VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

MONISHA H L (24BECS411)

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)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled “Object Oriented Java Programming (23CS3PCOOJ)” carried out by **MONISHA H L (24BECS411),** 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. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

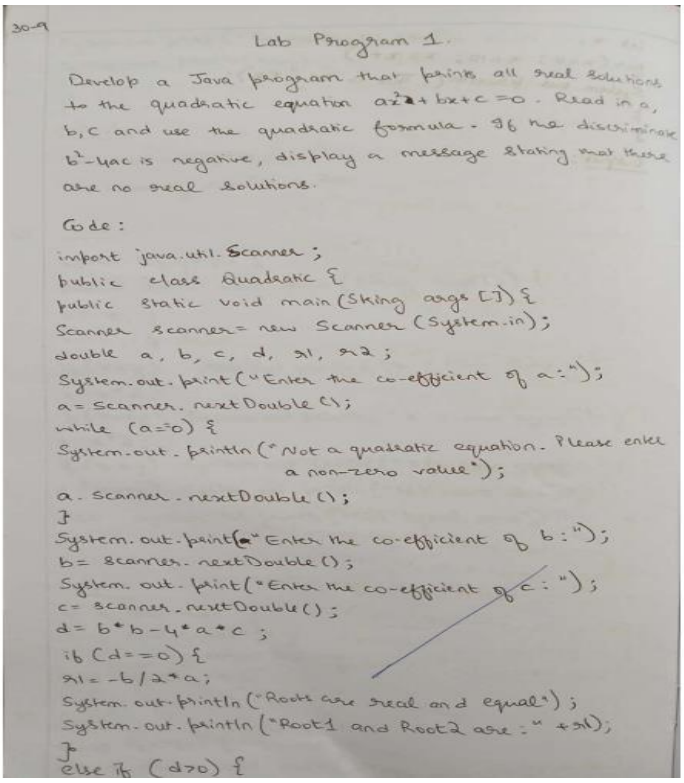
SRUSHTI C S Assistant Professor Department of CSE, BMSCE	Dr. JYOTHI S NAYAK Professor & HOD Department of CSE, BMSCE

Index

GITHUB LINK:
https://github.com/Monisha-HL/Java_Lab_Programs

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.



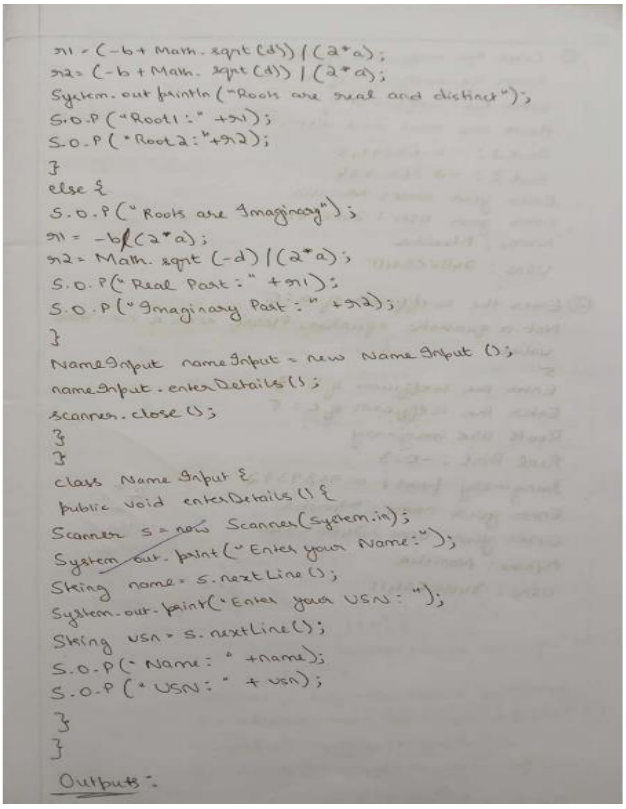
30-9

Lab Program 1.

Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

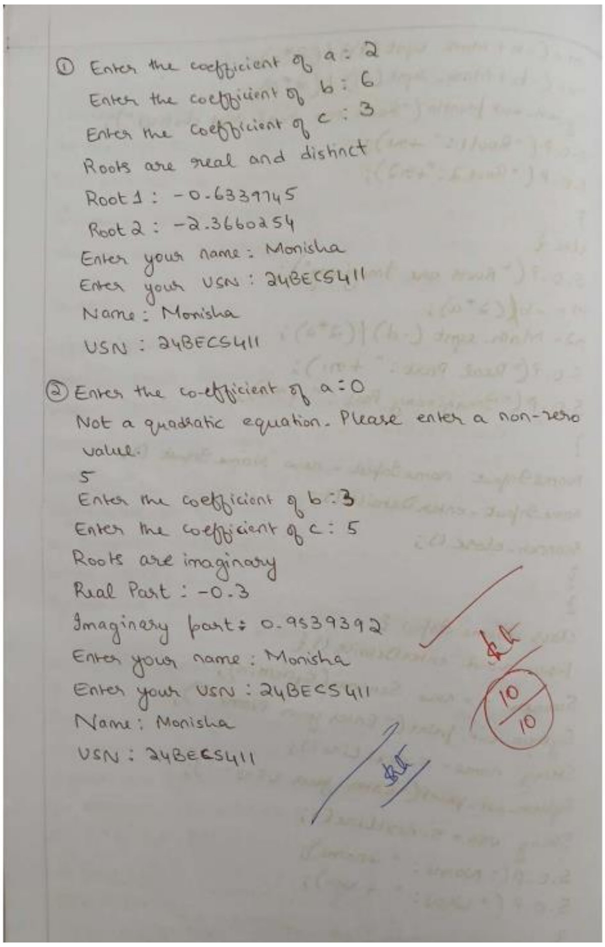
Code :

```
import java.util.Scanner ;
public class Quadratic {
public static void main (String args [ ]) {
Scanner scanner = new Scanner (System.in);
double a, b, c, d, r1, r2 ;
System.out.print ("Enter the co-efficient of a:");
a= scanner. nextDouble ();
while (a==0) {
System.out.println ("Not a quadratic equation. Please enter
                    a non-zero value");
a. scanner.nextDouble ();
}
System.out.print("Enter the co-efficient of b:");
b= scanner.nextDouble ();
System.out.print("Enter the co-efficient of c:");
c= scanner.nextDouble ();
d= b*b-4*a*c ;
if (d==0) {
r1= -b/2*a;
System.out.println ("Roots are real and equal");
System.out.println ("Root1 and Root2 are :" +r1);
}
else if (d>0) {
```



```
r1 = (-b+ Math.sqrt (d)) / (2*a);
r2= (-b + Math. sqrt (d)) / (2*a);
System.out println ("Roots are real and distinct");
S.O.P ("Root1 :" +r1);
S.O.P ("Root2:"+r2);
}
else {
S.O.P ("Roots are Imaginary");
r1 = -b/(2*a);
r2= Math. sqrt (-d) / (2*a);
S.O.P("Real Part :" +r1);
S.O.P ("Imaginary Part :" +r2);
}
NameInput nameInput = new NameInput ();
nameInput.enterDetails ();
scanner.close ();
}
}
class NameInput {
public void enterDetails () {
Scanner s = new Scanner(system.in);
System.out.print(" Enter your Name:");
String name= s.nextLine ();
System.out.print("Enter your USN:");
String usn = s.nextLine();
S.O.P(" Name: " +name);
S.O.P (" USN: " + usn);
}
}
```

Outputs :



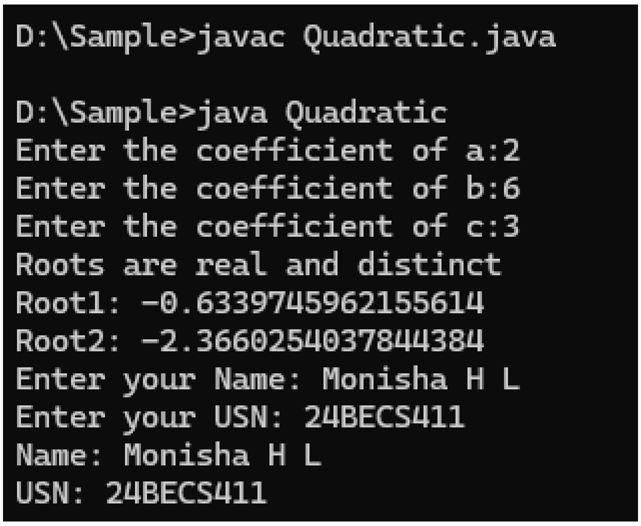
```
① Enter the coefficient of a : 2
  Enter the coefficient of b : 6
  Enter the coefficient of c : 3
  Roots are real and distinct
  Root 1 : -0.6339745
  Root 2 : -2.3660254
  Enter your name : Monisha
  Enter your USN : 24BECS411
  Name : Monisha
  USN : 24BECS411

② Enter the co-efficient of a : 0
  Not a quadratic equation. Please enter a non-zero
  value.
  5
  Enter the coefficient of b : 3
  Enter the coefficient of c : 5
  Roots are imaginary
  Real Part : -0.3
  Imaginary part : 0.9539392
  Enter your name : Monisha
  Enter your USN : 24BECS411
  Name : Monisha
  USN : 24BECS411
```

10/10

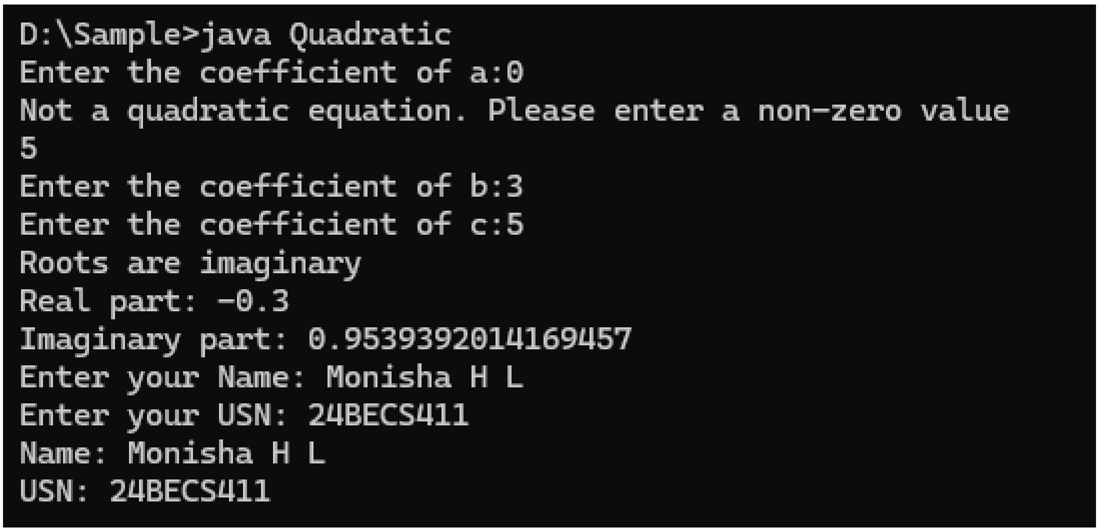
SOURCE CODE:

```
import java.util.Scanner;
public class Quadratic{
public static void main(String args[]){
Scanner scanner=new Scanner(System.in);
double a, b, c, d, r1, r2;

System.out.print("Enter the coefficient of a:");
a=scanner.nextDouble();

while (a==0){
System.out.println("Not a quadratic equation. Please enter a non-zero value");
a=scanner.nextDouble();
}

System.out.print("Enter the coefficient of b:");
b=scanner.nextDouble();

System.out.print("Enter the coefficient of c:");
c=scanner.nextDouble();

d=b*b-4*a*c;
if(d==0){
r1=-b/2*a;
System.out.println("Root are real and equal");
System.out.println("Root1 and Root2 are:" +r1);
}
else if (d>0){
r1=(-b+Math.sqrt(d))/(2*a);
r2=(-b-Math.sqrt(d))/(2*a);
System.out.println("Roots are real and distinct");
System.out.println("Root1: " +r1);
System.out.println("Root2: " +r2);
}
else{
System.out.println("Roots are imaginary");
r1 = -b / (2 * a);
r2 = Math.sqrt(-d) / (2 * a);
System.out.println("Real part: " + r1);
System.out.println("Imaginary part: " + r2);
}

NameInput nameInput=new NameInput();
nameInput.enterDetails();

scanner.close();
}
}

class NameInput{
public void enterDetails(){
Scanner s=new Scanner(System.in);
System.out.print("Enter your Name: ");
String name=s.nextLine();
System.out.print("Enter your USN: ");
String usn=s.nextLine();
System.out.println("Name: "+name);
System.out.println("USN: " +usn);
}
}
```

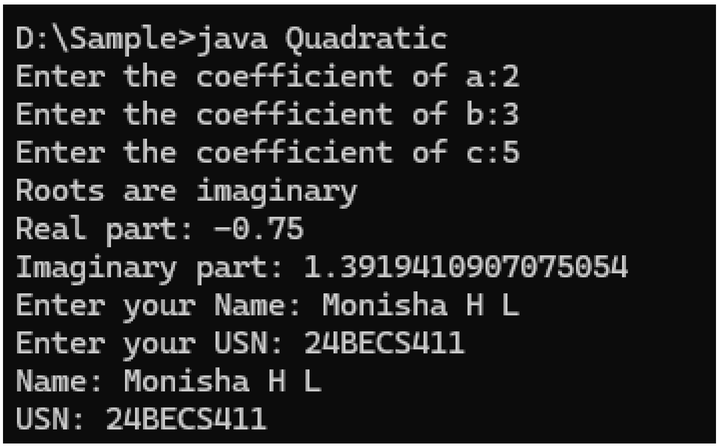
OUTPUTS:



```
D:\Sample>javac Quadratic.java

D:\Sample>java Quadratic
Enter the coefficient of a:2
Enter the coefficient of b:6
Enter the coefficient of c:3
Roots are real and distinct
Root1: -0.6339745962155614
Root2: -2.3660254037844384
Enter your Name: Monisha H L
Enter your USN: 24BECS411
Name: Monisha H L
USN: 24BECS411
```



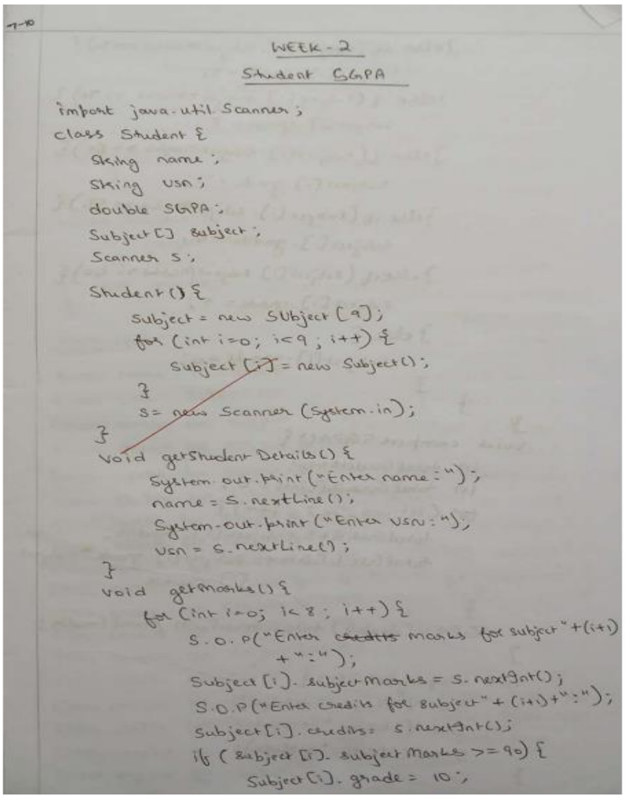
```
D:\Sample>java Quadratic
Enter the coefficient of a:0
Not a quadratic equation. Please enter a non-zero value
5
Enter the coefficient of b:3
Enter the coefficient of c:5
Roots are imaginary
Real part: -0.3
Imaginary part: 0.9539392014169457
Enter your Name: Monisha H L
Enter your USN: 24BECS411
Name: Monisha H L
USN: 24BECS411
```



```
D:\Sample>java Quadratic
Enter the coefficient of a:2
Enter the coefficient of b:3
Enter the coefficient of c:5
Roots are imaginary
Real part: -0.75
Imaginary part: 1.3919410907075054
Enter your Name: Monisha H L
Enter your USN: 24BECS411
Name: Monisha H L
USN: 24BECS411
```

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.

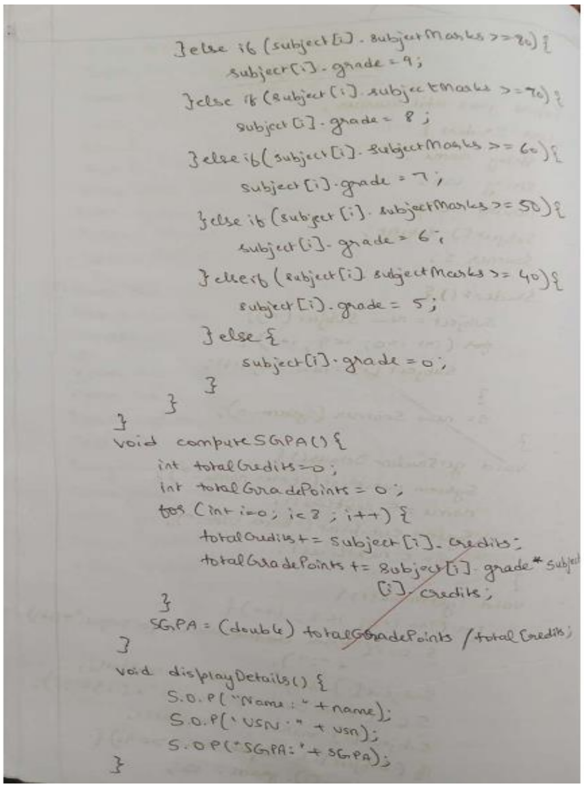


7-10

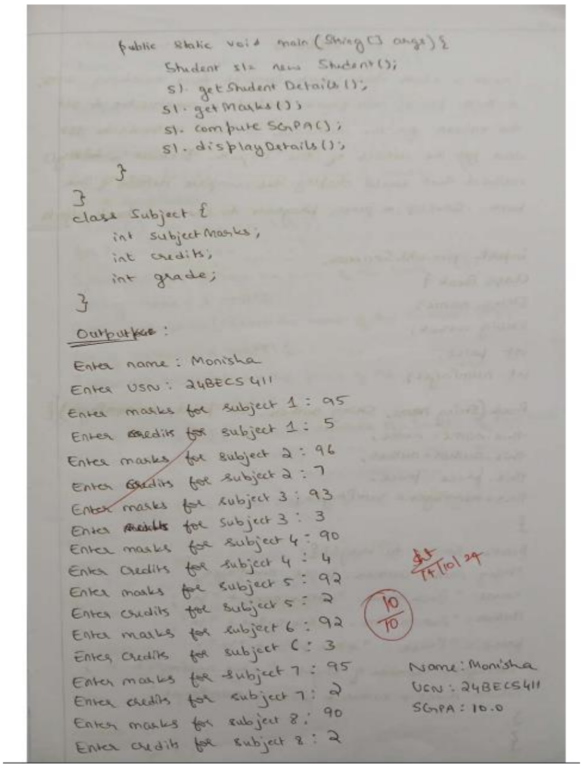
WEEK - 2

Student SGPA

```
import java.util.Scanner;
class Student {
    String name;
    String usn;
    double SGPA;
    Subject[] subject;
    Scanner s;

    Student() {
        subject = new SUbject[9];
        for (int i=0; i<9; i++) {
            subject[i] = new Subject();
        }
        s= new Scanner(System.in);
    }
    void getStudentDetails() {
        System.out.print("Enter name:");
        name = s.nextLine();
        System.out.print("Enter usn:");
        usn = s.nextLine();
    }
    void getMarks() {
        for (int i=0; i<8; i++) {
            S.O.P("Enter ~~credits~~ marks for subject"+(i+1)
                  +":");
            Subject[i].subjectMarks = s.nextInt();
            S.O.P("Enter credits for subject" + (i+1)+":");
            subject[i].credits= s.nextInt();
            if (subject[i].subjectMarks >=90) {
                Subject[i].grade= 10;
```



```
            }else if (subject[i].subjectMarks >=80) {
                subject[i].grade =9;
            }else if (subject[i].subjectMarks >=70) {
                subject[i].grade= 8;
            }else if(subject[i].subjectMarks >= 60) {
                subject[i].grade = 7;
            }else if (subject[i].subjectMarks >= 50) {
                subject[i].grade= 6;
            }elseif (subject[i].subjectMarks >= 40) {
                subject[i].grade = 5;
            }else {
                subject[i].grade =0;
            }
        }
    }
    void computeSGPA() {
        int totalCredits=0;
        int totalGradePoints = 0;
        for (int i=0; i<8; i++) {
            totalCredits += subject[i].credits;
            totalGradePoints += subject[i].grade* subject
                                [i].credits;
        }
        SGPA = (double) totalGradePoints /totalCredits;
    }
    void displayDetails() {
        S.O.P("Name:"+name);
        S.O.P("USN:" + usn);
        S.O.P("SGPA:"+ SGPA);
    }
```



```
    public static void main(String[] args){
        Student s1= new Student();
        s1. getStudent Details();
        s1. getMarks();
        s1. compute SGPA();
        s1. displayDetails();
    }
}
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}
```

Output :

```
Enter name : Monisha
Enter USN : 24BECS411
Enter marks for subject 1 : 95
Enter credits for subject 1 : 5
Enter marks for subject 2 : 96
Enter credits for subject 2 : 7
Enter marks for subject 3 : 93
Enter credits for subject 3 : 3
Enter marks for subject 4 : 90
Enter credits for subject 4 : 4
Enter marks for subject 5 : 92
Enter credits for subject 5 : 2
Enter marks for subject 6 : 92
Enter credits for subject 6 : 3
Enter marks for subject 7 : 95
Enter credits for subject 7 : 2
Enter marks for subject 8 : 90
Enter credits for subject 8 : 2
```

```
Name : Monisha
USN : 24BECS411
SGPA : 10.0
```

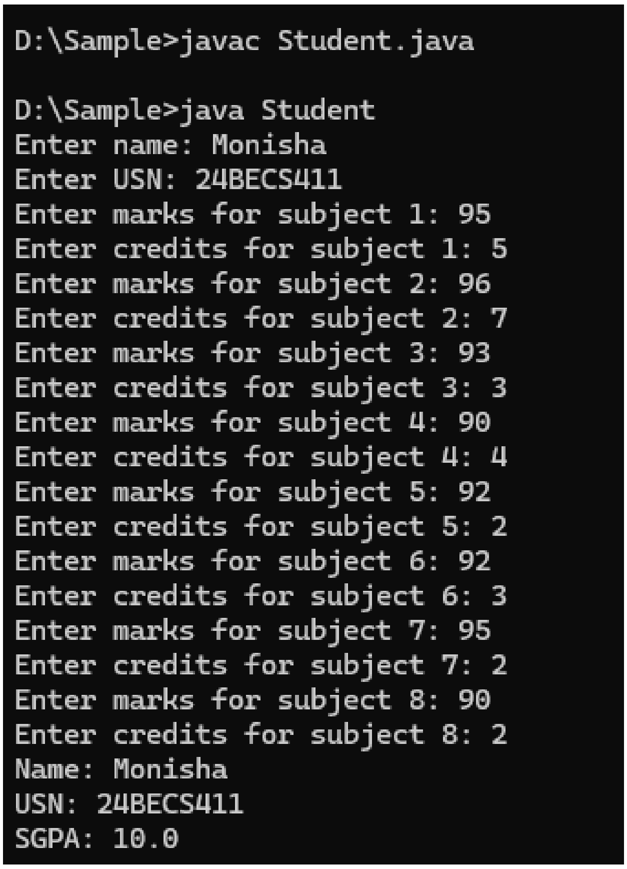
10/10

14/10/24

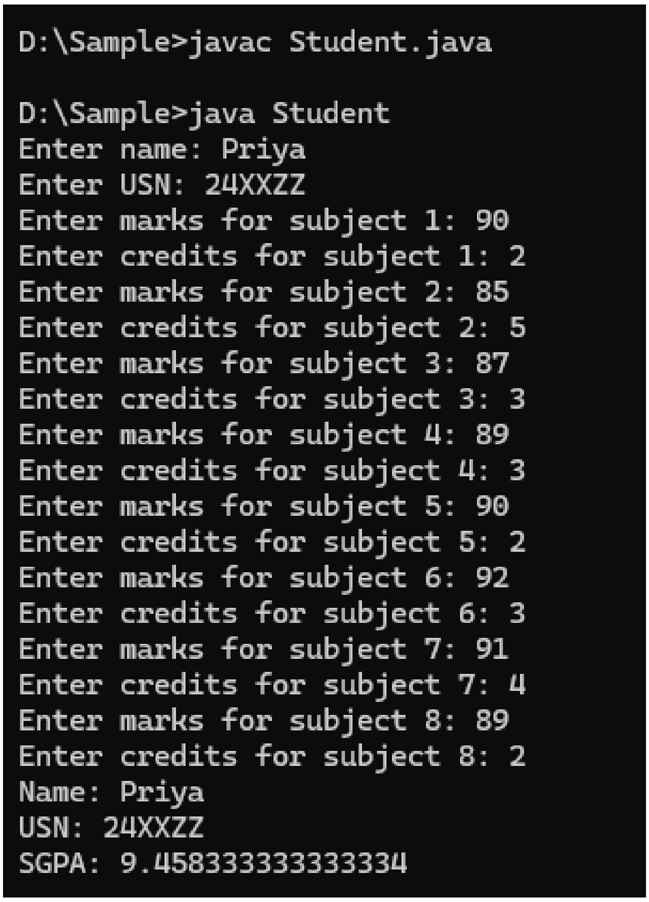
SOURCE CODE:

```
import java.util.Scanner;

class Student {
    String name;
    String usn;
    double SGPA;
    Subject[] subject;
    Scanner s;

    Student() {
        subject = new Subject[9];
        for (int i = 0; i < 9; i++) {
            subject[i] = new Subject();
        }
        s = new Scanner(System.in);
    }

    void getStudentDetails() {
        System.out.print("Enter name: ");
        name = s.nextLine();
        System.out.print("Enter USN: ");
        usn = s.nextLine();
    }

    void getMarks() {
        for (int i = 0; i < 8; i++) {
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            subject[i].subjectMarks = s.nextInt();
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            subject[i].credits = s.nextInt();
            if (subject[i].subjectMarks >= 90) {
                subject[i].grade = 10;
            } else if (subject[i].subjectMarks >= 80) {
                subject[i].grade = 9;
            } else if (subject[i].subjectMarks >= 70) {
                subject[i].grade = 8;
            } else if (subject[i].subjectMarks >= 60) {
                subject[i].grade = 7;
            } else if (subject[i].subjectMarks >= 50) {
                subject[i].grade = 6;
            } else if (subject[i].subjectMarks >= 40) {
                subject[i].grade = 5;
            } else {
                subject[i].grade = 0;
            }
        }
    }

    void computeSGPA() {
        int totalCredits = 0;
        int totalGradePoints = 0;
        for (int i = 0; i < 8; i++) {
            totalCredits += subject[i].credits;
            totalGradePoints += subject[i].grade * subject[i].credits;
        }
        SGPA = (double) totalGradePoints / totalCredits;
    }

    void displayDetails() {
        System.out.println("Name: " + name);
        System.out.println("USN: " + usn);
        System.out.println("SGPA: " + SGPA);
    }

    public static void main(String[] args) {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.displayDetails();
    }
}
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}
```

OUTPUTS:



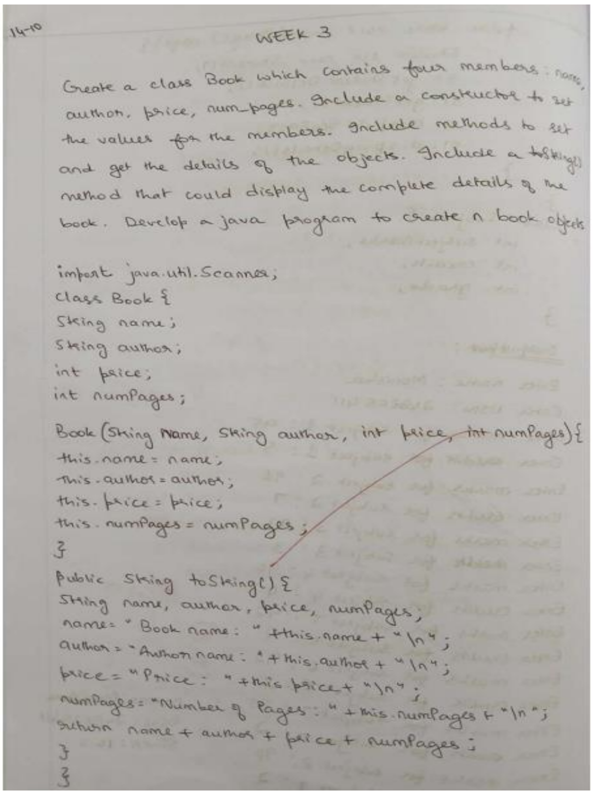
```
D:\Sample>javac Student.java

D:\Sample>java Student
Enter name: Monisha
Enter USN: 24BECS411
Enter marks for subject 1: 95
Enter credits for subject 1: 5
Enter marks for subject 2: 96
Enter credits for subject 2: 7
Enter marks for subject 3: 93
Enter credits for subject 3: 3
Enter marks for subject 4: 90
Enter credits for subject 4: 4
Enter marks for subject 5: 92
Enter credits for subject 5: 2
Enter marks for subject 6: 92
Enter credits for subject 6: 3
Enter marks for subject 7: 95
Enter credits for subject 7: 2
Enter marks for subject 8: 90
Enter credits for subject 8: 2
Name: Monisha
USN: 24BECS411
SGPA: 10.0
```



```
D:\Sample>javac Student.java

D:\Sample>java Student
Enter name: Priya
Enter USN: 24XXZZ
Enter marks for subject 1: 90
Enter credits for subject 1: 2
Enter marks for subject 2: 85
Enter credits for subject 2: 5
Enter marks for subject 3: 87
Enter credits for subject 3: 3
Enter marks for subject 4: 89
Enter credits for subject 4: 3
Enter marks for subject 5: 90
Enter credits for subject 5: 2
Enter marks for subject 6: 92
Enter credits for subject 6: 3
Enter marks for subject 7: 91
Enter credits for subject 7: 4
Enter marks for subject 8: 89
Enter credits for subject 8: 2
Name: Priya
USN: 24XXZZ
SGPA: 9.458333333333334
```

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.

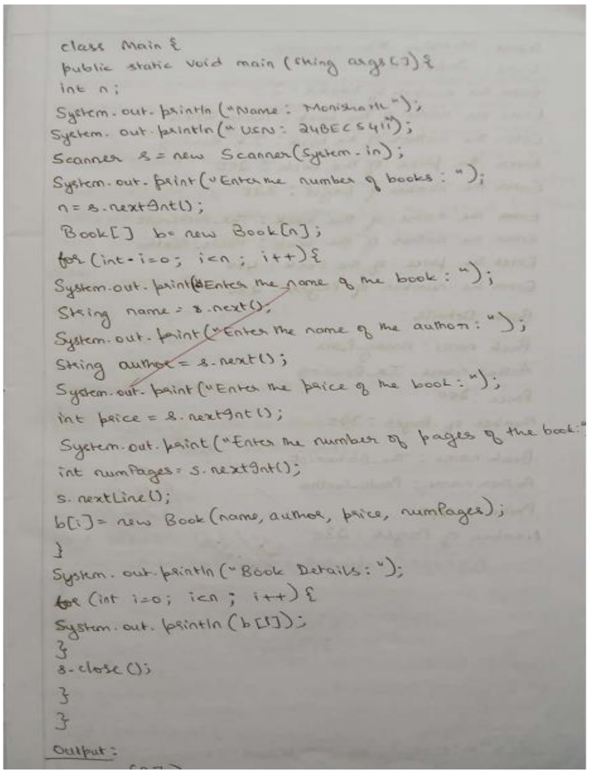


14-10

WEEK 3

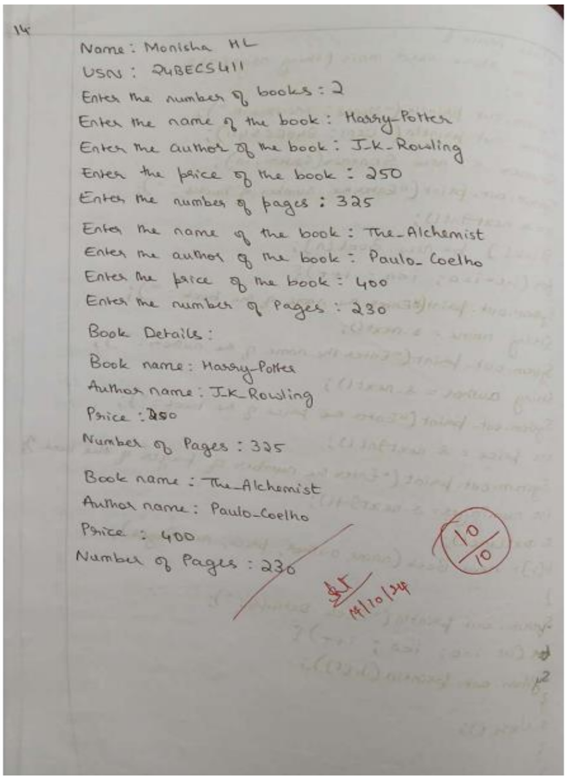
Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a java program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

```
import java.util.Scanner;
class Book {
String name;
String author;
int price;
int numPages;

Book (String Name, String author, int price, int numPages){
this.name = name;
This.author = author;
this.price = price;
this.numPages = numPages;
}
public String toString() {
String name, author, price, numPages;
name= " Book name: " +this.name + "\n";
author = "Author name: " + this.author + "\n";
price = "Price: " +this price+ "\n";
numPages= "Number of Pages: " + this.numPages + "\n";
return name + author + price + numPages;
}
}
```



```
class Main {
public static void main (String args[]){
int n;
System.out.println ("Name: Monisha.H.");
System. out.println(" USN: 24BECS411");
Scanner s = new Scanner(System.in);
System.out.print("Enter the number of books: ");
n = s.nextInt();
Book[] b= new Book[n];
for (int i=0; i<n; i++){
System.out.print("Enter the name of the book: ");
String name = s.next();
System.out.print("Enter the name of the author: ");
String author = s.next();
System.out.print("Enter the price of the book: ");
int price = s.nextInt();
System.out.print("Enter the number of pages of the book:"
int numPages= s.nextInt();
s.nextLine();
b[i]= new Book(name, author, price, numPages);
}
System.out.println("Book Details: ");
for (int i=0; i<n; i++) {
System.out.println (b[i]);
}
s.close();
}
}
```

Output:



14

```
Name : Monisha HL
USN : 24BECS411
Enter the number of books : 2
Enter the name of the book : Harry-Potter
Enter the author of the book : J-K-Rowling
Enter the price of the book : 250
Enter the number of pages : 325
Enter the name of the book : The-Alchemist
Enter the author of the book : Paulo- Coelho
Enter the price of the book : 400
Enter the number of Pages : 230
Book Details :
Book name : Harry-Potter
Author name : J-K-Rowling
Price : 250
Number of Pages : 325
Book name : The-Alchemist
Author name : Paulo-Coelho
Price : 400
Number of Pages : 230
```

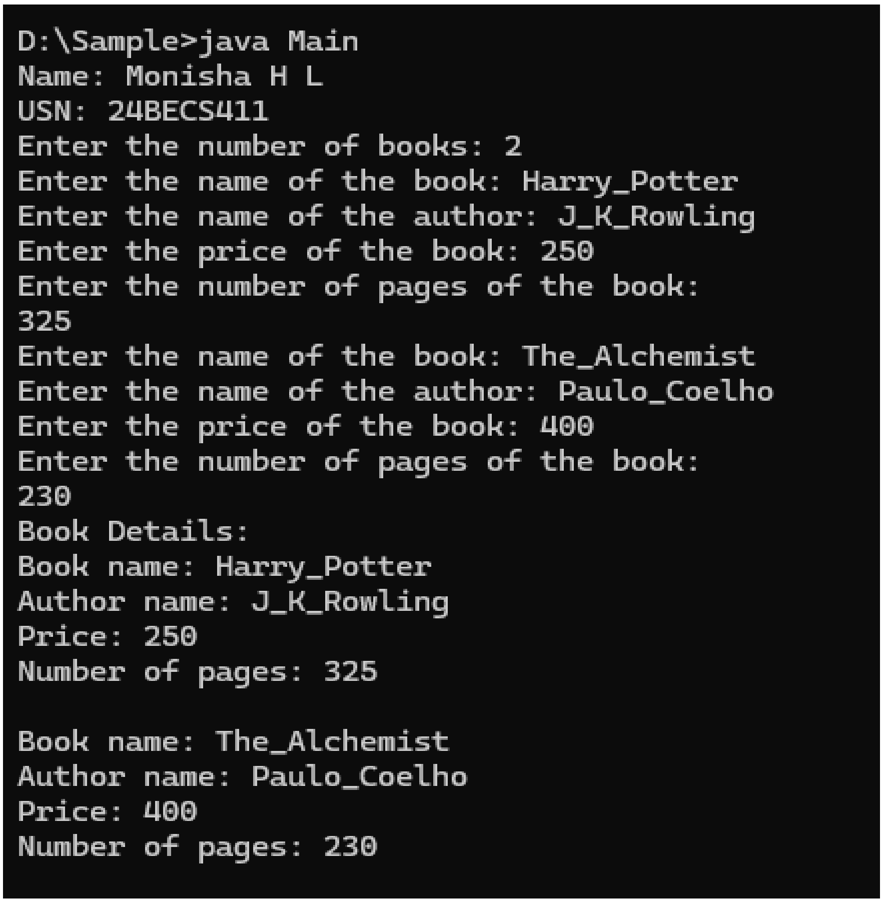
10/10

19/10/24

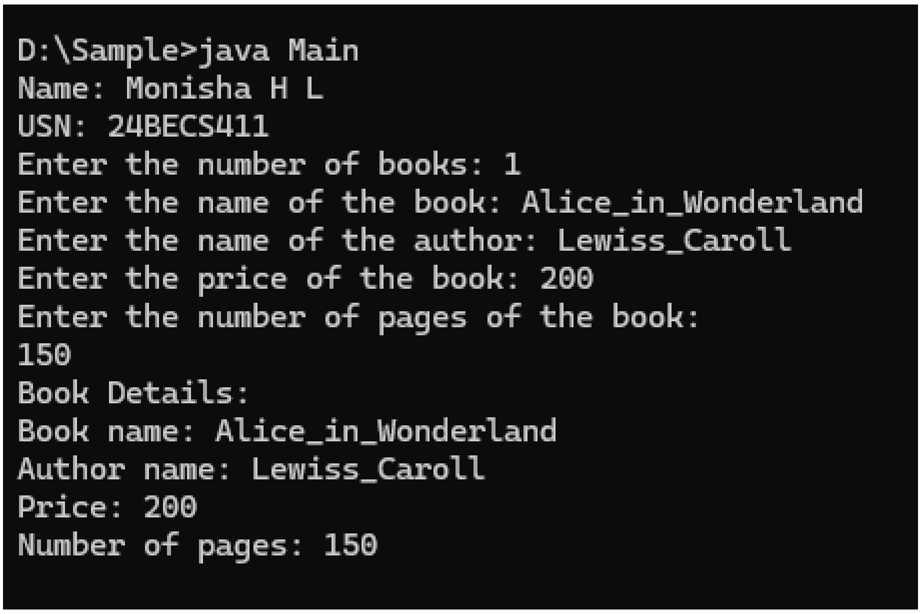
SOURCE CODE:

```
import java.util.Scanner;
class Book{
String name;
String author;
int price;
int numPages;

Book(String name, String author, int price, int numPages){
this.name=name;
this.author=author;
this.price=price;
this.numPages=numPages;
}

public String toString(){
String name, author, price, numPages;
name="Book name: "+this.name+ "\n";
author="Author name: "+this.author+ "\n";
price="Price: "+this.price+ "\n";
numPages="Number of pages: "+this.numPages+ "\n";
return name + author + price + numPages;
}
}

class Main{
public static void main(String args[]){
int n;
System.out.println("Name: Monisha H L");
System.out.println("USN: 24BECS411");
Scanner s=new Scanner(System.in);
System.out.print("Enter the number of books: ");
n=s.nextInt();
Book[] b=new Book[n];
for(int i=0; i<n; i++){
System.out.print("Enter the name of the book: ");
String name=s.next();
System.out.print("Enter the name of the author: ");
String author=s.next();
System.out.print("Enter the price of the book: ");
int price=s.nextInt();
System.out.println("Enter the number of pages of the book: ");
int numPages=s.nextInt();
s.nextLine();
b[i]=new Book(name, author, price, numPages);
}

System.out.println("Book Details: ");
for(int i=0; i<n; i++){
System.out.println(b[i]);
}
s.close();
}
}
```

OUTPUTS:



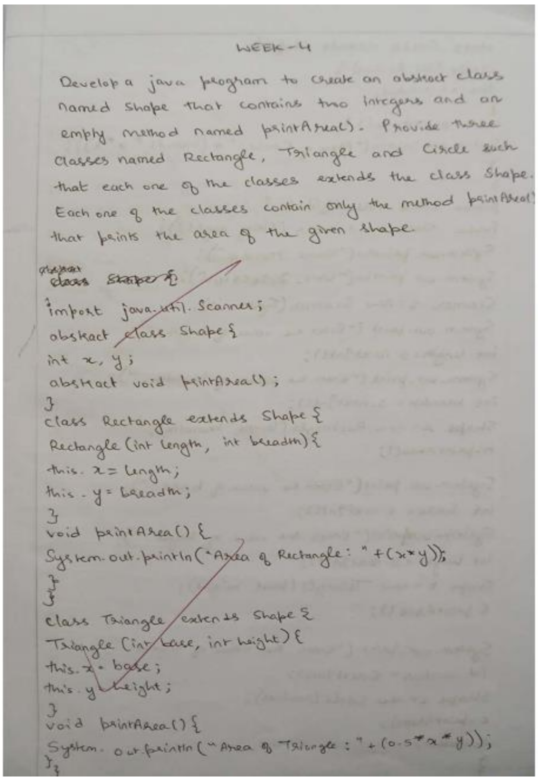
```
D:\Sample>java Main
Name: Monisha H L
USN: 24BECS411
Enter the number of books: 2
Enter the name of the book: Harry_Potter
Enter the name of the author: J_K_Rowling
Enter the price of the book: 250
Enter the number of pages of the book:
325
Enter the name of the book: The_Alchemist
Enter the name of the author: Paulo_Coelho
Enter the price of the book: 400
Enter the number of pages of the book:
230
Book Details:
Book name: Harry_Potter
Author name: J_K_Rowling
Price: 250
Number of pages: 325

Book name: The_Alchemist
Author name: Paulo_Coelho
Price: 400
Number of pages: 230
```



```
D:\Sample>java Main
Name: Monisha H L
USN: 24BECS411
Enter the number of books: 1
Enter the name of the book: Alice_in_Wonderland
Enter the name of the author: Lewiss_Caroll
Enter the price of the book: 200
Enter the number of pages of the book:
150
Book Details:
Book name: Alice_in_Wonderland
Author name: Lewiss_Caroll
Price: 200
Number of pages: 150
```

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.

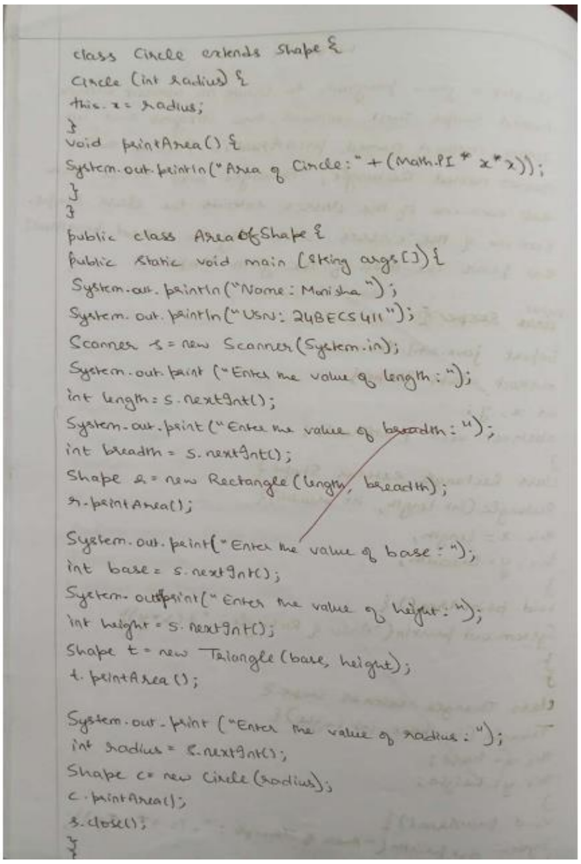


WEEK-4

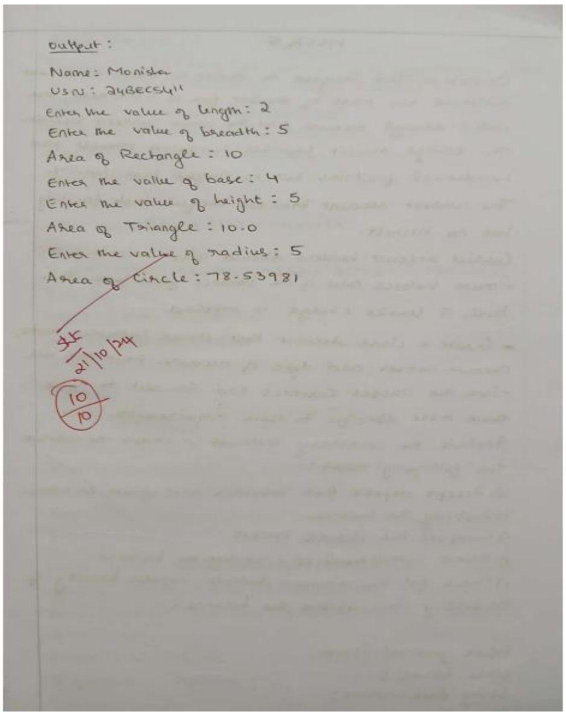
Develop a java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea()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

~~class Shape {~~

```
import java.util.Scanner;
abstract class Shape {
int x, y;
abstract void printArea();
}
class Rectangle extends Shape {
Rectangle (int length, int breadth){
this. x= length;
this. y= breadth;
}
void printArea() {
System.out.println("Area of Rectangle: "+(x*y));
}
}
class Triangle extends Shape {
Triangle (int base, int height) {
this. x= base;
this. y= height;
}
void printArea() {
System. out.println("Area of Triangle : "+(0.5*x*y));
}}
```



```
class Circle extends Shape {
Circle (int radius) {
this. x= radius;
}
void printArea() {
System.out.println("Area of Circle: "+(Math.PI * x*x));
}
}
public class AreaOfShape {
public static void main (String args[]) {
System.out. println("Name: Manisha");
System. out. println("USN: 24BECS411");
Scanner s = new Scanner(System.in);
System.out. print ("Enter the value of length: ");
int length= s.nextInt();
System.out.print("Enter the value of breadth: ");
int breadth = s.nextInt();
Shape r= new Rectangle(length, breadth);
r.printArea();

System.out. print("Enter the value of base: ");
int base= s.nextInt();
System.out.print("Enter the value of height: ");
int height = s.nextInt();
Shape t= new Triangle(base, height);
t. printArea();

System.out.print ("Enter the value of radius: ");
int radius = s.nextInt();
Shape c= new Circle(radius);
c.printArea();
s.close();
}
}
```



Output :

```
Name: Monisha
USN: 24BECS411
Enter the value of length: 2
Enter the value of breadth: 5
Area of Rectangle : 10
Enter the value of base: 4
Enter the value of height : 5
Area of Triangle : 10.0
Enter the value of radius: 5
Area of Circle : 78.53981
```

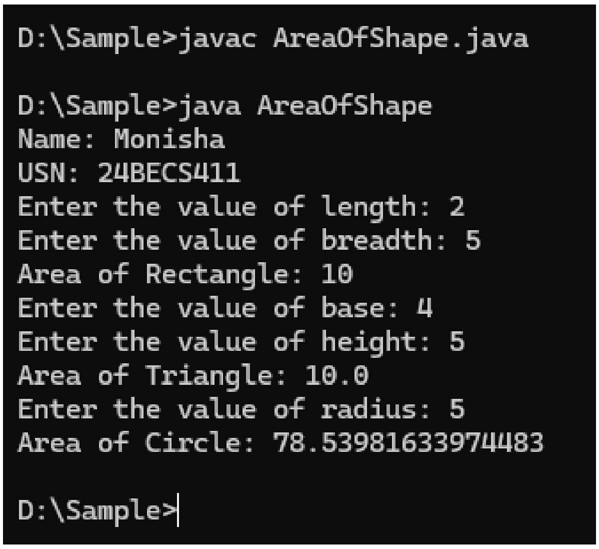
21/10/24

10/10

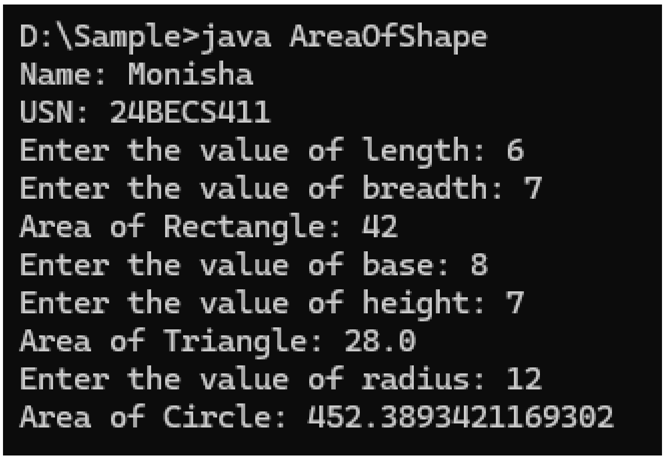
SOURCE CODE:

```
import java.util.Scanner;
abstract class Shape {
int x, y;
abstract void printArea();
}
class Rectangle extends Shape {
Rectangle(int length, int breadth) {
this.x = length;
this.y = breadth;
}
void printArea() {
System.out.println("Area of Rectangle: " + (x * y));
}
}
class Triangle extends Shape {
Triangle(int base, int height) {
this.x = base;
this.y = height;
}
void printArea() {
System.out.println("Area of Triangle: " + (0.5 * x * y));
}
}
class Circle extends Shape {
Circle(int radius) {
this.x = radius;
}
void printArea() {
System.out.println("Area of Circle: " + (Math.PI * x * x));
}
}
public class AreaOfShape {
public static void main(String[] args) {
System.out.println("Name: Monisha");
System.out.println("USN: 24BECS411");
Scanner s = new Scanner(System.in);
System.out.print("Enter the value of length: ");
int length = s.nextInt();
System.out.print("Enter the value of breadth: ");
int breadth = s.nextInt();
Shape r = new Rectangle(length, breadth);
r.printArea();

System.out.print("Enter the value of base: ");
int base = s.nextInt();
System.out.print("Enter the value of height: ");
int height = s.nextInt();
Shape t = new Triangle(base, height);
t.printArea();

System.out.print("Enter the value of radius: ");
int radius = s.nextInt();
Shape c = new Circle(radius);
c.printArea();

s.close();
}
}
```

OUTPUTS:



```
D:\Sample>javac AreaOfShape.java

D:\Sample>java AreaOfShape
Name: Monisha
USN: 24BECS411
Enter the value of length: 2
Enter the value of breadth: 5
Area of Rectangle: 10
Enter the value of base: 4
Enter the value of height: 5
Area of Triangle: 10.0
Enter the value of radius: 5
Area of Circle: 78.53981633974483

D:\Sample>
```



```
D:\Sample>java AreaOfShape
Name: Monisha
USN: 24BECS411
Enter the value of length: 6
Enter the value of breadth: 7
Area of Rectangle: 42
Enter the value of base: 8
Enter the value of height: 7
Area of Triangle: 28.0
Enter the value of radius: 12
Area of Circle: 452.3893421169302
```

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:

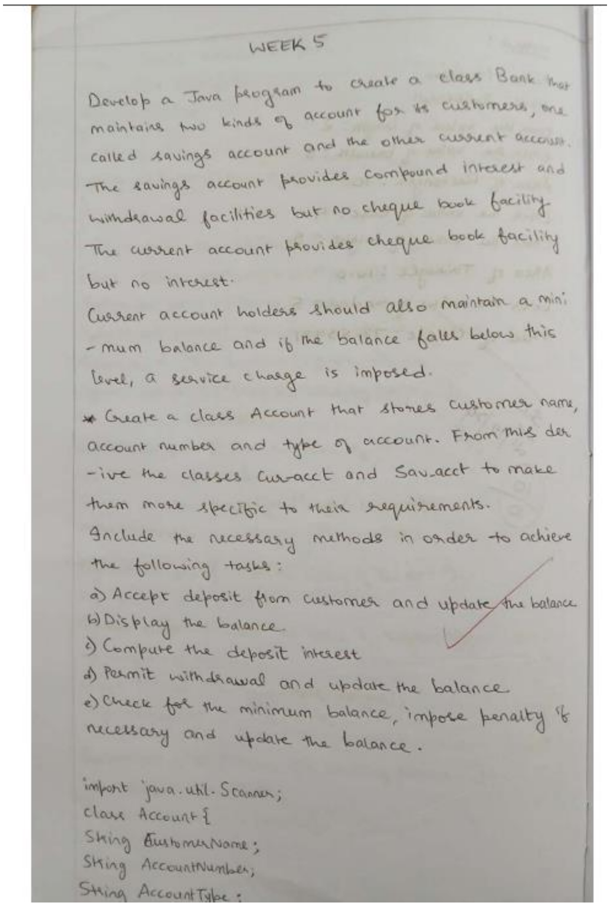
a)Accept deposit from customer and update the balance.

b)Display the balance.

c)Compute and deposit interest

d)Permit withdrawal and update the balance

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



WEEK 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

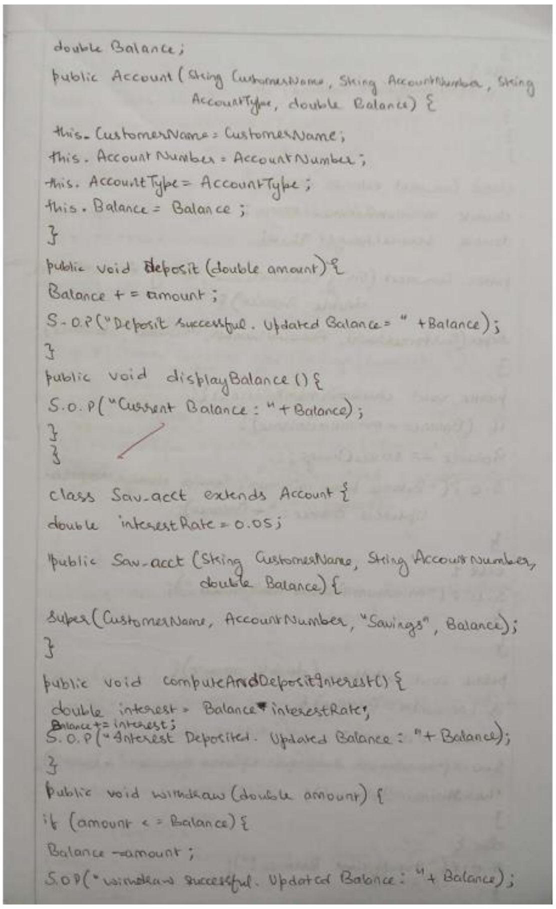
b) Display the balance

c) Compute the deposit interest

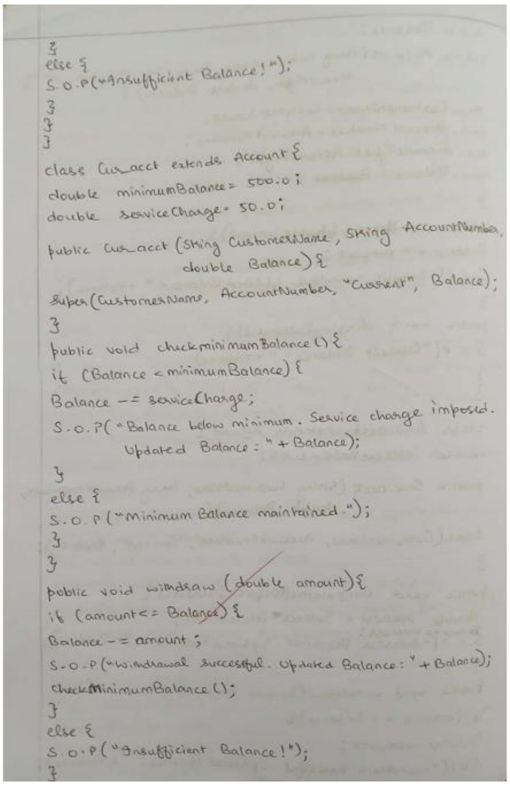
d) Permit withdrawal and update the balance

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

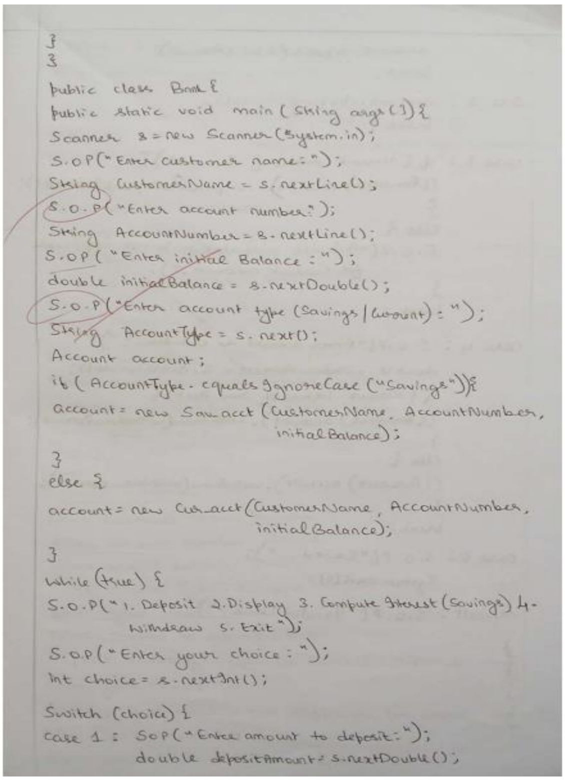
```
import java.util.Scanner;
class Account {
String CustomerName;
String AccountNumber;
String AccountType;
```



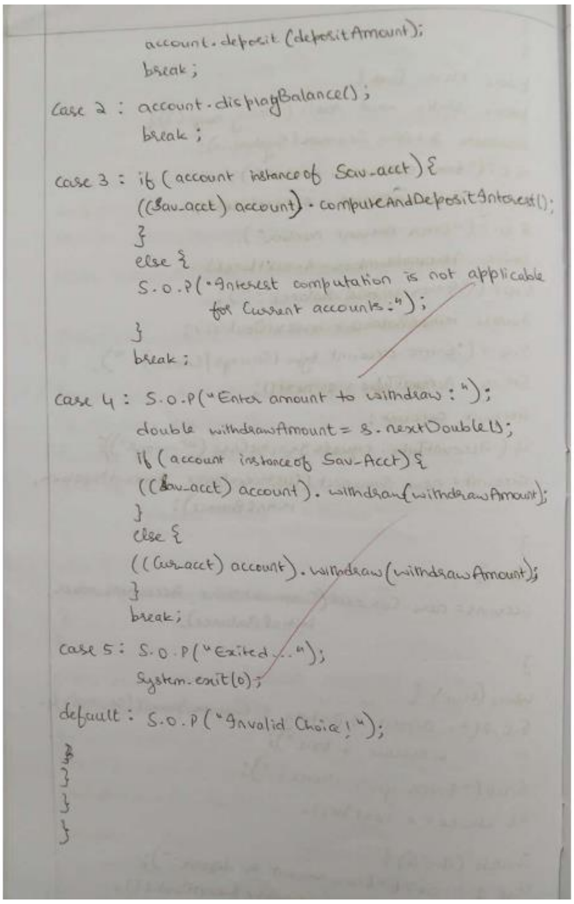
```
double Balance;
public Account (String CustomerName, String AccountNumber, String
                    AccountType, double Balance) {
this. CustomerName = CustomerName;
this. Account Number = AccountNumber;
this. AccountType = AccountType;
this. Balance = Balance;
}
public void deposit (double amount) {
Balance += amount;
S.O.P("Deposit successful. Updated Balance = " +Balance);
}
public void displayBalance () {
S.O.P("Current Balance: "+Balance);
}
}

class Sav-acct extends Account {
double interestRate = 0.05;
public Sav-acct (String CustomerName, String AccountNumber,
                    double Balance) {
super(CustomerName, AccountNumber, "Savings", Balance);
}
public void computeAndDepositInterest() {
double interest = Balance * interestRate;
Balance += interest;
S.O.P("Interest Deposited. Updated Balance: "+ Balance);
}
public void withdraw (double amount) {
if (amount <= Balance) {
Balance -= amount;
S.O.P("withdraw successful. Updated Balance: "+ Balance);
```



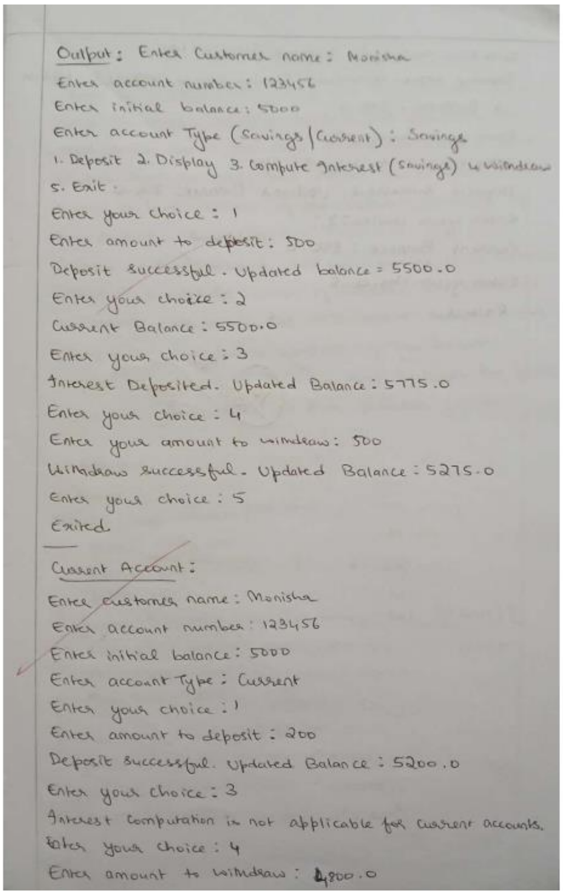
```
}
else {
S.O.P("Insufficient Balance!");
}
}
}

class Cur_acct extends Account {
double minimumBalance= 500.0;
double serviceCharge= 50.0;

public Cur_acct (String CustomerName, String AccountNumber,
                 double Balance) {

Super(CustomerName, AccountNumber, "Current", Balance);
}
public void checkminimumBalance() {
if (Balance <minimumBalance) {
Balance -= serviceCharge;
S.O.P("Balance below minimum. Service charge imposed.
        Updated Balance: " + Balance);
}
else {
S.O.P("Minimum Balance maintained.");
}
}
public void withdraw (double amount) {
if (amount<= Balance) {
Balance -= amount;
S.O.P("Withdrawal successful. Updated Balance: " + Balance);
checkMinimumBalance();
}
else {
S.O.P("Insufficient Balance!");
}
```



```
}
}

public class Bank {
public static void main (String args[]) {
Scanner s = new Scanner(System.in);
S.OP("Enter customer name:");
String CustomerName = s.nextLine();
S.O.P("Enter account number:");
String AccountNumber = s.nextLine();
S.OP("Enter initial Balance:");
double initialBalance = s.nextDouble();
S.O.P("Enter account type (Savings|Current):");
String AccountType = s.next();
Account account;
if (AccountType.equalsIgnoreCase("Savings")){
account= new Sav_acct (CustomerName, AccountNumber,
                        initialBalance);

}
else {
account= new Cur_acct(CustomerName, AccountNumber,
                       initialBalance);

}
while (true) {
S.O.P("1. Deposit 2.Display 3. Compute Interest (Savings) 4.
        Withdraw 5. Exit");
S.O.P("Enter your choice:");
int choice= s.nextInt();

Switch (choice) {
case 1 : SOP("Enter amount to deposit:");
         double depositAmount= s.nextDouble();
```



```
         account.deposit (depositAmount);
         break;
Case 2 : account.displayBalance();
         break;

case 3 : if (account instanceof Sav_acct) {
         ((Sav_acct) account).computeAndDepositInterest();
         }
         else {
         S.O.P("Interest computation is not applicable
                for Current accounts.");
         }
         break;

case 4 : S.O.P("Enter amount to withdraw:");
         double withdrawAmount = s.nextDouble();
         If (account instanceof Sav_Acct) {
         ((Sav_acct) account).withdraw(withdrawAmount);
         }
         else {
         ((Cur_acct) account).withdraw(withdrawAmount);
         }
         break;
case 5 : S.O.P("Exited...");
         System.exit(0);
default : S.O.P("Invalid Choice!");

}
}
}
}
```

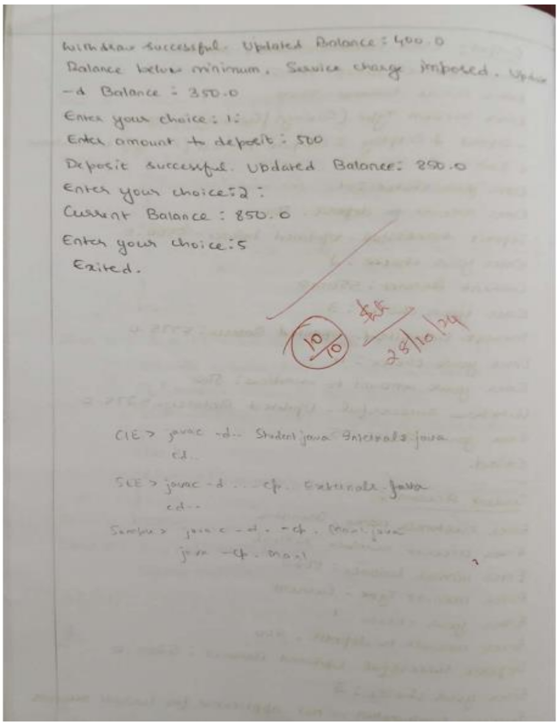


Output:

```
Enter Customer name: Monisha
Enter account number: 123456
Enter initial balance: 5000
Enter account Type (Savings|Current): Savings
1. Deposit 2. Display 3. Compute Interest (Savings) 4. Withdraw
5. Exit
Enter your choice: 1
Enter amount to deposit: 500
Deposit successful. Updated balance = 5500.0
Enter your choice: 2
Current Balance: 5500.0
Enter your choice: 3
Interest Deposited. Updated Balance: 5775.0
Enter your choice: 4
Enter your amount to withdraw: 500
Withdraw successful. Updated Balance: 5275.0
Enter your choice: 5
Exited
```

Current Account:

```
Enter customer name: Monisha
Enter account number: 123456
Enter initial balance: 5000
Enter account Type: Current
Enter your choice: 1
Enter amount to deposit: 200
Deposit successful. Updated Balance: 5200.0
Enter your choice: 3
Interest computation is not applicable for current accounts.
Enter your choice: 4
Enter amount to withdraw: 4800.0
```



```
Withdraw successful. Updated Balance: 400.0
Balance below minimum. Service charge imposed. Upda
-d Balance : 350.0
Enter your choice: 1:
Enter amount to deposit: 500
Deposit successful. Updated Balance: 850.0
Enter your choice:2 :
Current Balance : 850.0
Enter your choice:5
 Exited.
```

10/10 28/10/24

```
CIE > javac -d .. Student.java Internals.java
      cd..
SEE > javac -d .. -cp .. Externals.java
      cd--
Sample> javac -d . -cp . Main.java
        java -cp . Main
```

SOURCE CODE:

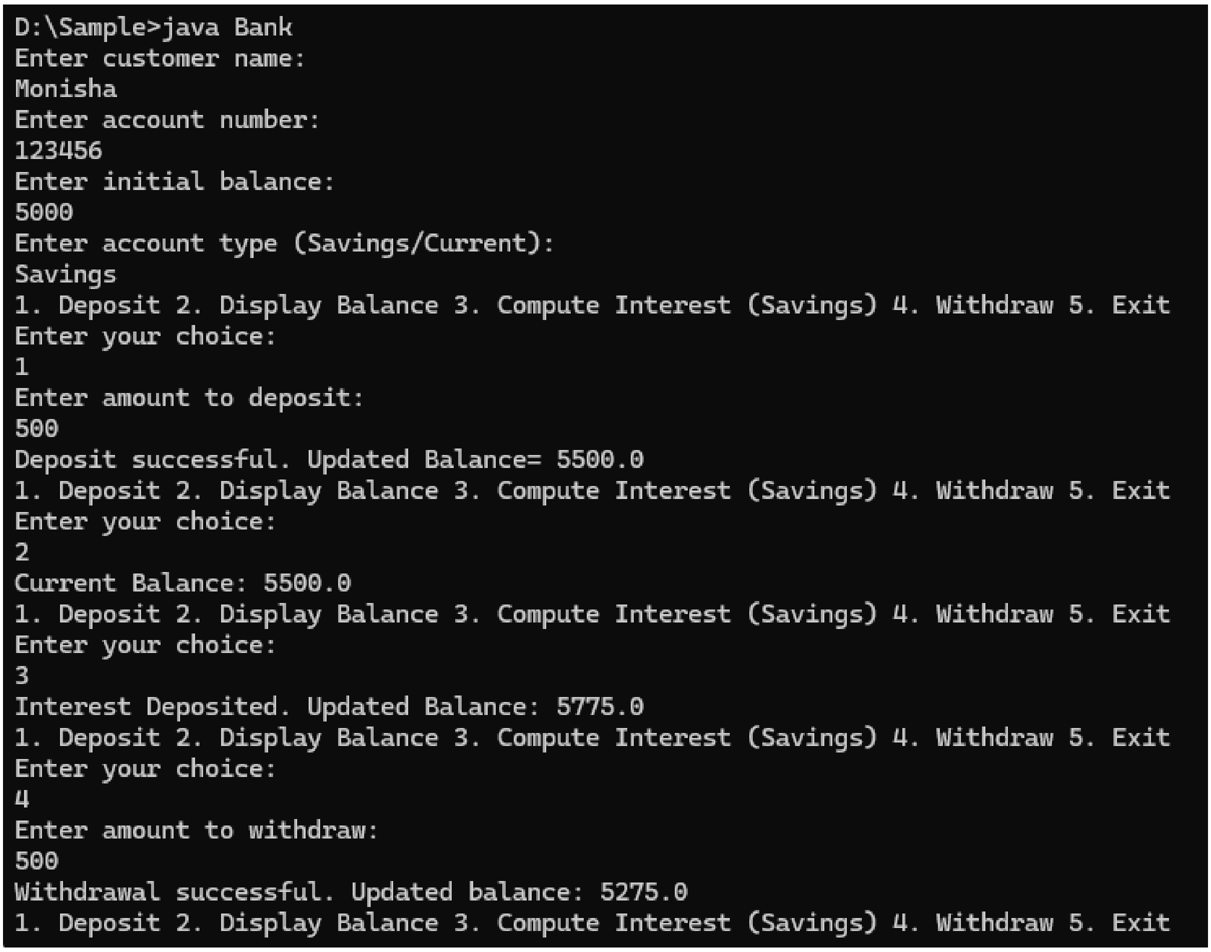
```
import java.util.Scanner;
class Account{
String CustomerName;
String AccountNumber;
String AccountType;
double Balance;

public Account(String CustomerName, String AccountNumber, String AccountType, double
Balance)
{
this.CustomerName=CustomerName;
this.AccountNumber=AccountNumber;
this.AccountType=AccountType;
this.Balance=Balance;
}
public void deposit(double amount){
Balance+=amount;
System.out.println("Deposit successful. Updated Balance= "+Balance);
}
public void displayBalance(){
System.out.println("Current Balance: "+Balance);
}
}

class Sav_acct extends Account{
double interestRate=0.05;

public Sav_acct(String CustomerName, String AccountNumber, double Balance){
super(CustomerName, AccountNumber, "Savings", Balance);
}
```

```
public void computeAndDepositInterest(){
double interest= Balance*interestRate;
Balance+=interest;
System.out.println("Interest Deposited. Updated Balance: "+Balance);
}
public void withdraw(double amount) {
if (amount <= Balance) {
Balance -= amount;
System.out.println("Withdrawal successful. Updated balance: " + Balance);
}
else {
System.out.println("Insufficient balance!");
}
}
}

class Cur_acct extends Account {
double minimumBalance = 500.0;
double serviceCharge = 50.0;

public Cur_acct(String CustomerName, String AccountNumber, double Balance) {
super(CustomerName, AccountNumber, "Current", Balance);
}

public void checkMinimumBalance() {
if (Balance < minimumBalance) {
Balance -= serviceCharge;
System.out.println("Balance below minimum. Service charge imposed. Updated balance: " +
Balance);
}
else {
System.out.println("Minimum balance maintained.");
}
}

public void withdraw(double amount) {
if (amount <= Balance) {
Balance -= amount;
System.out.println("Withdrawal successful. Updated balance: " + Balance);
checkMinimumBalance();
}
else {
System.out.println("Insufficient Balance!");
}
}
}

public class Bank{
public static void main(String args[]){
Scanner s = new Scanner(System.in);
```

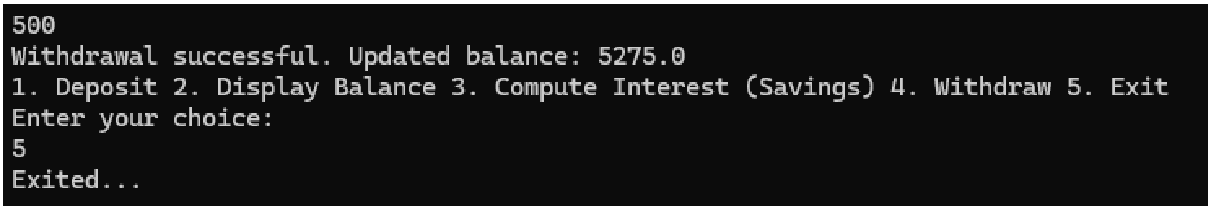
```
System.out.println("Enter customer name:");
String CustomerName = s.nextLine();

System.out.println("Enter account number:");
String AccountNumber = s.nextLine();
System.out.println("Enter initial balance:");
double initialBalance = s.nextDouble();
System.out.println("Enter account type (Savings/Current):");
String AccountType = s.next();
Account account;
if (AccountType.equalsIgnoreCase("Savings")){
account = new Sav_acct(CustomerName, AccountNumber, initialBalance);
}
else {
account = new Cur_acct(CustomerName, AccountNumber, initialBalance);
}

while (true) {
System.out.println("1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit");
System.out.println("Enter your choice:");
int choice = s.nextInt();

switch (choice) {
case 1:
    System.out.println("Enter amount to deposit:");
    double depositAmount = s.nextDouble();
    account.deposit(depositAmount);
    break;
case 2:
    account.displayBalance();
    break;
case 3:
    if (account instanceof Sav_acct) {
    ((Sav_acct) account).computeAndDepositInterest();
    } else {
    System.out.println("Interest computation is not applicable for Current accounts.");
    }
    break;
case 4:
    System.out.println("Enter amount to withdraw:");
    double withdrawAmount = s.nextDouble();
    if (account instanceof Sav_acct) {
    ((Sav_acct) account).withdraw(withdrawAmount);
    } else {
    ((Cur_acct) account).withdraw(withdrawAmount);
    }
    break;
case 5:
        System.out.println("Exited...");
        System.exit(0);
default:
        System.out.println("Invalid choice!");
}
}
}
}
```

OUTPUTS:

SAVINGS ACCOUNT

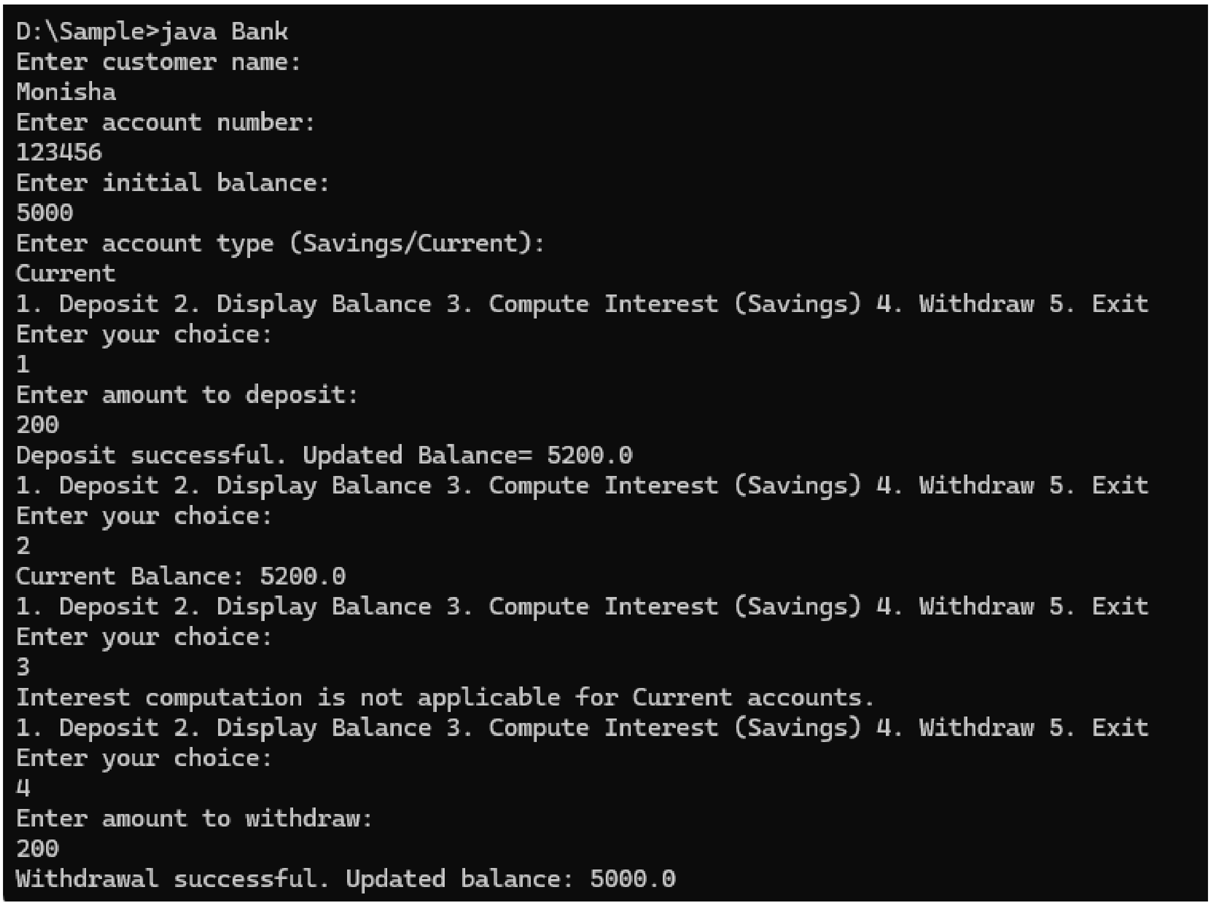


```
D:\Sample>java Bank
Enter customer name:
Monisha
Enter account number:
123456
Enter initial balance:
5000
Enter account type (Savings/Current):
Savings
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
1
Enter amount to deposit:
500
Deposit successful. Updated Balance= 5500.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
2
Current Balance: 5500.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
3
Interest Deposited. Updated Balance: 5775.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
4
Enter amount to withdraw:
500
Withdrawal successful. Updated balance: 5275.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
```

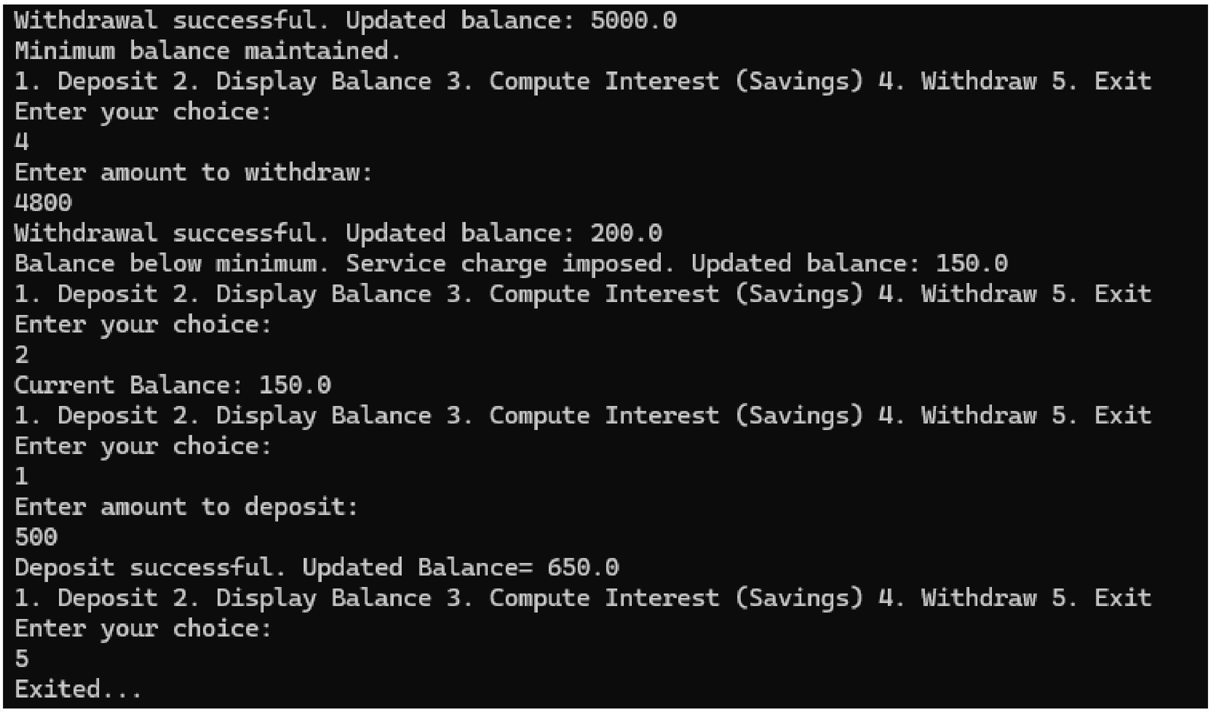


```
500
Withdrawal successful. Updated balance: 5275.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
5
Exited...
```

CURRENT ACCOUNT



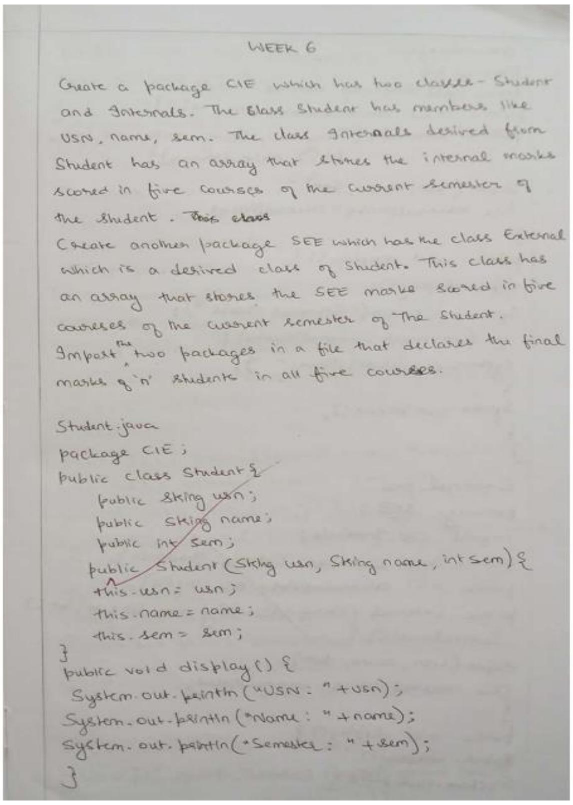
```
D:\Sample>java Bank
Enter customer name:
Monisha
Enter account number:
123456
Enter initial balance:
5000
Enter account type (Savings/Current):
Current
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
1
Enter amount to deposit:
200
Deposit successful. Updated Balance= 5200.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
2
Current Balance: 5200.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
3
Interest computation is not applicable for Current accounts.
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
4
Enter amount to withdraw:
200
Withdrawal successful. Updated balance: 5000.0
```



```
Withdrawal successful. Updated balance: 5000.0
Minimum balance maintained.
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
4
Enter amount to withdraw:
4800
Withdrawal successful. Updated balance: 200.0
Balance below minimum. Service charge imposed. Updated balance: 150.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
2
Current Balance: 150.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
1
Enter amount to deposit:
500
Deposit successful. Updated Balance= 650.0
1. Deposit 2. Display Balance 3. Compute Interest (Savings) 4. Withdraw 5. Exit
Enter your choice:
5
Exited...
```

PROGRAM 6

Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student 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.



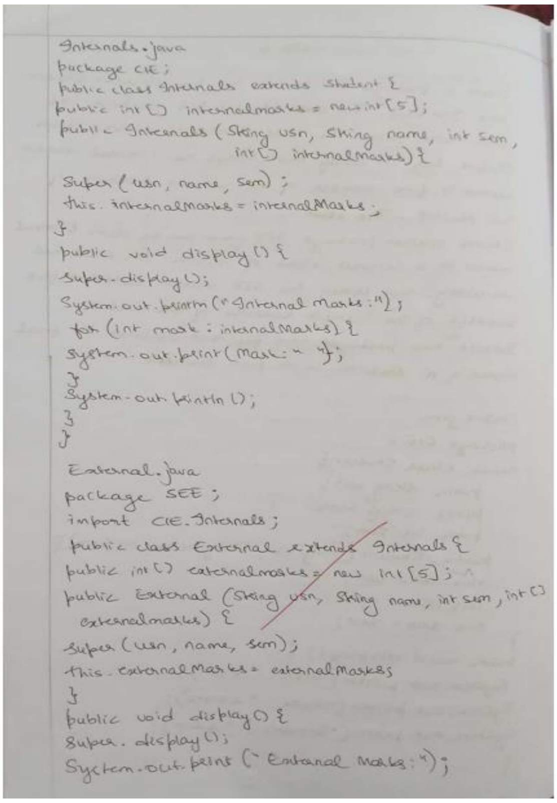
WEEK 6

Create a package CIE which has two classes- Student and Internals. The class Student has members like USN, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. ~~This class~~

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 Student. Import the two packages in a file that declares the final marks of 'n' students in all five courses.

Student.java

```
package CIE;
public class Student {
      public String usn;
      public String name;
      public int sem;
    public Student (String usn, String name, int sem) {
    this.usn = usn;
    this.name = name;
    this.sem = sem;
}
public void display () {
 System.out.println ("USN : " +usn);
System.out.println ("Name : " + name);
System.out.println("Semester : " + sem);
}
```

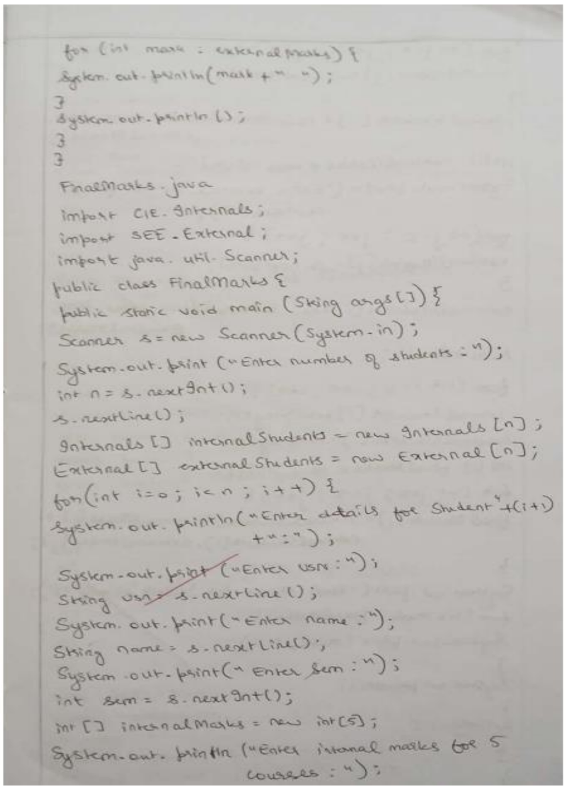


Internals.java

```
package CIE;
public class Internals extends Student {
public int[] internalmarks = new int[5];
public Internals (String usn, String name, int sem,
                         int[] internalmarks) {
 Super (usn, name, sem) ;
 this. internalmarks = internalMarks ;
}
public void display () {
Super.display();
System.out.println ("Internal marks :");
 for (int mask : internalMarks) {
 System.out.print(mask:" ");
 }
 System.out.println ();
 }
}
```

External.java

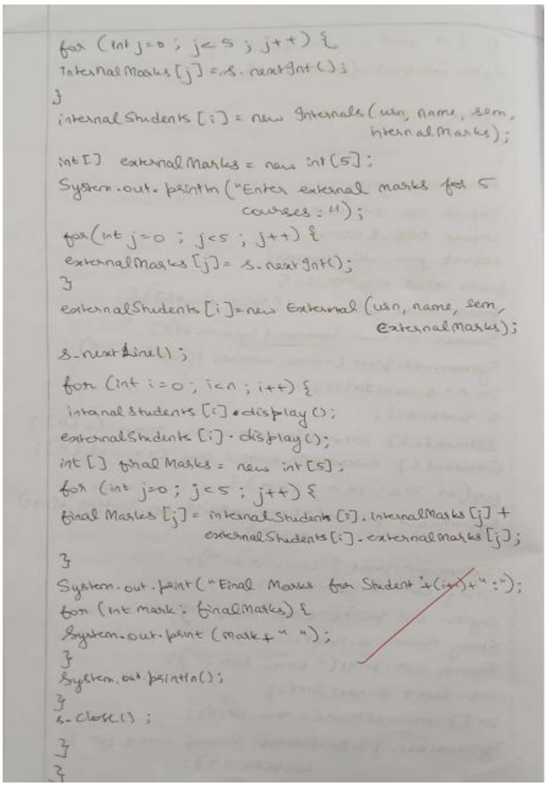
```
package SEE ;
import CIE.Internals ;
public class External extends Internals {
public int[] externalmarks = new int[5];
public External (String usn, String name, int sem, int[]
  externalmarks) {
Super (usn, name, sem);
this. externalMarks = externalMarks
}
public void display() {
Super. display();
System.out.print (" External Marks :");
```



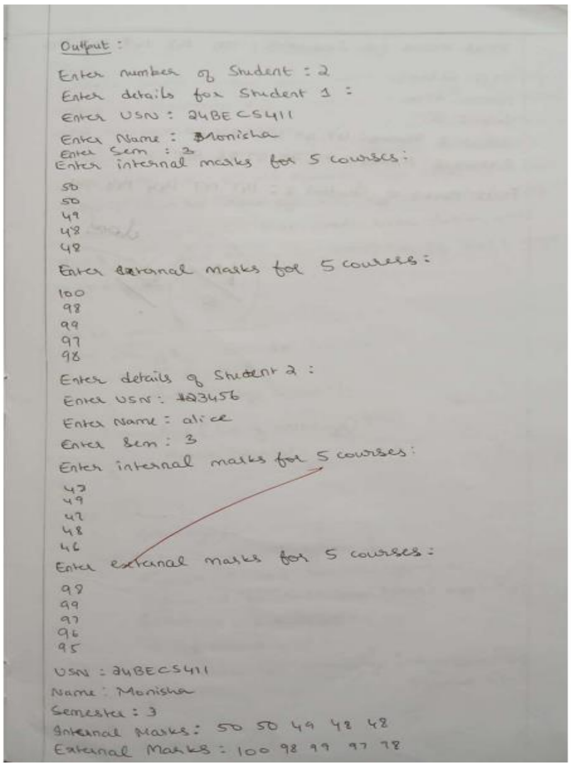
```
for (int mark : externalMarks) {
System.out.println(mark + " ");
}
System.out.println ();
}
}
```

FinalMarks.java

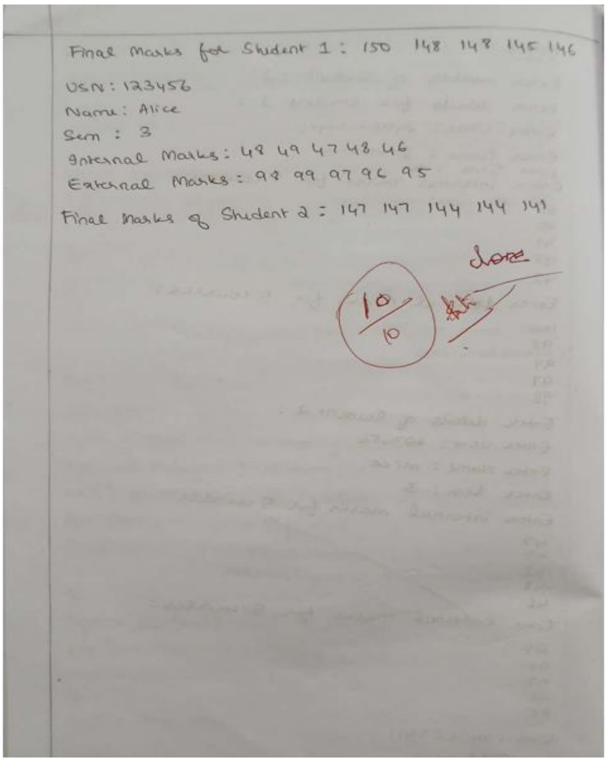
```
import CIE.Internals;
import SEE.External;
import java.util.Scanner;
public class FinalMarks {
public static void main (String args[]) {
Scanner s = new Scanner(System.in);
System.out.print ("Enter number of students : ");
int n = s.nextInt();
s.nextLine();
Internals [] internalStudents = new Internals [n];
External [] externalStudents = new External [n];
for(int i=0; i<n; i++) {
System.out.println("Enter details for Student"+(i+1)
                       + ":");
System.out.print ("Enter usn :");
String usn = s.nextLine();
System.out.print("Enter name :");
String name = s.nextLine();
System.out.print(" Enter sem :");
int sem = s.nextInt();
int [] internalMarks = new int[5];
System.out.println ("Enter internal marks for 5
                      courses :");
```



```
for (int j=0; j<5; j++) {
InternalMarks[j] = s.nextInt();
}
internalStudents [i] = new Internals (usn, name, sem,
                                       internalMarks);
int[] externalMarks = new int[5];
System.out.println ("Enter external marks for 5
                         courses :");
for(int j=0; j<5; j++) {
externalMarks [j] = s.nextInt();
}
externalStudents [i]=new External (usn, name, sem,
                                    externalMarks);
s.nextLine();
for (int i=0; i<n; i++) {
 internalStudents [i].display();
externalStudents [i].display();
int [] finalMarks = new int[5];
for (int j=0; j<5; j++) {
finalMarks [j] = internalStudents [i].internalMarks [j] +
              externalStudents[i].externalMarks [j];
}
System.out.print("Final Marks for Student "+(i+1)+" :");
for (int mark : finalMarks) {
System.out.print (mark+" ");
}
System.out.println();
}
s.close() ;
}
}
```



```
Output :
Enter number of Student : 2
Enter details for Student 1 :
Enter USN : 24BECS411
Enter Name : Monisha
Enter Sem : 3
Enter internal marks for 5 courses:
50
50
49
48
48
Enter external marks for 5 courses:
100
98
99
97
98
Enter details of Student 2 :
Enter USN : 123456
Enter Name : alice
Enter Sem : 3
Enter internal marks for 5 courses:
47
49
47
48
46
Enter external marks for 5 courses:
98
99
97
96
95
USN : 24BECS411
Name : Monisha
Semester : 3
Internal Marks: 50 50 49 48 48
External Marks : 100 98 99 97 98
```



```
Final Marks for Student 1 : 150 148 148 145 146
USN : 123456
Name : Alice
Sem : 3
Internal Marks : 48 49 47 48 46
External Marks : 98 99 97 96 95
Final Marks of Student 2 : 147 147 144 144 141
```

done

10/10

SOURCE CODE:

Student.java
```
package CIE;

public class Student {
    public String usn;
    public String name;
    public int sem;

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

    public void display() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}
```

Internals.java
```
package CIE;

public class Internals extends Student {
    public int[] internalMarks = new int[5];
public Internals(String usn, String name, int sem, int[] internalMarks) {
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }

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

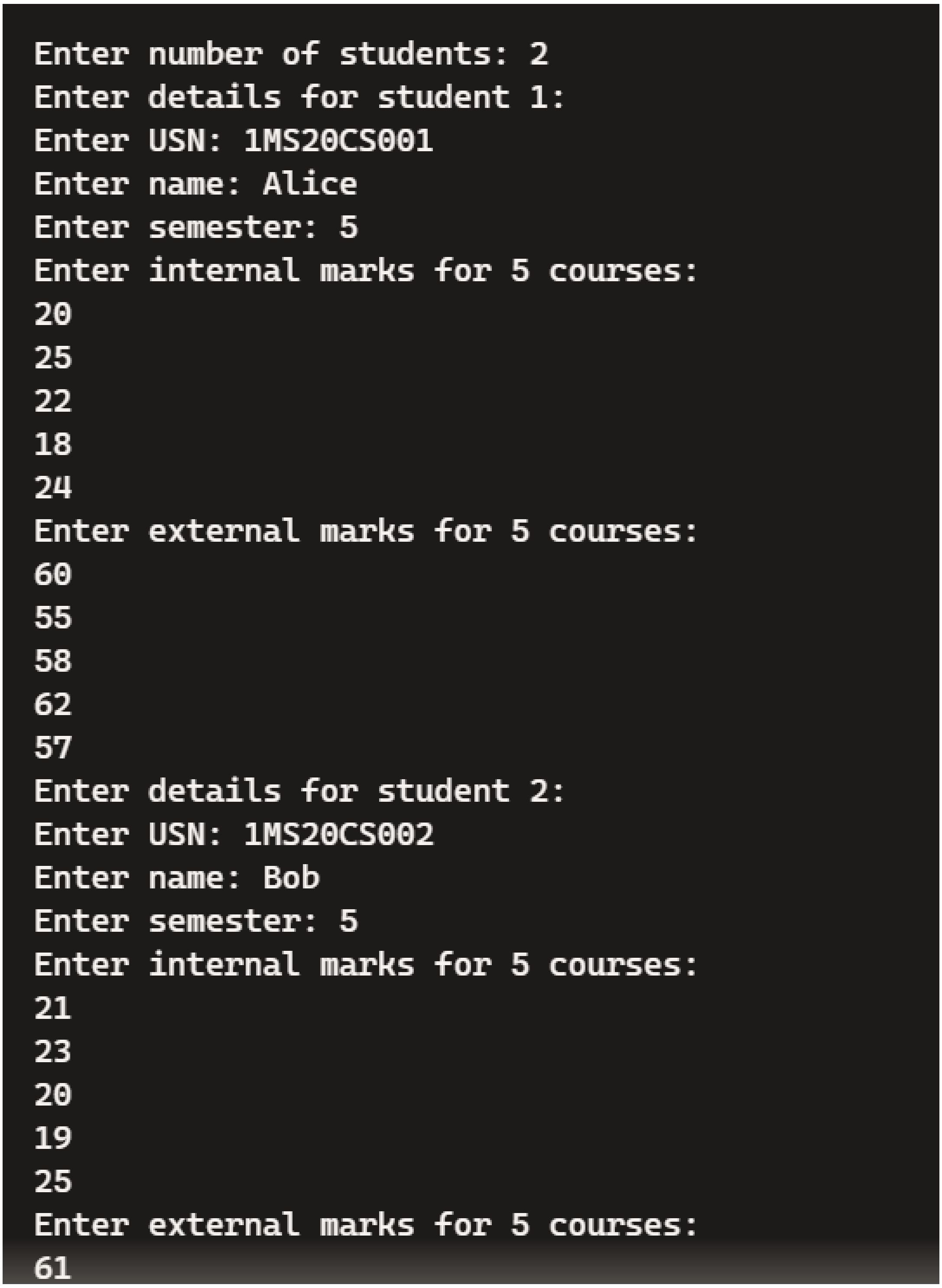
External.java

```
package SEE;

import CIE.Student;

public class External extends Student {
    public int[] externalMarks = new int[5];

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

    @Override
    public void display() {
        super.display();
        System.out.print("External Marks: ");
        for (int mark : externalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
    }
}
```

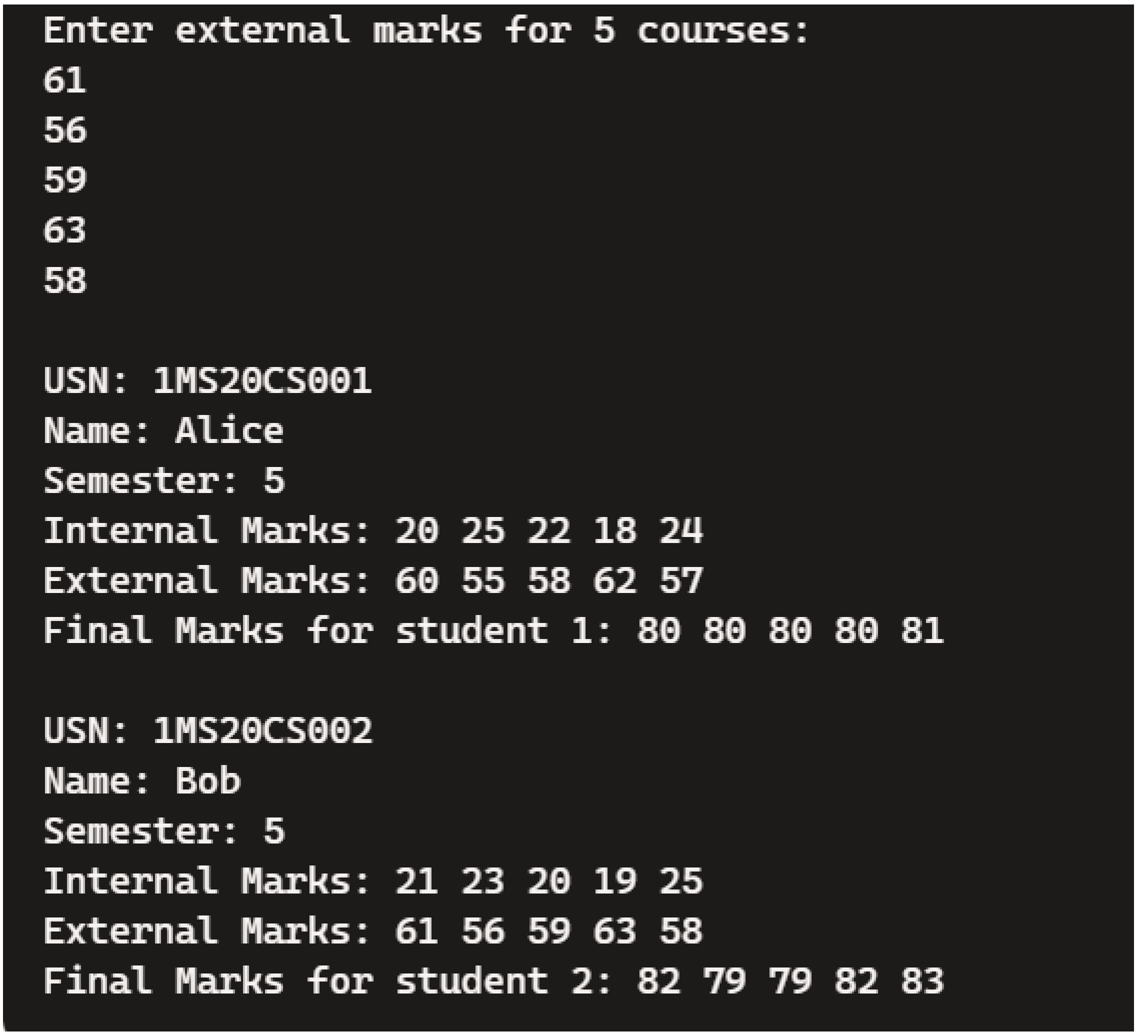
FinalMarks.java

```
import CIE.Internals;
import SEE.External;
import java.util.Scanner;

public class FinalMarks {
    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();  // Consume newline
        Internals[] internalStudents = new Internals[n];
        External[] externalStudents = new External[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for student " + (i + 1) + ":");

            System.out.print("Enter USN: ");
            String usn = scanner.nextLine();

            System.out.print("Enter name: ");
            String name = scanner.nextLine();

            System.out.print("Enter 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();
            }

            internalStudents[i] = new Internals(usn, name, sem, internalMarks);

            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();
            }

            externalStudents[i] = new External(usn, name, sem, externalMarks);
            scanner.nextLine();  // Consume newline
        }

        // Display student details and final marks
        for (int i = 0; i < n; i++) {
            internalStudents[i].display();
            externalStudents[i].display();

            int[] finalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                finalMarks[j] = internalStudents[i].internalMarks[j] +
    externalStudents[i].externalMarks[j];
            }

            System.out.print("Final Marks for student " + (i + 1) + ": ");
            for (int mark : finalMarks) {
                System.out.print(mark + " ");
            }
        System.out.println();
        }
        scanner.close();
    }
}
```

OUTPUTS:



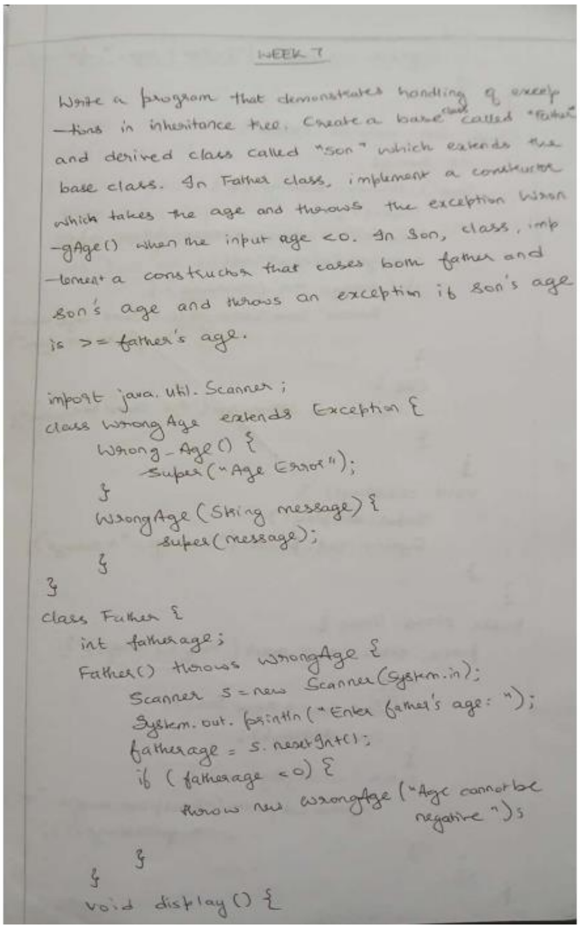
```
Enter number of students: 2
Enter details for student 1:
Enter USN: 1MS20CS001
Enter name: Alice
Enter semester: 5
Enter internal marks for 5 courses:
20
25
22
18
24
Enter external marks for 5 courses:
60
55
58
62
57
Enter details for student 2:
Enter USN: 1MS20CS002
Enter name: Bob
Enter semester: 5
Enter internal marks for 5 courses:
21
23
20
19
25
Enter external marks for 5 courses:
61
```



```
Enter external marks for 5 courses:
61
56
59
63
58

USN: 1MS20CS001
Name: Alice
Semester: 5
Internal Marks: 20 25 22 18 24
External Marks: 60 55 58 62 57
Final Marks for student 1: 80 80 80 80 81

USN: 1MS20CS002
Name: Bob
Semester: 5
Internal Marks: 21 23 20 19 25
External Marks: 61 56 59 63 58
Final Marks for student 2: 82 79 79 82 83
```

PROGRAM 7

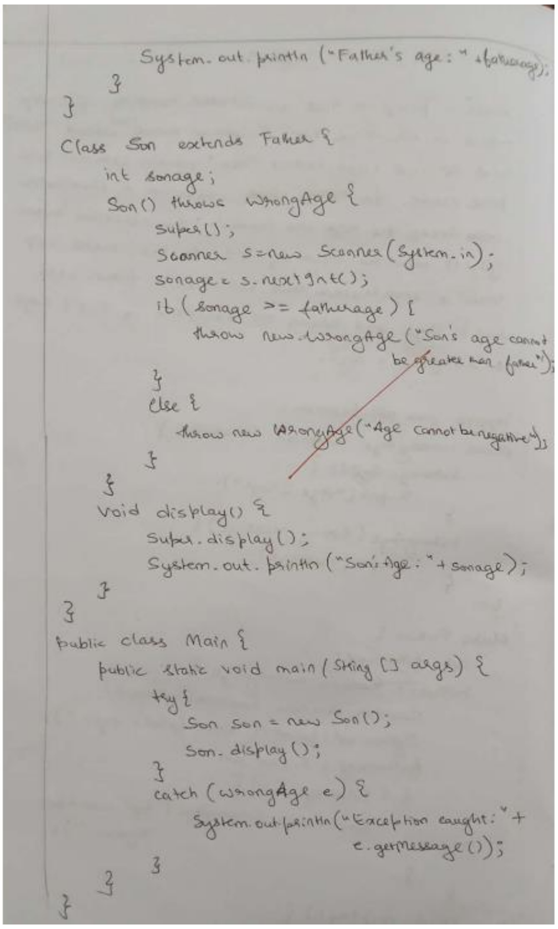
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son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



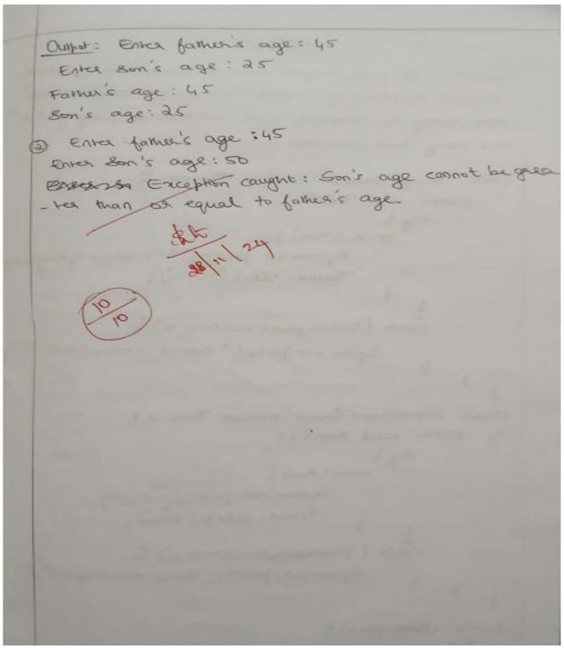
WEEK 7

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 cases both father and son's age and throws an exception if son's age is >= father's age.

```
import java. util. Scanner;
class WrongAge extends Exception {
      Wrong_Age() {
          super("Age Error");
      }
      WrongAge(String message) {
          super(message);
      }
}
class Father {
    int fatherage;
    Father() throws WrongAge {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter father's age: ");
        fatherage = s.nextInt();
        if (fatherage <0) {
            throw new WrongAge("Age cannot be
                                  negative");
        }
    }
    void display() {
```



```
        System.out.println("Father's age:" +fatherage);
    }
}
Class Son extends Father {
    int sonage;
    Son() throws WrongAge {
        super();
        Scanner s=new Scanner(System.in);
        sonage= s.nextInt();
        if (sonage >= fatherage) {
            throw new WrongAge("Son's age cannot
                              be greater than father");
        }
        else {
          throw new WrongAge("Age cannot be negative");
        }
    }
    void display() {
        super.display();
        System.out.println("Son's Age:"+ sonage);
    }
}
public class Main {
    public static void main(String [] args) {
        try {
            Son son = new Son();
            Son.display();
        }
        catch (WrongAge e) {
            System.out.println("Exception caught:" +
                                   e.getMessage());
        }
    }
}
```



```
Output: Enter father's age: 45
 Enter son's age: 25
Father's age: 45
Son's age: 25
② Enter father's age :45
Enter Son's age: 50
Exception caught: Son's age cannot be grea
-ter than or equal to father's age.
```

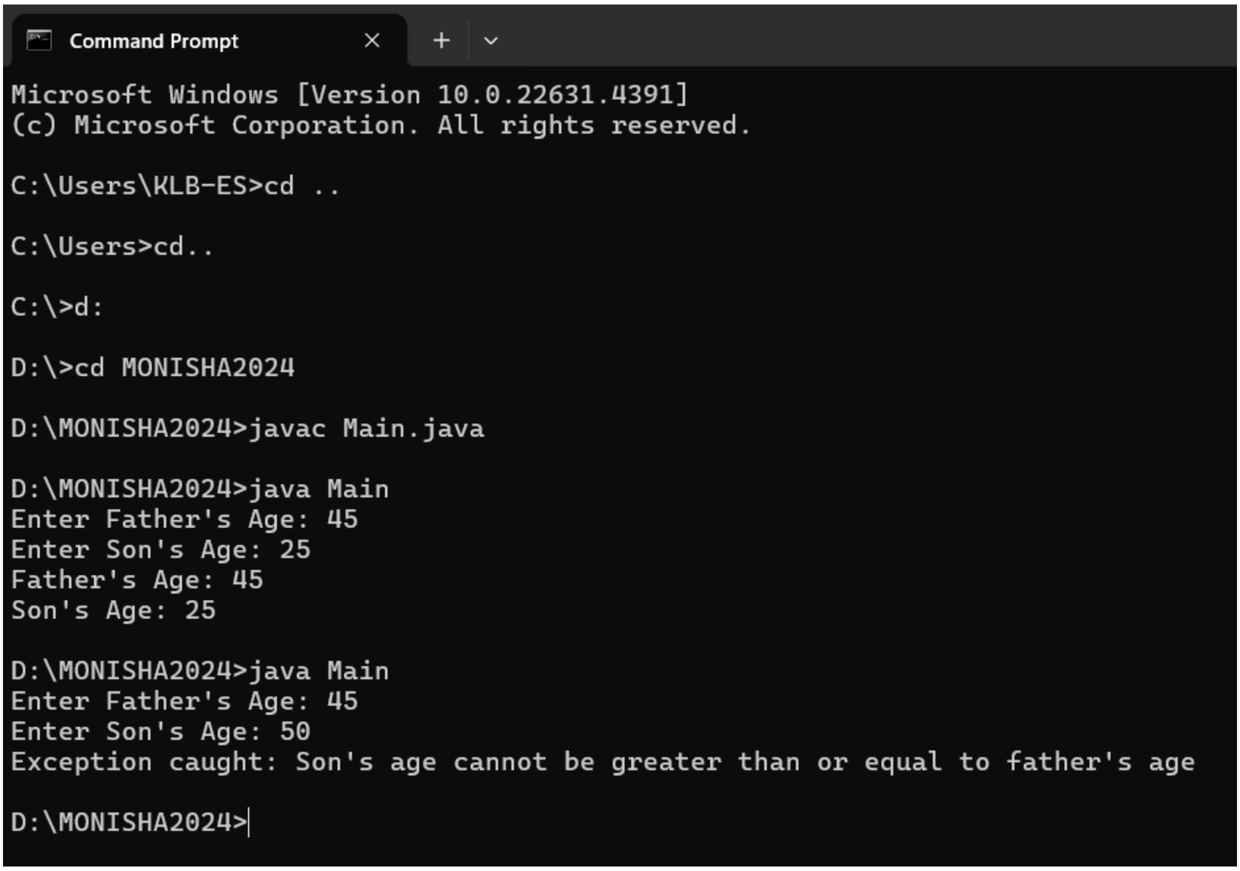
28/11/24

10/10

SOURCE CODE:

```
import java.util.Scanner;

class WrongAge extends Exception {
    WrongAge() {
        super("Age Error");
    }

    WrongAge(String message) {
        super(message);
    }
}

class Father {
    int fatherAge;

    Father() throws WrongAge {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter Father's Age: ");
        fatherAge = s.nextInt();
        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }

    void display() {
        System.out.println("Father's Age: " + fatherAge);
    }
}

class Son extends Father {
    int sonAge;

    Son() throws WrongAge {
        super(); // Calling the constructor of Father class
        Scanner s = new Scanner(System.in);
        System.out.print("Enter Son's Age: ");
        sonAge = s.nextInt();
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to father's age");
        } else if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }


    void display() {
        super.display(); // Display father's age
        System.out.println("Son's Age: " + sonAge);
}
}

public class Main {
    public static void main(String[] args) {
        try {
            Son son = new Son(); // Creating an instance of Son
            son.display(); // Displaying ages
        } catch (WrongAge e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

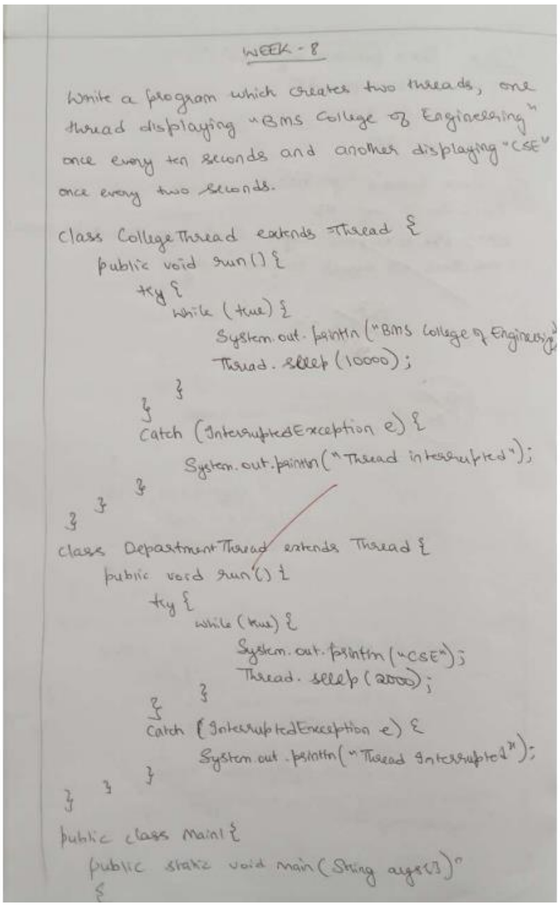
OUTPUTS:



```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\KLB-ES>cd ..

C:\Users>cd..

C:\>d:

D:\>cd MONISHA2024

D:\MONISHA2024>javac Main.java

D:\MONISHA2024>java Main
Enter Father's Age: 45
Enter Son's Age: 25
Father's Age: 45
Son's Age: 25

D:\MONISHA2024>java Main
Enter Father's Age: 45
Enter Son's Age: 50
Exception caught: Son's age cannot be greater than or equal to father's age

D:\MONISHA2024>
```

PROGRAM 8

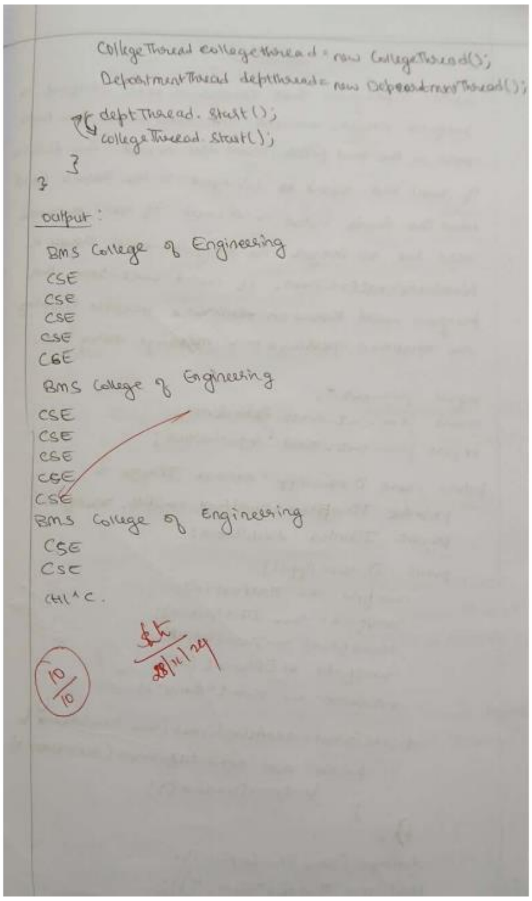
Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.



WEEK - 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.

```
class CollegeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            }
        }
        Catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
}
class DepartmentThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        }
        Catch (InterruptedException e) {
            System.out.println("Thread Interrupted");
        }
    }
}
public class Main {
    public static void main(String args[])
    {
```



```
        CollegeThread collegethread = new CollegeThread();
        DepartmentThread deptthread = new DepartmentThread();

        deptThread.start();
        collegeThread.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
Ctrl^C.
```

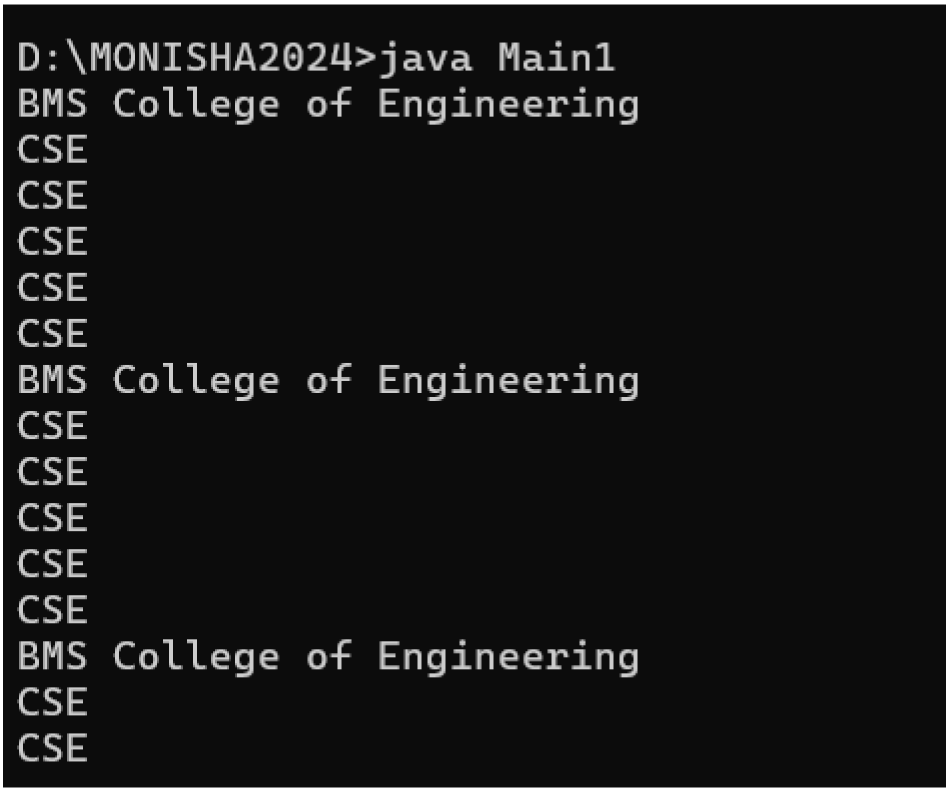
10/10

28/11/24

SOURCE CODE:

```
class CollegeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000); // Sleep for 10 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
}

class DepartmentThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000); // Sleep for 2 seconds
            }
        }
catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
}

public class Main1 {
    public static void main(String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        DepartmentThread departmentThread = new DepartmentThread();


        departmentThread.start();
        collegeThread.start();
    }
}
```

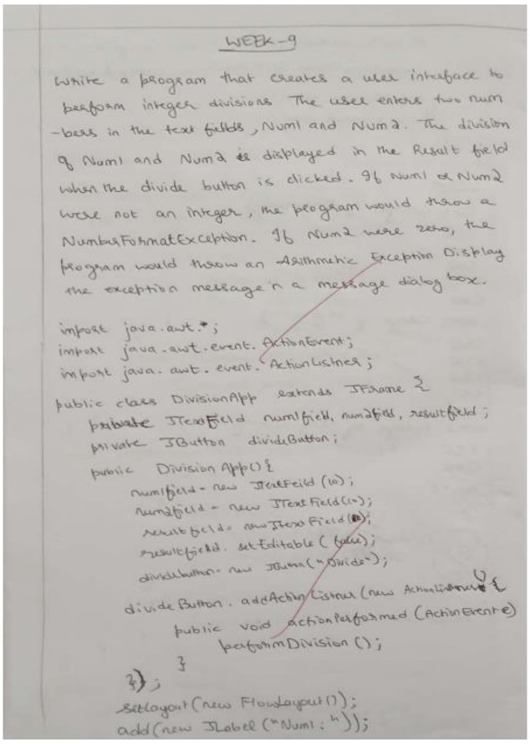
OUTPUTS:



```
D:\MONISHA2024>java Main1
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
```

PROGRAM 9

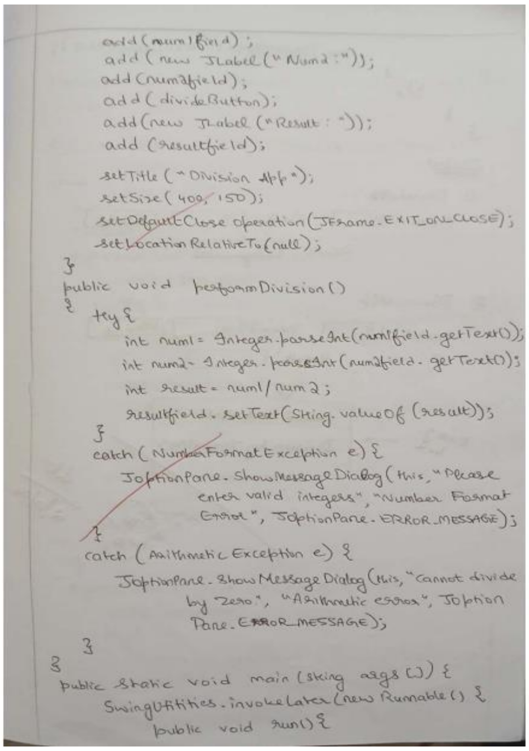
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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



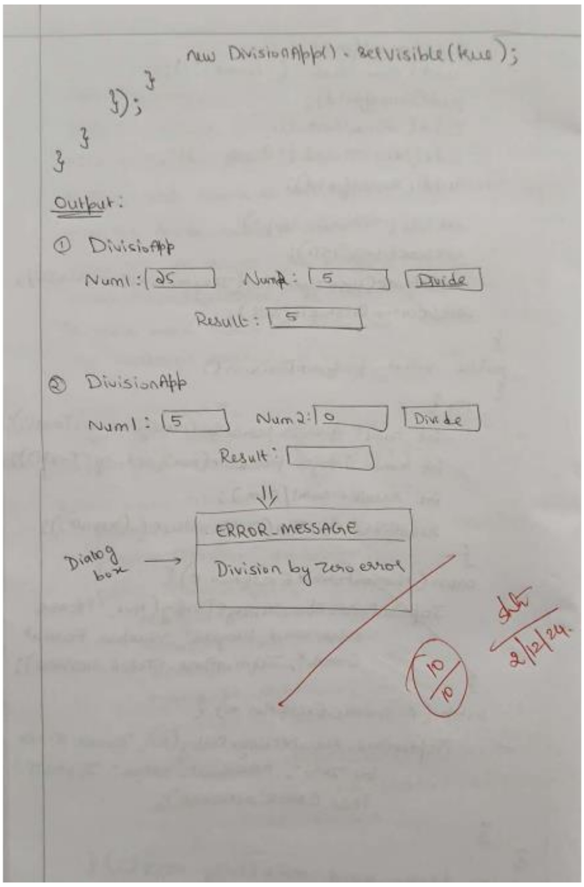
WEEK-9

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 message in a message dialog box.

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListner;
public class DivisionApp extends JFrame {
    private JTextField num1field, num2field, resultfield;
    private JButton divideButton;
    public DivisionApp(){
        num1field = new JTextFeild (10);
        num2field = new JText Field(10);
        resultfield = new JText Field(10);
        resultfield. setEditable ( false);
        divideButton = new JButton("Divide");
      divideButton. addActionListner (new ActionListener() {
          public void actionPerformed (ActionEvent e)
              performDivision ();
          }
      });
     setLayout(new FlowLayout());
     add(new JLabel ("Num1 : "));
```



```
        add(num1field);
        add(new JLabel("Num2 :"));
        add(num2field);
        add(divideButton);
        add(new JLabel ("Result : "));
        add (resultfield);
        setTitle ("Division App");
        setSize( 400, 150);
        setDefaultClose Operation(JFrame.EXIT_ON_CLOSE);
        setLocation RelativeTo(null);
}
public void performDivision()
{
    try {
        int num1= Integer.parseInt(num1field.getText());
        int num2= Integer.parseInt(num2field.getText());
        int result= num1/num2;
        resultfield.setText(String.valueOf (result));
    }
    catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Please
                enter valid integers", "Number Format
                Error", JOptionPane.ERROR_MESSAGE);
    }
    catch (ArithmeticException e) {
        JOptionPane.showMessageDialog(this, "cannot divide
                by zero", "Arithmetic error", JOption
                Pane.ERROR_MESSAGE);
    }
}
public static void main (String args[]) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run(){
```



```
            new DivisionApp().setVisible(true);
        }
    });
  }
}
```

Output:

① DivisioApp

Num1 : 25   Num2 : 5   Divide

Result : 5

② DivisionApp

Num1 : 5   Num2: 0   Divide

Result :

ERROR_MESSAGE

Division by zero error

Dialog box →

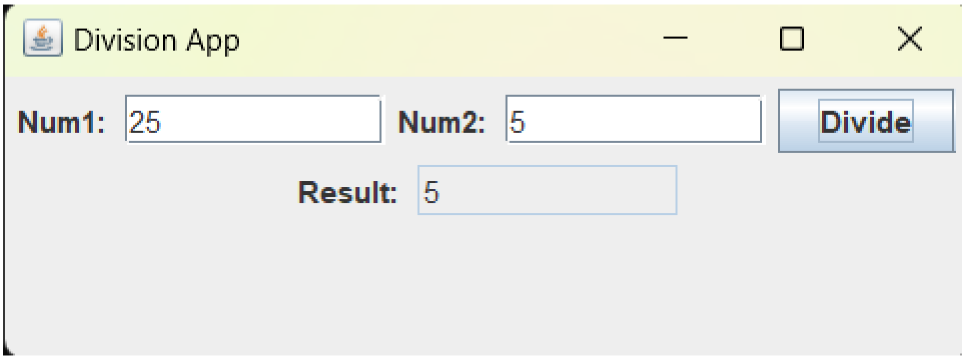
10/10

2/12/24.

SOURCE CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class DivisionApp extends JFrame {
    private JTextField num1Field, num2Field, resultField;
    private JButton divideButton;

    public DivisionApp() {
        // Create UI elements
        num1Field = new JTextField(10);
        num2Field = new JTextField(10);
        resultField = new JTextField(10);
        resultField.setEditable(false); // Make result field non-editable
        divideButton = new JButton("Divide");

        // Add action listener to the button
        divideButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                performDivision();
            }
        });

        // Set up the layout
        setLayout(new FlowLayout());
        add(new JLabel("Num1: "));
        add(num1Field);
        add(new JLabel("Num2: "));
        add(num2Field);
        add(divideButton);
        add(new JLabel("Result: "));
        add(resultField);

        // Set up the frame
        setTitle("Division App");
        setSize(400, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Center the frame
    }

    private void performDivision() {
        try {
            // Get the input values
            int num1 = Integer.parseInt(num1Field.getText());
            int num2 = Integer.parseInt(num2Field.getText());
            // Perform the division
            int result = num1 / num2;

            // Display the result
            resultField.setText(String.valueOf(result));
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter valid integers.", "Number Format
Error", JOptionPane.ERROR_MESSAGE);
        } catch (ArithmeticException e) {
            JOptionPane.showMessageDialog(this, "Cannot divide by zero.", "Arithmetic Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new DivisionApp().setVisible(true);
            }
        });
    }
}
```

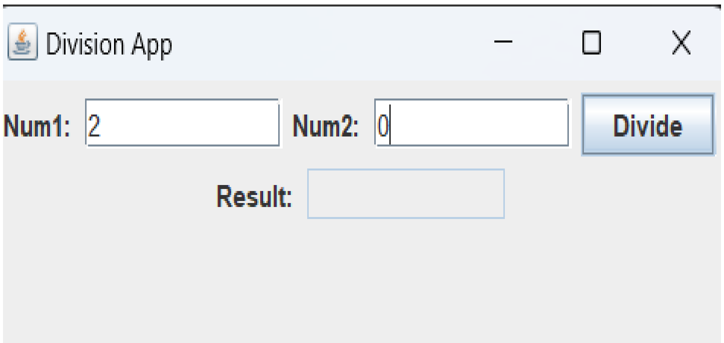
OUTPUTS:



Division App

Num1: 25 Num2: 5

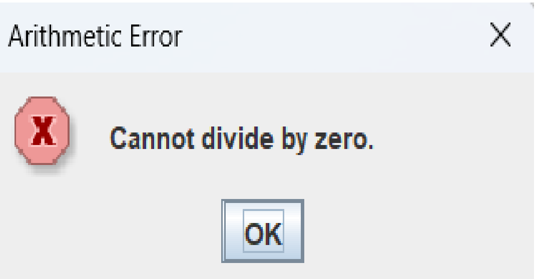
Result: 5



Division App

Num1: 2 Num2: 0

Result:



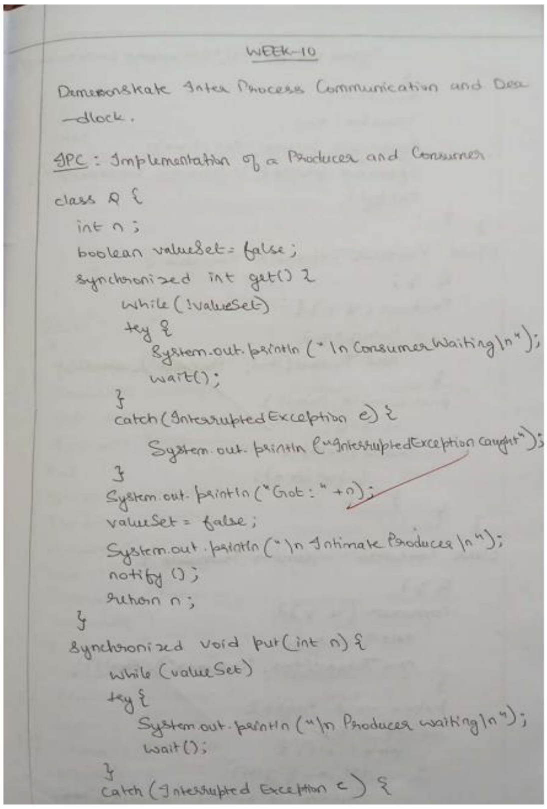
Arithmetic Error

Cannot divide by zero.

PROGRAM 10

Demonstrate Inter process Communication and deadlock

INTER PROCESS COMMUNICATION

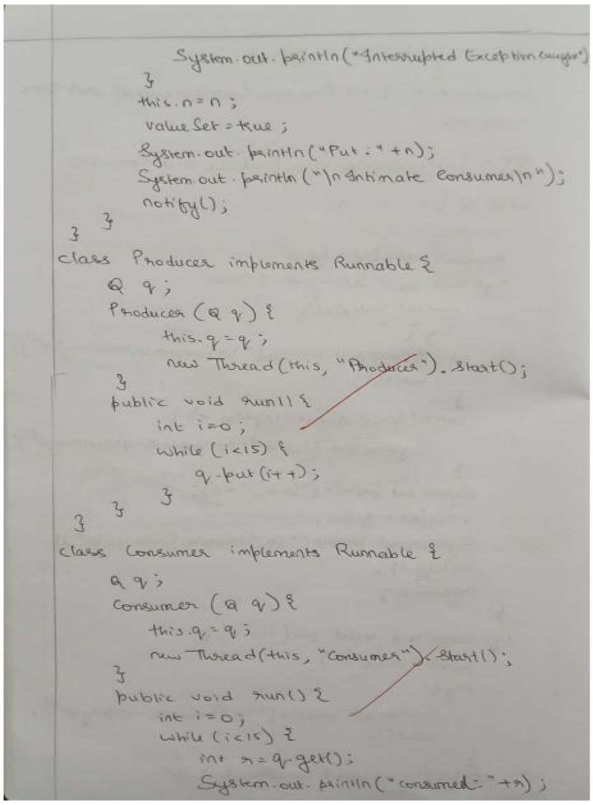


WEEK-10

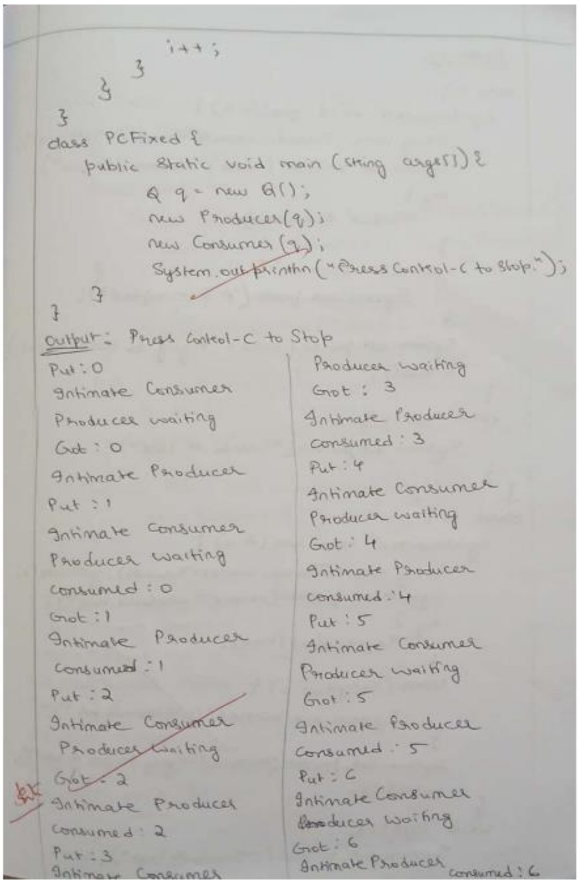
Demonstrate Inter Process Communication and Dea-dlock.

IPC : Implementation of a Producer and Consumer

```
class Q {
  int n;
  boolean valueSet= false;
  synchronized int get() {
      while (!valueSet)
      try {
        System.out.println ("\n ConsumerWaiting\n");
        wait();
      }
      catch(InterruptedException e) {
          System.out.println ("InterruptedException caught");
      }
     System.out.println ("Got : " +n);
     valueSet = false;
     System.out.println ("\n Intimate Producer \n");
     notify ();
     return n;
  }
 synchronized void put(int n) {
     while (valueSet)
     try {
        System.out.println ("\n Producer waiting\n");
        wait();
     }
     catch (Interrupted Exception e) {
```



```
            System.out.println("Interrupted Exception caught")
        }
        this.n=n;
        valueSet = true;
        System.out.println("Put : " +n);
        System.out println ("\n Intimate 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                     Producer waiting
Intimate Consumer           Got : 3
Producer waiting            Intimate Producer
Got : 0                     consumed : 3
Intimate Producer           Put : 4
Put : 1                     Intimate Consumer
Intimate Consumer           Producer waiting
Producer waiting            Got : 4
consumed : 0                Intimate Producer
Got : 1                     consumed : 4
Intimate Producer           Put : 5
consumed : 1                Intimate Consumer
Put : 2                     Producer waiting
Intimate Consumer           Got : 5
Producer waiting            Intimate Producer
Got : 2                     Consumed : 5
Intimate Producer           Put : 6
consumed : 2                Intimate Consumer
Put : 3                     Producer waiting
Intimate Consumer           Got : 6
                            Intimate Producer consumed : 6
```

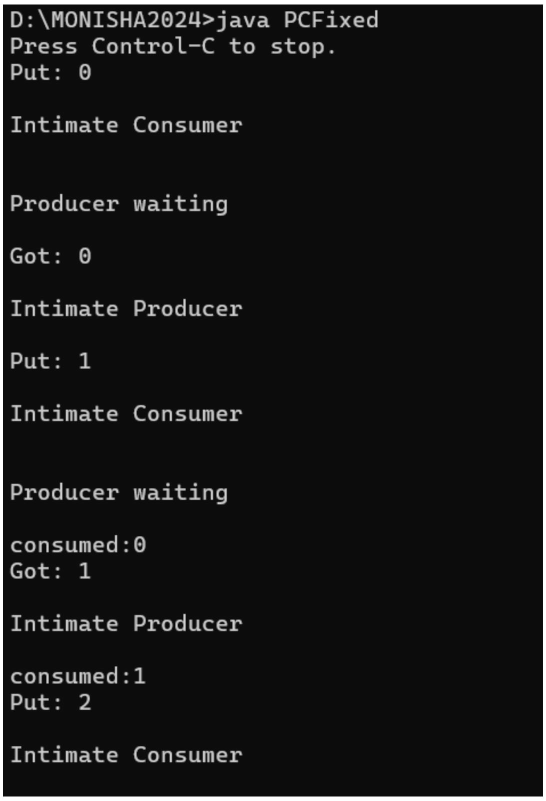
SOURCE CODE:

```
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;
```

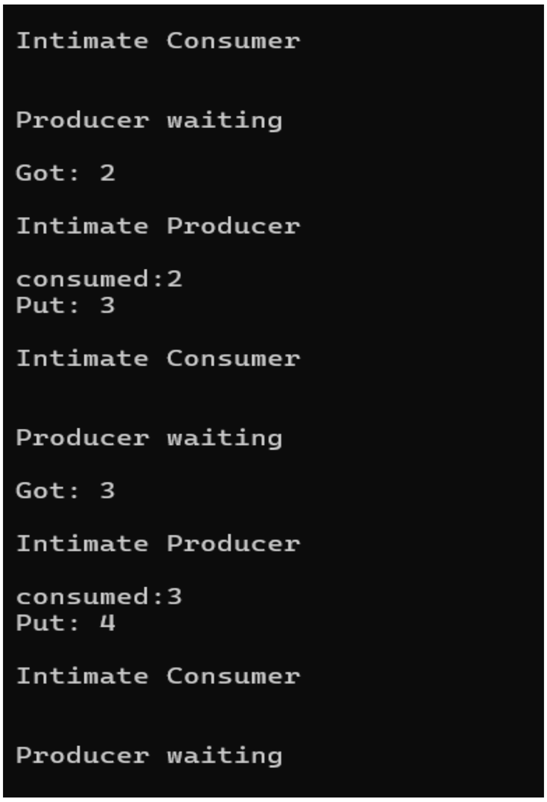
```
valueSet = true;
System.out.println("Put: " + n);

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.");

}

}
```

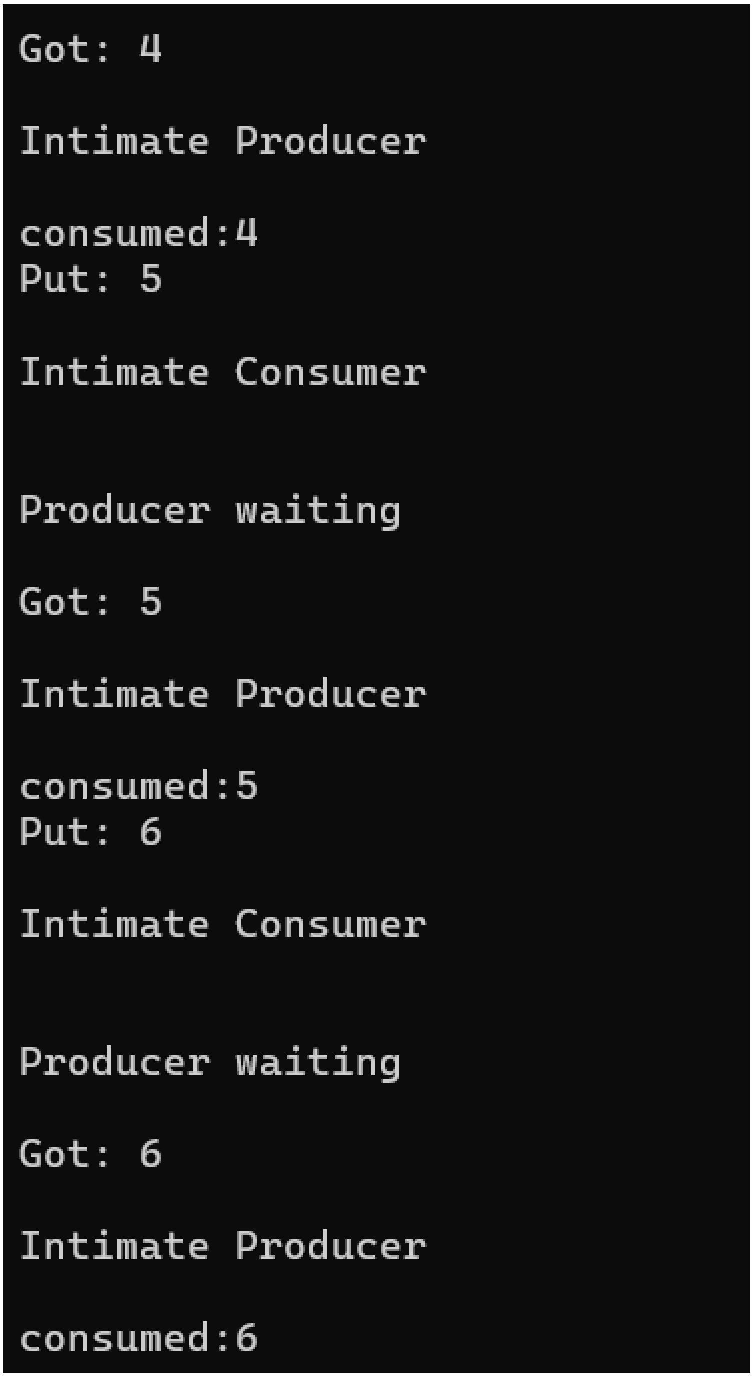
OUTPUTS:



```
D:\MONISHA2024>java PCFixed
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
```

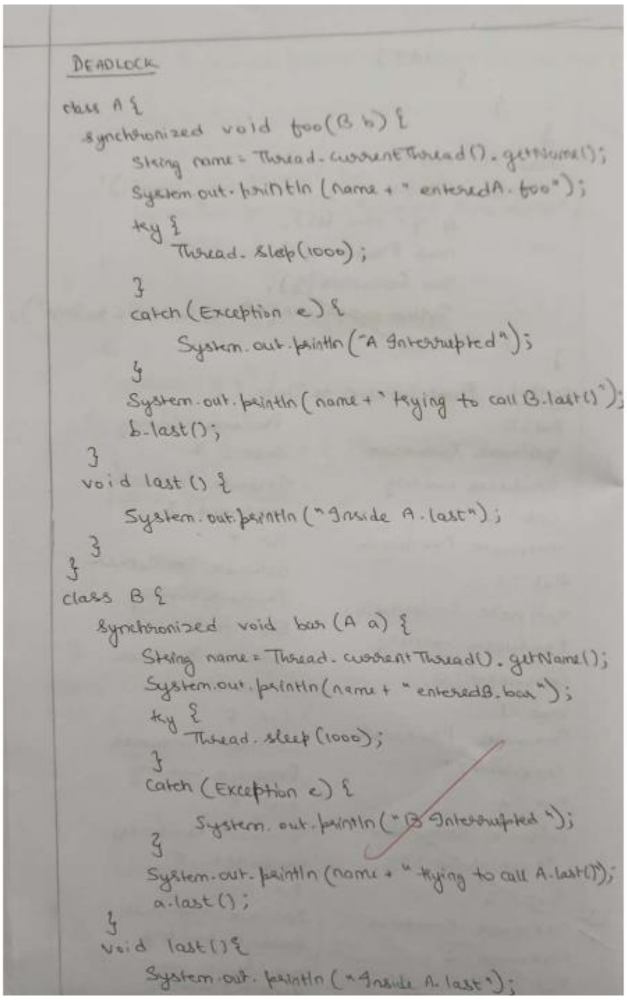


```
Intimate Consumer


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
```



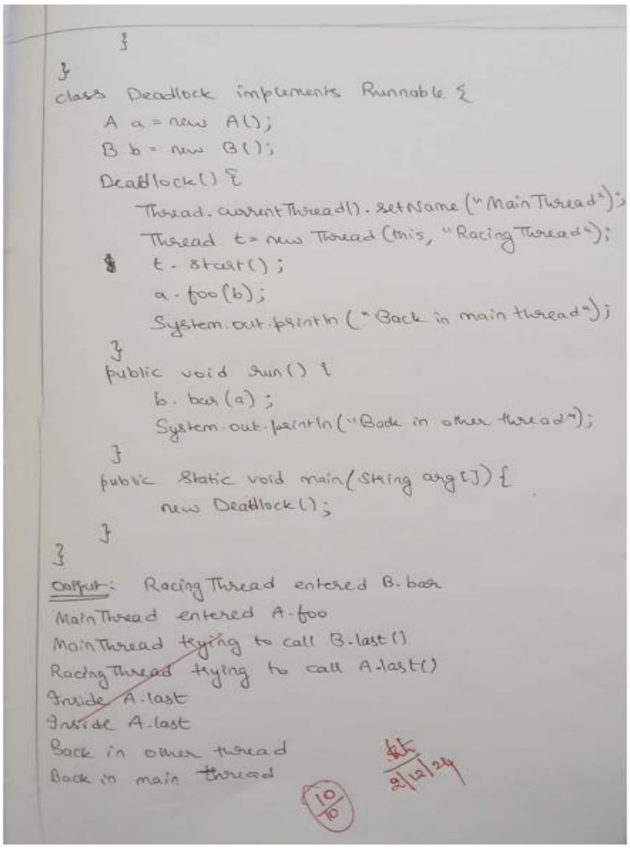
```
Got: 4

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:6
```

DEADLOCK



DEADLOCK

```
class A {
 synchronized void foo(B b) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " enteredA.foo");
      try {
          Thread.Slep(1000);
      }
      catch(Exception e) {
          System.out.println("A Interrupted");
      }
      System.out.println(name + " trying to call B.last()");
      b.last();
  }
  void last() {
      System.out.println("Inside A.last");
  }
}
class B {
  synchronized void bar(A a) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " enteredB.bar");
      try {
          Thread.sleep(1000);
      }
      catch(Exception e) {
          System.out.println("B Interrupted");
      }
      System.out.println(name + " trying to call A.last()");
      a.last();
  }
  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);
          System.out.println("Back in main thread");
      }
      public void run() {
          b.bar(a);
          System.out.println("Back in other thread");
      }
      public static void main(String arg[]) {
          new Deadlock();
      }
}
```

Output:

```
RacingThread entered B.bar
MainThread entered A.foo
MainThread trying to call B.last()
RacingThread trying to call A.last()
Inside A.last
Inside A.last
Back in other thread
Back in main thread
```

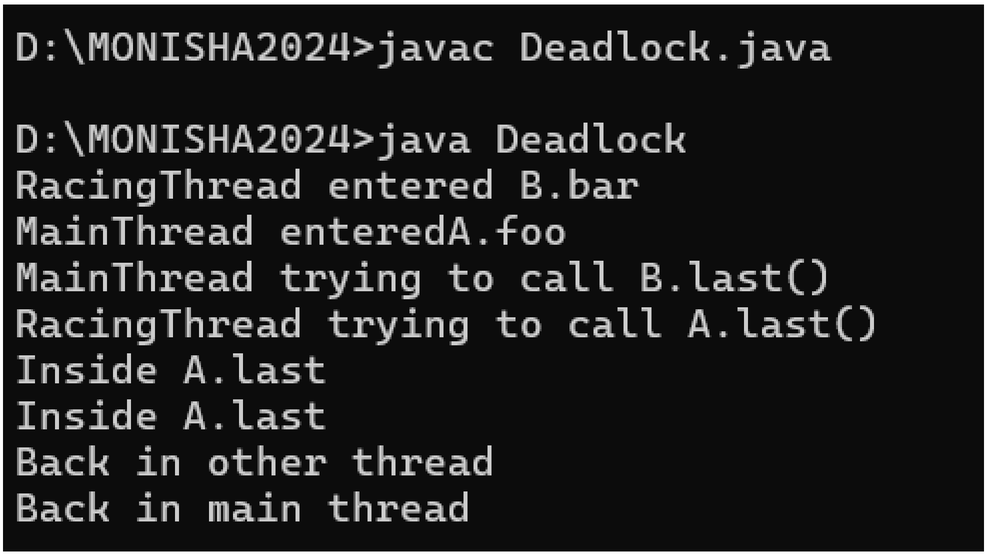
10/10

2/12/24

SOURCE CODE:

```
class A {

synchronized void foo(B b) {

String name =
Thread.currentThread().getName();

System.out.println(name + " enteredA.foo");

try {

Thread.sleep(1000);

} catch(Exception e) {

System.out.println("A Interrupted");

}
System.out.println(name + " trying to call B.last()");

b.last();

}

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();

}
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();

}

}
```

OUTPUTS:



```
D:\MONISHA2024>javac Deadlock.java

D:\MONISHA2024>java Deadlock
RacingThread entered B.bar
MainThread enteredA.foo
MainThread trying to call B.last()
RacingThread trying to call A.last()
Inside A.last
Inside A.last
Back in other thread
Back in main thread
```