VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SUJAN G E (1BM23CS347)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

B. M. S. College of Engineering, Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

CERTIFICATE

This is to certify that the Lab work entitled “OBJECT ORIENTED JAVA PROGRAMMING” carried out by SUJAN G E(1BM23CS347), who is bonafide student of B. M. S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Object-Oriented Java Programming Lab - (23CS3PCOOJ) work prescribed for the said
degree.

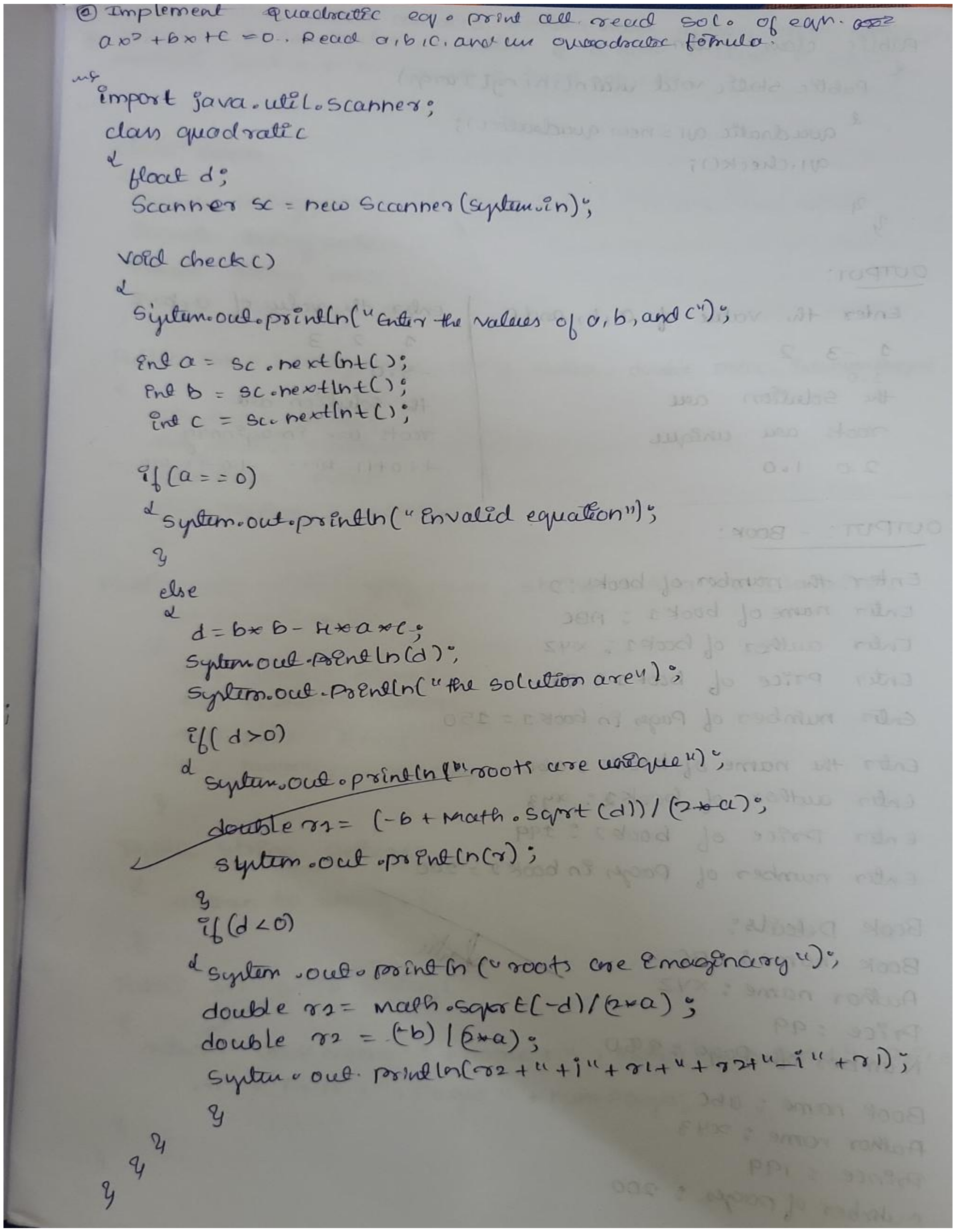
Dr. Nandhini Vineeth	Dr. Kavitha Sooda
Associate Professor, Department of CSE, CSE	Professor and Head, Department of
BMSCE, Bengaluru	BMSCE, Bengaluru

INDEX

PROGRAM 1:

Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions

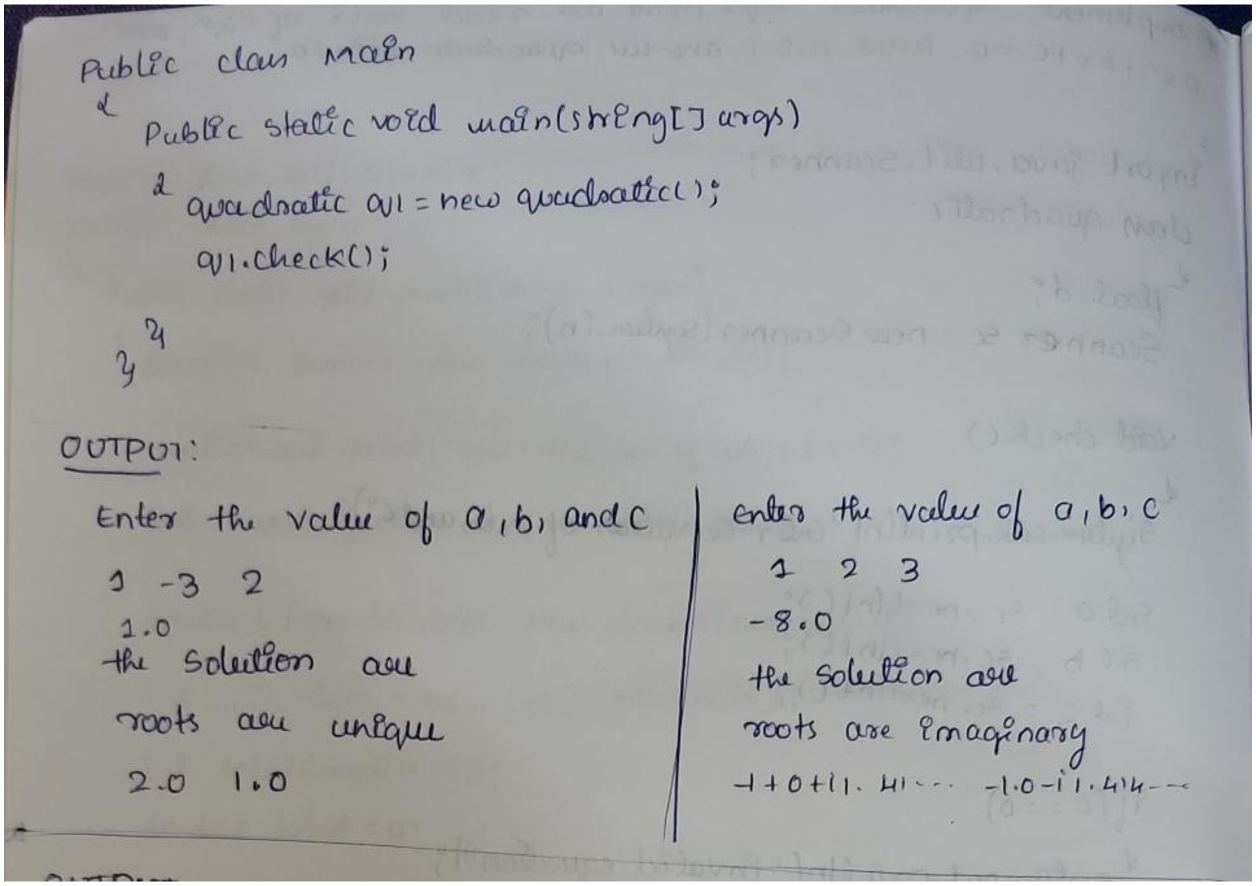
OBSERVATION :



(a) Implement quadratic eq. print all real sol. of eqn. $ax^2+bx+c=0$. Read a, b, c, and use quadratic formula.

```
import java.util.scanner;
class quadratic
{
  float d;
  Scanner sc = new Scanner(system.in);

 void check()
 {
  System.out.println("Enter the values of a, b, and c");
  int a = sc.nextInt();
  int b = sc.nextInt();
  int c = sc.nextInt();

  if(a==0)
  { system.out.println("Invalid equation");
  }
  else
  {
   d=b*b- 4*a*c;
   System.out.println(d);
   System.out.println("the solution are");
   if( d>0)
   { System.out.println("roots are unique");
    double r1= (-b + Math.sqrt(d))/(2*a);
    system.out.println(r);
   }
   if(d<0)
   { system.out.println("roots are imaginary");
   double r1= math.sqrt(-d)/(2*a);
   double r2 = (-b)/(2*a);
   system.out.println(r2+"+i"+r1+"+r2+"-i"+r1);
   }
  }
 }
}
```

4



```
Public class Main
{
  Public static void main(String[] args)
  {
   quadratic q1 = new quadratic();
   q1.check();

  }
}
```

OUTPUT:

```
Enter the value of a, b, and c
 1  -3  2
 1.0
the solution are
 roots are unique
  2.0   1.0
```

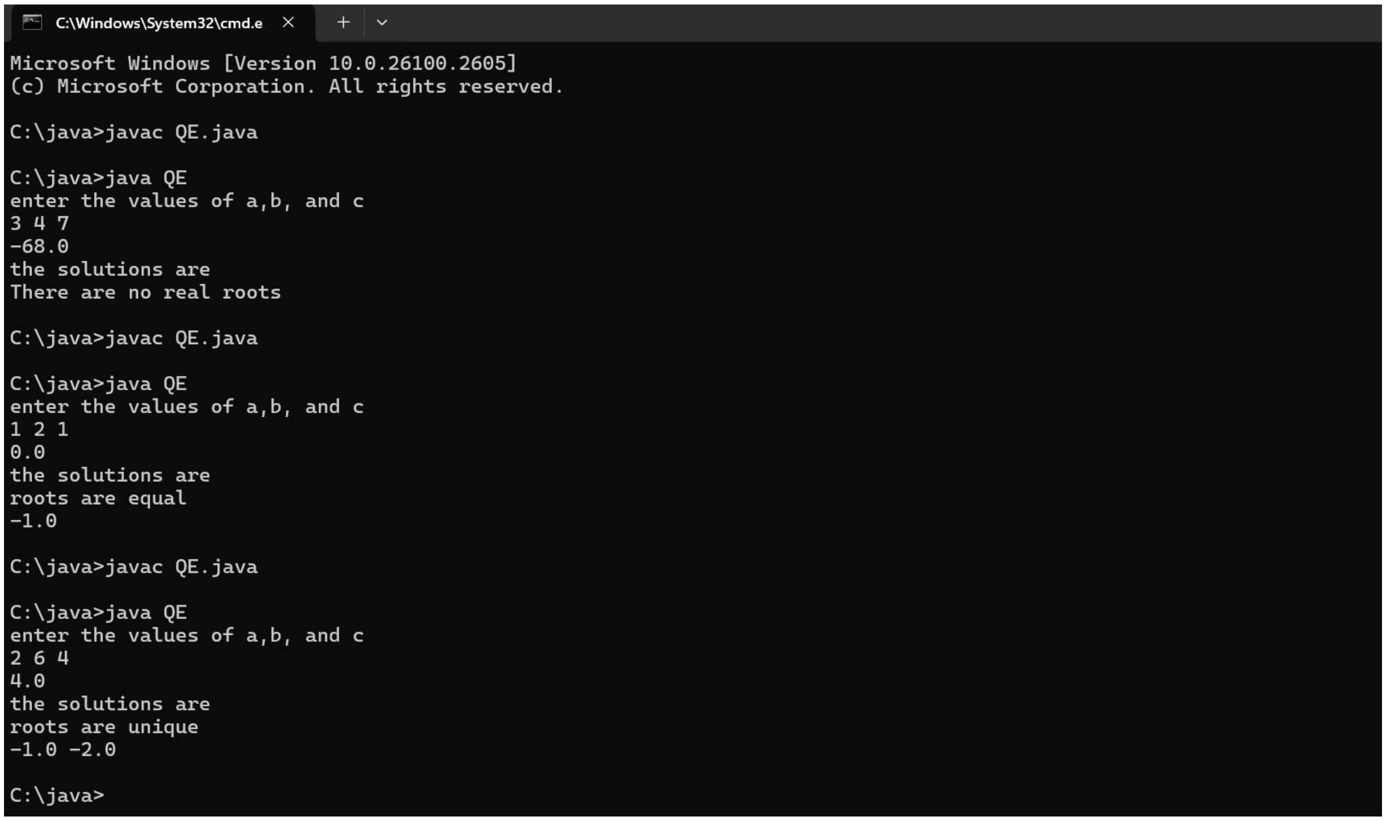
```
enter the value of a, b, c
  1  2  3
 -8.0
 the solution are
 roots are imaginary
 -1+0+i1.41...  -1.0-i1.414...
```

Source Code:

```
 import java.util.Scanner;
class Quadratic {
float d;
    Scanner sc = new Scanner(System.in);
     void
solver()

    {
        System.out.println("enter the values of a,b, and
c");          int a = sc.nextInt();          int b =
sc.nextInt();          int c = sc.nextInt();


        if (a == 0) {
            System.out.println("invalid equation");
        }
else{
            d= b*b - 4*a*c;
            System.out.println(d);
            System.out.println("the solutions are");
if(d>0){
                System.out.println("roots are unique ");
double r1 = (-b+Math.sqrt(d))/(2*a);                 double
r2 = (-b-Math.sqrt(d))/(2*a);
                System.out.println(r1 +" " + r2);
            }
            if(d==0){
                System.out.println("roots are equal ");
double r = -b/(2*a);
                System.out.println(r);
            }
if(d<0){
                System.out.println("There are no real roots" );
            }
        }

    }



}



public class QE {
    public static void main(String[] args) {
Quadratic q1 = new Quadratic();          q1.solver();
```

```
  }
}
```

OUTPUT:

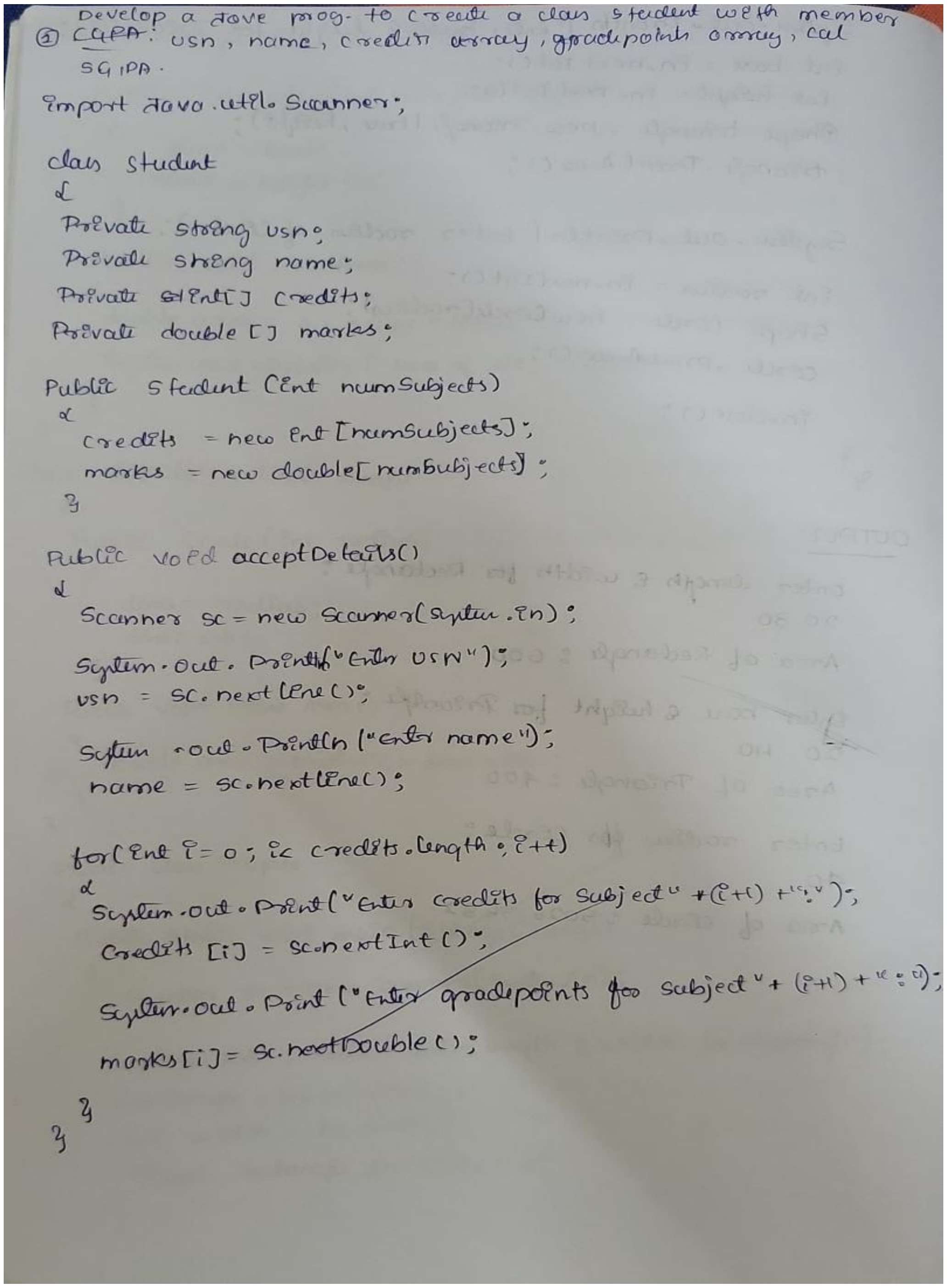


```
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\java>javac QE.java

C:\java>java QE
enter the values of a,b, and c
3 4 7
-68.0
the solutions are
There are no real roots

C:\java>javac QE.java

C:\java>java QE
enter the values of a,b, and c
1 2 1
0.0
the solutions are
roots are equal
-1.0

C:\java>javac QE.java

C:\java>java QE
enter the values of a,b, and c
2 6 4
4.0
the solutions are
roots are unique
-1.0 -2.0

C:\java>
```

PROGRAM 2:

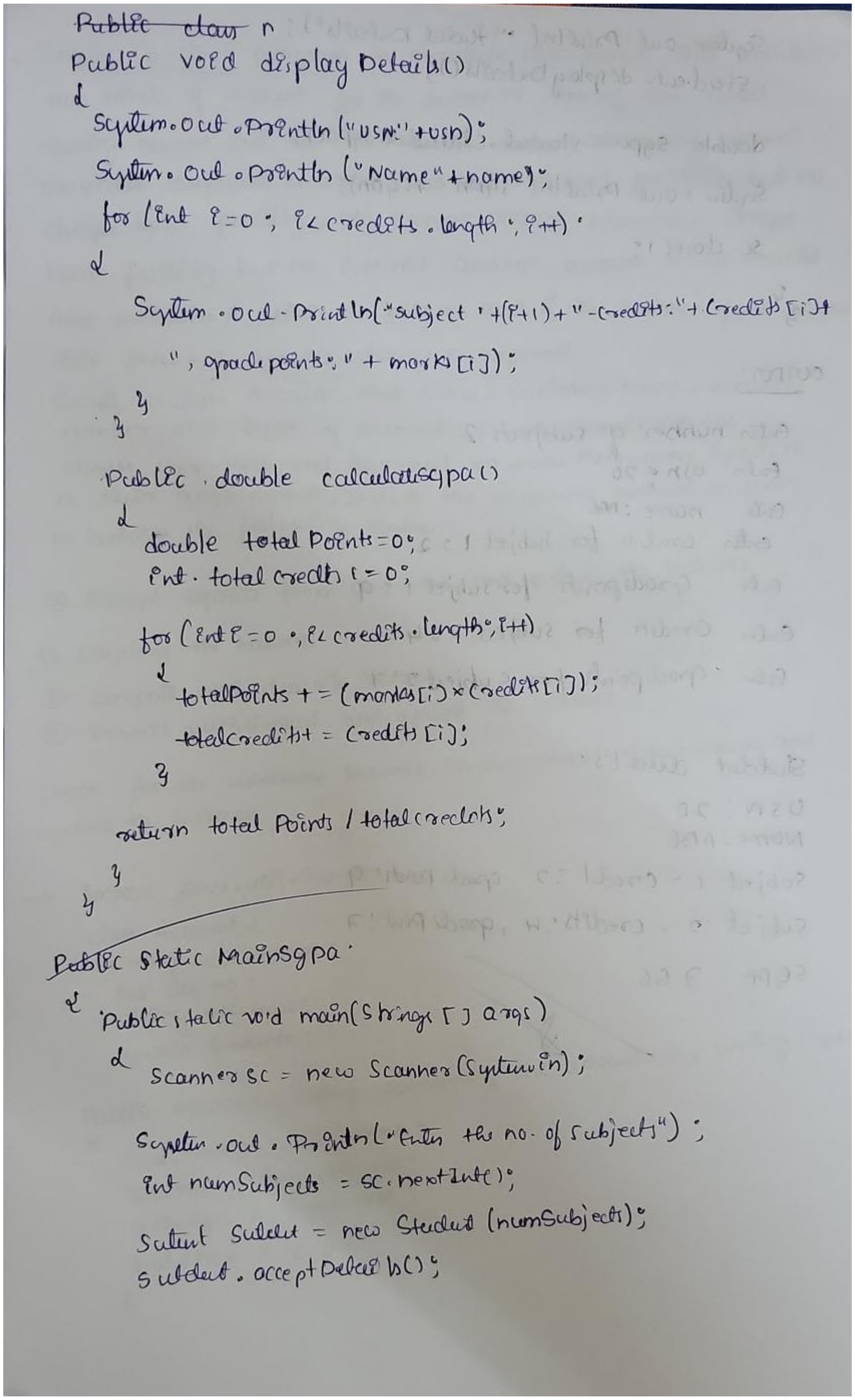
Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

OBSERVATION:



Develop a Jave prog. to create a class student with member
① CGPA: usn, name, credits array, gradepoints array, cal SGIPA.

```
import java.util.Scanner;

class student
{
 Private string usn;
 Private string name;
 Private int[] credits;
 Private double [] marks;

Public Student (int numSubjects)
 {
  credits  = new int [numSubjects];
  marks  = new double[numSubjects];
 }

Public void acceptDetails()
{
  Scanner sc = new Scanner(System.in);
  System.out.Println("Enter USN");
  usn = sc.nextLine();
  System.out.Println("Enter name");
  name = sc.nextLine();

  for(int i= 0; i< credits.length; i++)
  {
   System.out.Print("Enter credits for Subject" +(i+1) +":");
   Credits [i] = sc.nextInt();
   System.out.Print ("Enter gradepoints for subject"+ (i+1) + ":");
   marks[i]= Sc.nextDouble();
  }
}
```



```
~~Public class~~ n
Public void displayDetails()
{
  System.out.Println("USN:"+usn);
  System.out.Println("Name"+name);
  for (int i=0; i<credits.length; i++)
  {
     System.out.Println("subject "+(i+1)+"-credits:"+ Credits[i]+
        ", grade points:" + marks[i]);
  }
}
 Public double calculatesgpa()
  {
    double totalPoints=0;
    int totalcredits=0;

    for (int i=0, i<credits.length; i++)
     {
       totalPoints += (marks[i] * (credits[i]);
       totalcredits+ = (credits[i];
     }
   return totalPoints / totalcredits;
  }
}
Public Static Mainsgpa
{
  Public static void main(Strings [] args)
  {
    Scanner sc = new Scanner(System.in);

    System.out.Println("Enter the no. of subjects");
    int numSubjects = sc.nextInt();
    Sutent Sudent = new Student (numSubjects);
    sudent.acceptDetails();
System.out.Println("Student Details");
Student.displayDetails();

double sgpa = student.calculateSgpa();
System.out.Println("SGPA" +SGPA);

sc.close();

}
}
```

OUTPUT:

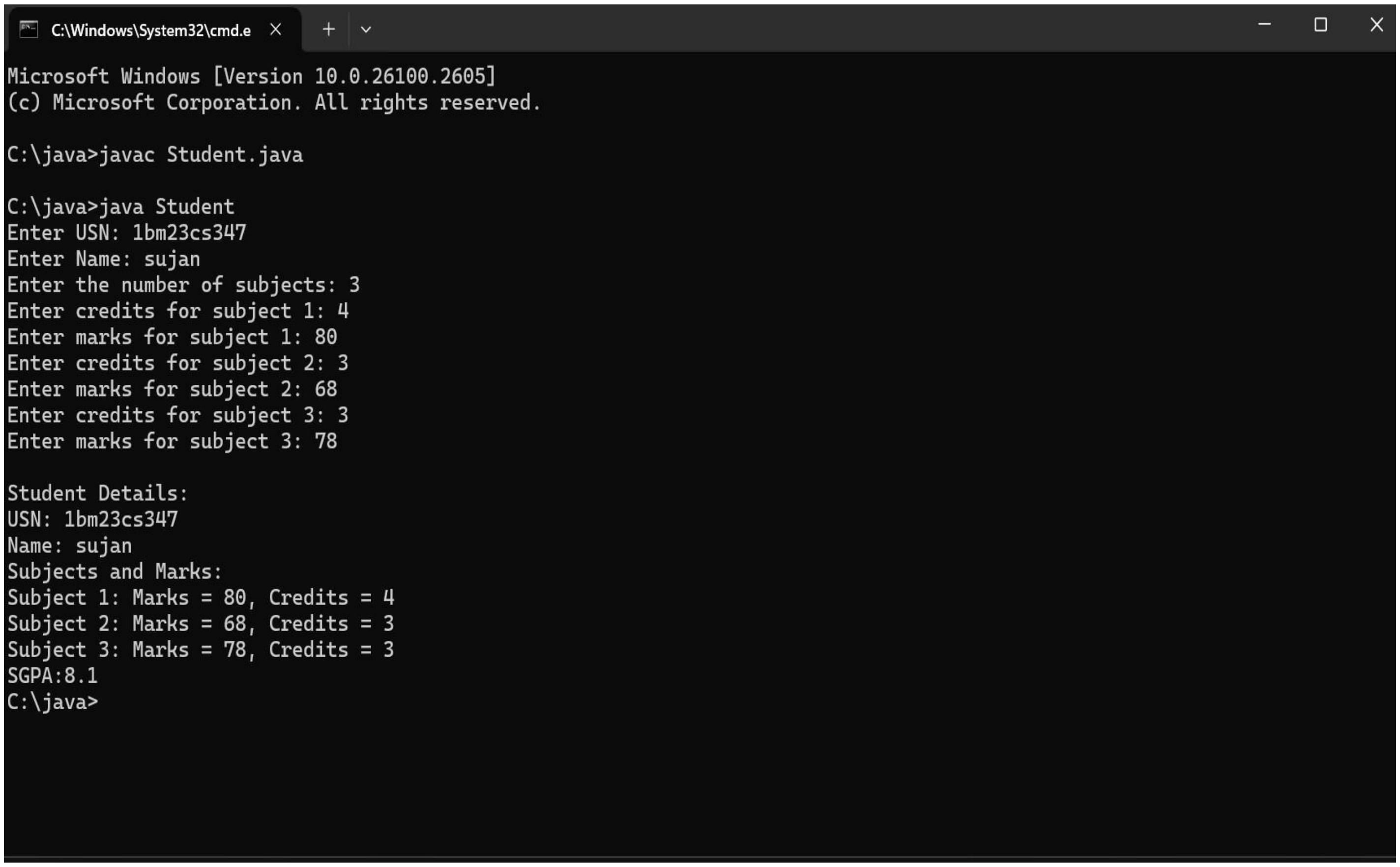
```
Enter number of subjects 2
Enter USN : 20
Enter name : ABC
enter credits for Subjet 1 : 2
Enter Gradepoints for subjet 1 : 9
Enter Credits for subject 2 : 4
Enter gradpoints for Subject 2 : 7

Student details.
USN : 20
Name : ABC
Subject 1 - Credit : 2 , grade point : 9
Subjet 2 - credits : 4 , grade point : 7
SGPA = 7.66.
```

Source Code:

```
import
java.util.Scanner; class
Student {      String
usn;      String name;
int numSubjects;
int[] credits;      int[]
marks;      double sgpa;


    public void acceptDetails() {
Scanner sc = new Scanner(System.in);
        System.out.print("Enter USN: ");
usn = sc.nextLine();

        System.out.print("Enter Name: ");
name = sc.nextLine();

        System.out.print("Enter    the    number    of    subjects:    ");
numSubjects = sc.nextInt();
         credits = new
int[numSubjects];          marks = new
int[numSubjects];

        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
credits[i] = sc.nextInt();
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
marks[i] = sc.nextInt();
        }
    }

    public void displayDetails() {
        System.out.println("\nStudent Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subjects and Marks:");
         for (int i = 0; i < numSubjects; i++)
{
```

```
            System.out.println("Subject " + (i + 1) + ": Marks = " + marks[i]
+ ", Credits = " + credits[i]);
        }
    }
          public void
calculateSGPA() {          int
totalCredits = 0;          int
totalGradePoints = 0;
         for (int i = 0; i < numSubjects; i++) {
int grade = calculateGrade(marks[i]);
totalGradePoints += grade * credits[i];
totalCredits += credits[i];
        }
         sgpa = (double) totalGradePoints /
totalCredits;
    }     private int calculateGrade(int
marks) {          if (marks >= 90) {
return 10;
        } else if (marks >= 80) {
return 9;
        } else if (marks >= 70) {
return 8;
        } else if (marks >= 60) {
return 7;
        } else if (marks >= 50) {
return 6;
        } else if (marks >= 40)
{              return 5;
} else {              return 0;
        }
    }

    public void displaySGPA() {
        System.out.printf("SGPA:" + sgpa);
    }
    public static void main(String[] args) {
Student student = new Student();
student.acceptDetails();
student.displayDetails();








        student.calculateSGPA();
student.displaySGPA();
    }
}
```

OUTPUT :



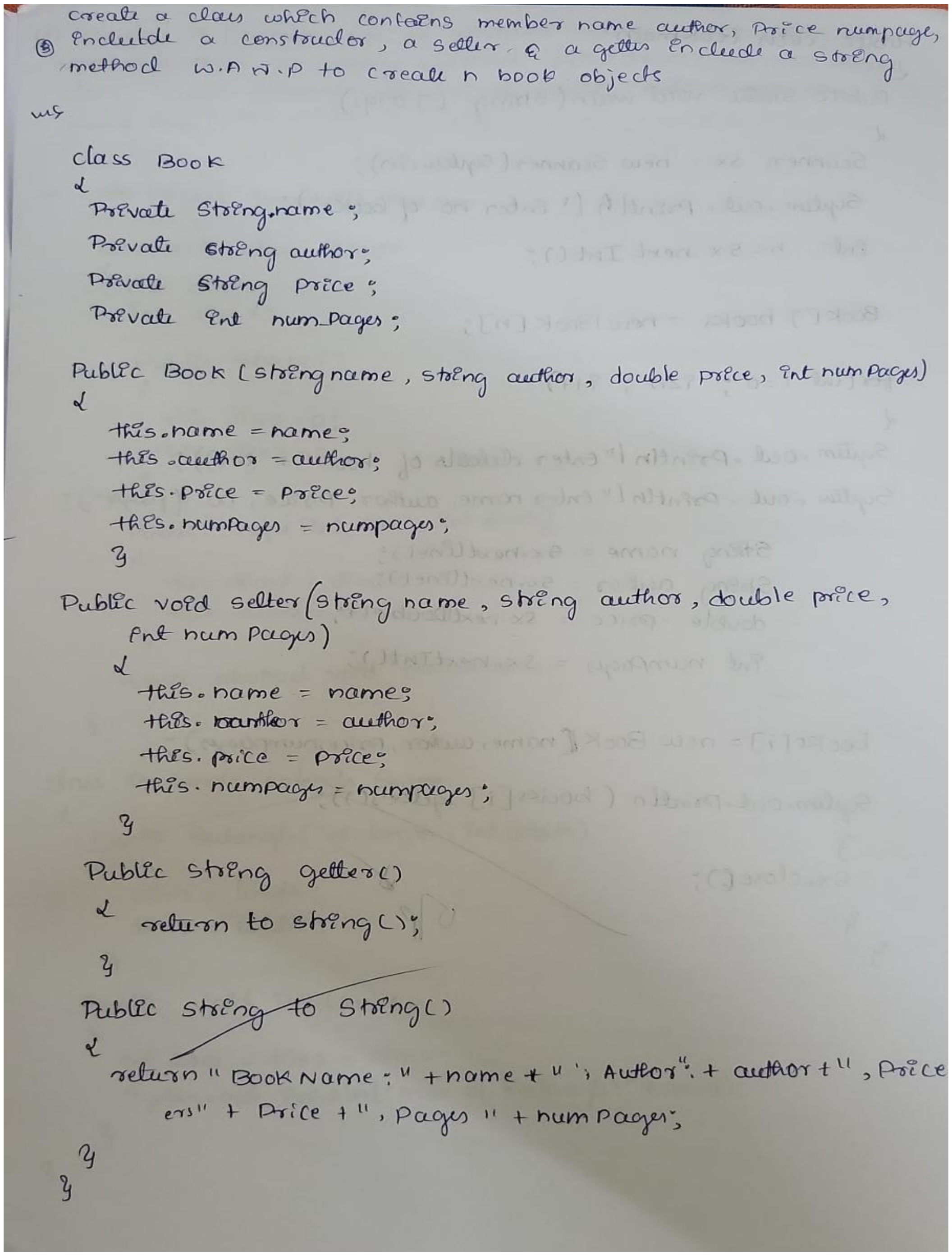
C:\Windows\System32\cmd.e

```
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\java>javac Student.java

C:\java>java Student
Enter USN: 1bm23cs347
Enter Name: sujan
Enter the number of subjects: 3
Enter credits for subject 1: 4
Enter marks for subject 1: 80
Enter credits for subject 2: 3
Enter marks for subject 2: 68
Enter credits for subject 3: 3
Enter marks for subject 3: 78

Student Details:
USN: 1bm23cs347
Name: sujan
Subjects and Marks:
Subject 1: Marks = 80, Credits = 4
Subject 2: Marks = 68, Credits = 3
Subject 3: Marks = 78, Credits = 3
SGPA:8.1
C:\java>
```

PROGRAM 3:

Create a class Book which contains four members: name,author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program to create n book objects.

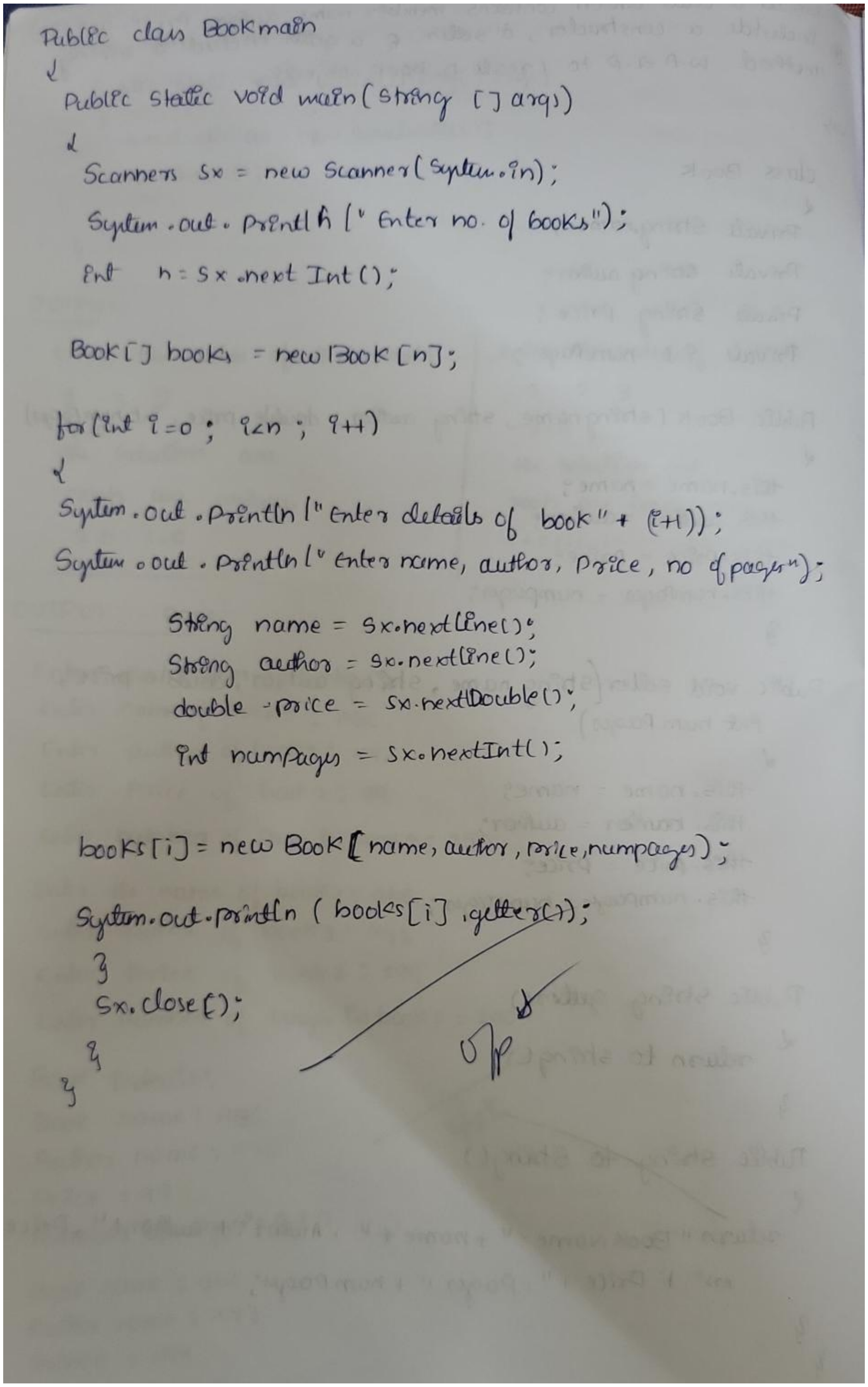
OBSERVATION:



(3) create a class which contains member name author, Price numpage, Include a constructor, a setter & a getter include a string method W.A J.P to create n book objects

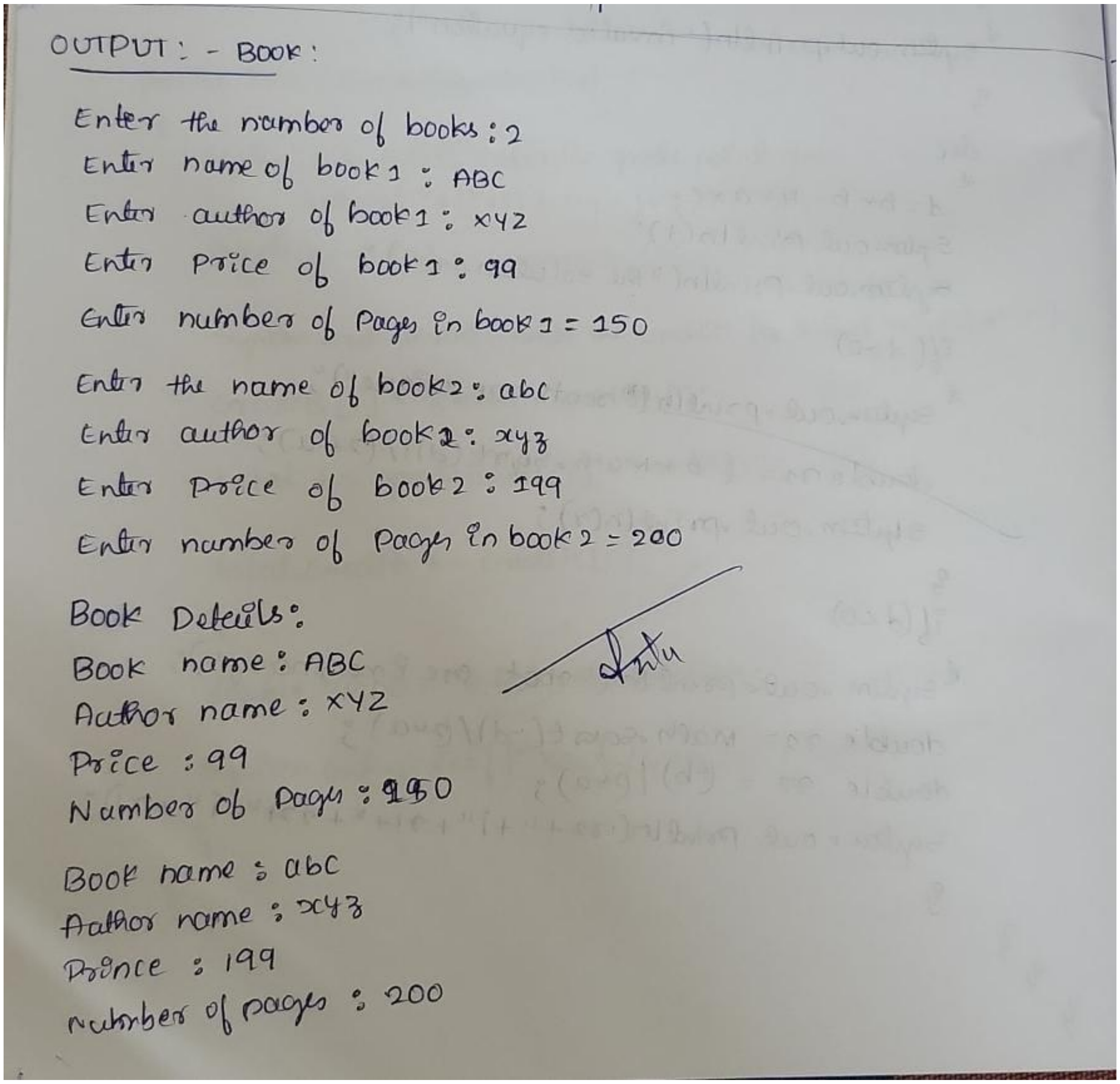
ws

```
class Book
{
  Private String.name ;
  Private string author;
  Private String price ;
  Private int num_Pages ;

  Public Book (string name , string author , double price, int num Pages)
  {
    this.name = name;
    this.author = author;
    this.price = price;
    this.numPages = numpages;
    }
 Public void setter(string name , string author, double price,
    int num Pages)
    {
     this.name = name;
     this.author = author;
     this.price = price;
     this.numpages = numpages;
    }
  Public string getter()
   {  return to string();
   }
  Public string to String()
  {
   return " Book Name : " + name + " ; Autor". + author + " , Price
        ers" + Price + " , Pages " + num Pages;
  }
}
```



```
Public class Bookmain
{
 Public static void main(string [] args)
 {
  Scanners sx = new Scanner(System.in);
  System.out.Println(" Enter no. of books");
  int  n = sx.next Int();

  Book[] books = new Book[n];

 for(int i=0 ; i<n ; i++)
 {
 System.out.Println(" Enter details of book" + (i+1));
 System.out.Println(" Enter name, author, Price, no of pages");

         String name = sx.nextLine();
         String author = sx.nextLine();
         double price = sx.nextDouble();
         int numPages = sx.nextInt();

  books[i] = new Book[name, author, price,numpages);

  System.out.println ( books[i] .getter());
   }
   Sx.close();
  }
 }
```

[illegible]



OUTPUT : - BOOK :

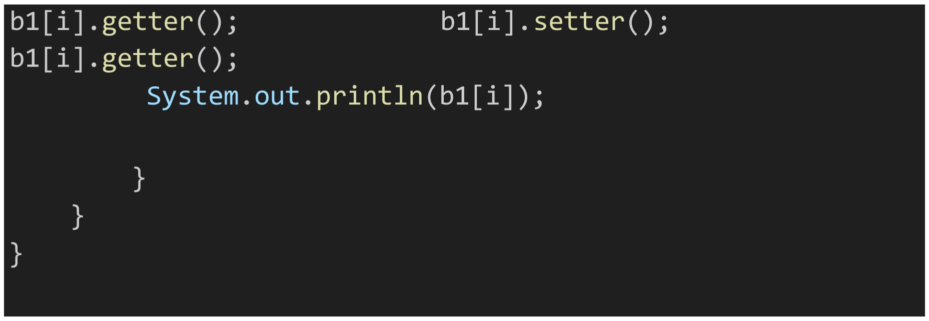
```
Enter the number of books : 2
Enter name of book 1 : ABC
Enter author of book 1 : xYZ
Enter Price of book 1 : 99
Enter number of Pages in book 1 = 150

Enter the name of book 2 : abc
Enter author of book 2 : xyz
Enter Price of book 2 : 199
Enter number of Pages in book 2 = 200

Book Details :
Book name : ABC
Author name : XYZ
Price : 99
Number of Pages : 150

Book name : abc
Author name : xyz
Prince : 199
number of pages : 200
```

[illegible]

Source Code:

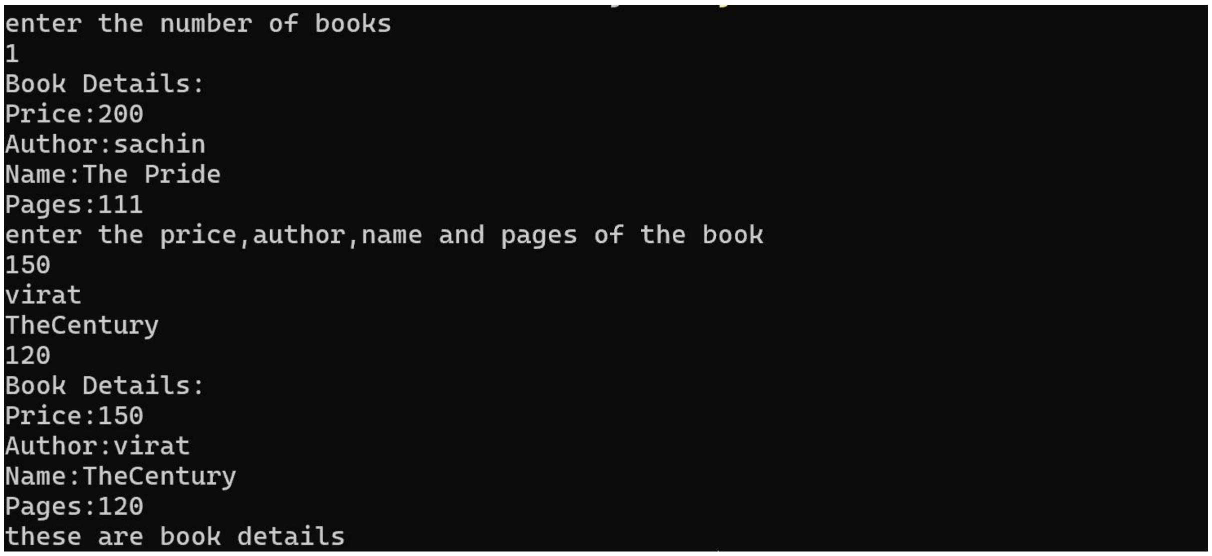
```
import java.util.Scanner;
 class Book {
int price;
String author;
String name;
int pages;
     public Book(int price, String author, String name, int pages)
{          this.price = price;          this.author = author;
this.name = name;          this.pages = pages;
    }
    public void setter() {
        System.out.println("enter the price,author,name and pages of the
book");         Scanner sc = new Scanner(System.in);
this.price=sc.nextInt();           this.author= sc.next();
this.name=sc.next();           this.pages=sc.nextInt();
    }
    public void getter() {
        System.out.println("Book Details:");
        System.out.println("Price:"+price);
        System.out.println("Author:"+author);
        System.out.println("Name:"+name);
        System.out.println("Pages:"+pages);
    }
    public String toString() {
                return "these are book details";
    }
}


public class Pro {
    public static void main(String[] args) {
Scanner s1 = new Scanner(System.in);
        System.out.println("enter the number of books");
int n = s1.nextInt();
```

```
       Book []b1 = new Book[n];
                 for(int i=0;i<n;i++){             b1[i] =
new Book(200,"sachin","The Pride",111);
```



```
b1[i].getter();             b1[i].setter();
b1[i].getter();
         System.out.println(b1[i]);

        }
    }
}
```

OUTPUT:

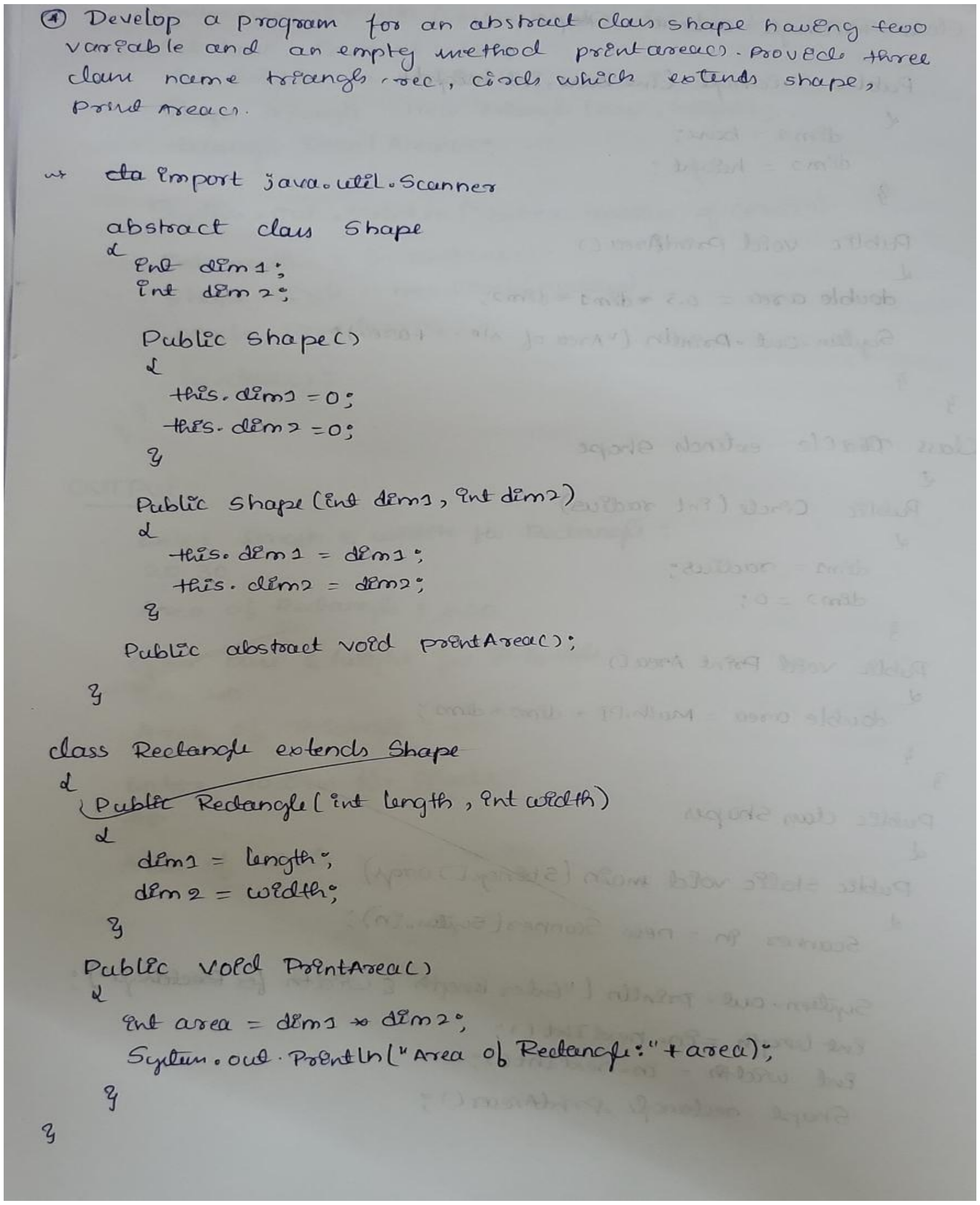


```
enter the number of books
1
Book Details:
Price:200
Author:sachin
Name:The Pride
Pages:111
enter the price,author,name and pages of the book
150
virat
TheCentury
120
Book Details:
Price:150
Author:virat
Name:TheCentury
Pages:120
these are book details
```

PROGRAM 4 :

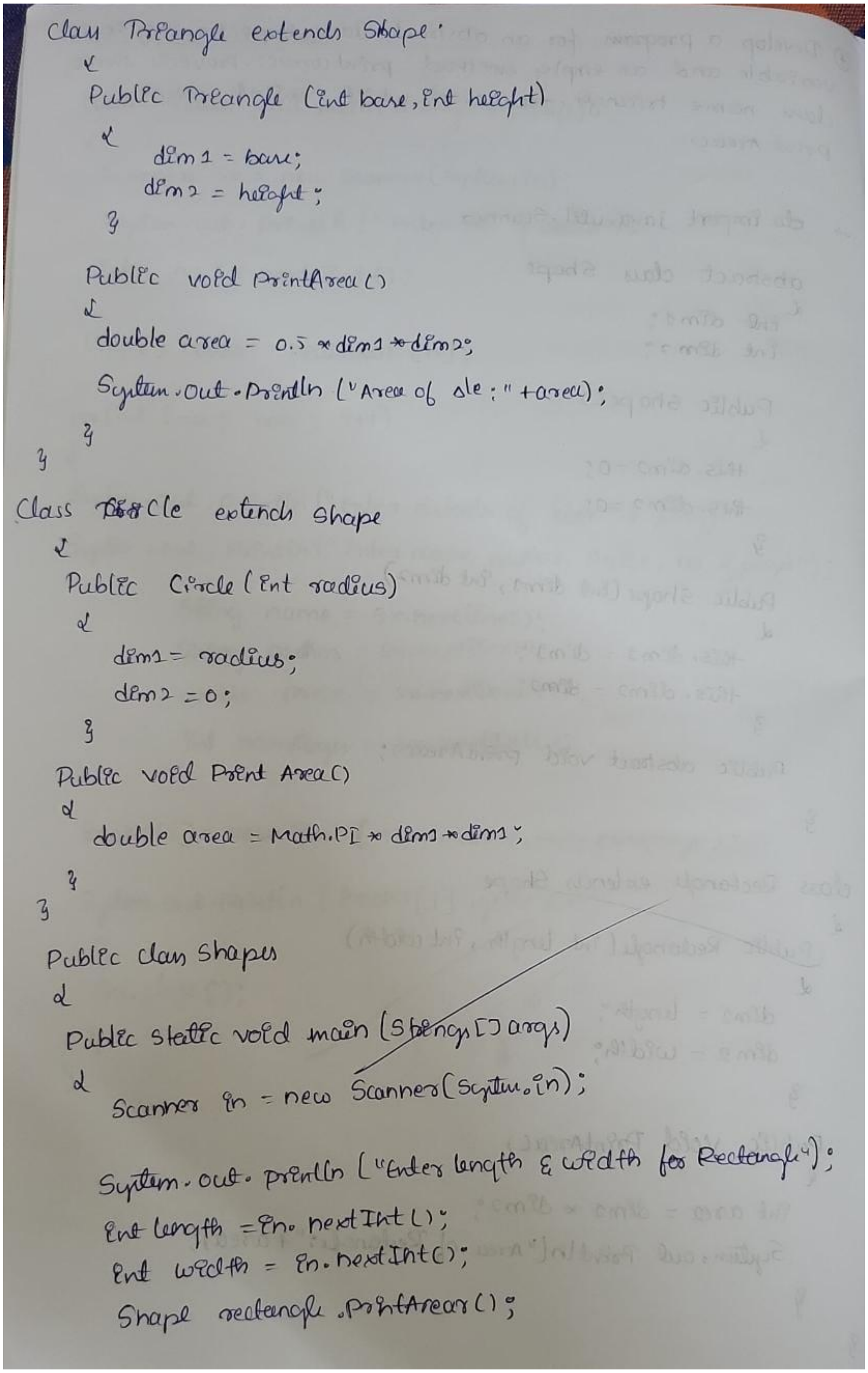
Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea( ). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.

OBSERVATION:

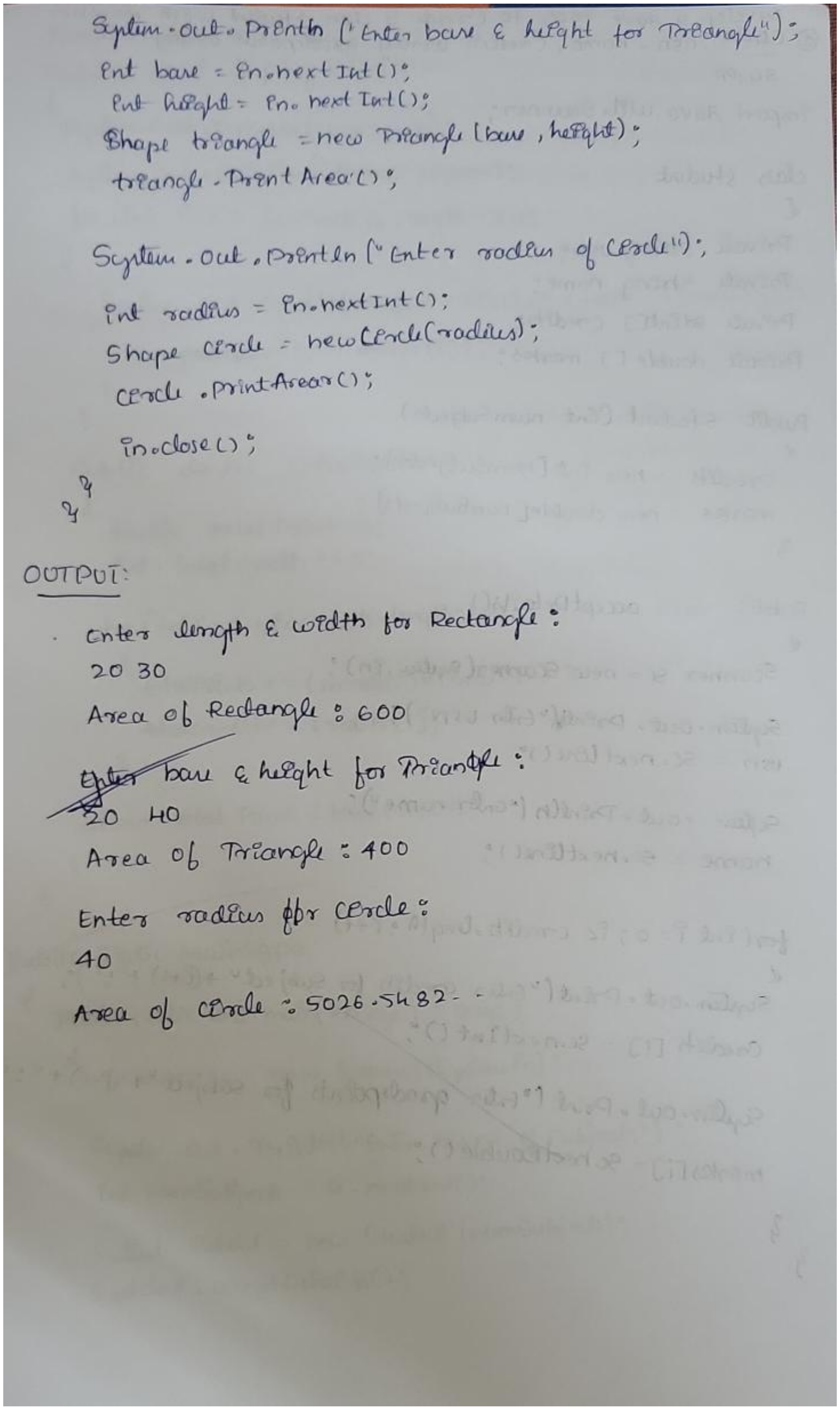


④ Develop a program for an abstract class shape having two variable and an empty method printarea(). Provide three class name triangle, rec, circle which extends shape, Print Area().

```
→   ~~cla~~ import java.util.Scanner

    abstract class Shape
    {
      int dim1;
      int dim2;

      Public Shape()
      {
        this.dim1 = 0;
        this.dim2 = 0;
      }
      Public Shape(int dim1, int dim2)
      {
        this.dim1 = dim1;
        this.dim2 = dim2;
      }
     Public abstract void printArea();
  }

class Rectangle extends Shape
{
  Public Rectangle(int length, int width)
  {
    dim1 = length;
    dim2 = width;
  }
  Public void PrintArea()
  {
    int area = dim1 * dim2;
    System.out.Println("Area of Rectangle:"+area);
  }
}
```



```
class Triangle extends Shape
  {
   Public Triangle (int base, int height)
    {
        dim1 = base;
       dim2 = height;
    }

   Public void PrintArea()
   {
    double area = 0.5 * dim1 * dim2;
    System.out.Println ("Area of tle :" +area);
   }
}
Class ~~Tri~~Cle extends Shape
  {
   Public Circle (int radius)
    {
      dim1= radius;
      dim2 =0;
    }
  Public void Print Area()
   {
    double area = Math.PI * dim1 * dim1;
   }
}
 Public class Shapes
  {
   Public static void main (String[] args)
    {
      Scanner in = new Scanner(System.in);

     System.out.println ("Enter length & width for Rectangle");
     int length =in.nextInt();
     int width = in.nextInt();
     Shape rectangle.PrintArear();
```



```
System.out.Println("Enter base & height for Triangle");
int base = in.nextInt();
int height = in.nextInt();
Shape triangle = new Triangle(base, height);
triangle.PrintArea();

System.out.Println("Enter radius of circle");
int radius = in.nextInt();
Shape circle = new Circle(radius);
circle.printArear();

in.close();
}
}
```

<u>OUTPUT:</u>

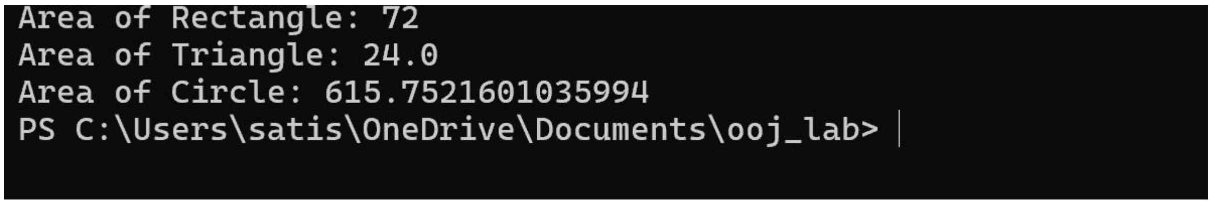
```
Enter length & width for Rectangle:
20 30
Area of Rectangle : 600
Enter base & height for Triangle:
20 40
Area of Triangle : 400
Enter radius for circle:
40
Area of circle : 5026.5482..
```

Source Code:

```
abstract class Shape {
int dim1;      int
dim2;
        abstract void
printArea();
}  class Rectangle extends Shape {
public Rectangle(int length, int width) {
this.dim1 = length;         this.dim2 =
width;
    }
         void
printArea() {
        int area = dim1 * dim2;
        System.out.println("Area of Rectangle: " + area);
    }
}  class Triangle extends Shape {
public Triangle(int base, int height) {
this.dim1 = base;         this.dim2 =
height;
    }      void
printArea() {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
    }
}
class Circle extends Shape {
public Circle(int radius) {
this.dim1 = radius;
this.dim2 = 0;
    }
         void
printArea() {
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: " + area);
```

```java
    }
}  public class
Main {
    public static void main(String[] args) {
Shape rectangle = new Rectangle(8,9);
        Shape triangle = new Triangle(8, 6);
        Shape circle = new Circle(14);

rectangle.printArea();
triangle.printArea();
circle.printArea();
    }
}
```

OUTPUT:



```
Area of Rectangle: 72
Area of Triangle: 24.0
Area of Circle: 615.7521601035994
PS C:\Users\satis\OneDrive\Documents\ooj_lab>
```

PROGRAM 5 :

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

a) Accept deposit from customer and update the balance.

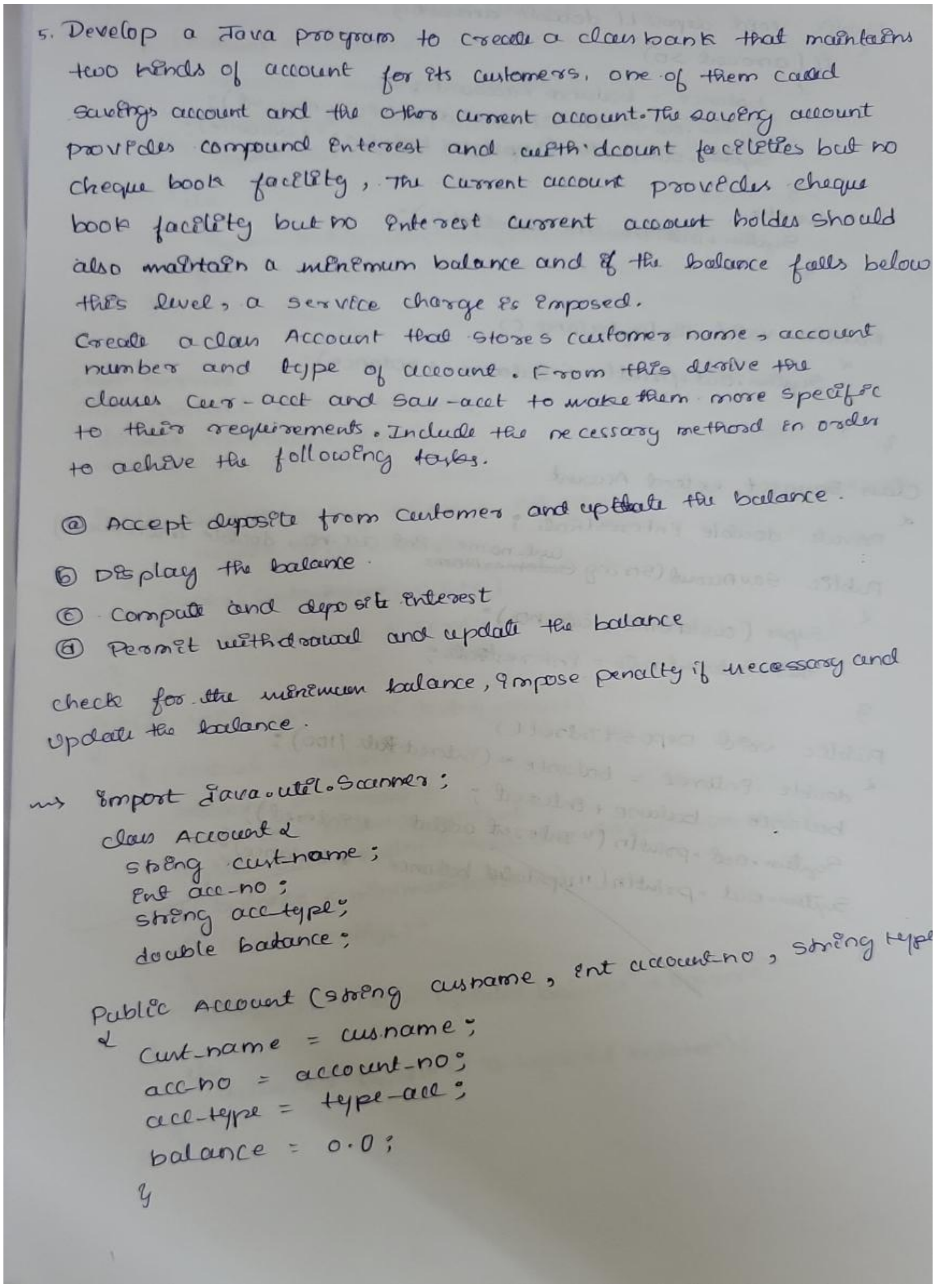
b) Display the balance.

c) Compute and deposit interest

d) Permit withdrawal and update the balance

Check for the minimum balance, impose penalty if necessary and update the balance.

OBSERVATION:



5. Develop a Java program to create a class bank that maintains two kinds of account for its customers. one of them called savings account and the other current account. The saving account provides compound interest and withdraw facilities but no cheque book facility, The current account provides cheque book facility but no interest current account holder should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.
Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achive the following tasks.

ⓐ Accept deposite from customer and update the balance.

ⓑ Display the balance.

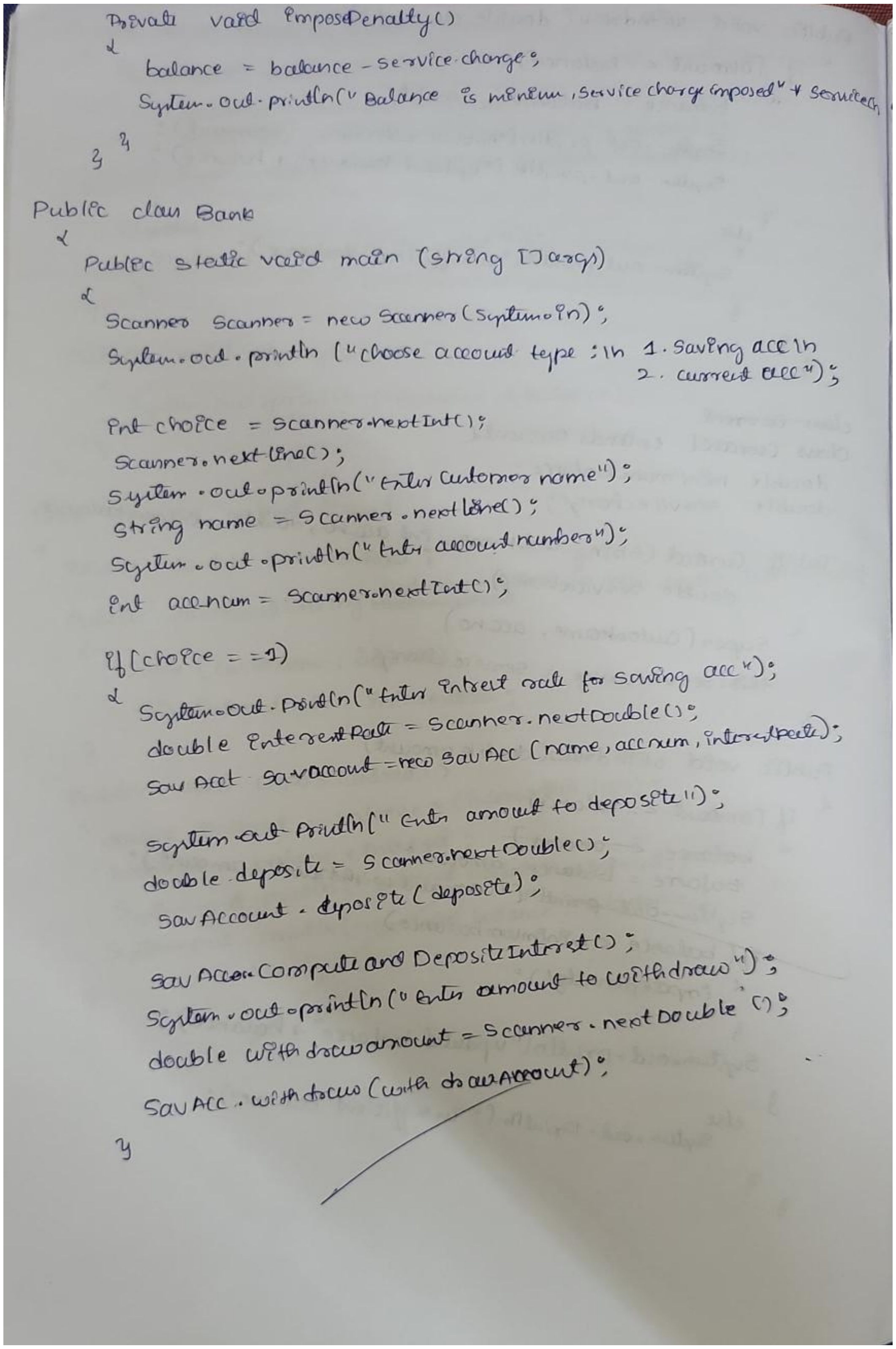
ⓒ Compute and deposite interest

ⓓ Permit withdrawal and update the balance

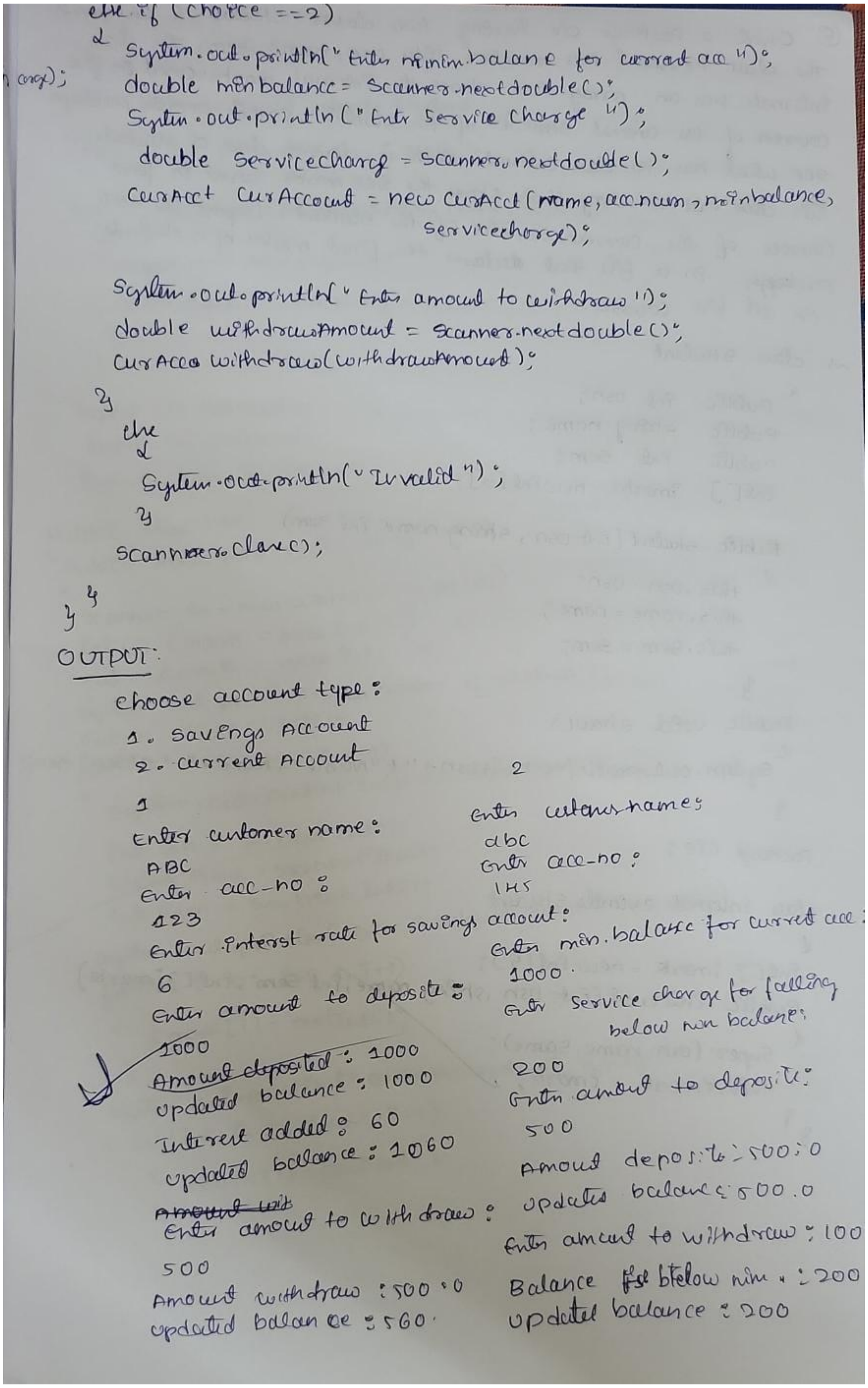
check for the minimum balance, impose penalty if necessary and update the balance.

```
→ import java.util.Scanner;
  class Account {
    string cust-name;
    int acc-no;
    string acc-type;
    double balance;

  Public Account (string cusname, int account-no, string type
  {
     Cust-name = cus.name;
     acc-no = account-no;
     acc-type = type-acc;
     balance = 0.0;
  }
Public void deposit( double amount )
 {
    if (amount >0)
     {
        balance = balance +account ;
        System.out.println ("Amount deposited:" +amount);
        System.out.println (" updated balance:"+balance);
     }
    el~~se~~
     System.out.println("Invalid");
     }
 }
Public void displaybalance ()
 {
   System.out.println ("Balance"+ balance);
 }
}
class Savacct extend Account
{
 Private double interestRate ;
 Public SavAccount(String ~~CustomerName~~ cust_name , int acc-no , double Interestrate)
 {
   Super (custname , acc-no );
   this.interestrate = interestRate;
 }
 Public void DepositInterest()
 {
  double interest = balance * (interestRate /100) ;
  balance = balane + interest ;
  System.out.println (" Interest added:" + interest);
  System.out.println("updated balance:"+ balance);
 }
```

```
Public void withdraw( double amount)
 {
   if (amount <= balance)
     {
        balance = balance - amont;
        System.out.println("Amount withdraw: " +amount);
        System.out.println ("updated balance" + balance);
     }
     else
      {
       System.out.println("Insufficient balance");
      }
   }
 }

~~class Current~~
Class Curacct extends Account{
  double minimumbalance;
  double servicecharge;

  Public Curacct (String custname, int acc_no, double minimumbalance,
        double servicecharge)
   {
     Super(Custoname, accno)
     this.servicecharge = service charge;
   }
  Public void withdraw(double amount)
  {
    if (amount <= balance)
     {
       ~~balance <= amount}~~
       balance = balance - amount;
       System.out.println(" Amount withdraw" +amount);
       if( balance < minimumbalance)
       { imposepenalty();
       }
       System.out.println("updated balance" +balance);
    }
     else
       System.out.Println (" Insufficient balance");

     }
  }
```



```
  Private void imposePenalty()
  {
     balance = balance - service charge;
     System.out.println(" Balance is minimum, Service charge imposed" + serviceCh
   }
 }

Public class Bank
 {
  Public static void main (String []args)
  {
   Scanner Scanner = new Scanner(System.in);
   System.out.println ("choose account type :\n 1. Saving acc\n
                                               2. current acc");

   int choice = Scanner.nextInt();
   Scanner.nextLine();
   System.out.println("Enter Customer name");
   String name = Scanner.nextLine();
   System.out.println("Enter account number");
   int accnum = Scanner.nextInt();

   if(choice ==1)
   {
     System.out.Println("Enter interest rate for saving acc");
     double interestRate = Scanner.nextDouble();
     Sav Acct savaccount = new SavAcc (name, accnum, interestRate);

     system.out.Println(" Enter amount to deposite");
     double deposite = Scanner.nextDouble();
     SavAccount . deposite ( deposite);

     SavAcc.ComputeandDepositInterest();
     System.out.println(" Enter amount to withdraw");
     double withdrawamount = Scanner.nextDouble ();
     SavAcc . withdraw (withdrawAmount);

   }
```



```
else if (choice ==2)
 {
   System.out.println(" Enter minim.balance for current acc ");
   double minbalance= Scanner.nextdouble();
   System.out.println(" Enter service charge ");
    double servicecharge = Scanner.nextdouble();
   CurAcct CurAccount = new CurAcct(name, acc.num, minbalance,
                              servicecharge);

  System.out.println(" Enter amount to withdraw ");
  double withdrawAmount = Scanner.nextdouble();
  CurAcco withdraw(withdrawAmount);
 }
  else
  {
   System.out.println(" Invalid ");
  }
  Scanner.close();
 }
}
```

OUTPUT:

```
choose account type:
 1. Savings Account
 2. current Account
 1
 Enter customer name:
 ABC
 Enter acc-no :
 123
 Enter interest rate for savings account:
 6
 Enter amount to deposit :
 1000
 Amount deposited : 1000
 updated balance : 1000
 Interest added : 60
 updated balance : 1060
 ~~Amount wit~~
 Enter amount to withdraw :
 500
 Amount withdraw : 500.0
 updated balance : 560.
```

```
2
Enter customer name:
abc
Enter acc-no :
145
Enter min. balance for current acc :
1000.
Enter service charge for falling
      below min balance:
200
Enter amount to deposit:
500
Amount deposit : 500.0
updated balance : 500.0
Enter amount to withdraw : 100
Balance ~~fal~~ below min. : 200
updated balance : 200
```

Source Code :

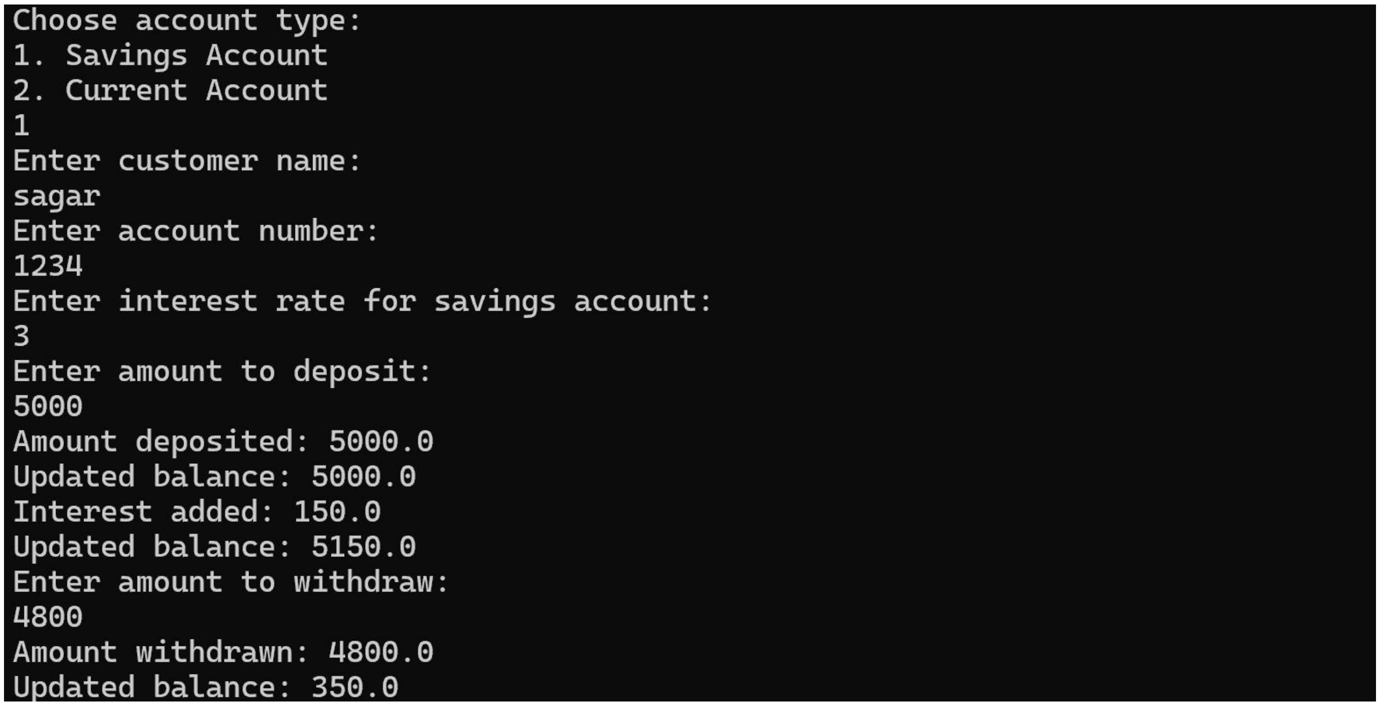
```
import java.util.Scanner;

class Account {
     String customerName;
int accountNumber;
String accountType;
double balance;
     public Account(String customerName, int accountNumber, String accountType)
{         this.customerName = customerName;         this.accountNumber =
accountNumber;         this.accountType = accountType;         this.balance =
0.0;
    }      public void deposit(double
amount) {         if (amount > 0) {
balance += amount;
            System.out.println("Amount deposited: " + amount);
System.out.println("Updated balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount!");
        }
    }
    public void displayBalance() {
        System.out.println("Balance: " + balance);
    }
}  class SavAcct extends
Account {     private double
interestRate;
     public SavAcct(String customerName, int accountNumber, double interestRate)
{         super(customerName, accountNumber, "Savings");
this.interestRate = interestRate;
    }      public void computeAndDepositInterest() {
double interest = balance * (interestRate / 100);
balance += interest;
```

```
        System.out.println("Interest added: " + interest);
System.out.println("Updated balance: " + balance);
    }      public void withdraw(double
amount) {         if (amount <= balance)
{             balance -= amount;
            System.out.println("Amount withdrawn: " + amount);
System.out.println("Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance!");
        }
    }
}  class CurAcct extends
Account {    double
minimumBalance;     double
serviceCharge;
     public CurAcct(String customerName, int accountNumber, double
minimumBalance, double serviceCharge) {         super(customerName,
accountNumber, "Current");         this.minimumBalance = minimumBalance;
this.serviceCharge = serviceCharge;
    }      public void withdraw(double
amount) {         if (amount <= balance)
{             balance -= amount;
            System.out.println("Amount withdrawn: " +
amount);             if (balance < minimumBalance) {
imposePenalty();
            }
            System.out.println("Updated balance: " + balance);
} else {
            System.out.println("Insufficient balance!");
        }
    }      private void
imposePenalty() {         balance
-= serviceCharge;
        System.out.println("Balance fell below minimum. Service charge imposed: "
+ serviceCharge);
    }
}
```

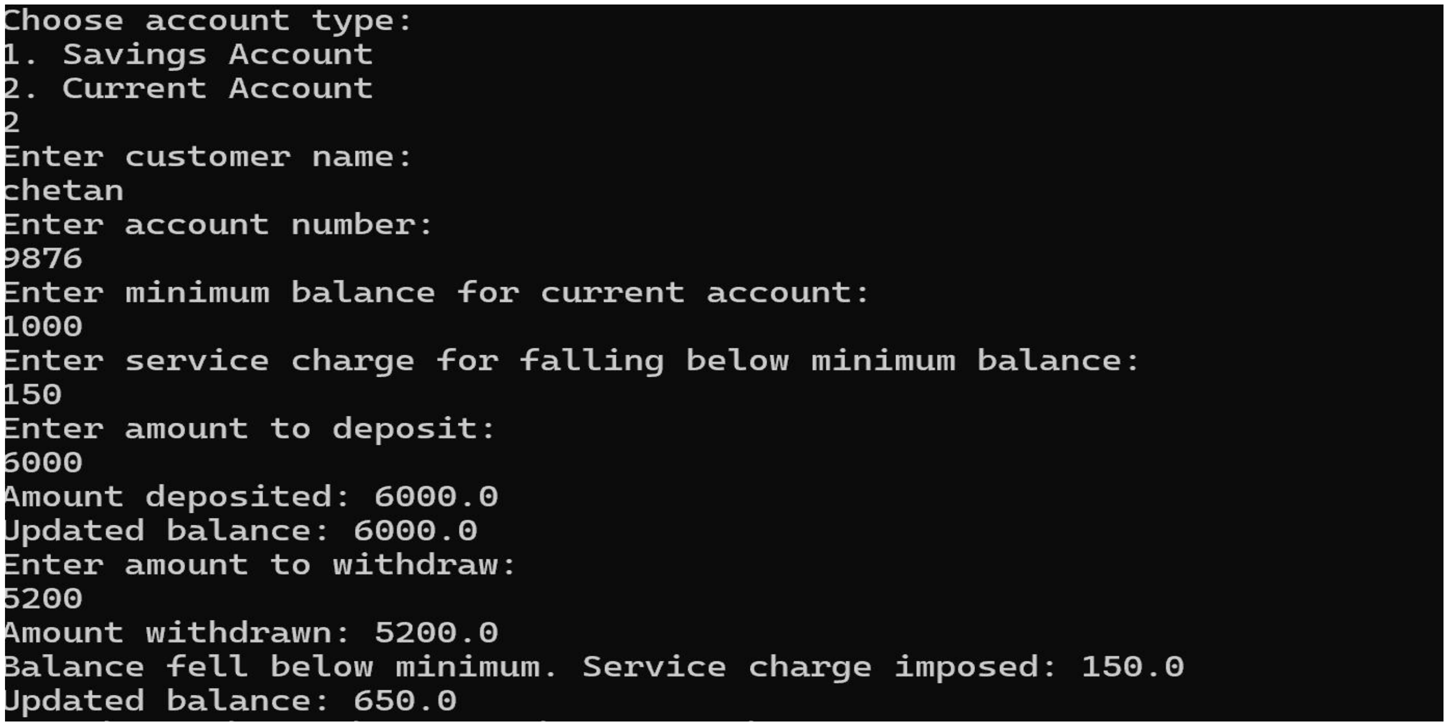
```
 public class Bank
{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Choose account type:\n1. Savings Account\n2.
Current Account");          int choice = scanner.nextInt();
scanner.nextLine();

        System.out.println("Enter customer name: ");
        String name = scanner.nextLine();
        System.out.println("Enter account number: ");
int accNum = scanner.nextInt();

        if (choice == 1) {
            System.out.println("Enter interest rate for savings account: ");
double interestRate = scanner.nextDouble();
            SavAcct savAccount = new SavAcct(name, accNum, interestRate);
            System.out.println("Enter amount to deposit: ");
double deposit = scanner.nextDouble();
savAccount.deposit(deposit);

            savAccount.computeAndDepositInterest();
            System.out.println("Enter amount to withdraw: ");
double withdrawAmount = scanner.nextDouble();
savAccount.withdraw(withdrawAmount);

        } else if (choice == 2) {
            System.out.println("Enter minimum balance for current account: ");
double minBalance = scanner.nextDouble();
            System.out.println("Enter  service  charge  for  falling  below  minimum
balance: ");
            double serviceCharge = scanner.nextDouble();
            CurAcct curAccount = new CurAcct(name, accNum, minBalance,
serviceCharge);

            System.out.println("Enter amount to deposit: ");
double deposit = scanner.nextDouble();
curAccount.deposit(deposit);

            System.out.println("Enter amount to withdraw: ");
double withdrawAmount = scanner.nextDouble();
curAccount.withdraw(withdrawAmount);
```

```java
        } else {
            System.out.println("Invalid account type selected.");
        }

scanner.close();
    }
}
```

Output :



```
Choose account type:
1. Savings Account
2. Current Account
1
Enter customer name:
sagar
Enter account number:
1234
Enter interest rate for savings account:
3
Enter amount to deposit:
5000
Amount deposited: 5000.0
Updated balance: 5000.0
Interest added: 150.0
Updated balance: 5150.0
Enter amount to withdraw:
4800
Amount withdrawn: 4800.0
Updated balance: 350.0
```

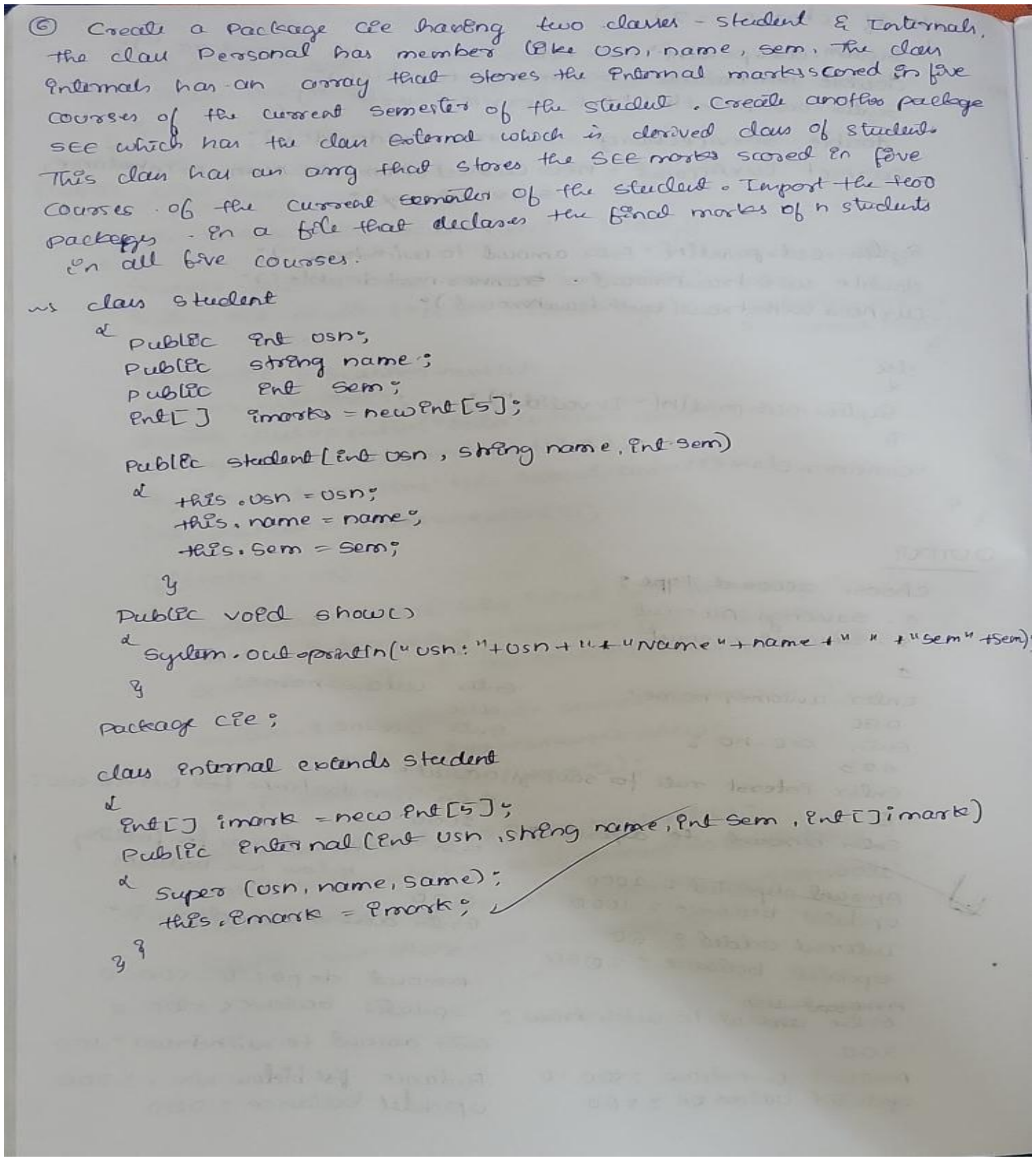


```
Choose account type:
1. Savings Account
2. Current Account
2
Enter customer name:
chetan
Enter account number:
9876
Enter minimum balance for current account:
1000
Enter service charge for falling below minimum balance:
150
Enter amount to deposit:
6000
Amount deposited: 6000.0
Updated balance: 6000.0
Enter amount to withdraw:
5200
Amount withdrawn: 5200.0
Balance fell below minimum. Service charge imposed: 150.0
Updated balance: 650.0
```

PROGRAM 6 :

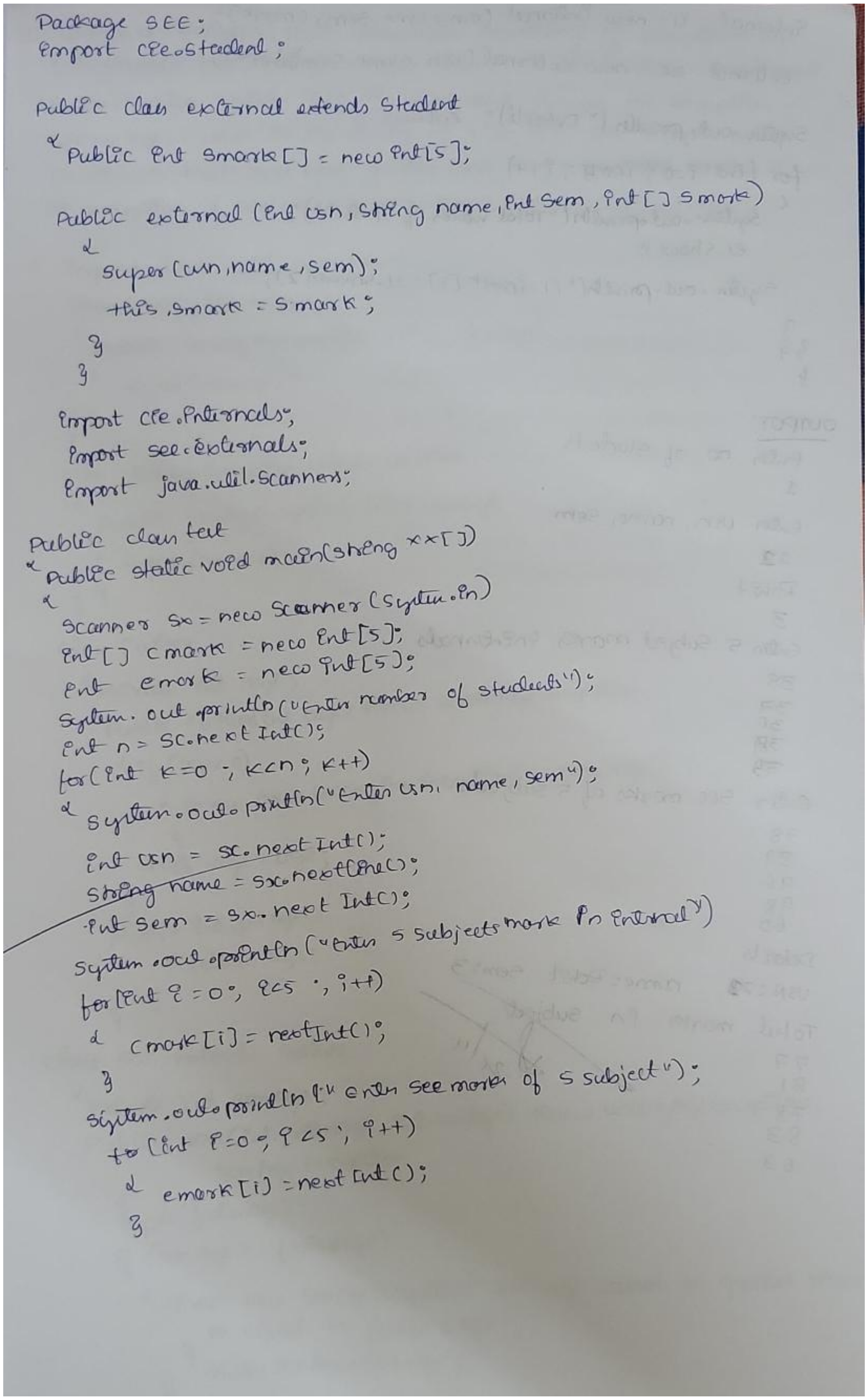
Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

OBSERVATION:

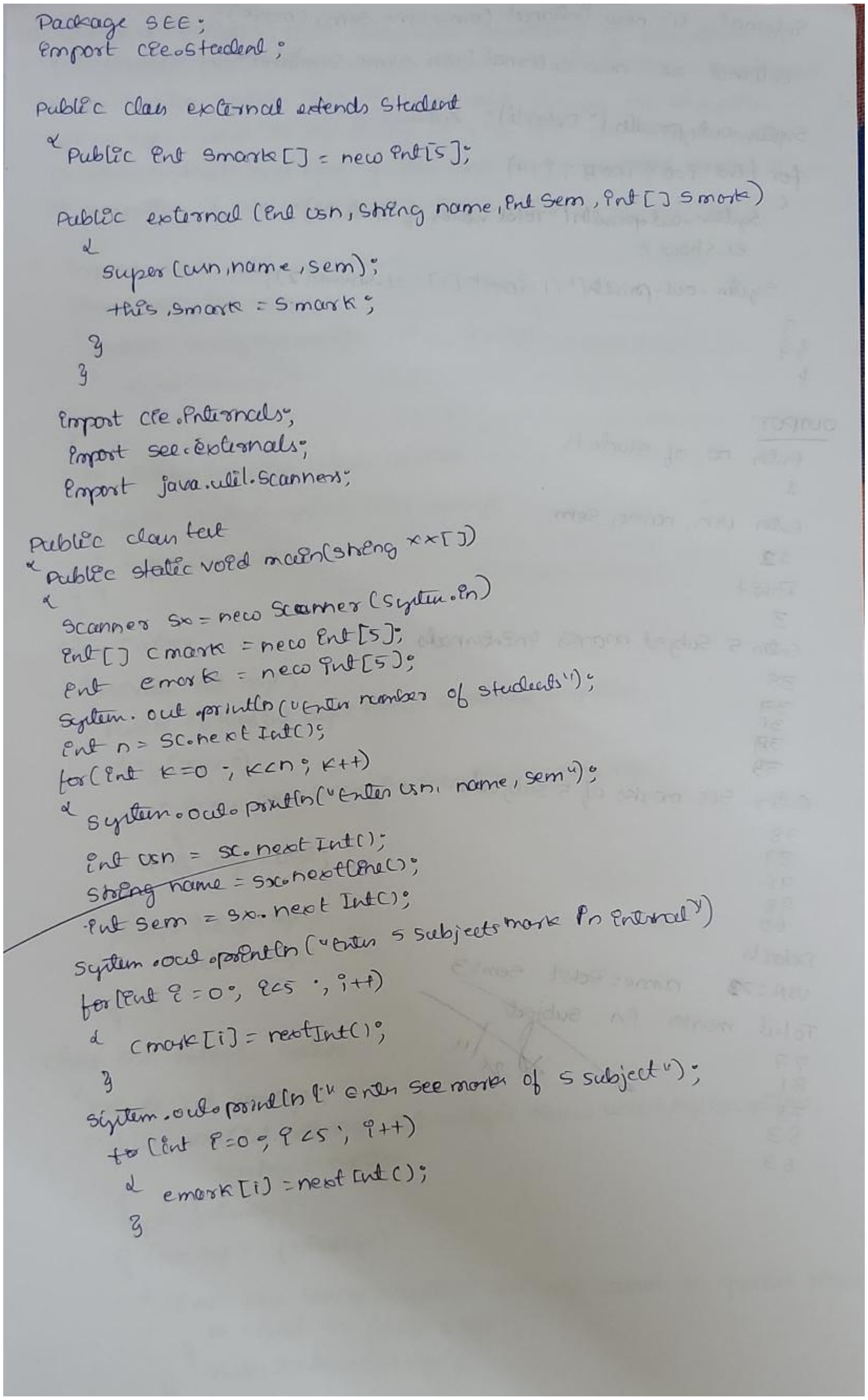


⑥ Create a package cie having two classes - student & Internals, the class Personal has member like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is derived class of student. This class has an arry that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
→ class student
  {
    public int usn;
    public string name;
    public int sem;
    int[] imarks = new int[5];

    public student[int usn, string name, int sem)
    {
       this.usn = usn;
       this.name = name;
       this.sem = sem;
     }
   public void show()
   {
    system.out.println("usn: "+usn+""+"Name"+name+" "+"sem"+sem)
   }
 package cie;

 class internal extends student
 {
  int[] imark = new int[5];
  public internal(int usn, string name, int sem, int[] imark)
  {
    super (usn, name, same);
    this.imark = imark;
  }
 }
```



```
Package SEE;
import cie.student;

public class external extends Student
{
 public int smark[] = new int[5];

  public external (int usn, String name, int sem, int[] smark)
  {
    super(usn, name, sem);
    this.smark = smark;
  }
 }

 import cie.internals;
 import see.externals;
 import java.util.Scanner;

public class test
{ public static void main(String xx[])
 {
  Scanner sx = new Scanner(System.in)
  int[] cmark = new int[5];
  int emark = new int[5];
  System.out.println("Enter number of students");
  int n = sc.nextInt();
  for(int k=0 ; k<n; k++)
  { System.out.println("Enter usn, name, sem");
    int usn = sc.nextInt();
    String name = sx.nextLine();
    int sem = sx.nextInt();
   System.out.println("Enter 5 subjects mark in internal")
   for(int i=0; i<5 ; i++)
   { cmark[i] = nextInt();
   }
  System.out.println("enter see marks of 5 subject");
   for(int i=0; i<5; i++)
    { emark[i] = next int();
    }
```



```
Package SEE;
import cie.student;

Public class external extends Student
{ Public int smark[] = new int[5];

  Public external (int usn, String name, int sem, int[] smark)
   {
     super (usn, name, sem);
     this.smark = smark;

   }
  }

 import cie.internals;
 import see.externals;
 import java.util.scanner;

Public class test
{ public static void main(String xx[])
 {
  Scanner sx = new Scanner (System.in)
  int[] cmark = new int[5];
  int  emark = new int[5];
  System. out.println ("Enter number of students");
  int n = Sc.next Int();
  for(int k=0 ; k<n ; k++)
  {
    System.out.println("Enter usn, name, sem");
    int usn = sc.next Int();
    String name = sx.nextLine();
    int sem = sx. next Int();
   System.out.println ("Enter 5 subjects mark in internal")
   for(int i = 0; i<5 ; i++)
   {  cmark[i] = nextInt();
    }
   System.out.println ("enter see marks of 5 subject");
    for (int i=0; i<5 ; i++)
     {  emark[i] = next int();
     }
internal i1 = new internal (usn,name,sem, cmark);

  external e1 = new external (usn ,name, sem,smark);

System.out.println ("Details"); e.show()
for (int i=0; i<5; i++)
{
   System.out.println("Total marks of student");
    e1.show();
  System.out.println(" i1.imark [i] + e1.smark[2]);

 }
}}
}
```

OUTPUT:

```
Enter no of students
1
Enter usn, name, sem
 21
Rohit
 3
Enter 5 subject marks internals
38
37
30
34
39
Enter see marks of 5 subjects.
78
89
96
98
60
Details
USN : 21      name: Rohit  sem: 3
Total marks in subject
77
81
78
83
63
```

[illegible]

Source Code :

```
import CIE.Internals;
import SEE.External;
import java.util.Scanner;
 public class Studentmarks
{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter number of students:
");         int n = scanner.nextInt();
scanner.nextLine();


        Internals[] cieStudents = new Internals[n];
        External[] seeStudents = new External[n];
                 for (int i = 0; i <
n; i++) {

            System.out.println("Enter details for CIE Student " + (i + 1) + ":
");
            System.out.print("USN: ");
            String usn = scanner.nextLine();
            System.out.print("Name: ");
            String name = scanner.nextLine();
System.out.print("Semester: ");
int sem = scanner.nextInt();             int[]
internalMarks = new int[5];
            System.out.println("Enter internal marks for 5 courses: ");
for (int j = 0; j < 5; j++) {
                internalMarks[j] = scanner.nextInt();
            }
            cieStudents[i] = new Internals(usn, name, sem, internalMarks);
scanner.nextLine();


            System.out.println("Enter details for SEE Student " + (i + 1) + ":
");
            System.out.print("USN: ");
usn = scanner.nextLine();
System.out.print("Name: ");
name = scanner.nextLine();
```

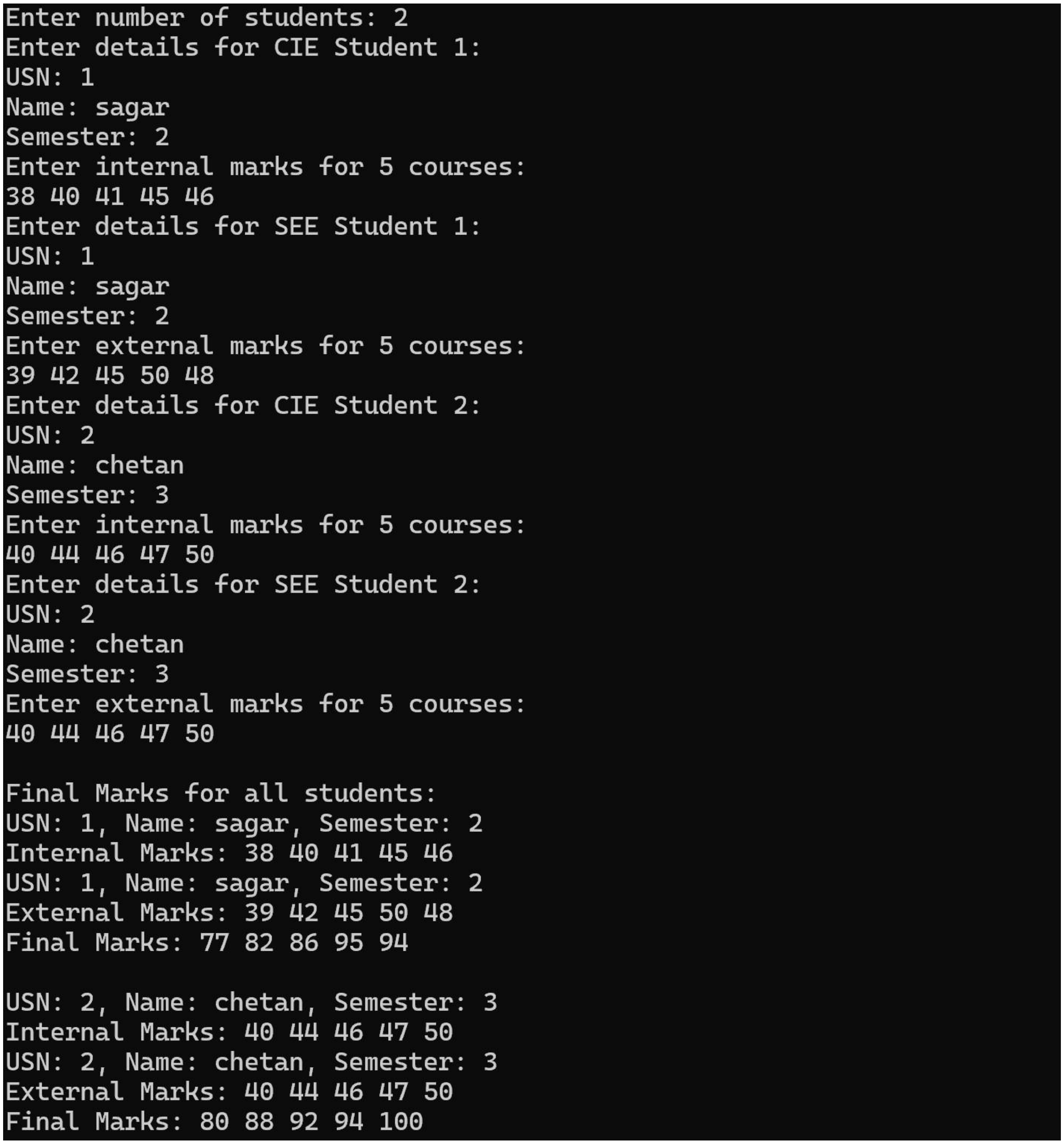
```
            System.out.print("Semester: ");
sem = scanner.nextInt();
            int[] externalMarks = new int[5];
            System.out.println("Enter external marks for 5 courses:
");             for (int j = 0; j < 5; j++) {
externalMarks[j] = scanner.nextInt();
            }             seeStudents[i] = new External(usn, name, sem,
externalMarks);               scanner.nextLine();
        }


        System.out.println("\nFinal Marks for all students:");
         for (int i = 0; i < n; i++)
{

cieStudents[i].displayStudentDetails();
cieStudents[i].displayInternalMarks();


seeStudents[i].displayStudentDetails();
seeStudents[i].displayExternalMarks();

                         int[] internalMarks =
cieStudents[i].getInternalMarks();             int[] externalMarks
= seeStudents[i].getExternalMarks();             int[] finalMarks =
new int[5];
             for (int j = 0; j < 5; j++) {
finalMarks[j] = internalMarks[j] + externalMarks[j];             }


            System.out.print("Final Marks: ");
for (int mark : finalMarks) {
                System.out.print(mark + " ");
            }
            System.out.println("\n");
        }

        scanner.close();
    }
}
```

```
package CIE;
 public class Internals extends Student
{
        private int[] internalMarks = new
int[5];
         public Internals(String usn, String name, int sem, int[]
internalMarks) {          super(usn, name, sem); // Call parent constructor
this.internalMarks = internalMarks;
    }

    public void displayInternalMarks() {
System.out.print("Internal Marks: ");
for (int mark : internalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
    }

    public int[] getInternalMarks() {
return internalMarks;
    }
}
```

```
package CIE;
 public class Student
{

    protected String usn;
protected String name;
protected int sem;
     public Student(String usn, String name, int sem)
{         this.usn = usn;         this.name = name;
this.sem = sem;
    }

    public void displayStudentDetails() {
        System.out.println("USN: " + usn + ", Name: " + name + ", Semester: " +
sem);



    }
}
```

```
package SEE;
 import
CIE.Student;
 public class External extends Student {
private int[] externalMarks = new int[5];
     public External(String usn, String name, int sem, int[] externalMarks)
{          super(usn, name, sem);          this.externalMarks =
externalMarks;
    }
    public void displayExternalMarks() {
System.out.print("External Marks: ");
for (int mark : externalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
    }      public int[]
getExternalMarks() {          return
externalMarks;
    }
}
```

Output :

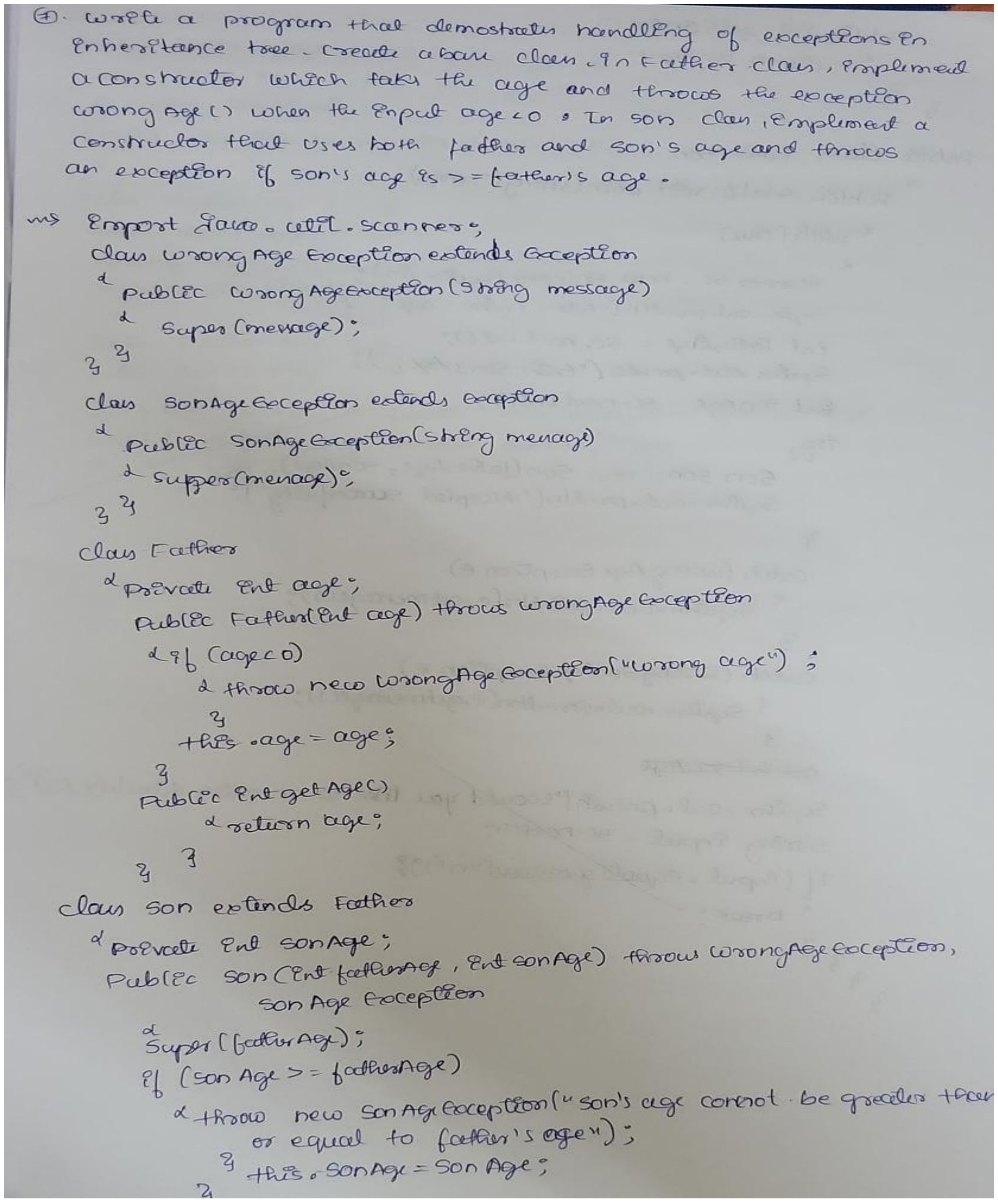


```
Enter number of students: 2
Enter details for CIE Student 1:
USN: 1
Name: sagar
Semester: 2
Enter internal marks for 5 courses:
38 40 41 45 46
Enter details for SEE Student 1:
USN: 1
Name: sagar
Semester: 2
Enter external marks for 5 courses:
39 42 45 50 48
Enter details for CIE Student 2:
USN: 2
Name: chetan
Semester: 3
Enter internal marks for 5 courses:
40 44 46 47 50
Enter details for SEE Student 2:
USN: 2
Name: chetan
Semester: 3
Enter external marks for 5 courses:
40 44 46 47 50

Final Marks for all students:
USN: 1, Name: sagar, Semester: 2
Internal Marks: 38 40 41 45 46
USN: 1, Name: sagar, Semester: 2
External Marks: 39 42 45 50 48
Final Marks: 77 82 86 95 94

USN: 2, Name: chetan, Semester: 3
Internal Marks: 40 44 46 47 50
USN: 2, Name: chetan, Semester: 3
External Marks: 40 44 46 47 50
Final Marks: 80 88 92 94 100
```

PROGRAM 7 :

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that uses both father and son’s age and throws an exception if son’s age is >=father’s age.

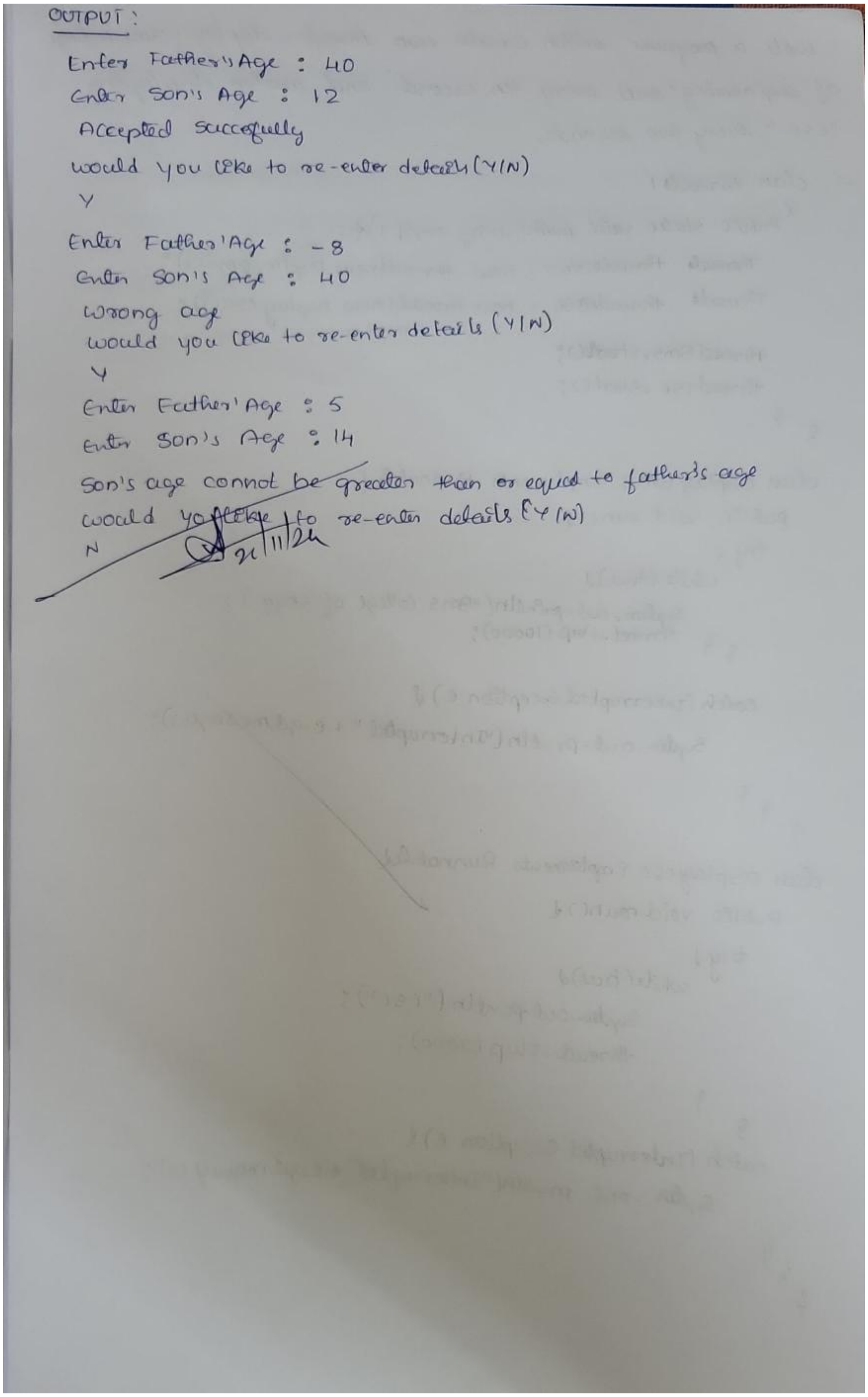
OBSERVATION:



7. Write a program that demostrates handling of exceptions in inheritance tree. Create a base class. In Father class, implement a constructor which takes the age and throws the exception wrong Age () when the input age<0. In son class, implement a constructor that uses both father and son's age and throws an exception if son's age is >= father's age.

```
ans Import java.util.Scanner;
    class wrongAgeException extends Exception
     {
      public wrongAgeException(String message)
       {
         Super(message);
    }  }
    class SonAgeException extends exception
     {
      Public SonAgeException(String message)
      { Supper(message);
     } }
    class Father
     { Private int age;
      Public Father(int age) throws wrongAgeException
       { if (age<0)
          { throw new WrongAgeException("Wrong age");
          }
         this.age = age;
       }
      Public int getAge()
           { return age;
      }   }
   class Son extends Father
    { Private int sonAge;
     Public Son(int fatherAge, int sonAge) throws WrongAgeException,
                 Son Age Exception
      {
      Super(fatherAge);
      if (son Age >= fatherAge)
        { throw new SonAgeException("son's age cannot be greater than
               or equal to father's age");
        } this.SonAge = Son Age;
      }
```

```
        Public int getSonAge()
          { return SonAge;
        }
    }

Public class FatherSon
    {
      Public static void main(String[] args)
        { while(true)
          {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter Father's age : ")
           int FatherAge = sc.nextInt();
           System.out.println("Enter Son's Age : ");
           int sonAge = sc.nextInt();
           try{
                Son son = new Son(fatherAge, sonAge);
                System.out.println("Accepted successfully");
           }
            catch (wrongAgeException e)
              { System.out.println(e.getMessage());
             }
           catch (~~wrong~~ SonAgeException e)
              { System.out.println(e.getMessage());
              }
           ~~catch (sonAge~~
         System.out.print("would you like to re-enter details(Y/N)"
         String input = sc.next();
         if (input.equalsIgnoreCase("n")){
           { break;
         }
      }
    }
  }
```



OUTPUT :

```
Enter Father's Age : 40
Enter Son's Age : 12
Accepted Succefully
would you like to re-enter details(Y/N)
Y
Enter Father'Age : -8
Enter Son's Age : 40
Wrong age
would you like to re-enter details (Y/N)
Y
Enter Father'Age : 5
Enter Son's Age : 14
Son's age connot be greater than or equal to father's age
would you like to re-enter details (Y/N)
N
```

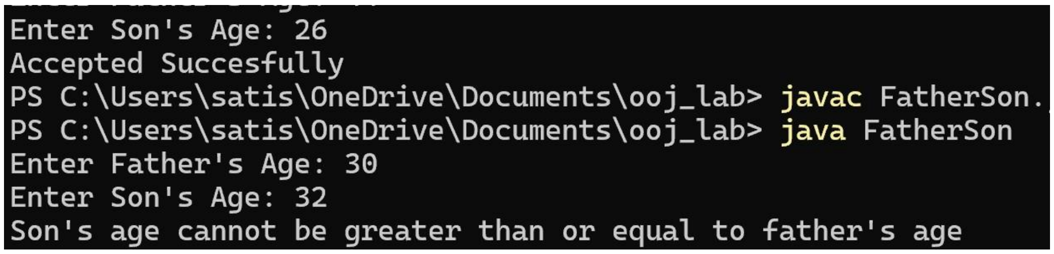
21/11/24

Source Code :

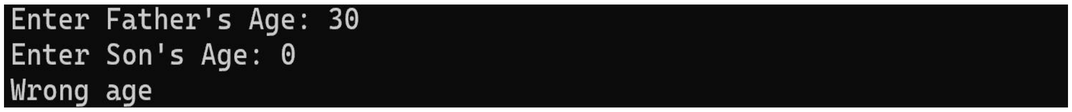
```
import java.util.Scanner;
 class WrongAgeException extends Exception {
public WrongAgeException(String message) {
super(message);
    }
}  class SonAgeException extends Exception
{     public SonAgeException(String
message) {          super(message);
    }
}  class Father {      int age;     public
Father(int age) throws WrongAgeException {
if (age <= 0) {             throw new
WrongAgeException("Wrong age");
        }
this.age = age;
    }     public int
getAge() {
return age;
    }
}  class Son extends
Father {      int sonAge;
    public Son(int fatherAge, int sonAge) throws
WrongAgeException, SonAgeException {          super(fatherAge);
if (sonAge >= fatherAge) {
            throw new SonAgeException("Son's age cannot be greater than or equal
to father's age");
        }
if(sonAge <= 0){
        throw new WrongAgeException("Wrong age");
        }
this.sonAge = sonAge;
    }     public int
getSonAge() {
return sonAge;
```

```
    }
}

public class FatherSon{
    public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
System.out.print("Enter Father's Age: ");             int
fatherAge = sc.nextInt();
            System.out.print("Enter Son's Age: ");
int sonAge = sc.nextInt();             try {
                Son son = new Son(fatherAge, sonAge);
                System.out.println("Accepted Succesfully");
            }
            catch (WrongAgeException e) {
                System.out.println(e.getMessage());
            }
            catch (SonAgeException e) {
                System.out.println(e.getMessage());
            }

        }
  }
```

Output :



```
Enter Son's Age: 26
Accepted Succesfully
PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac FatherSon.
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java FatherSon
Enter Father's Age: 30
Enter Son's Age: 32
Son's age cannot be greater than or equal to father's age
```

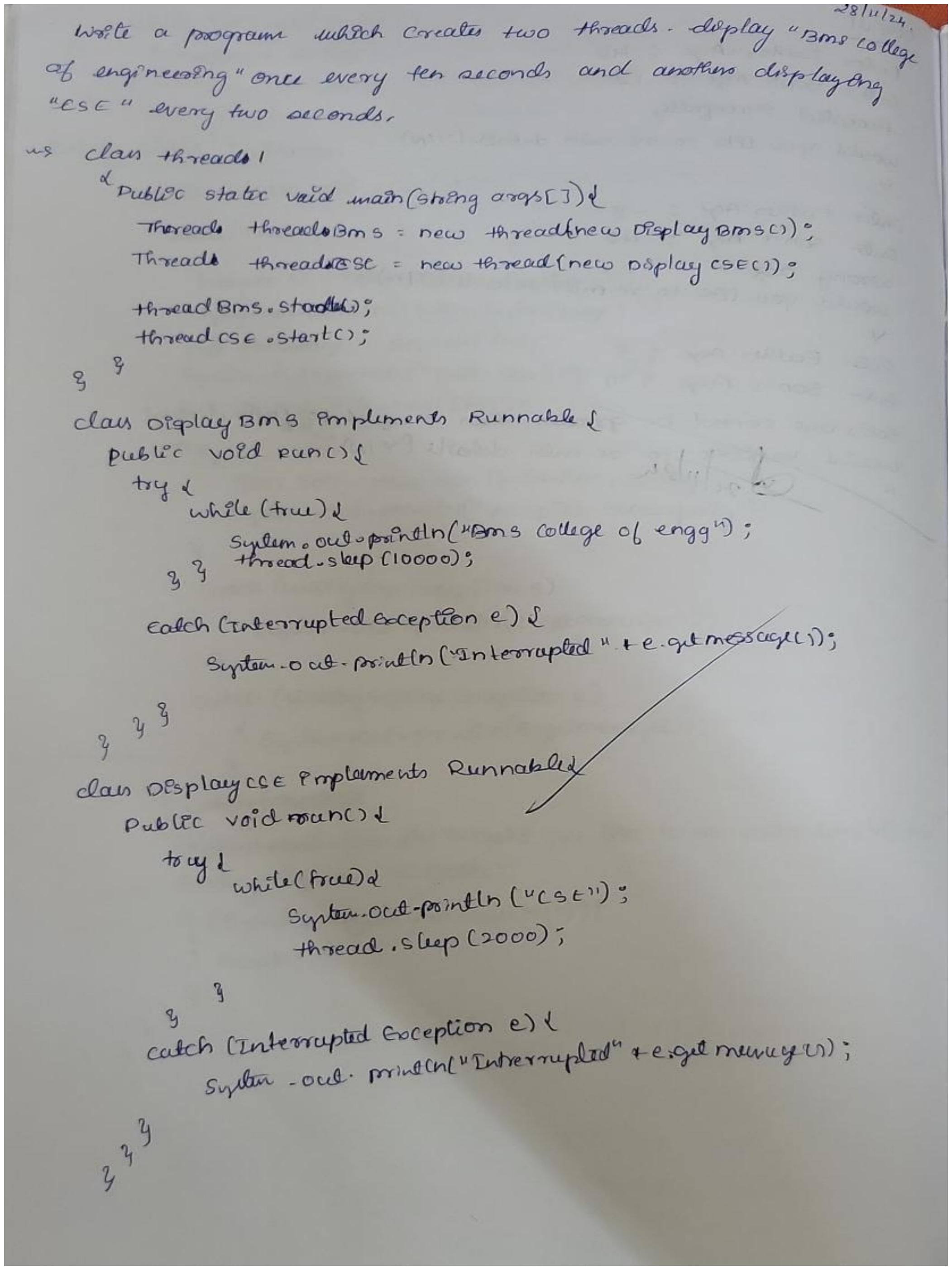


```
Enter Father's Age: 30
Enter Son's Age: 0
Wrong age
```

PROGRAM 8 :

Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

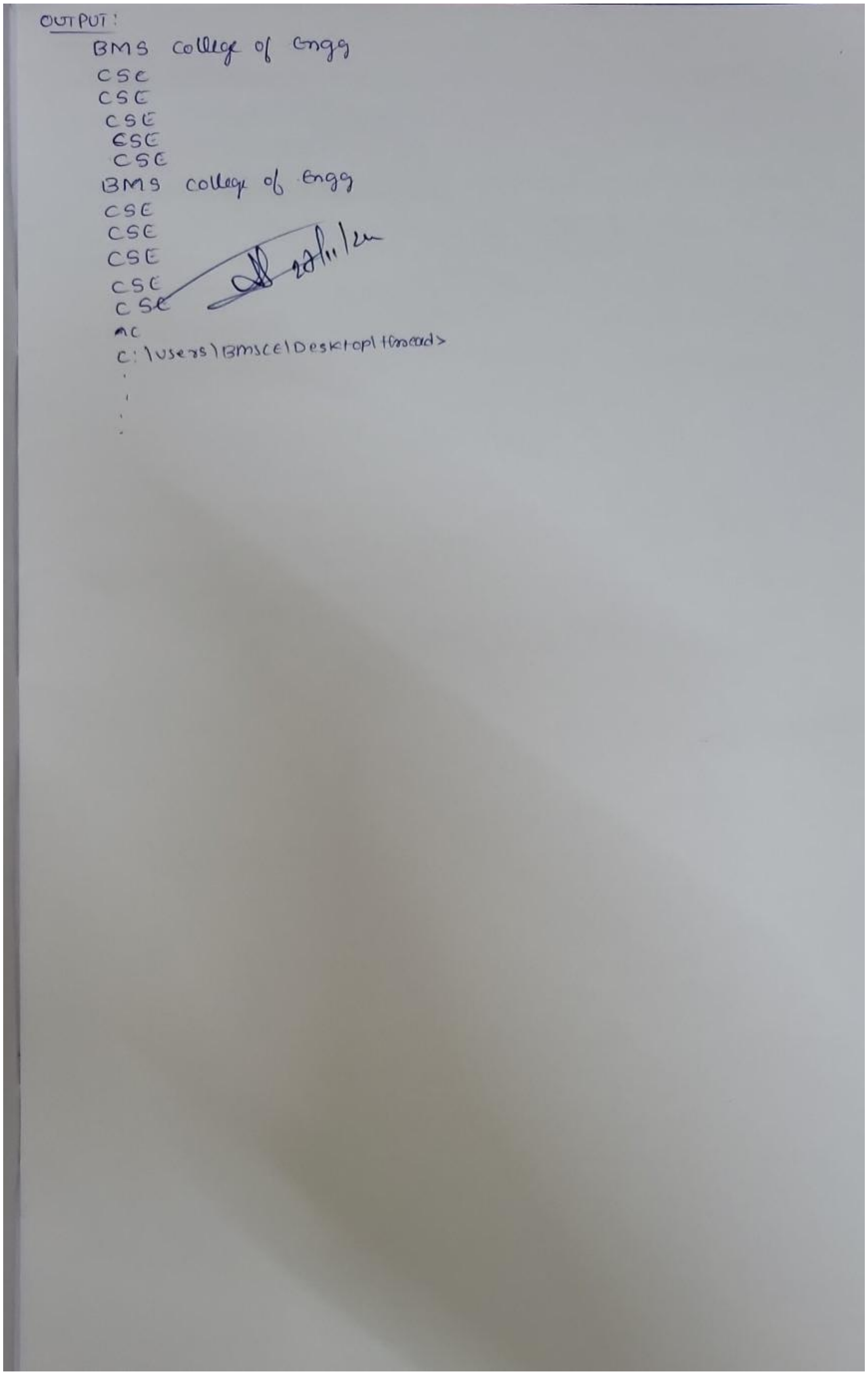
OBSERVATION:



28/11/24.

Write a program which creates two threads. display "Bms college of engineering" once every ten seconds and another displaying "CSE" every two seconds.

```
class threads1
 {
  Public static void main(String args[]){
    Threads threadsBms = new thread(new DisplayBms());
    Threads threadsCSC = new thread(new Display CSE());

    threadBms.start();
    threadCSE.start();
  }
}

class DisplayBms implements Runnable{
   public void run(){
    try{
       while(true){
          System.out.println("Bms college of engg");
          thread.sleep(10000);
    }}
     catch(InterruptedException e){
         System.out.println("Interrupted" + e.getMessage());

  }}}
}
class DisplayCSE implements Runnable{
   Public void run(){
     try{
        while(true){
              System.out.println("CSE");
              thread.sleep(2000);

        }
     }
     catch(Interrupted Exception e){
         System.out.println("Interrupted" + e.getMessage());

 }}}
```


OUTPUT:

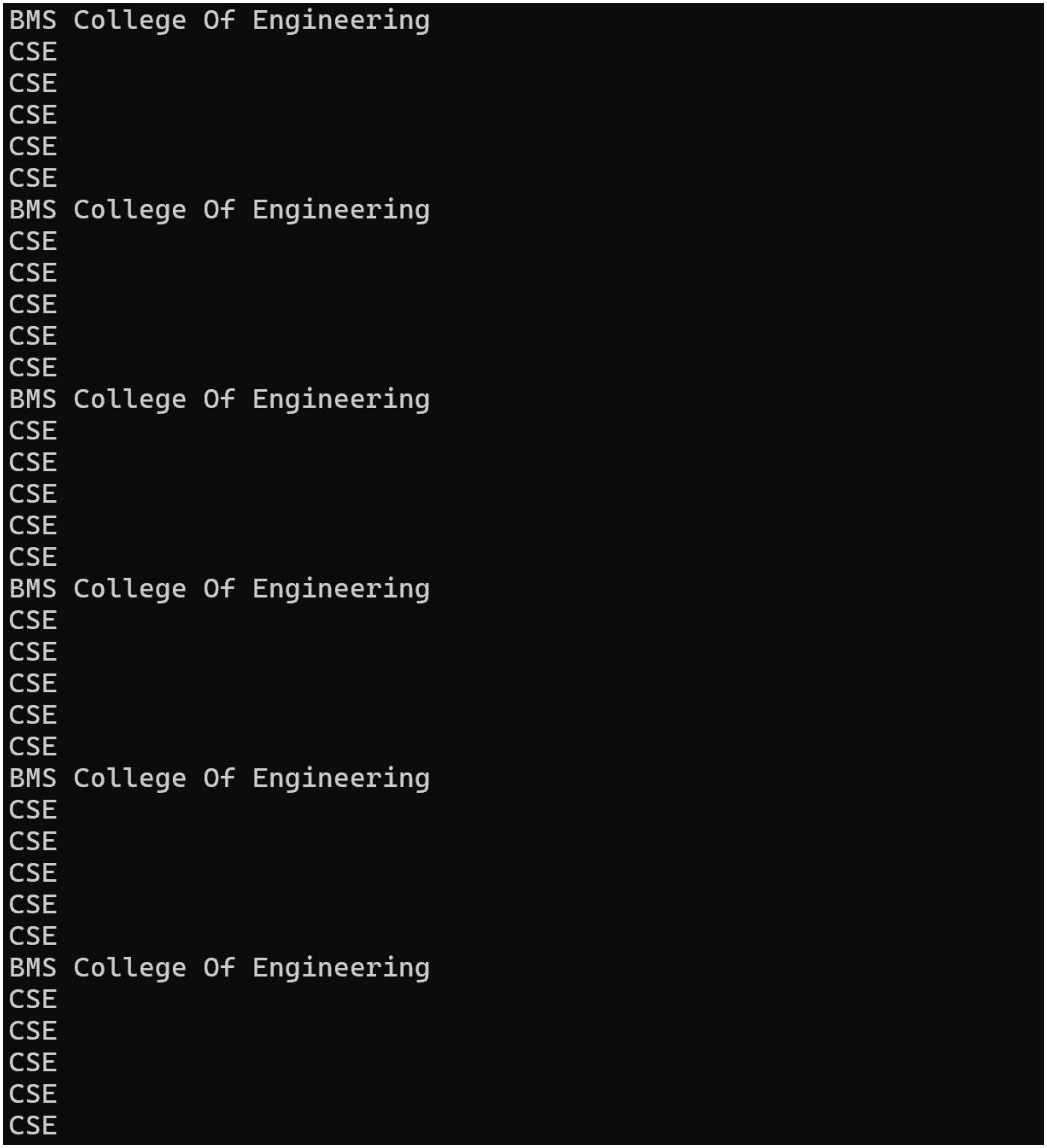
```
BMS college of Engg
CSE
CSE
CSE
CSE
CSE
BMS college of Engg
CSE
CSE
CSE
CSE
CSE
^C
C:\users\BMSCE\Desktop\thread>
.
.
.
.
```

[illegible]

Source Code :

```
class ThreadDemo extends Thread{
public void run(){     while(true){
    System.out.println("BMS College Of Engineering");
try{
    Thread.sleep(10000);
    }
catch(InterruptedException e){
    e.printStackTrace();
    }
     }
  }
}  class CSEThread extends
Thread{     public void run(){
while(true){
    System.out.println("CSE");
try{
    Thread.sleep(2000);
    }
catch(InterruptedException e){
    e.printStackTrace();
    }
     }
  }
}  public class
Demo{
    public static void main(String[] args){
    ThreadDemo t1 = new
ThreadDemo();     CSEThread t2 = new
CSEThread();     t1.start();
t2.start();
     }
 }
```

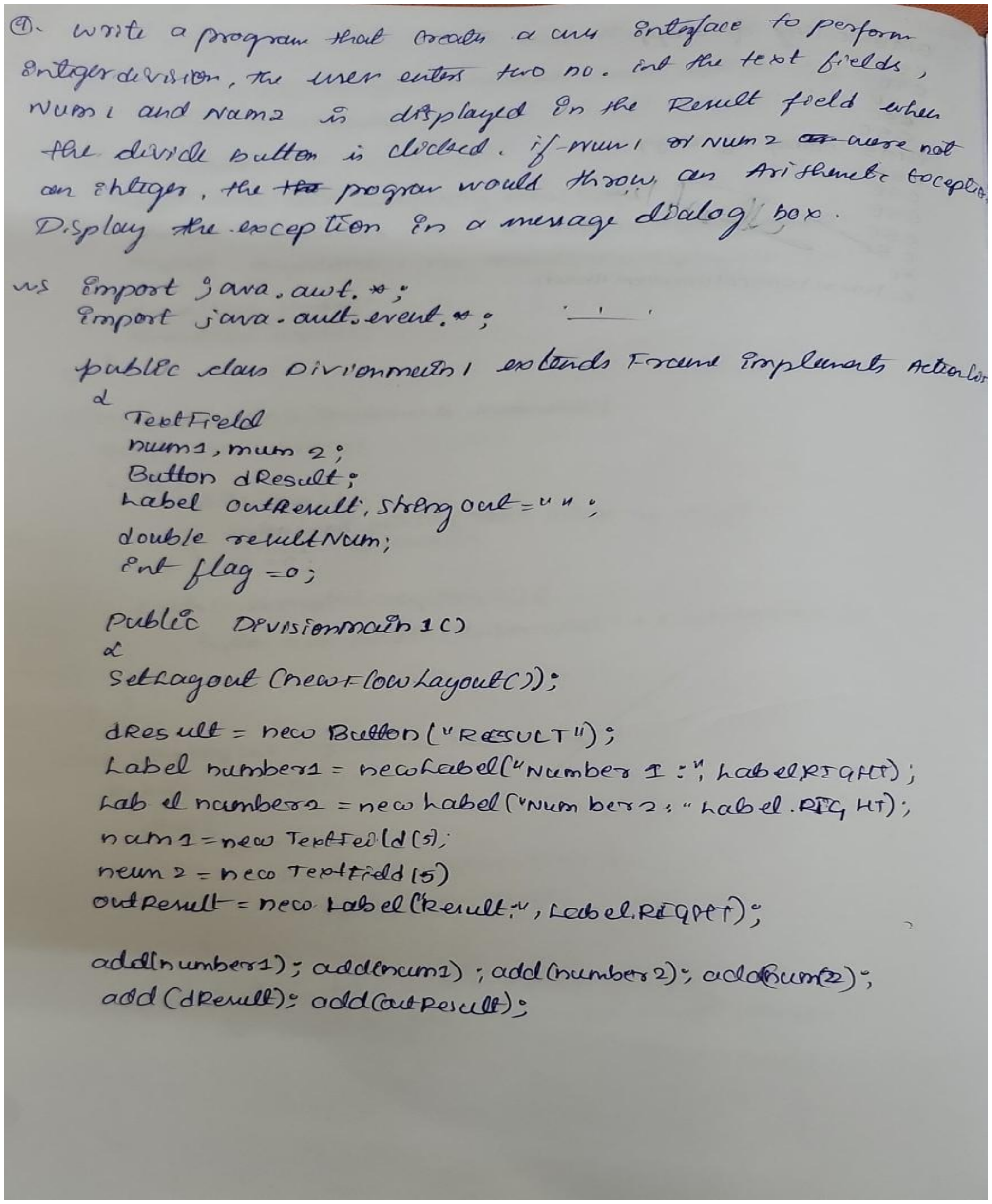
Output :


```
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
```

PROGRAM 9 :

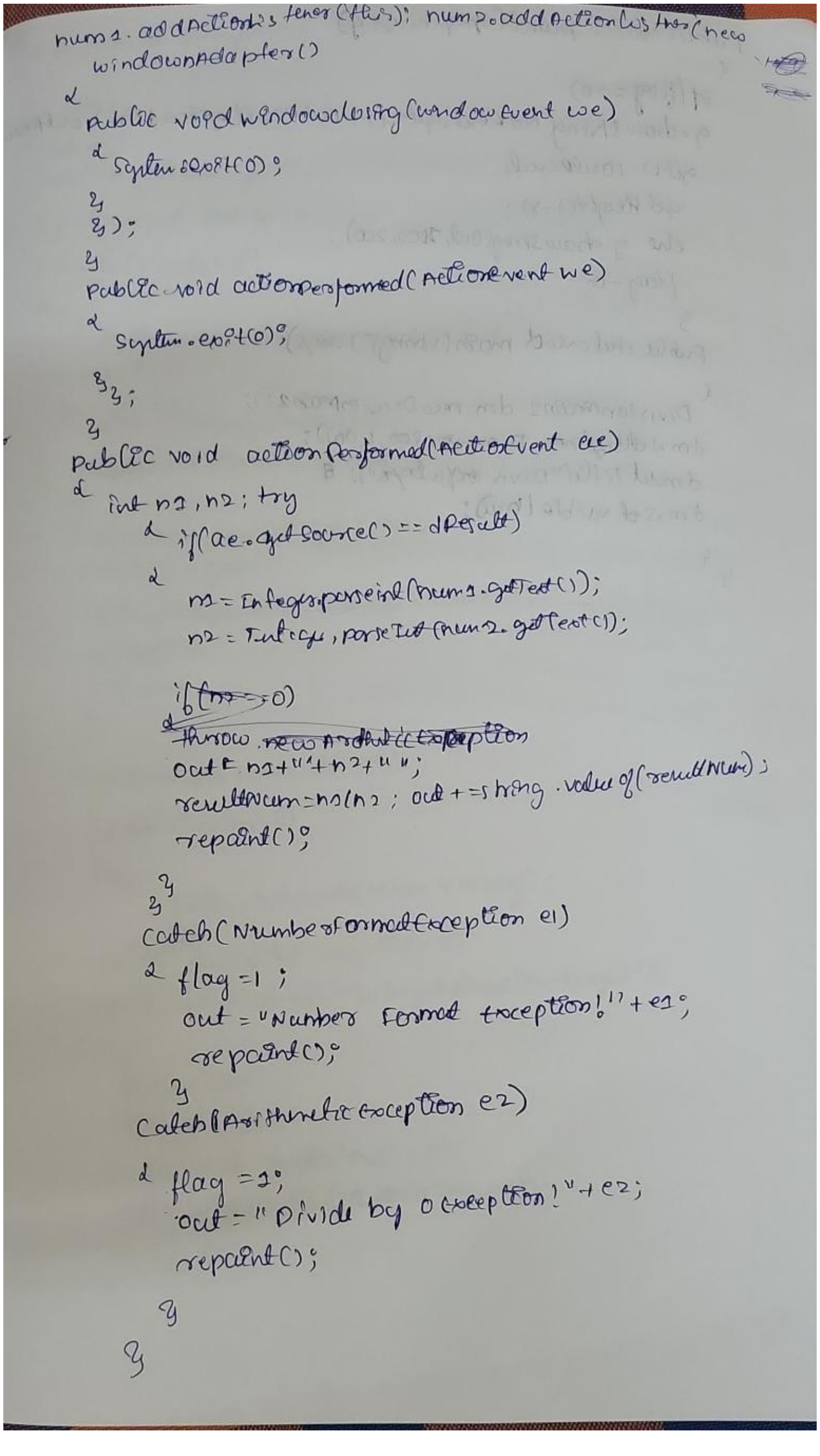
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialogbox.

Observation :

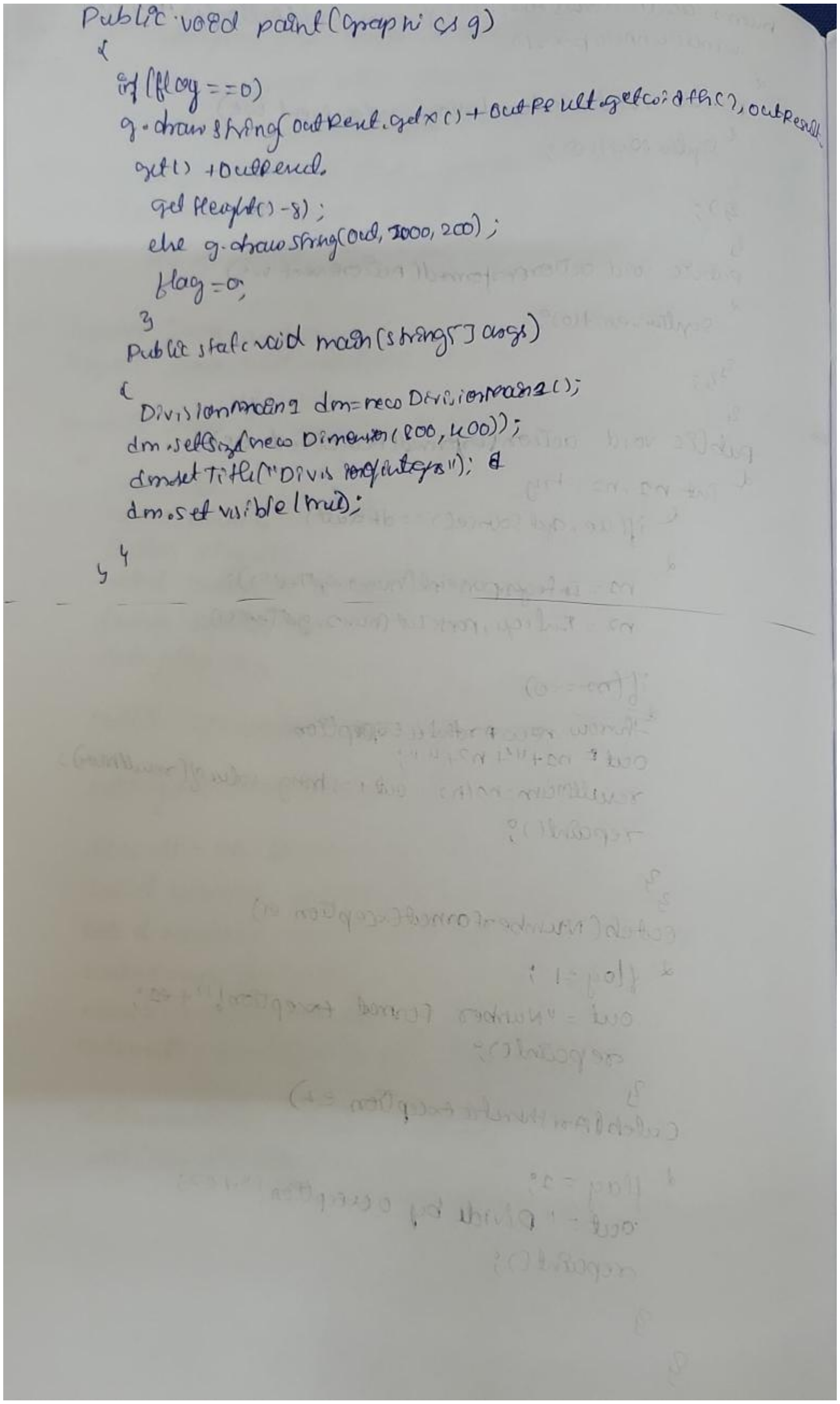


9. write a program that creats a user interface to perform integer division, the user enters two no. int the text fields, Num1 and Num2 is displayed in the Result field when the divide button is clicked. if Num1 or Num2 ~~or~~ were not an integer, the ~~the~~ program would throw an Arithmetic Exception Display the exception in a message dialog box.

```
as  import java.awt.*;
    import java.awt.event.*;

    public class Divisionmain1 extends Frame implements ActionLis
     {
       TextField
       num1, num 2;
       Button dResult;
       Label outResult, String out = " ";
       double resultNum;
       int flag = 0;

      Public Divisionmain 1()
      {
      SetLayout (new FlowLayout());

      dResult = new Button ("RESULT");
      Label number1 = new Label("Number 1 :", Label.RIGHT);
      Label number2 = new Label ("Number2 : " Label.RIGHT);
      num1 = new TextFeild(5);
      neum 2 = neo TextField (5)
      outResult = neo Label ("Result:", Label.RIGHT);

      add(number1); add(num1) ; add (number 2); add(num(2));
      add (dResult); add(outResult);
```



```
nums1.addActionListener(this); num2.addActionListener(new
    windowAdapter()
{
   public void windowclosing(windowEvent we)
   { System.exit(0);
   }
   });
  }
  public void actionPerformed(ActionEvent we)
  {
     System.exit(0);
   }};
  }
public void actionPerformed(ActionEvent ae)
{
   int n1,n2; try
     { if(ae.getSource()==dResult)
       {
          n1=Integer.parseInt(num1.getText());
          n2=Integer.parseInt(num2.getText());

          ~~if(n2==0)~~
          ~~{ throw new ArithmeticException~~
          out= n1+" "+n2+" ";
          resultNum=n1/n2; out+=String.valueOf(resultNum);
          repaint();
       }
      }
      catch(NumberFormatException e1)
      { flag=1;
        out="Number Format Exception! "+e1;
        repaint();
       }
      catch(ArithmeticException e2)

      { flag=1;
        out="Divide by 0 Exception! "+e2;
        repaint();
       }
   }
```



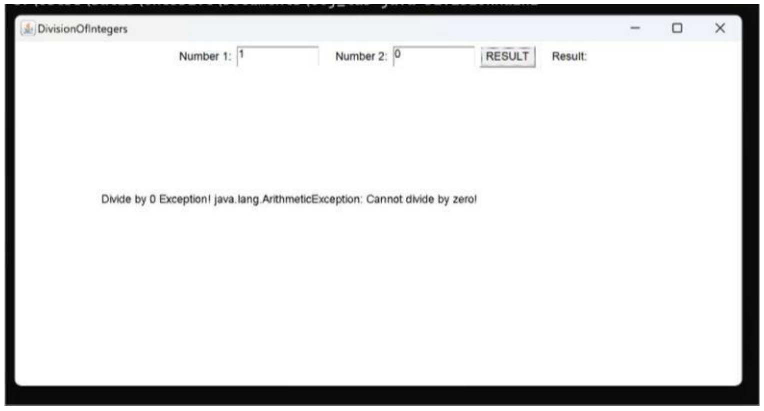
```
Public void paint(Graphics g)
{
  if(flag==0)
  g.drawString(outResult.getx()+outResult.getWidth(),outResult.
   get()+outResult.
    getHeight()-8);
   else g.drawString(out,1000,200);
    flag=0;
  }
 Public static void main(String[] args)

 {
  DivisionMain1 dm=new DivisionMain1();
 dm.setSize(new Dimension(800,400));
 dm.setTitle("Division of integers"); 
 dm.setVisible(true);

 }
}
```

Source Code :

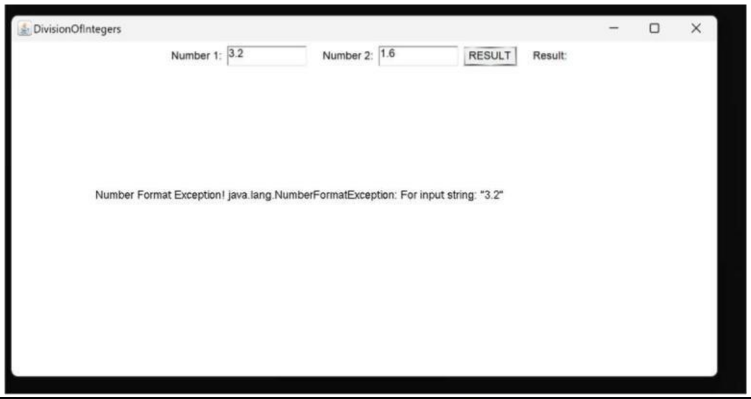
```
import java.awt.*;  import java.awt.event.*; public class DivisionMain1 extends Frame
implements ActionListener {
   TextField num1, num2;
   Button dResult;
   Label outResult;      String out =
"";      double resultNum;      int
flag = 0;       public
DivisionMain1()
   {        setLayout(new FlowLayout());


      dResult = new Button("RESULT");
      Label number1 = new Label("Number 1:", Label.RIGHT);         Label number2
= new Label("Number 2:", Label.RIGHT);         num1 = new
TextField(5);        num2 = new TextField(5);        outResult = new
Label("Result:",   Label.RIGHT);                          add(number1);
add(num1);       add(number2);       add(num2);       add(dResult);
add(outResult);                              num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this);             addWindowListener(new
WindowAdapter()
      {           public void windowClosing(WindowEvent we)
         {
            System.exit(0);
         }
      });
       setTitle("Division Calculator");        setSize(300,
200);          setVisible(true);
   }      public void actionPerformed(ActionEvent ae)
   {        int n1, n2;          try          {             if
(ae.getSource() == dResult)
          {                 n1 = Integer.parseInt(num1.getText());                  n2 =
Integer.parseInt(num2.getText());
               if (n2 == 0)
              {                    throw new ArithmeticException("Cannot divide by zero!");                 }

              out = n1 + " / " + n2 + " = ";                  resultNum =
(double) n1 / n2;                  out += String.valueOf(resultNum);                  repaint();
          }         }        catch(NumberFormatException e1)
       {
          flag = 1;             out = "Number Format Exception! " + e1;             repaint();
       }
       catch(ArithmeticException e2)
       {            flag = 1;             out = "Divide by 0 Exception! " + e2;
repaint();
       }
   }
   public void paint(Graphics g)
   {         if(flag
== 0)
       {
          g.drawString(out, outResult.getX() + outResult.getWidth(), outResult.getY() +
outResult.getHeight() - 8);
       }        else
{
          g.drawString(out, 100, 200);             flag = 0;
}
   }
```

```
   public static void main(String[] args)
   {
      DivisionMain1  dm=new  DivisionMain1();               dm.setSize(new  Dimension(800,400));
dm.setTitle("DivisionOfIntegers");         dm.setVisible(true);

    }
}
```

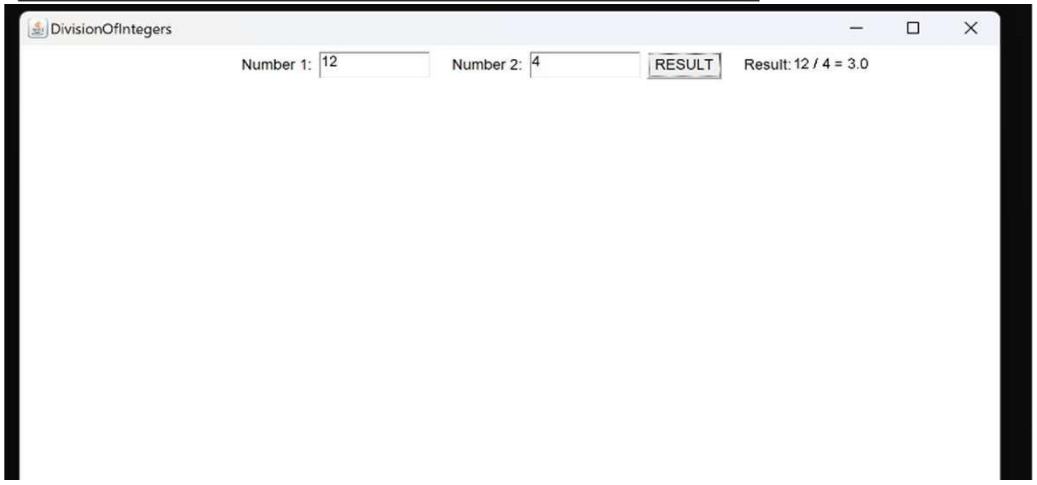
Output:



```
DivisionOfIntegers
Number 1: 1    Number 2: 0    RESULT    Result:

Divide by 0 Exception! java.lang.ArithmeticException: Cannot divide by zero!
```



```
DivisionOfIntegers
Number 1: 3.2    Number 2: 1.6    RESULT    Result:

Number Format Exception! java.lang.NumberFormatException: For input string: "3.2"
```



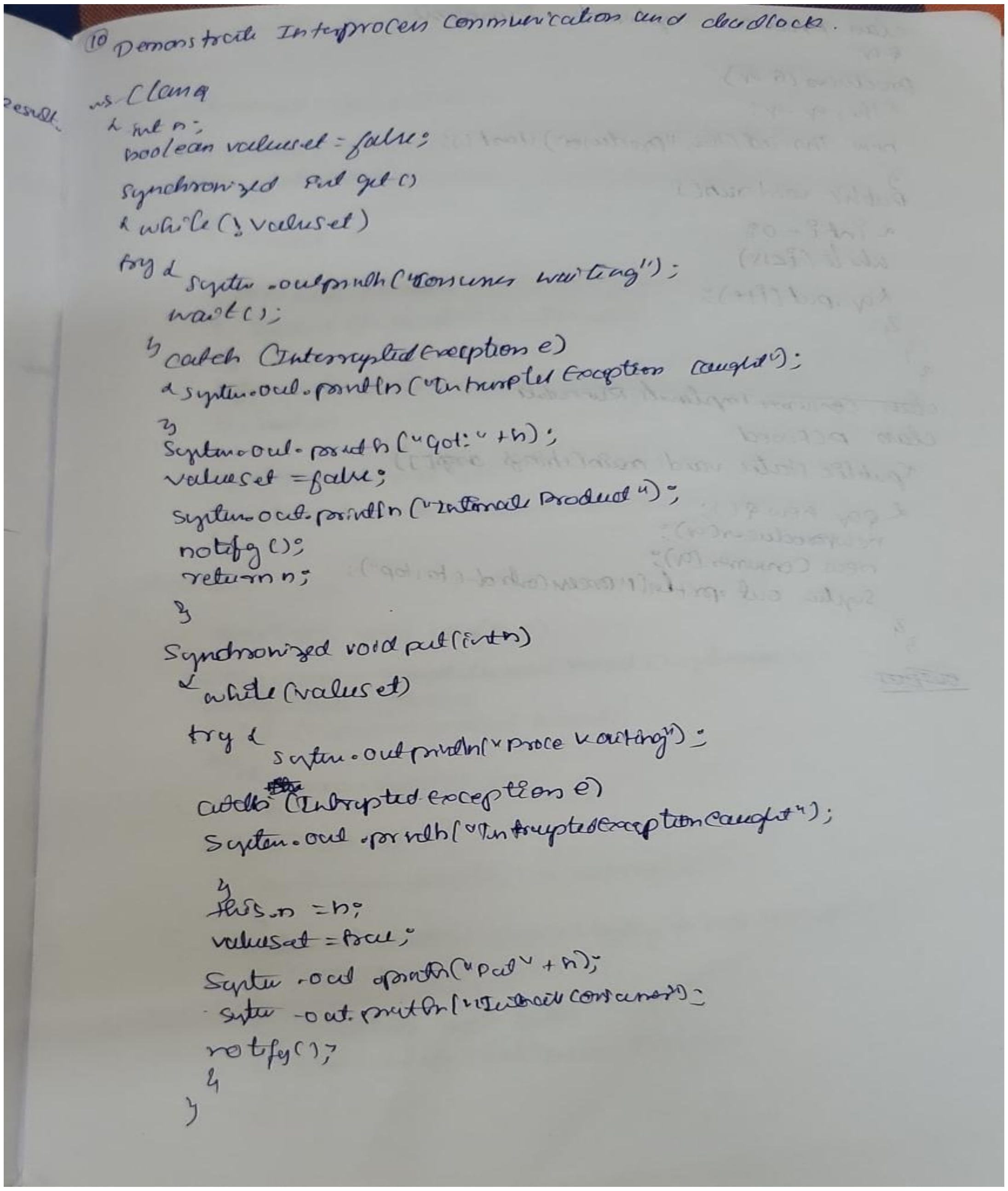
```
DivisionOfIntegers
Number 1: 12    Number 2: 4    RESULT    Result: 12 / 4 = 3.0
```

PROGRAM 10 :

Demonstrate Inter process Communication and deadlock

1. Demonstration of Inter process Communication Observation

OBSERVATION:

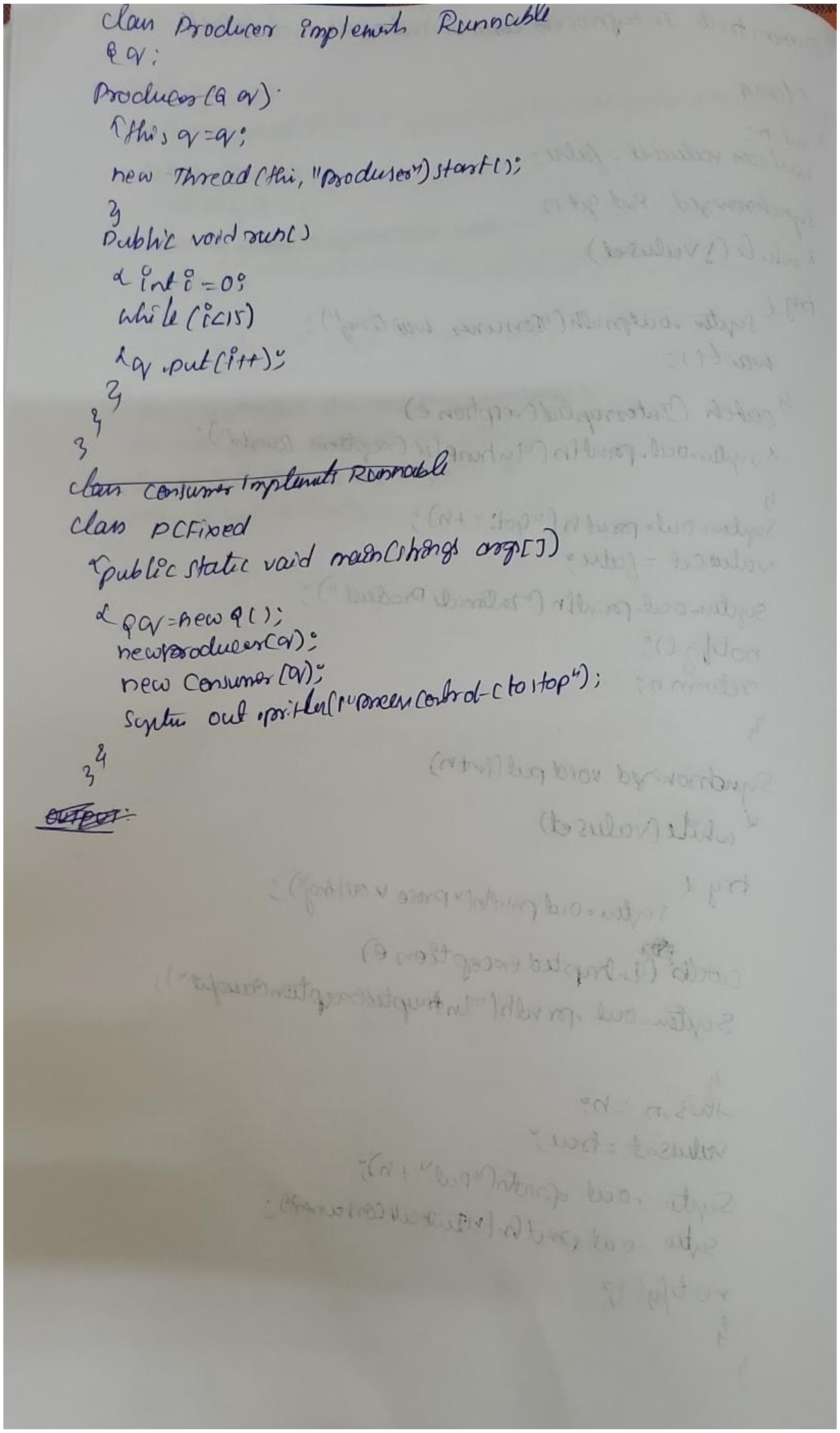


⑩ Demonstrate Interprocess Communication and deadlock.

Result

```
class Q
{ int n;
  boolean valueset = false;
  synchronized int get()
  { while(!valueset)
  try {
      System.out.println("Consumer waiting");
      wait();
    } catch (InterruptedException e)
    { system.out.println("Interrupted Exception caught");
    }
    System.out.println("got:" +n);
    valueset = false;
    system.out.println("Intimate Producer");
    notify();
    return n;
  }
  Synchronized void put(int n)
  { while (valueset)
    try {
          system.out.println("Producer waiting");
     catch (Intrupted exception e)
     System.out.println("InterruptedException caught");

     }
     this.n = n;
     valueset = true;
     System.out.println("Put" + n);
     system.out.println("Intimate Consumer");
     notify();
     }
}
```

60

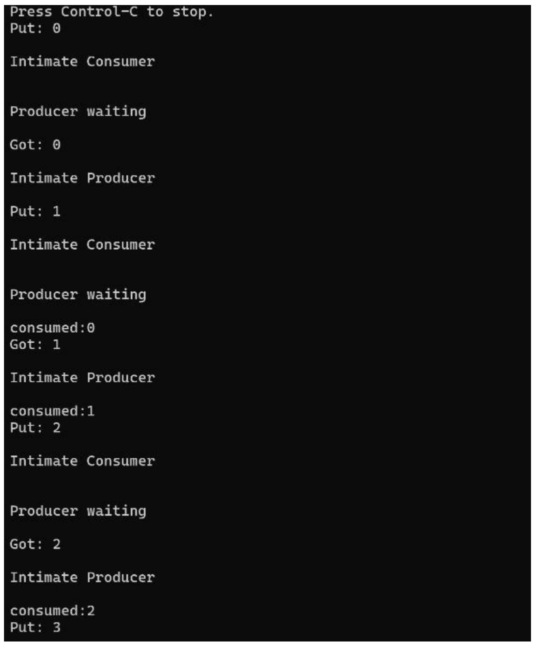


```
class Producer implements Runnable
{ Q q;
Producer(Q q)
 {this.q=q;
 new Thread(this,"Producer").start();
 }
 public void run()
 { int i=0;
 while (i<15)
 { q.put(i++);
 }
 }
}
~~class Consumer implements Runnable~~
class PCFixed
 {public static void main(String args[])
 { Q q=new Q();
 new Producer(q);
 new Consumer(q);
 System.out.println("press control-c to stop");
 }
}
```

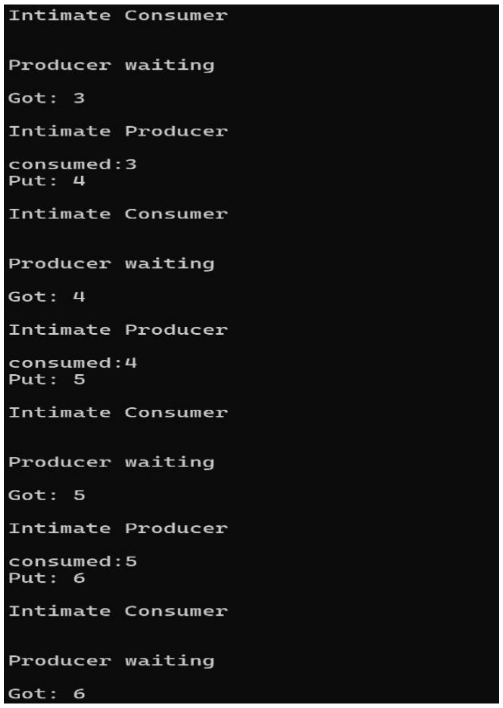
~~OUTPUT:~~

```
class Q { int n; boolean valueSet
= false;   synchronized
int get() {
while(!valueSet) try {
System.out.println("\nConsumer waiting\n"); wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught"); }
System.out.println("Got: " + n); valueSet = false;
System.out.println("\nIntimate Producer\n"); notify(); return
n;
}  synchronized void put(int n) { while(valueSet)
try {
System.out.println("\nProducer waiting\n"); wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
} this.n = n; valueSet
= true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n"); notify();
}  }
class Producer implements Runnable {
Q q;
Producer(Q q) { this.q = q; new Thread(this,
"Producer").start();
}  public void run() { int i =
0; while(i<15) {
q.put(i++);
}
}  }
class Consumer implements Runnable {
Q q;
Consumer(Q q) { this.q = q; new Thread(this,
"Consumer").start();  }
public void run() {  int
i=0; while(i<15) { int
r=q.get();
System.out.println("consumed:"+r); i++; }
}
}  class PCFixed {  public static void main(String
args[]) { Q q = new Q(); new Producer(q); new
Consumer(q);
System.out.println("Press Control-C to stop.");
}
}
```

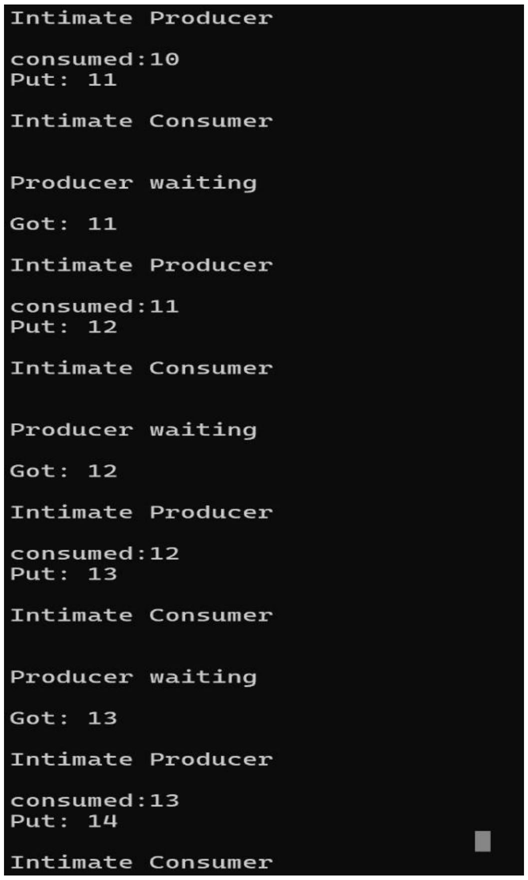
OUTPUT :



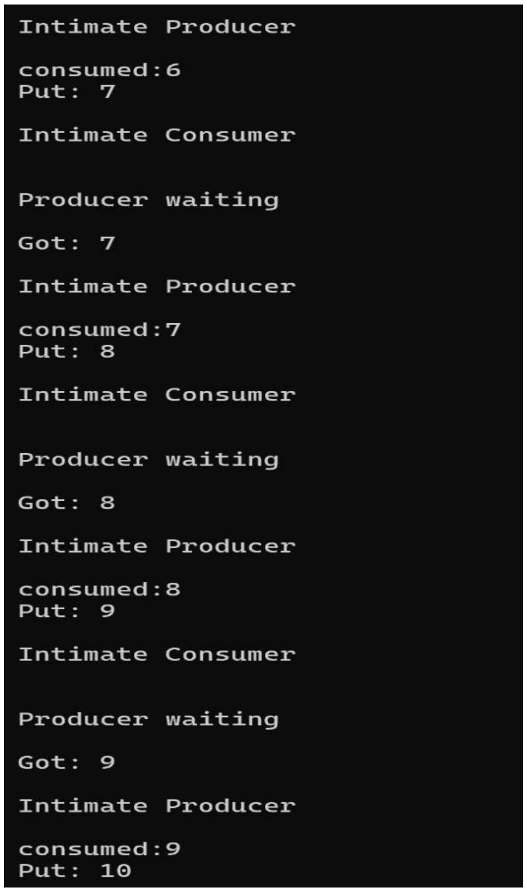
```
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

consumed:0
Got: 1

Intimate Producer

consumed:1
Put: 2

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

consumed:2
Put: 3
```



```
Intimate Consumer


Producer waiting

Got: 3

Intimate Producer

consumed:3
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer

consumed:4
Put: 5

Intimate Consumer


Producer waiting

Got: 5

Intimate Producer

consumed:5
Put: 6

Intimate Consumer


Producer waiting

Got: 6
```



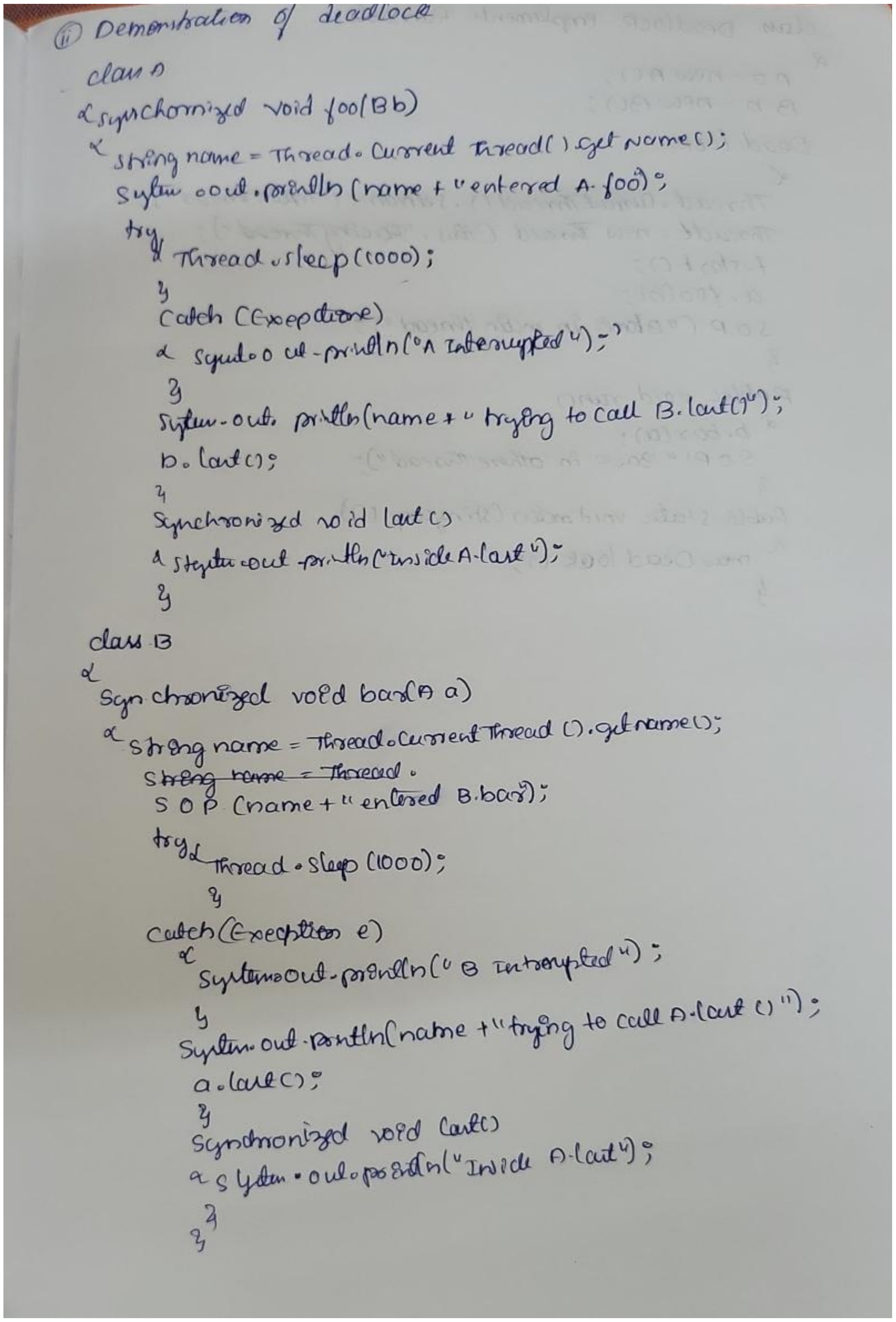
```
Intimate Producer

consumed:10
Put: 11

Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

consumed:11
Put: 12

Intimate Consumer


Producer waiting

Got: 12

Intimate Producer

consumed:12
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

consumed:13
Put: 14

Intimate Consumer
```



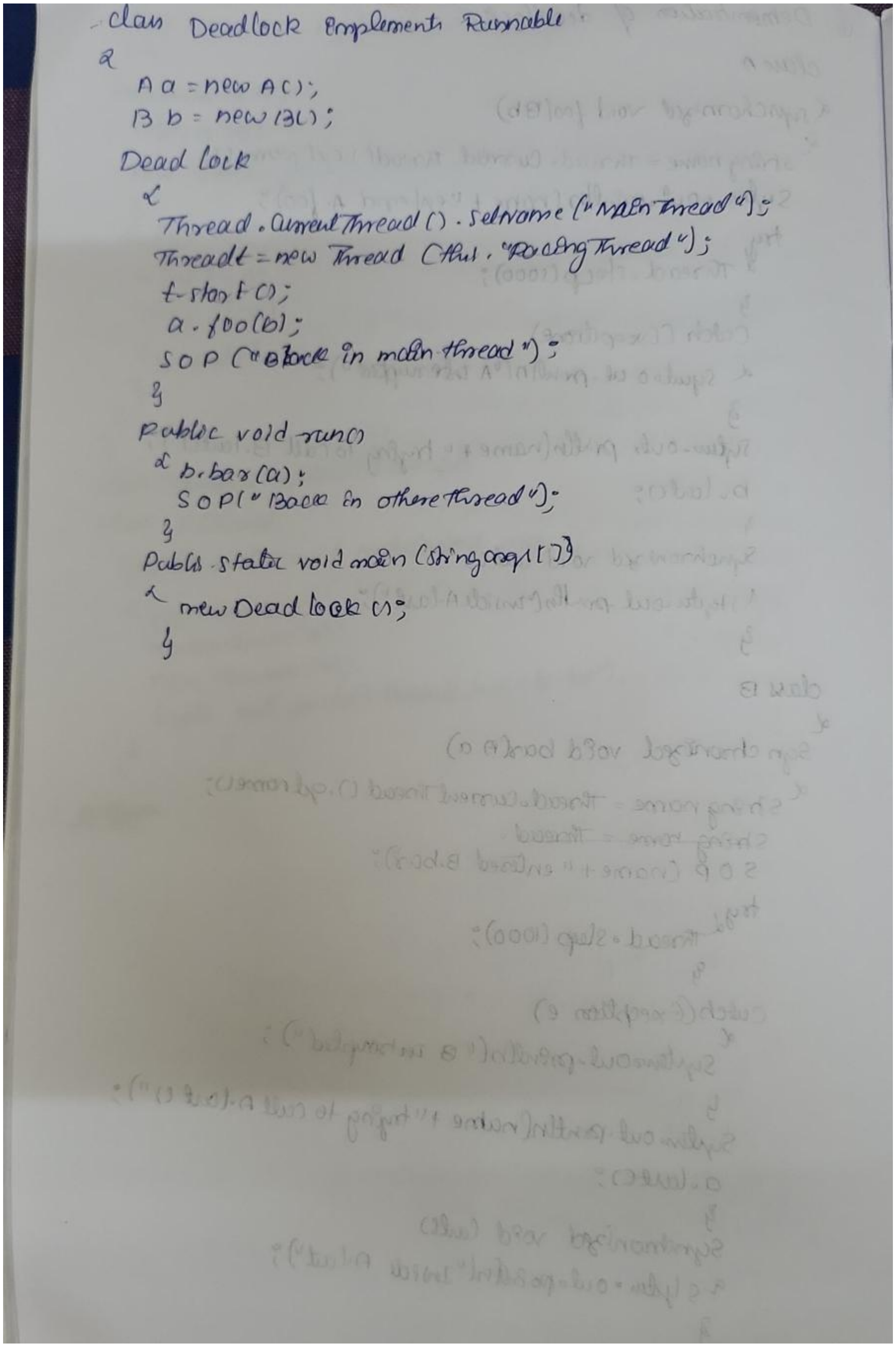
```
Intimate Producer

consumed:6
Put: 7

Intimate Consumer


Producer waiting

Got: 7

Intimate Producer

consumed:7
Put: 8

Intimate Consumer


Producer waiting

Got: 8

Intimate Producer

consumed:8
Put: 9

Intimate Consumer


Producer waiting

Got: 9

Intimate Producer

consumed:9
Put: 10
```

2.Demonstration of Deadlock   Observation

OBSERVATION:



```
(ii) Demonstration of deadlock
class A
{synchronized void foo(B b)
{ string name = Thread. Current thread( ).get Name();
  System.out.println(name + "entered A.foo");
  try
   { Thread.sleep(1000);
   }
   catch (Exception e)
   { System.out.println("A Interrupted");
   }
   System.out.println(name + " trying to call B.last()");
   b.last();
   }
   Synchronized void last()
   { System.out.println("inside A.last");
   }
class B
{
 Synchronized void bar(A a)
 { string name = Thread.Current Thread ().getname();
   ~~string name = Thread.~~
   S O P (name + "entered B.bar");
   try{ Thread.sleep (1000);
      }
   catch(Exception e)
    { System.out.println(" B Interrupted");
    }
   System.out.println(name + "trying to call A.last() ");
    a.last();
    }
    Synchronized void last()
    { System.out.println("Inside A.last");
     }
    }
```

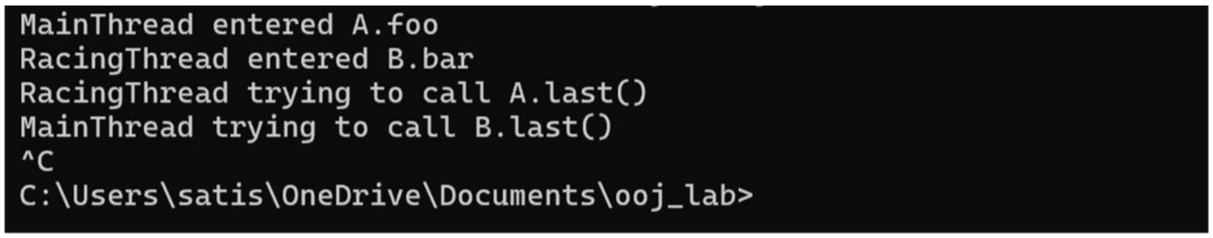


```
class Deadlock Implements Runnable
{
   A a = new A();
   B b = new B();
 Dead lock
  {
   Thread.CurrentThread().setname("MAin Thread");
   Threadt = new Thread (this, "RacingThread");
   t.start();
   a.foo(b);
   SOP ("Back in main thread");
  }
 public void run()
  { b.bar(a);
    SOP(" Back in othere thread");
  }
 Public static void main (String args[])
 { new Dead lock ();
 }
```

SOURCE CODE:

```
 class A

{   synchronized void foo(B b)
  { String name = Thread.currentThread().getName();
System.out.println(name + " entered A.foo");       try { Thread.sleep(1000); }
     catch(Exception e) { System.out.println("A Interrupted"); }       System.out.println(name + " trying to call
B.last()"); b.last(); }       synchronized void last() { System.out.println("Inside A.last"); }   }
class B {
   synchronized void bar(A a) {
   String name = Thread.currentThread().getName();
System.out.println(name + " entered B.bar");     try { Thread.sleep(1000); }
catch(Exception e) { System.out.println("B Interrupted"); }  System.out.println(name + " trying to
call A.last()"); a.last(); }  synchronized void last() { System.out.println("Inside A.last"); }
}

class Deadlock implements Runnable
 {
 A a = new A(); B b = new B();
 Deadlock( ) {
   Thread.currentThread().setName("MainThread");
   Thread t = new Thread(this, "RacingThread");
    t.start(); a.foo(b); // get lock on a in this thread.      System.out.println("Back in main thread");
}   public void run() { b.bar(a); // get lock on b in other thread.
 System.out.println("Back in other thread");
 }
public static void main(String args[]) { new Deadlock(); }
              }
```

OUTPUT :



```
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
^C
C:\Users\satis\OneDrive\Documents\ooj_lab>
```