. If the account is not

// available at the bank, it should print a message.

public void printAccountInfo(int accountNum, int n) {

for (int i =0; i<numOfAccounts; i++) {

if (accountNum == accounts[i].getAccountNum() ) {

System.out.println(accounts[i].getAccountInfo());

System.out.println(accounts[i].getTransactionInfo(n));

return;

}

}

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

}

}

static class BankAccount{

private int accountNum;

private String customerName;

private double balance;

private double[] transactions;

private String[] transactionsSummary;

private int numOfTransactions;

private static int noOfAccounts=0;

public String getAccountInfo(){

return "Account number: " + accountNum + "\nCustomer Name: " + customerName + "\nBalance:" + balance +"\n";

}

public String getTransactionInfo(int n)

{

String transaction = transactionsSummary[n];

if (transaction == null) {

return "No transaction exists with that number.";

}

else {

return transaction;

}

}

public BankAccount(String abc, double xyz){

customerName = abc;

balance = xyz;

noOfAccounts ++;

accountNum = noOfAccounts;

transactions = new double[100];

transactionsSummary = new String[100];

transactions[0] = balance;

transactionsSummary[0] = "A balance of : $" + Double.toString(balance) + " was deposited.";

numOfTransactions = 1;

}

public int getAccountNum(){

return accountNum;

}

public int getNumberOfTransactions() {

return numOfTransactions;

}

public void deposit(double amount){

if (amount<=0) {

System.out.println("Amount to be deposited should be positive");

} else {

balance = balance + amount;

transactions[numOfTransactions] = amount;

transactionsSummary[numOfTransactions] = "$" + Double.toString(amount) + " was deposited.";

numOfTransactions++;

}

}

public void withdraw(double amount)

{

if (amount<=0){

System.out.println("Amount to be withdrawn should be positive");

}

else

{

if (balance < amount) {

System.out.println("Insufficient balance");

} else {

balance = balance - amount;

transactions[numOfTransactions] = amount;

transactionsSummary[numOfTransactions] = "$" + Double.toString(amount) + " was withdrawn.";

numOfTransactions++;

}

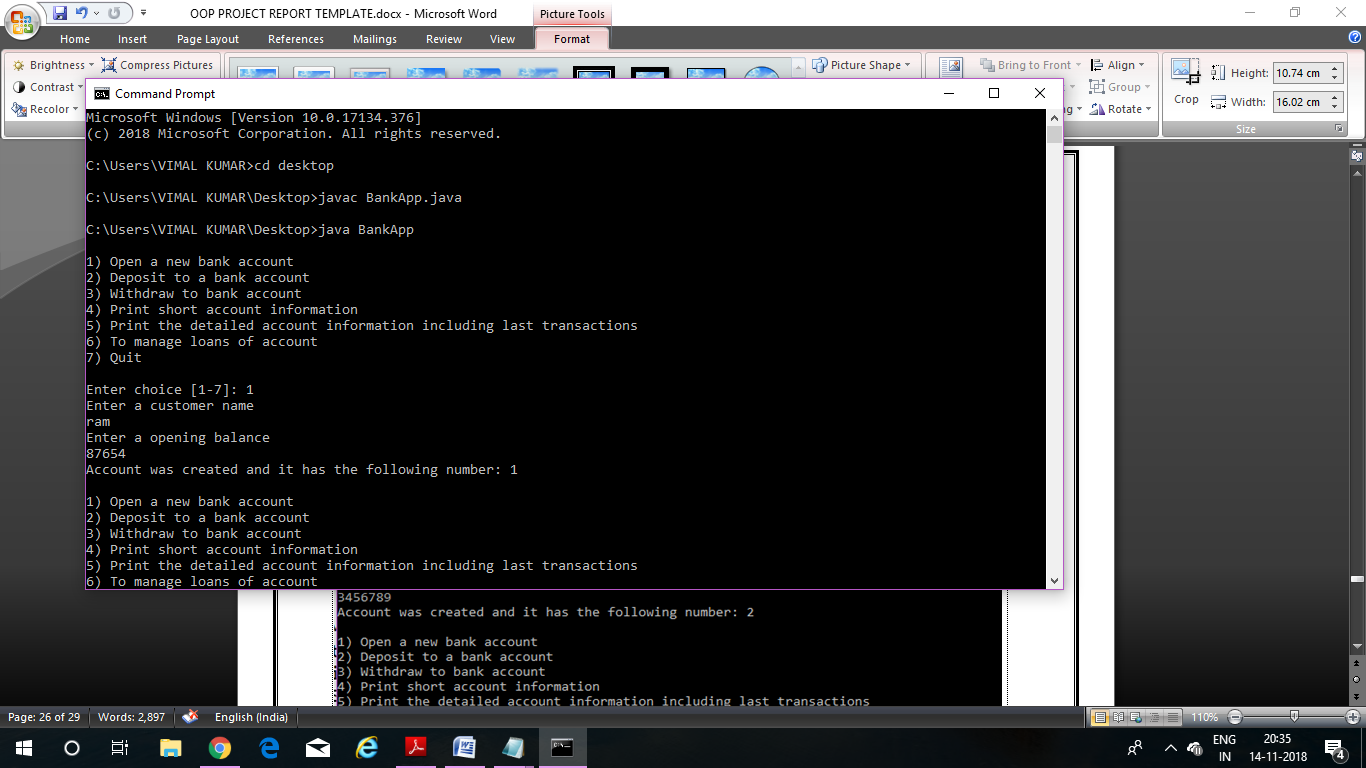
}

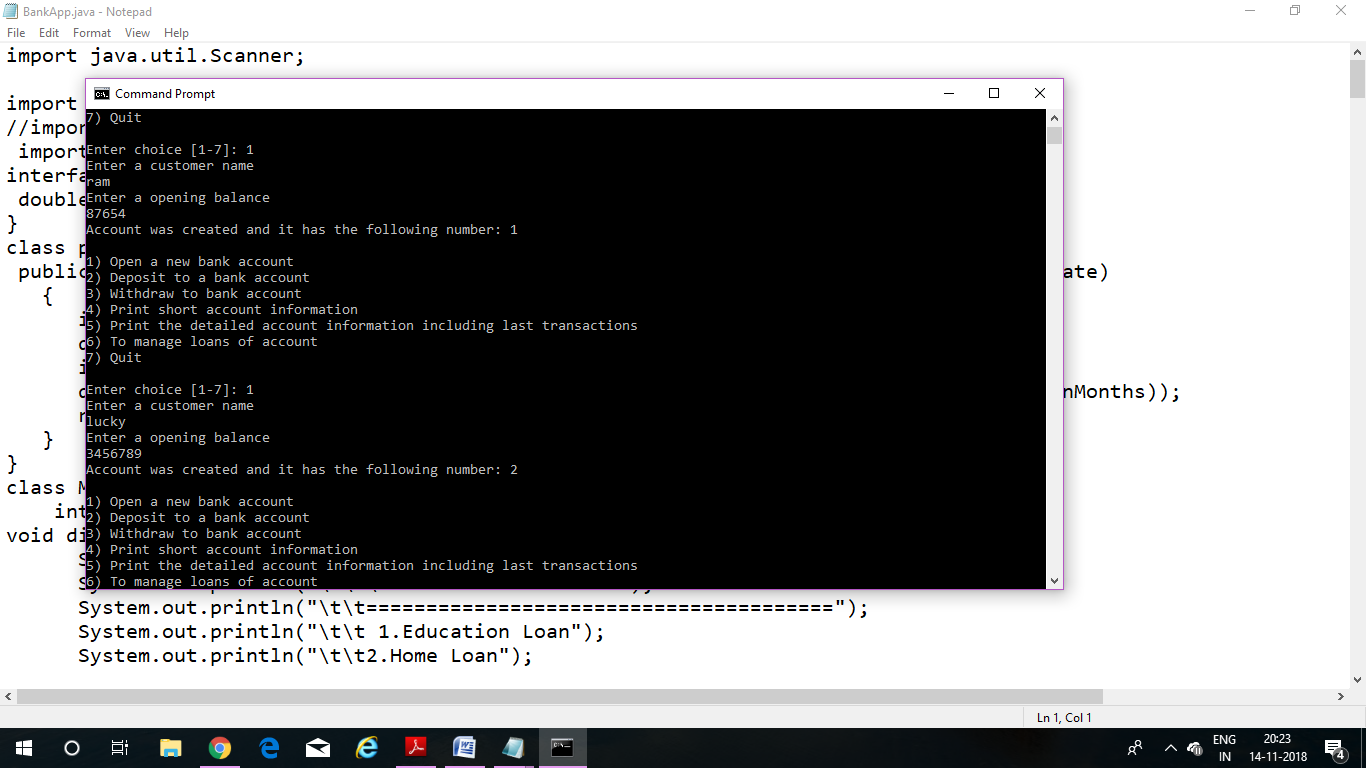
}

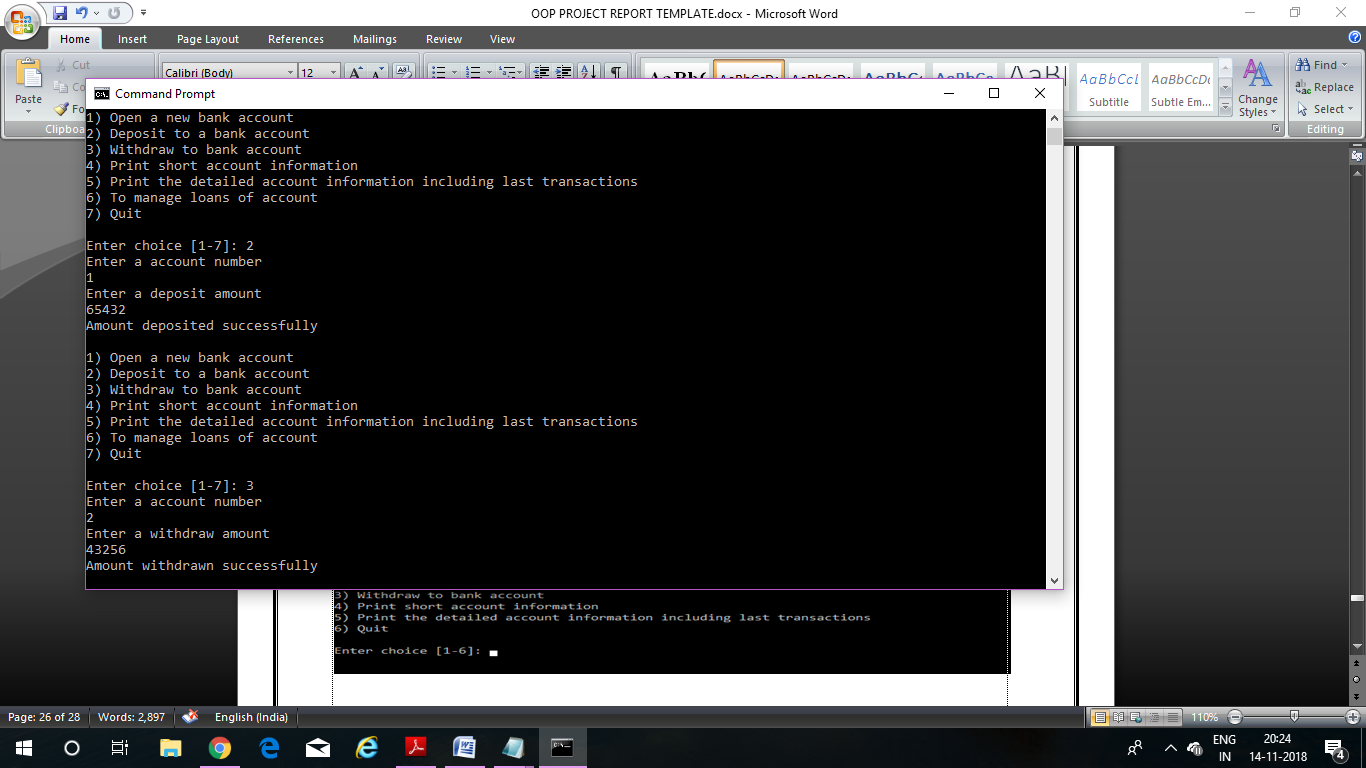
}//end of class

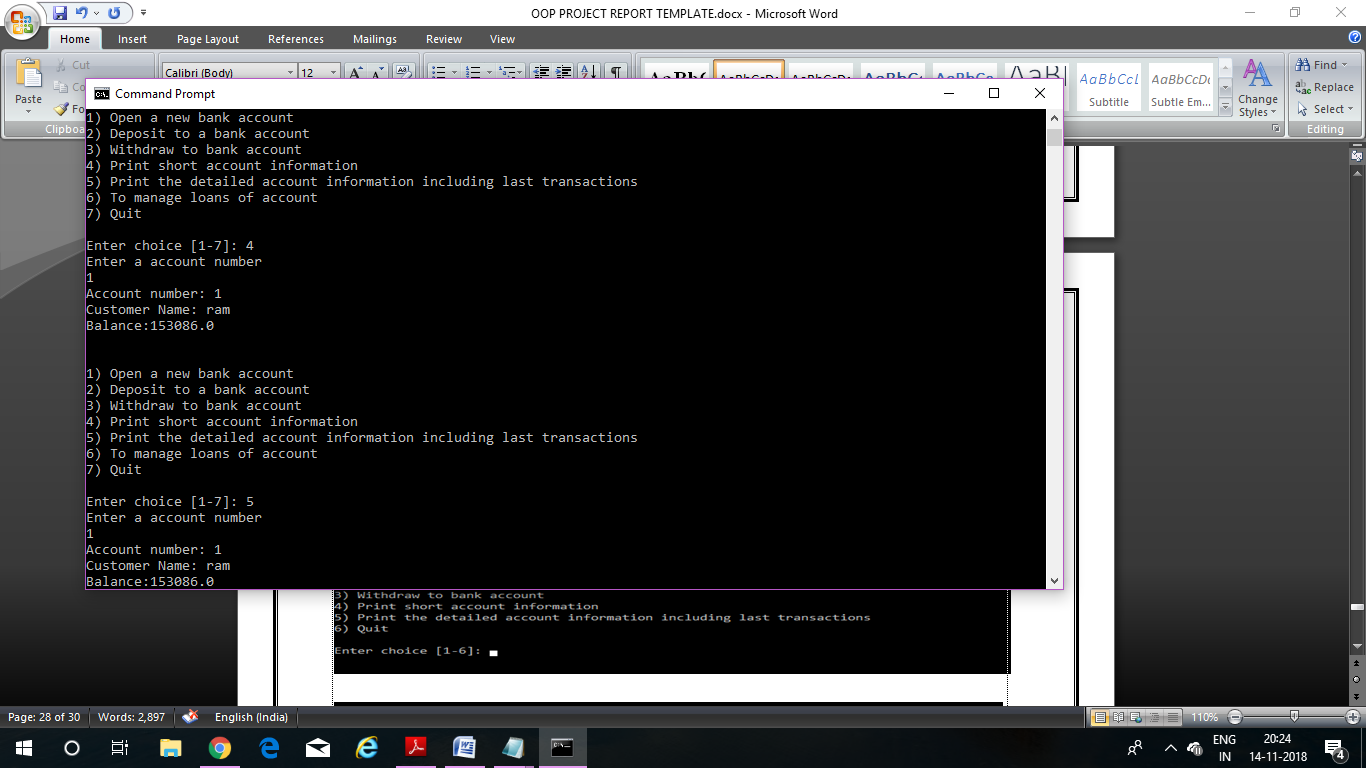
}

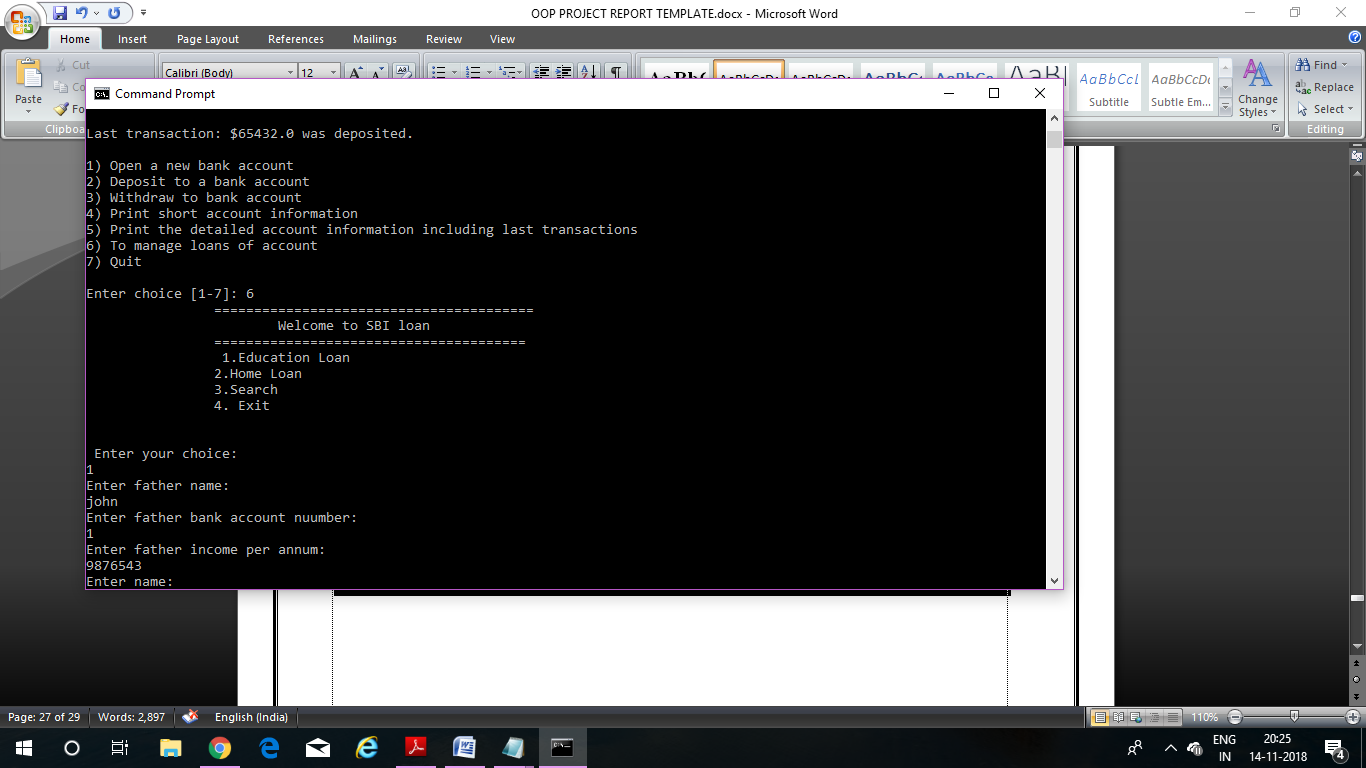
**OUTPUT SCREENS**

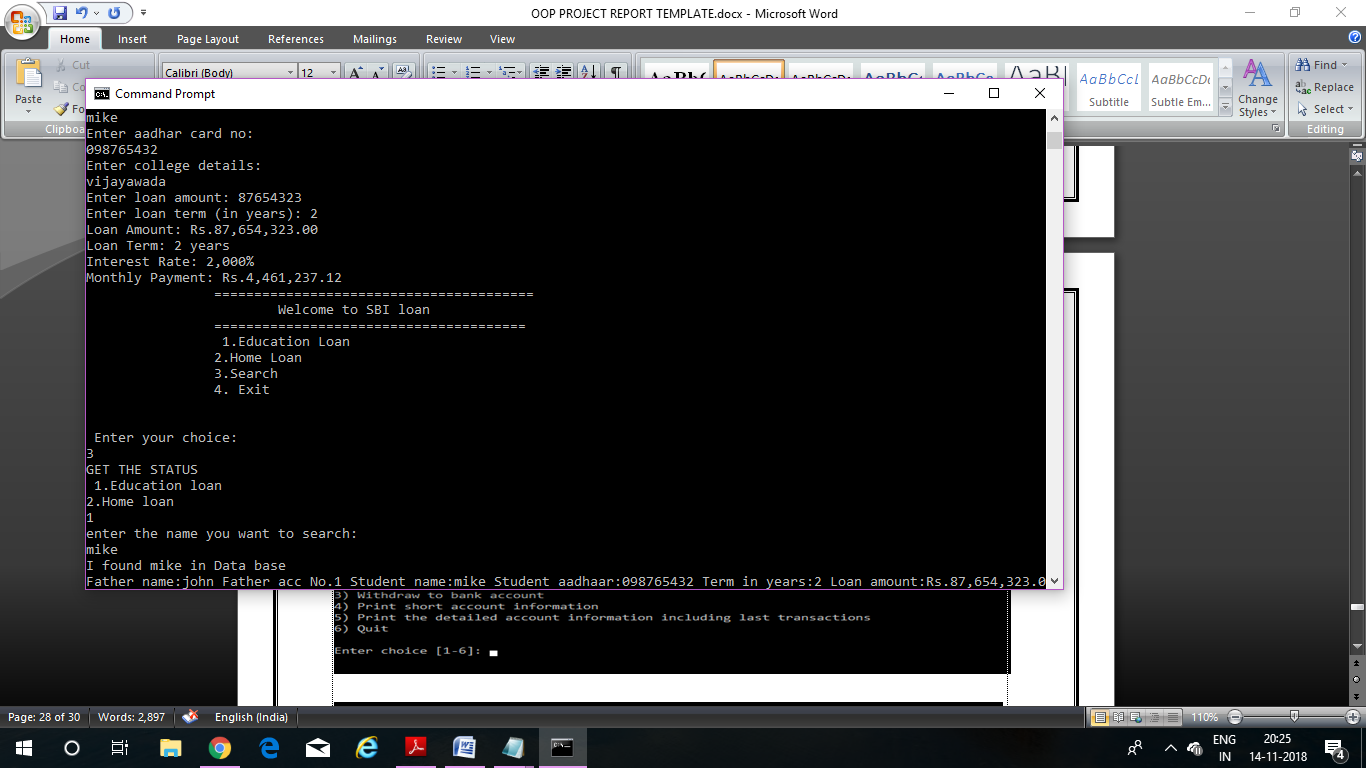
****

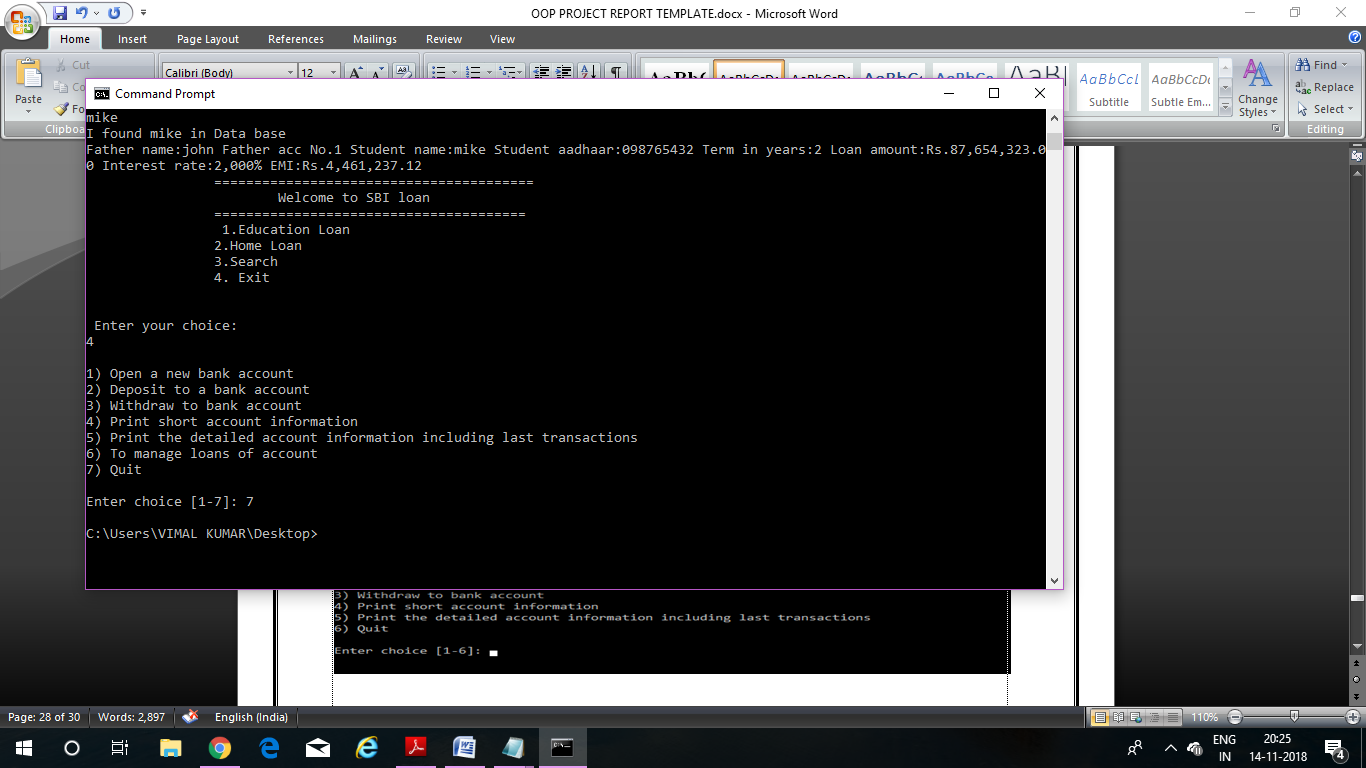
****

****

****

****

****

****

**CONCLUSION**

By using this program we can easily handle the banking system details. By this we can easily enquire the details of their account. Hence, the desired output as per the given task is obtained

REFERENCES

* Henry F Korth, Abraham Silberschatz, ‘’Database system concepts’’, McGrsw-Hill Internation editions, Computer Science Series (1991). Second Ed.
* Software Engineering ,McGrsw-Hill Internation editions.
* [www.slideshare.net](http://www.slideshare.net)
* [www.google.com](http://www.google.com)
* <http://www.gktoday.in/reference/brief-history-of-banking-in-india/>
* <http://en.wikipedia.org/wiki/Banking_in_India>