

Comprehensive Definition:

Create a program to simulate the working of ATM system. A user can have an account in the bank and the ATM asks for the user pin the user can withdraw at most Rs. 20,000 from the account which is linked with her bank system. Also, it is not possible to withdraw the amount greater than the total amount present in the linked bank account. The pin of the user's debit card should not be visible. Use appropriate concepts of OOP to handle the user created exceptions.

Java Program files.

Name of file: **User_portal**

File purpose: The main purpose of this java file is that it asks the user its current state. Like he is opening an account or already has an account.

Java file code:

```
import java.io.*;
import java.util.Scanner;
public class User_portal
{
    public static void main(String args[]) throws IOException
    {
        //handles an exception
        Welcome();
        user();//forming a static interface for user
    }

    static void user() throws IOException
    {
        int choice;
        Scanner sc=new Scanner(System.in);
        System.out.println("\n1. Sign Up");
        System.out.println("\n2. Login");
        System.out.println("\n3. Exit");
    }
}
```

```

        System.out.print("\nEnter Your Current State: ");
        choice=sc.nextInt();//asking the choice

        switch(choice)
        {
            case 1:
                New_User user = new New_User();
                user.data();//calling function of New_User class

            case 2:
                Old_User user1 = new Old_User();
                user1.operation();//calling function of Old_User class

            case 3:
                System.out.println("\n----- THANKYOU -----
\n");

                System.exit(0);

            default:
                System.out.println("\nEnter a valid Choice\n");

        }

        sc.close();
    }

    static void Welcome() {
        System.out.println(
            "-----
\n");

        System.out.println(" \t\t\t\t\t\t\tWELCOME TO KANISHA'S ATM ");
        System.out.println(
            "-----
\n");

        System.out.println(" \t\t\t\t\t\t\tTHE MOST EASIEST WAY FOR TRANSACTION");
        System.out.println(
            "-----
\n\n");

    }
}

```

Name of file: **New_User**

File purpose: When the user is new to the bank, he initially deposits some amount of money into it's account by creating it with is class. Where he has to choose for a PIN no. which is one of the identities of user.

Java file code:

```
import java.io.*;
import java.util.Scanner;
public class New_User
{
    String User_Name;
    private String pin;//for privary purpose only child can activate this dat
a
    String Bank_Name;
    private String Card_Num;//same for the card Number
    int Balance;

    Scanner sc=new Scanner(System.in);

    void data() throws IOException
    {
        File f1 = new File("Record.txt");//locating a pointer at the start of
a file
        FileWriter f = new FileWriter(f1, true); // Here due to lack of charc
ter forming error can't use this
        PrintWriter pw = new PrintWriter(f);// It is comparitively reliable

        System.out.print("\n-> Full Name: ");
        User_Name= sc.nextLine();

        System.out.print("\n-> Bank Name: ");
        Bank_Name=sc.nextLine();

        System.out.print("\n-> Amount to be Deposited: ");
        Balance=sc.nextInt();
    }
}
```

```

String s="\n-> Enter Your Pin number: ";
Pin_Hack p =new Pin_Hack();//calling a class for scanning PIN
pin=p.Pin(s);
pin=pin.trim();//trimming for safety purpose

String pin2=pin;//storing for checking again

System.out.println();
s="\n-> Please confirm Your Pin: ";
pin=p.Pin(s);
pin=pin.trim();

if(pin2.equals(pin)==true)//if they are same
{
    System.out.println("\n----- Pin Confirmed ----- \n");
}

else
{
    System.out.println("\n----- Please Enter Pin correctly ----
----- \n");
    s="\n-> Enter Your Pin again: ";
    pin=p.Pin(s);
    pin=pin.trim();//Giving a chance
}

int num =(int)(Math.random()*1000000000);//generating a random no. fo
r Card_Number
Card_Num = Integer.toString(num);

System.out.println("\n-----
- Please Note Down Your Alloted Card Number ----- \n");
System.out.println("\n Your Card Number is: "+Card_Num);

pw.write("\nCard Number: "+Card_Num);
pw.write("\nPin: "+pin);
pw.write("\n"+User_Name+"'s Balance: " +Balance);
pw.write("\nUser Name: " +User_Name);
pw.write("\nBank Name: " +Bank_Name);//writing whole data into the fi
le called Record

pw.write("\n***** \n");

pw.flush();//to clear any buffer if present

```

```

        pw.close();//closing the file
    }

}

```

Name of file: **Old_User**

File purpose: When the user already has a bank account, and comes to the ATM then he is asked for the transactions he need to perform i.e. Transfer of money, depositing money, withdrawing money and Enquiry of Balance and thus one of its method calls the respective class using object.

Java file code:

```

import java.io.*;
import java.util.Scanner;
public class Old_User
{
    int choice;
    Scanner sc=new Scanner(System.in);

    void operation() throws IOException
    {
        System.out.println("\n***** Select Your Transaction *****
*****\n");
        System.out.print("\n1. Transfer \t\t\t 2. Balance Enquiry");
        System.out.println("\n\n3. Deposit Money \t\t 4. Withdraw Money");
        System.out.println("\n5. Exit");
        System.out.print("\n-> Enter Your Choice: "); //asking choice
        choice=sc.nextInt();

        switch(choice)
        {
            case 1:

```

```

        System.out.println("\n----- Welcome to Transfer Mode -
-----\n");
        Transfer t = new Transfer();
        t.transfer_amt();//calling specific function
        break;

    case 2:
        System.out.println("\n-----
- Welcome to Balance Inquiry Mode -----\n");
        Balance_Inquiry bi =new Balance_Inquiry();
        bi.inquiry();//calling its function
        break;

    case 3:
        System.out.println("\n----- Welcome to Deposit Mode -
-----\n");
        Deposit b=new Deposit();
        b.deposit_mon();
        break;

    case 4:
        System.out.println("\n----- Welcome to Withdraw Mode-
-----\n");
        Withdraw w=new Withdraw();
        w.withdraw_mon();
        break;

    case 5:
        System.exit(0);
        break;//exiting the module
    default:
        System.out.println("Invalid Choice");
        //user();

    }

}

}

}

```

Name of file: **Transfer**

File purpose: Here it first checks the account holder's balance using File pointer, while traversing the whole file finds the correct place and stores the value similarly the one whose account money is deposited is found by the same process into this and balance is modulated accordingly.

Java file code:

```
import java.io.*;
import java.util.Scanner;
public class Transfer extends Old_User
{
    Scanner sc=new Scanner(System.in);
    int flag=0;
    String pin,name,name1;
    String number=new String();
    String temp1="",temp2="";
    int present_amt_acc1,trans_amt,amt,present_amt_acc2,curr_bal_acc2,curr_bal_acc1;

    String Old_data="",New_data="";

    void transfer_amt() throws IOException
    {
        int val=validate();//calling a function to validate the state of user

        if(val==1)
        {
            System.out.print("\n-
> Enter the amount to be Transferred (in Rs.):  ");
            amt=sc.nextInt();//while such user exist, asking the amount to be transferred

            String arr[]=temp1.split(":");//here using split() splitting the string i.e. balance
            temp1=arr[1].trim();//trimming for safety purpose
            present_amt_acc1=Integer.parseInt(temp1);//converting to int

            if(present_amt_acc1>=amt)
```

```

        {
            System.out.println("\n----- Transaction is possible -----
-----\n");
            transaction(amt); //if lunpsum amount is present then only it is g
oing further
        }

        else
        {
            System.out.println("\n-----
- Balance is not sufficient -----");
            operation(); //returns to main menu
        }

    }
    operation();
}

int validate() throws IOException
{
    File f1 = new File("Record.txt");
    BufferedReader br = new BufferedReader(new FileReader(f1)); //opening a fi
le in read mode

    System.out.println("\n----- Personal Details -----
\n");

    System.out.print("\n-> Please Enter Your Card Number: ");
    number=sc.nextLine().trim();

    System.out.print("\n-> Card Holder's Name: ");
    name=sc.nextLine().trim(); //scanning details

    String initial = br.readLine(); //here it stores the first line of file an
d then goes further

    while(initial != null && flag==0)
    {
        if(("Card Number: "+number).equals(initial)) //checking with the name
        {
            flag=1;
            initial=br.readLine();
            temp2=initial; //storing pin
            temp1=br.readLine(); //storing current amount

```



```

    }

    else
    {
        initial=br.readLine();
    }

}
br.close();
if(flag==0)
{
    System.out.println("\n----- No such Card Number exist -----
-----");
    return 0;
}

else
{
    String s="\n-> Enter Your Pin number: ";
    Pin_Hack p1 =new Pin_Hack();
    pin=p1.Pin(s);
    pin=pin.trim();//asking pin

    if(("Pin: "+pin).equals(temp2))//if it gets equal to the data present
    in file transaction is possible
    {
        System.out.println("\n-----
- Pin Matched Successfully ----- \n");
        return 1;
    }

    else
    {
        System.out.println("\n-----
- Please enter your Pin again ----- \n");
        s="\n-> Enter Your Pin: ";
        pin=p1.Pin(s);
        pin=pin.trim();//giving a chance

        if(("Pin: "+pin).equals(temp2))
        {
            System.out.println("\n-----
- Pin Matched Successfully ----- \n");
            return 1;
        }
    }
}

```

```

        else
        {
            System.out.println("\n----- Incorrect Pin -
-----\n");
            return 0;
        }
    }
}

}

void transaction(int amt) throws IOException
{
    int flag=0;
    String card_num2;
    sc.nextLine();
    System.out.print("\n-
> Enter in which Account you want to transfer Money(in Rs.):  ");
    card_num2=sc.nextLine();

    System.out.print("\n-> Card Holder's Name:  ");
    name1=sc.nextLine();//asking details of the person in which it should be
transferred

    File f1 = new File("Record.txt");
    BufferedReader br = new BufferedReader(new FileReader(f1));//110 -
> 100 as a string read

    String initial = br.readLine();//again reading the first line

    while(initial != null && flag==0)
    {
        if(("Card Number: "+card_num2).equals(initial))
        {
            flag=1;
            initial=br.readLine();
            temp1=br.readLine();//storing balance
        }

        else
        {

```

```

        initial=br.readLine();
    }
}
br.close();//closing file

if(flag==0)
{
    System.out.println("\n-----
- No such Card Number exist -----\n");
    operation();//if not returns to the last menu
}

else
{
    String arr[]=temp1.split(":");

    temp1=arr[1].trim();

    present_amt_acc2=Integer.parseInt(temp1);//for the person in which am
t adds

    curr_bal_acc2 = amt + present_amt_acc2;

    f1 = new File("Record.txt");
    br = new BufferedReader(new FileReader(f1));
    String hack=br.readLine();//again opening the file

    while(hack!=null)
    {
        Old_data=Old_data+hack+System.lineSeparator();//storing whole fil
e data into the string

        hack=br.readLine();
    }

    String ex1=name1+"'s Balance: "+temp1;//forming string to be replaced
    String ex2=name1+"'s Balance: "+curr_bal_acc2;
    New_data=Old_data.replace(ex1,ex2);

    PrintWriter pw = new PrintWriter("Record.txt");// 100 -> 100

    pw.write(New_data);//writing whole data again

    br.close();
    pw.flush();

```

```

        pw.close();//closing all pointers

    }

    // The person who transfers the money
    f1 = new File("Record.txt");
    br = new BufferedReader(new FileReader(f1));//110 -> 100 as a string read

    initial = br.readLine();
    int flag1=0;

    while(initial != null && flag1==0)
    {
        if(("Card Number: "+number).equals(initial))
        {
            flag1=1;
            initial=br.readLine();
            temp2=br.readLine();

        }

        else
        {
            initial=br.readLine();
        }
    }
    br.close();
    if(flag1==0)
    {
        System.out.println("\n-----
- No such Card Number exist -----");
        operation();
    }

    else
    {
        String arr[]=temp2.split(":");
        temp2=arr[1].trim();
        present_amt_acc1=Integer.parseInt(temp2);//holder

        trans_amt=present_amt_acc1-amt;//modifying its balance

        File f2 = new File("Record.txt");

```

```

        br = new BufferedReader(new FileReader(f2)); //110 -
> 100 as a string read

        String Old_data2="";
        String hack=br.readLine();

        while(hack!=null)
        {
            Old_data2=Old_data2+hack+System.lineSeparator();

            hack=br.readLine();
        }

        String ex1=name+"'s Balance: "+temp2;
        String ex2=name+"'s Balance: "+trans_amt;
        New_data=Old_data2.replace(ex1,ex2); //replacing the string after find
ing

        PrintWriter pw = new PrintWriter("Record.txt"); // 100 -> 100

        pw.print(New_data); //overwriting whole data into the file

        br.close();
        pw.flush();
        pw.close();
        //closing all pointers
        System.out.println("\n----- Transaction Successfull --
-----\n");
        operation();
    }
}
}

```

Name of file: **Balance_Inquiry**

File purpose: Here the user can fetch it's current Account Balance just by entering certain details.

Java file code:

```
import java.io.*;
public class Balance_Inquiry extends Transfer
{
    Balance_Inquiry() throws IOException
    {

    }

    void inquiry() throws IOException
    {
        int flag=0,p2=0;

        String Bank_name="",Balance="";
        File f1 = new File("Record.txt");

        BufferedReader br = new BufferedReader(new FileReader(f1));//110 -
> 100 as a string read

        System.out.print("\n-> Please Enter Your Card Number: ");
        number=sc.nextLine();

        System.out.print("\n-> Card Holder's Name: ");
        name=sc.nextLine();//scanning details

        String initial = br.readLine();

        while(initial != null && flag==0)
        {
            if(("Card Number: "+number).equals(initial))
            {
                flag=1;
                initial=br.readLine();
                temp1=initial;//storing pin
                Balance=br.readLine();//stores name
                Bank_name=br.readLine();//stores bank name
                Bank_name=br.readLine();//stores bank name
            }

            else
            {
                initial=br.readLine();
            }
        }
    }
}
```

```

    }

    br.close();
    if(flag==0)
    {
        System.out.println("\n-----
- No such Card Number exist -----\n");
        operation();
    }

    else
    {
        String s="\n-> Enter Your Pin: ";
        Pin_Hack p1 =new Pin_Hack();
        pin=p1.Pin(s);
        pin=pin.trim();//validating pin

        if(("Pin: "+pin).equals(temp1))
        {
            System.out.println("\n-----
- Pin Matched Successfully -----\n");
            p2=1;
        }

        else
        {
            System.out.println("\n-----
- Please enter your Pin again -----\n");
            s="\n-> Enter Your Pin: ";
            pin=p1.Pin(s);
            pin=pin.trim();//giving a chance

            if(("Pin: "+pin).equals(temp1))
            {
                System.out.println("\n-----
- Pin Matched Successfully -----\n");
                p2=1;
            }

            else
            {
                System.out.println("\n----- Incorrect Pin -
-----\n");
                operation();
            }
        }
    }
}

```

```

    }

    if(p2==1)
    {
        Bank_name=Bank_name.substring(11);//removing initial data

        System.out.println("\n*Details: \n");
        System.out.println("\nAccount Holder: "+name);
        System.out.println("\nCard Number: "+number);
        System.out.println("\nBank Name: "+Bank_name);

        String arr[]=Balance.split(":");
        Balance=arr[1].trim();
        present_amt_acc1=Integer.parseInt(Balance);

        System.out.println("\nYour Current Balance is: "+present_amt_acc1);
    };//printing details
    operation();
}

}

}
}

```

Name of file: Deposit

File purpose: Here when the user wants to Deposit money into its account, first he/she needs to verify using their PIN and then the amount is modulated accordingly.

Java file code:

```

import java.io.*;
public class Deposit extends Transfer

```



```

{
    Deposit() throws IOException
    {

    }
    void deposit_mon() throws IOException
    {
        int x= validate();

        if(x==1)
        {
            System.out.print("\n-
> Enter the amount to be Deposited (in Rs.): \n\n");
            amt=sc.nextInt();//scanning amount to be deposited

            String arr[]=temp1.split(":");
            temp1=arr[1].trim();
            present_amt_acc1=Integer.parseInt(temp1);

            if(amt>0)//if fixes this constrainnt
            {
                int res=deposit(amt);//asking status

                if(res==1)
                {
                    System.out.println(amt +" Rs. Deposited successfully");//prin
ting status
                    operation();
                }

                else
                {
                    operation();
                }
            }

            else
            {
                System.out.println("\n-----
- Can't Deposit This Amount ----- \n");
                operation();
            }
        }

        else

```

```

    {

    }

}

int deposit(int amt) throws IOException
{
    int flag=0;

    File f1 = new File("Record.txt");
    BufferedReader br = new BufferedReader(new FileReader(f1)); //110 -
> 100 as a string read

    String initial = br.readLine(); //taking data line by line

    while(initial != null && flag==0)
    {
        if(("Card Number: "+number).equals(initial))
        {
            flag=1;
            initial=br.readLine();
            temp1=br.readLine(); //balance
        }

        else
        {
            initial=br.readLine();
        }
    }
    br.close(); //closing file

    if(flag==0)
    {
        System.out.println("\n-----
- No such Card Number exist ----- \n");
        return 0;
    }

    else
    {
        String arr[]=temp1.split(":");

        temp1=arr[1].trim();

        present_amt_acc2=Integer.parseInt(temp1); //converting balance to int
    }
}

```

```

        int curr_bal_acc2 = amt + present_amt_acc2;

        f1 = new File("Record.txt");
        br = new BufferedReader(new FileReader(f1)); //opening a file

        String hack=br.readLine();

        String Old_data=""; //assigning it to null
        while(hack!=null)
        {
            Old_data=Old_data+hack+System.lineSeparator(); //storing data which
            //even takes care of \r and \n

            hack=br.readLine();
        }

        String ex1=name+"'s Balance: "+temp1;
        String ex2=name+"'s Balance: "+curr_bal_acc2;
        New_data=Old_data.replace(ex1,ex2); //replacing data from old to new

        PrintWriter pw = new PrintWriter("Record.txt");

        pw.write(New_data); //writing whole data

        br.close();
        pw.flush();
        pw.close();
        //closing all file pointers
        return 1;
    }
}

```

Name of file: **Withdraw**

File purpose: Here the user can withdraw maximum 20,000 from his account and validating with some constraints it works accordingly for modulating balance

Java file code:

```
import java.io.*;
public class Withdraw extends Transfer
{
    Withdraw() throws IOException
    {
    }

    void withdraw_mon() throws IOException
    {
        int x= validate();
        if(x==1)
        {
            System.out.println("\n-> Enter the amount to be Withdrawn: \n");
            amt=sc.nextInt();

            String arr[]=temp1.split(":");
            temp1=arr[1].trim();
            present_amt_acc1=Integer.parseInt(temp1);//converting to int

            if(amt>20000)//it should not exceed
            {
                System.out.println("\n-----
- More than 20,000 Rs. can't be Withdrawn ----- \n");
                operation();
            }

            else
            {
                int status=withdraw();

                if(status==1)
                {
                    System.out.println(amt +" Rs. successfully withdrawn\n");//gi
ving the update
                    operation();
                }
            }
        }
    }
}
```

```

        else
        {
            operation();
        }
    }
}

int withdraw() throws IOException
{
    int flag1=0;
    sc.nextLine();

    File f1 = new File("Record.txt");
    BufferedReader br = new BufferedReader(new FileReader(f1)); //110 -
    > 100 as a string read

    String initial = br.readLine(); //storing first line of file

    while(initial != null && flag1==0)
    {
        if(("Card Number: "+number).equals(initial))
        {
            flag1=1;
            initial=br.readLine();
            temp2=br.readLine(); //storing balance

        }

        else
        {
            initial=br.readLine();
        }
    }
    br.close(); //closing file
    if(flag1==0)
    {
        System.out.println("\n-----
- No such Card Number exist ----- \n");
        return 0;
    }

    else
    {
        String arr[]=temp2.split(":");

```

```

        temp2=arr[1].trim();
        int present_amt_acc1=Integer.parseInt(temp2);//converting to int

        int trans_amt=present_amt_acc1-amt;//modifying balance

        File f2 = new File("Record.txt");
        br = new BufferedReader(new FileReader(f2));//110 -
> 100 as a string read

        String Old_data2="";//initialising with null
        String hack=br.readLine();

        while(hack!=null)
        {
            Old_data2=Old_data2+hack+System.lineSeparator();//storing whole f
ile data into the string

            hack=br.readLine();
        }

        String ex1=name+"'s Balance: "+temp2;
        String ex2=name+"'s Balance: "+trans_amt;//forming strings to be chan
ged

        New_data=Old_data2.replace(ex1,ex2);//replacing line

        PrintWriter pw = new PrintWriter("Record.txt");// 100 -> 100

        pw.print(New_data);
        br.close();
        pw.flush();
        pw.close();
        //closing all files
        return 1;

    }

}

```

Name of file: **Password**

File purpose: Using inbuilt functions of Thread and using readPassword() password is taken from the console and simultaneously "*" is printed while the original PIN is stored in file.

Java file code:

```
import java.io.*;
public class Password {

    public static String readPassword (String prompt)
    {
        EraserThread et = new EraserThread(prompt); //using inbuilt class which erases the thread
        Thread mask1 = new Thread(et);
        mask1.start(); //calls run()

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        String password = ""; //initialing password

        try
        {
            password = in.readLine();
        }
        catch (IOException inx)
        {
            System.out.println("Interrupted");
        }

        et.stopMasking(); //while it ends

        return password;
    }
}
```

Name of file: **Pin_Hack**

File purpose: It is used specifically for handling simultaneous behavior of threads. Where some inbuilt functions are used.

Java file code:

```
class EraserThread implements Runnable//inbuilt classes
{
    private boolean stop;//to stop while password ends

    public EraserThread(String prompt)
    {
        System.out.print(prompt);
    }

    public void run () //called by start()
    {
        stop = true;//changing status
        while (stop)
        {
            System.out.print("\010*");
            try
            {
                Thread.currentThread();//calling currentThread()
                Thread.sleep(1);//using inbuilt functions here sleeping for 1ms
            }
            catch (InterruptedException ie)
            {
                System.out.println("Interrupted");
            }
        }
    }

    public void stopMasking()
    {
        this.stop = false;//while password ends
    }
}

class Pin_Hack
{
    String Pin(String s)
    {
        String password = Password.readPassword(s);
    }
}
```



```
    return password;  
}  
}
```

Text Files:

Name of file: **Record.txt**

File purpose: It maintains the whole ATM record and transactions followed.

Content:

Card Number: 401283854

Pin: 1234

Kanisha Shah's Balance: 3900

User Name: Kanisha Shah

Bank Name: SBI

Card Number: 560241145

Pin: 4512

Kahan Sheth's Balance: 5000

User Name: Kahan Sheth

Bank Name: SBI

19BCE253

Card Number: 31615769

Pin: 12345

Nivan Sheth's Balance: 6000

User Name: Nivan Sheth

Bank Name: SBI

Card Number: 250053751

Pin: 1452

Kartik Vaghela's Balance: 15487

User Name: Kartik Vaghela

Bank Name: Dena

Card Number: 976760542

Pin: 7845

Vivek Shah's Balance: 478512

User Name: Vivek Shah

Bank Name: Nutan

Card Number: 714296779

Pin: 4875

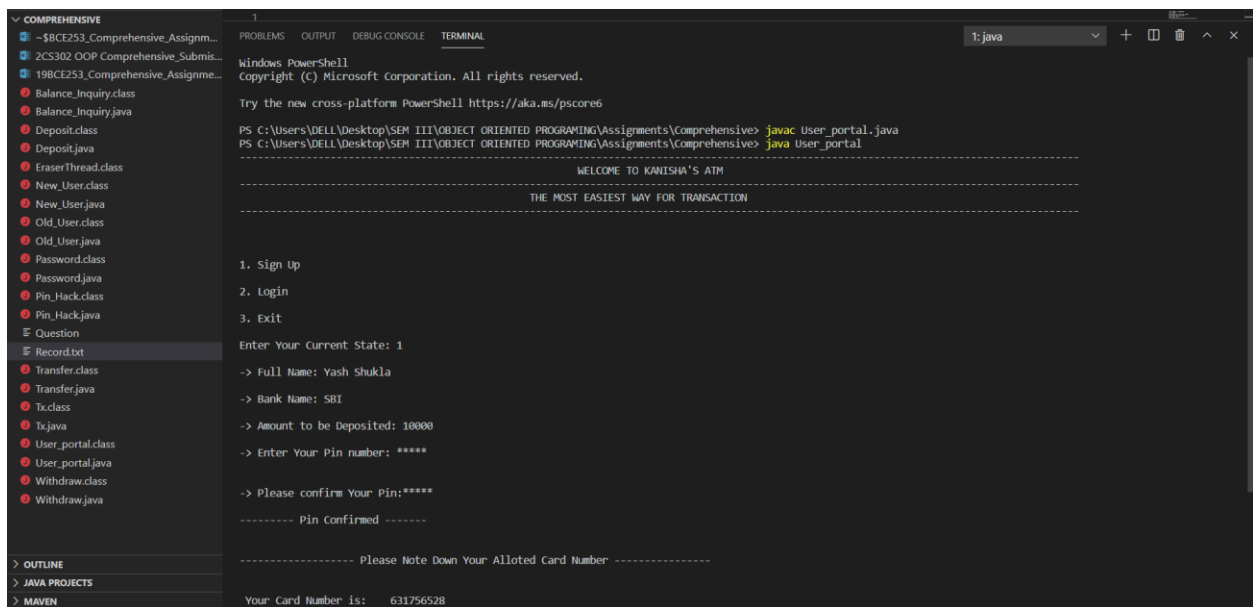
19BCE253

Rahul Shukla's Balance: 100000

User Name: Rahul Shukla

Bank Name: SBI

Screen Shots:



```
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\DELL\Desktop\SEM III\OBJECT ORIENTED PROGRAMING\Assignments\comprehensive> javac User_portal.java
PS C:\Users\DELL\Desktop\SEM III\OBJECT ORIENTED PROGRAMING\Assignments\comprehensive> java User_portal

-----
WELCOME TO KANISHA'S ATM
-----

THE MOST EASIEST WAY FOR TRANSACTION

-----

1. Sign Up
2. Login
3. Exit

Enter Your Current State: 1
-> Full Name: Yash Shukla
-> Bank Name: SBI
-> Amount to be Deposited: 10000
-> Enter Your Pin number: *****

-> Please confirm Your Pin:*****

----- Pin Confirmed -----

----- Please Note Down Your Alloted Card Number -----

Your Card Number is: 631756528
```

19BCE253

```
Record.txt
22 user name: RAHUL SHUKLA
34 Bank Name: Nutan
35 *****
36
37 Card Number: 714296779
38 Pin: 4875
39 Rahul Shukla's Balance: 100000
40 User Name: Rahul Shukla
41 Bank Name: SBI
42 *****
43
44 Card Number: 631756528
45 Pin: 1234
46 Yash Shukla's Balance: 10000
47 User Name: Yash Shukla
48 Bank Name: SBI
49 *****
50

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: java

----- Please Note Down Your Alloted Card Number -----

Your Card Number is: 631756528

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money           4. Withdraw Money
5. Exit

-> Enter Your Choice: 
```

```
Record.txt
1
2 Card Number: 401283854
3 Pin: 1234
4 Kanisha Shah's Balance: 3900
5 User Name: Kanisha Shah
6 Bank Name: SBI
7 *****
8
9 Card Number: 560241145
10 Pin: 4512
11 Kahan Sheth's Balance: 5000
12 User Name: Kahan Sheth
13 Bank Name: SBI

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: java

3. Deposit Money           4. Withdraw Money
5. Exit

-> Enter Your Choice: 2

----- Welcome to Balance Inquiry Mode -----

-> Please Enter Your Card Number: 40128385
-> Card Holder's Name: Kanisha Shah

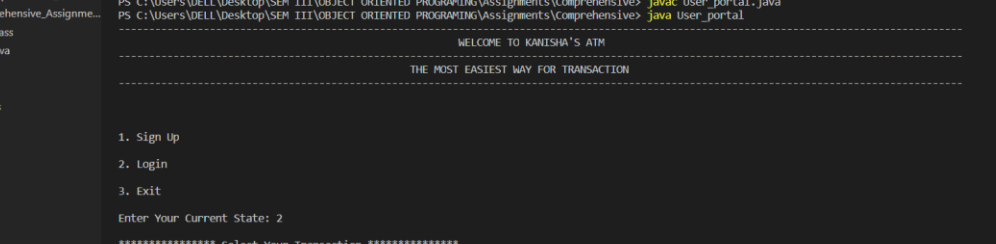
----- No such Card Number exist -----

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money           4. Withdraw Money
5. Exit

-> Enter Your Choice: 
```

19BCE253



```
COMPREHENSIVE
- $BCE253_Comprehensive_Assignm...
2CS302_OOP_Comprehensive_Submis...
19BCE253_Comprehensive_Assignm...
  Balance_Inquiry.class
  Balance_Inquiry.java
  Deposit.class
  Deposit.java
  EraserThread.class
  New_User.class
  New_User.java
  Old_User.class
  Old_User.java
  Password.class
  Password.java
  Pin_Hack.class
  Pin_Hack.java
  Question
Record.txt
  Transfer.class
  Transfer.java
  Tx.class
  Tx.java
  User_portal.class
  User_portal.java
  Withdraw.class
  Withdraw.java

> OUTLINE
> JAVA PROJECTS
> MAVEN

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
z: java
PS C:\Users\DELL\Desktop\SEM III\OBJECT ORIENTED PROGRAMING\Assignments\Comprehensive> javac User_portal.java
PS C:\Users\DELL\Desktop\SEM III\OBJECT ORIENTED PROGRAMING\Assignments\Comprehensive> java User_portal

-----
WELCOME TO KANISHA'S ATM
-----

THE MOST EASIEST WAY FOR TRANSACTION
-----

1. Sign Up
2. Login
3. Exit

Enter Your Current State: 2

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money          4. Withdraw Money
5. Exit

-> Enter Your Choice: 2

----- Welcome to Balance Inquiry Mode -----

-> Please Enter Your Card Number: 401283854

-> Card Holder's Name: Kanisha Shah

-> Enter Your Pin: *****

----- Pin Matched Successfully -----

Bank Name: SBI
```

The screenshot shows an IDE with a project named 'COMPREHENSIVE'. The left sidebar lists various Java files, including 'Record.txt' which is currently selected. The main editor window displays the content of 'Record.txt', which is a list of transactions: 1. Transfer, 2. Balance Enquiry, 3. Deposit Money, and 4. Withdraw Money. Below the editor, a terminal window shows the output of a Java program. The program prompts the user to enter a card number (401283854), a name (Kanisha Shah), and a pin (****). It then displays a message: '----- Pin Matched Successfully -----'. The terminal also shows the details of the transaction: 'Bank Name: SBI', 'Account Holder: Kanisha Shah', 'Card Number: 401283854', 'Bank Name: SBI', and 'Your Current Balance is: 3900'. Finally, it prompts the user to 'Select Your Transaction *****'.

19BCE253

```
1 Card Number: 401283854
2 Pin: 1234
3 Kanisha Shah's Balance: 3900
4 User Name: Kanisha Shah
5 Bank Name: SBI
6 *****
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: java

Card Number: 401283854

Bank Name: SBI

Your Current Balance is: 3900

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money           4. Withdraw Money
5. Exit

-> Enter Your Choice: 3

----- Welcome to Deposit Mode -----

----- Personal Details -----

-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****
*
----- Pin Matched Successfully -----

-> Enter the amount to be Deposited (in Rs.):
```

```
1 Card Number: 401283854
2 Pin: 1234
3 Kanisha Shah's Balance: 4000
4 User Name: Kanisha Shah
5 Bank Name: SBI
6 *****
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: java

5. Exit

-> Enter Your Choice: 3

----- Welcome to Deposit Mode -----

----- Personal Details -----

-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****
*
----- Pin Matched Successfully -----

-> Enter the amount to be Deposited (in Rs.):
100
100 Rs. Deposited successfully

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money           4. Withdraw Money
5. Exit

-> Enter Your Choice: █
```

19BCE253

```
COMPREHENSIVE
~$BCE253_Comprehensive_Assignm...
2CS302 OOP Comprehensive_Submis...
19BCE253_Comprehensive_Assignme...
Balance_Inquiry.class
Balance_Inquiry.java
Deposit.class
Deposit.java
EraserThread.class
New_User.class
New_User.java
Old_User.class
Old_User.java
Password.class
Password.java
Pin_Hack.class
Pin_Hack.java
Question
Record.txt
Transfer.class
Transfer.java
Tx.class
Tx.java
User_portal.class
User_portal.java
Withdraw.class
Withdraw.java

OUTLINE
JAVA PROJECTS
MAVEN

1
2 Card Number: 401283854
3 Pin: 1234
4 Kanisha Shah's Balance: 4000
5 User Name: Kanisha Shah

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: java

1. Transfer 2. Balance Enquiry
3. Deposit Money 4. Withdraw Money
5. Exit
-> Enter Your Choice: 4
----- Welcome to Withdraw Mode-----

----- Personal Details -----
-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****
*
----- Pin Matched Successfully -----

-> Enter the amount to be Withdrawn:
100000
----- More than 20,000 Rs. can't be Withdrawn -----

***** Select Your Transaction *****

1. Transfer 2. Balance Enquiry
```

```
COMPREHENSIVE
~$BCE253_Comprehensive_Assignm...
2CS302 OOP Comprehensive_Submis...
19BCE253_Comprehensive_Assignme...
Balance_Inquiry.class
Balance_Inquiry.java
Deposit.class
Deposit.java
EraserThread.class
New_User.class
New_User.java
Old_User.class
Old_User.java
Password.class
Password.java
Pin_Hack.class
Pin_Hack.java
Question
Record.txt
Transfer.class
Transfer.java
Tx.class
Tx.java
User_portal.class
User_portal.java
Withdraw.class
Withdraw.java

OUTLINE
JAVA PROJECTS
MAVEN

1
2 Card Number: 401283854
3 Pin: 1234
4 Kanisha Shah's Balance: 4000
5 User Name: Kanisha Shah

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2. Login
3. Exit

Enter Your Current State: 2

***** Select Your Transaction *****

1. Transfer 2. Balance Enquiry
3. Deposit Money 4. Withdraw Money
5. Exit
-> Enter Your Choice: 4
----- Welcome to Withdraw Mode-----

----- Personal Details -----
-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****
----- Pin Matched Successfully -----

-> Enter the amount to be Withdrawn:
1000
```

19BCE253

```
1 Card Number: 401283854
2 Pin: 1234
3 Kanisha Shah's Balance: 3000
4 User Name: Kanisha Shah
5

5. Exit
-> Enter Your Choice: 4

----- Welcome to Withdraw Mode-----

----- Personal Details -----

-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****

----- Pin Matched Successfully -----

-> Enter the amount to be Withdrawn:
1000
1000 Rs. successfully withdrawn

***** Select Your Transaction *****

1. Transfer                2. Balance Enquiry
3. Deposit Money          4. Withdraw Money
5. Exit
-> Enter Your Choice: █
```

```
2 Card Number: 401283854
3 Pin: 1234
4 Kanisha Shah's Balance: 3000
5 User Name: Kanisha Shah
6 Bank Name: SBI
7 *****
8
9 Card Number: 560241145
10 Pin: 4512
11 Kahan Sheth's Balance: 5000
12 User Name: Kahan Sheth
13 Bank Name: SBI
14 *****

5. Exit
-> Enter Your Choice: 1

----- Welcome to Transfer Mode -----

----- Personal Details -----

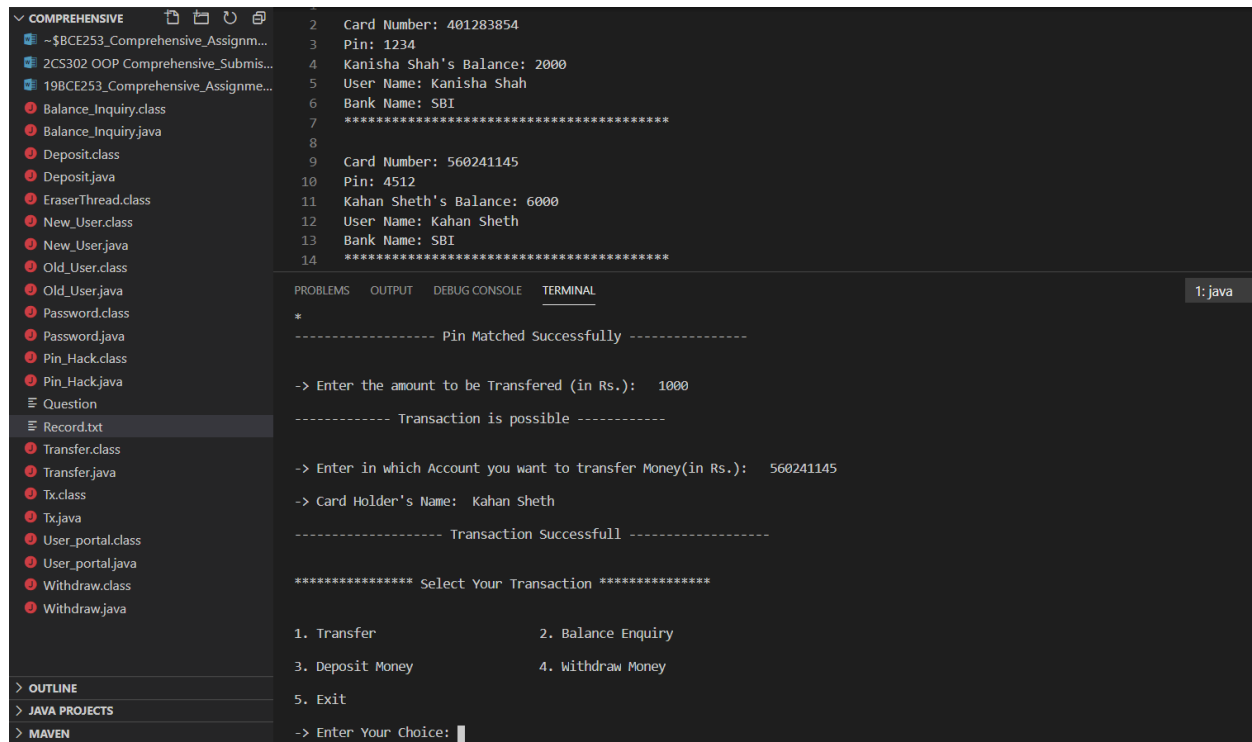
-> Please Enter Your Card Number: 401283854
-> Card Holder's Name: Kanisha Shah
-> Enter Your Pin number: *****
*
----- Pin Matched Successfully -----

-> Enter the amount to be Transferred (in Rs.): 1000

----- Transaction is possible -----

-> Enter in which Account you want to transfer Money(in Rs.): █
```


19BCE253



The screenshot shows an IDE with a project named '19BCE253'. The left sidebar contains a file explorer with the following files:

- COMPREHENSIVE
- ~\$BCE253_Comprehensive_Assignm...
- 2CS302 OOP Comprehensive_Submis...
- 19BCE253_Comprehensive_Assignme...
- Balance_Inquiry.class
- Balance_Inquiry.java
- Deposit.class
- Deposit.java
- EraserThread.class
- New_User.class
- New_User.java
- Old_User.class
- Old_User.java
- Password.class
- Password.java
- Pin_Hack.class
- Pin_Hack.java
- Question
- Record.txt
- Transfer.class
- Transfer.java
- Tx.class
- Tx.java
- User_portal.class
- User_portal.java
- Withdraw.class
- Withdraw.java

The main editor displays the following Java code:

```
1  
2 Card Number: 401283854  
3 Pin: 1234  
4 Kanisha Shah's Balance: 2000  
5 User Name: Kanisha Shah  
6 Bank Name: SBI  
7 *****  
8  
9 Card Number: 560241145  
10 Pin: 4512  
11 Kahan Sheth's Balance: 6000  
12 User Name: Kahan Sheth  
13 Bank Name: SBI  
14 *****
```

The terminal window shows the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: java  
*  
----- Pin Matched Successfully -----  
  
-> Enter the amount to be Transferred (in Rs.): 1000  
----- Transaction is possible -----  
  
-> Enter in which Account you want to transfer Money(in Rs.): 560241145  
-> Card Holder's Name: Kahan Sheth  
----- Transaction Successfull -----  
  
***** Select Your Transaction *****  
  
1. Transfer 2. Balance Enquiry  
3. Deposit Money 4. Withdraw Money  
5. Exit  
-> Enter Your Choice: 
```

19BCE253

```
2 Card Number: 401283854
3 Pin: 1234
4 Kanisha Shah's Balance: 2000
5 User Name: Kanisha Shah
6 Bank Name: SBI
7 *****
8
9 Card Number: 560241145
10 Pin: 4512
11 Kahan Sheth's Balance: 6000
12 User Name: Kahan Sheth
13 Bank Name: SBI
14 *****
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

5. Exit

-> Enter Your Choice: 1

----- Welcome to Transfer Mode -----

----- Personal Details -----

-> Please Enter Your Card Number: 123451

-> Card Holder's Name: kanisha

----- No such Card Number exist -----

***** Select Your Transaction *****

1. Transfer 2. Balance Enquiry

3. Deposit Money 4. Withdraw Money

5. Exit

-> Enter Your Choice: █