VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

K Indu (1BM23CS131)

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 **K Indu (1BM23CS131),** 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.

Seema Patil Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/Indu7777/JAVA-LAB-PROGRAMS

Program 1
Implement Quadratic Equation

Algorithm:

(1)
```
import java.util.Scanner;
class quadratic
{
 int a,b,c;
 double r1,r2,d;
 void get()
 {
  Scanner in=new Scanner (System.in);
  System.out.print("Enter the coefficient
      of a");
  int a= in.nextInt();
  System.out.println ("Enter b");
  int b=in.nextInt();
  System.out.println ("Enter c");
  int c= in.nextInt();
 }
 void compute()
 {
  if (a==0)
  {
```

System.out.println("a should not be zero");
}
else
{
d = (b*b) - (4*a*c);
if (d==0)
{
r1=r2=(-b)/(2*a);
System.out.println("The roots are equal");
System.out.println("root1="+r1+"root2="+r2);
}
else if(d>0)
{
r1=(-b+Math.sqrt(d))/(2*a);
r2=(-b-Math.sqrt(d))/(2*a);
System.out.println("the roots are real and
distinct and root1="+r1+"root2="+r2);
}
else
{

```
double realpart = -b/(2*a);
double imagpart = Math.sqrt(-d)/(2*a);
System.out.println("No real roots:
Roots are complex, root1 = "+realpart+"
+"+imagpart+"i");
system.out.println("root2 = "+realpart
+" - "+imagpart+"i");
}
}
}
class quadraticMain2
{
public static void main(String args[])
{
quadratic q = new quadratic();
q.get();
q.compute();
System.out.println("
```

Output:

Enter a: 1

Enter b: 2

Enter c: 4

The roots are equal

The root1 = -1

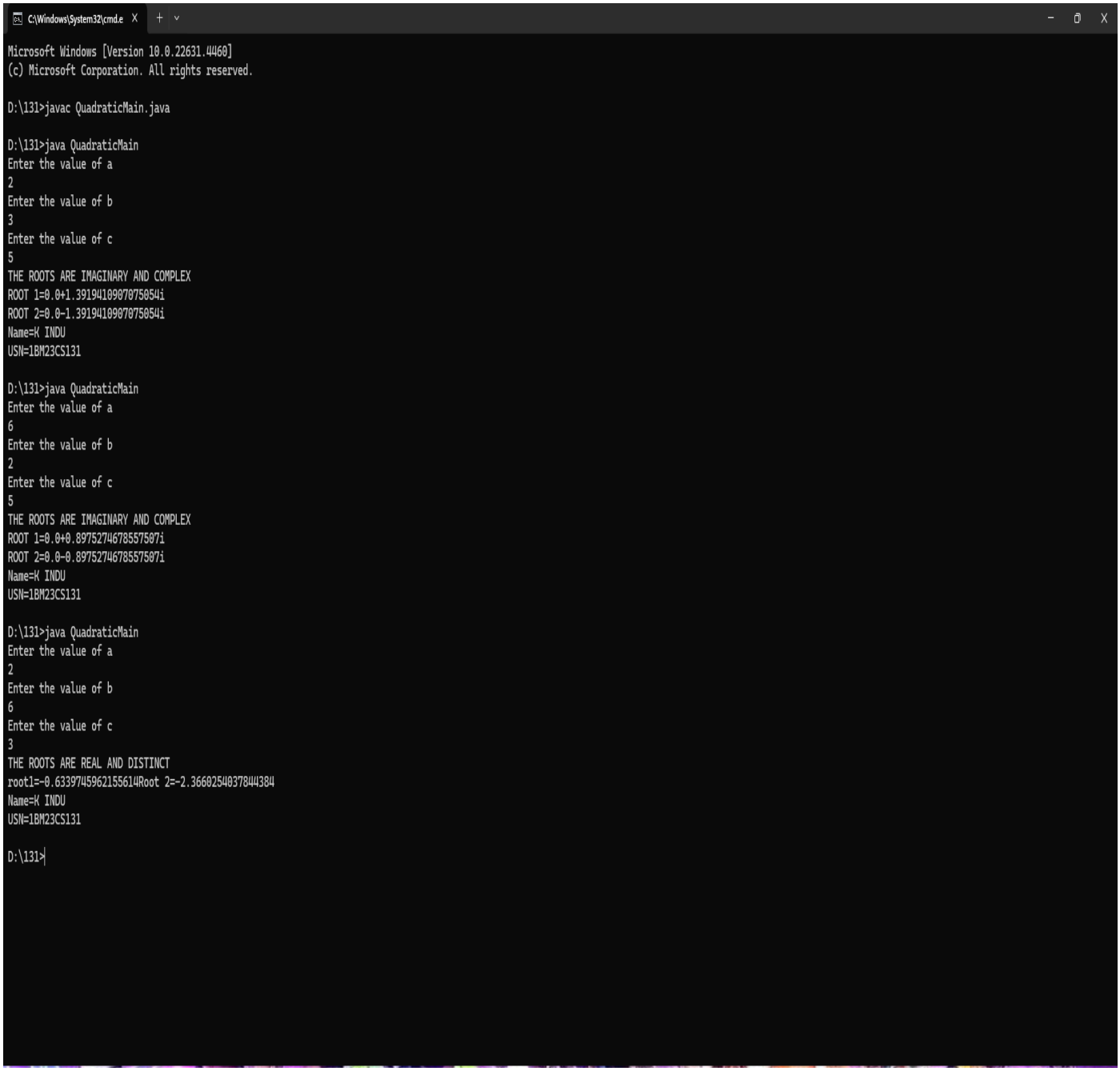
The root2 = -1

Date: 9-24

Code:

```
import java.util.Scanner;
class Quadratic{
int a,b,c;
double r1,r2,d;
void get(){
Scanner in=new Scanner(System.in);
System.out.println("Enter the value of a");
a=in.nextInt();
System.out.println("Enter the value of b");
b=in.nextInt();
System.out.println("Enter the value of c");
c=in.nextInt();}
void compute()
{
if(a==0)
{
System.out.println("The value of a shouldn't be zero");
}
else
{
d=(b*b)-(4*a*c);
if(d>0)
{
r1=(-b+Math.sqrt(d))/(2*a);
r2=(-b-Math.sqrt(d))/(2*a);
System.out.println("THE ROOTS ARE REAL AND DISTINCT");
System.out.println("root1="+r1+"Root 2="+r2);
}
else if(d==0)
{
r1=-b/(2*a);
System.out.println("THE ROOTS ARE EQUAL");
System.out.println("root1 = root2 ="+r1);
}
else
{
double real=-b/(2*a);
double imag=Math.sqrt(-d)/(2*a);
System.out.println("THE ROOTS ARE IMAGINARY AND COMPLEX");
System.out.println("ROOT 1="+real+"+"+imag+"i");
System.out.println("ROOT 2="+real+"-"+imag+"i");
}}
}}
class QuadraticMain{
public static void main(String args[])
{
```

```
Quadratic q=new Quadratic();
q.get();
q.compute();
System.out.println("Name=K INDU");
System.out.println("USN=1BM23CS131");
}}
```



Program 2
Calculation Of SGPA

Algorithm:

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.

```
import java.util.Scanner;
class Subject
{
int SubjectMarks;
int credits;
int grade;
public void calculategrade()
{
 if (SubjectMarks>=90)
  {
   grade=10;
  }
```

```
else
{
  grade=0;
}}}

class Student{
String name;
String usn;
double sgpa;
Subject sub[];
Scanner s;

public Student()
{
int i;
sub= new Subject[8];
for(i=0;i<8;i++)
{
 sub[i]= new Subject();
}
s= new Scanner(System.in);
```

```
System.out.println("Student Name:"+name);
System.out.println("USN:"+usn);
System.out.println("SGPA:"+sgpa);
}
}
class Main
{
public static void main(String args[]
{
Student student = new Student();
for(int i=0; i<3; i++)
{
student.getstudentdetails();
student.getmarks();
student.sgpa();
student.display();
}
}
}
```

Output:

Enter the Student Name: Sonu
Enter the Student usn: 156
Enter the marks for subject1: 95
Enter the credits for subject1: 4
Enter the marks for subject 2: 95
Enter the credits for subject 2: 4
Enter the marks for subject 3: 93
Enter the credits for subject 3: 3
Enter the marks for subject 4: 92
Enter the credits for subject 4: 3
Enter the marks for subject 5: 85
Enter the credits for subject 5: 3
Enter the marks for subject 6: 95
Enter the credits for subject 6: 1
Enter the marks for subject 7: 91
Enter the credits for subject 7: 1
Enter the marks for subject 8: 93
Enter the credits for subject 8: 1

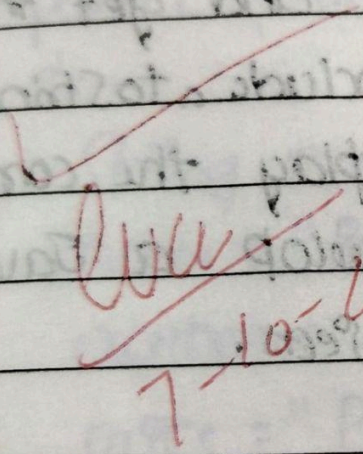
Page No.

Date: / /

Student Name: Sonu

USN: 156

SGPA: 9.85



Code:

```
import java.util.Scanner;

class Student {
    Scanner sc = new Scanner(System.in);
    String name, usn;
    double marks[] = new double[8];
    double credit[] = new double[8];

    void stdinfo() {
        System.out.print("Enter the name of the student:");
        name = sc.next();
        System.out.print("Enter the USN of the student:");
        usn = sc.next();

        System.out.print("Enter the marks of the student");
        for (int i = 0; i < 8; i++) {
   System.out.print("Subject "+(i+1)+":");
            marks[i] = sc.nextDouble();
        }
System.out.print("Enter the credits of the subjects respectively");
for (int i = 0; i < 8; i++) {
   System.out.print("Subject "+(i+1)+":");
   credit[i] = sc.nextDouble();
}
    }

    void printinfo() {
        System.out.print("Name of the student is " + name);
        System.out.print("USN: " + usn);
        System.out.print("The marks and credits of the subjects are:");
        for (int i = 0; i < 8; i++) {
            System.out.println("Subject" + (i + 1) + ":" + marks[i] + " Credit: " + credit[i]);
        }
    }

    double sgpa() {
        double sum = 0;
        double cred = 0;
        for (int i = 0; i < 8; i++) {
            sum += calculateGrade(marks[i]) * credit[i];
            cred += credit[i];
        }
        return cred == 0 ? 0 : sum / cred;
```

```
    }

    int calculateGrade(double mark) {
        if (mark >= 90) return 10;
        if (mark >= 80) return 9;
        if (mark >= 70) return 8;
        if (mark >= 60) return 7;
        if (mark >= 50) return 6;
        if (mark >= 40) return 5;
        return 0;
    }
}

class Display {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.stdinfo();
        s1.printinfo();
        double sgh = s1.sgpa();
        System.out.println("The SGPA of the student is: " + sgh);
System.out.println("Name: K. Indu");
System.out.println("USN:1BM23CS131");
    }
}
```

```
D:\131>javac Display.java

D:\131>java Display
Enter the name of the student:indu
Enter the USN of the student:131
Enter the marks of the studentSubject 1:45
Subject 2:55
Subject 3:54
Subject 4:78
Subject 5:67
Subject 6:66
Subject 7:88
Subject 8:87
Enter the credits of the subjects respectivelySubject 1:3
Subject 2:3
Subject 3:4
Subject 4:4
Subject 5:3
Subject 6:2
Subject 7:3
Subject 8:2
Name of the student is induUSN: 131The marks and credits of the subjects are:Subject1:45.0 Credit: 3.0
Subject2:55.0 Credit: 3.0
Subject3:54.0 Credit: 4.0
Subject4:78.0 Credit: 4.0
Subject5:67.0 Credit: 3.0
Subject6:66.0 Credit: 2.0
Subject7:88.0 Credit: 3.0
Subject8:87.0 Credit: 2.0
The SGPA of the student is: 7.041666666666667
Name: K. Indu
USN:1BM23CS131

D:\131>
```

Program 3
Creating ‘n’ Book Objects

Algorithm:

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

```
import java.util.Scanner;
class Books {
String name;
 String name;
int price;
int numPages;
 Books ( 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[])
{
Scanner s = new Scanner(System.in);
```

```
int n;
String name;
String author;
int price;
int numpages;
System.out.println("Enter no.of Book name:");
n=s.nextInt();
Books b[]=new Books[n];
for(int i=0; i<n; i++)
{
System.out.println("Enter the name of the books");
name=s.next();
System.out.println("Enter author name");
author=s.next();
System.out.println("Enter price of book");
price=s.nextInt();
System.out.println("Enter no.of pages");
numpages=s.nextInt();
b[i]=new Books(name
```

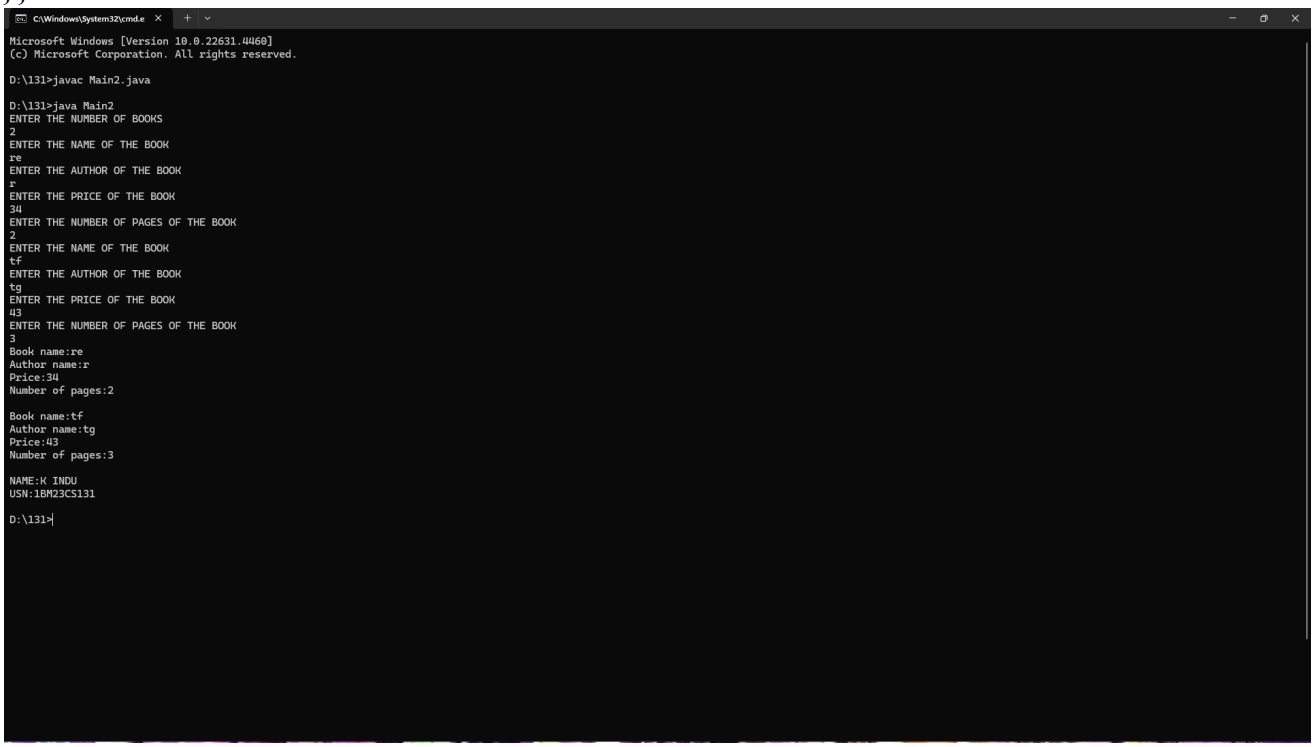
```
}
for(int i=0;i<n;i++)
{
System.out.println(b[i].tostring());
}
}
```

Output:

```
Enter no of Books: 1
Enter the name of the book: HOD
Enter author name: Michel
Enter price of book: 456
Enter no of pages: 500

Book name: HOD
author name: Michel
Price: 456
Number of pagy: 500
```

```
Code:import java.util.Scanner;
class Books{
String name;
String author;
int price;
int numPages;
Books(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[]){
Scanner s =new Scanner(System.in);
int n;
String name;
String author;
int price;
int numPages;
System.out.println("ENTER THE NUMBER OF BOOKS");
n=s.nextInt();
Books b[]=new Books[n];
for(int i=0;i<n;i++)
{
System.out.println("ENTER THE NAME OF THE BOOK");
name=s.next();
System.out.println("ENTER THE AUTHOR OF THE BOOK");
author=s.next();

System.out.println("ENTER THE PRICE OF THE BOOK");
price=s.nextInt();

System.out.println("ENTER THE NUMBER OF PAGES OF THE BOOK");
numPages=s.nextInt();
```

```
b[i]=new Books(name,author,price,numPages);
}
for(int i=0;i<n;i++)
{
System.out.println(b[i].toString());}
System.out.println("NAME:K INDU");
System.out.println("USN:1BM23CS131");
}}
```



Program 4

Area of shapes

Algorithm:

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, 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.

```
import java.util.Scanner();
abstract class Shape{
double a,b, result;
void printArea()
{
  Scanner xx=new Scanner(System.in);
  System.out.println("Enter two parameters");
  a = xx.nextDouble();
```

```
    b=xx.nextDouble();
    f abstract void cal();
}
class Rectangle extends shape{
  void cal()
  {
   result= l*b;
   System.out.println("Area of Rectangle Is
     +result);
  }
}

class Triangle extends Shape{
void cal()
{
 result = (l*b)/2;
 System.out.println ("Area of Triangle Is"
  +result);
 }
}
```

```
class Circle extends Shape {
void calc()
{
Scanner sn=new Scanner(System.in);
System.out.println("Enter radius:");
int r=sn.nextInt();
result= 3.142*r*r;
System.out.println("Area of circle is"+result);
}
}

class Main()
{
public static void main(String args[])
{
Rectangle r=new Rectangle();
r.printArea();
r.calc();
Triangle t=new Triangle();
t.printArea();
t.calc();
```

```
Circle c = new Circle();
c.calc();
}
}
```

Output:

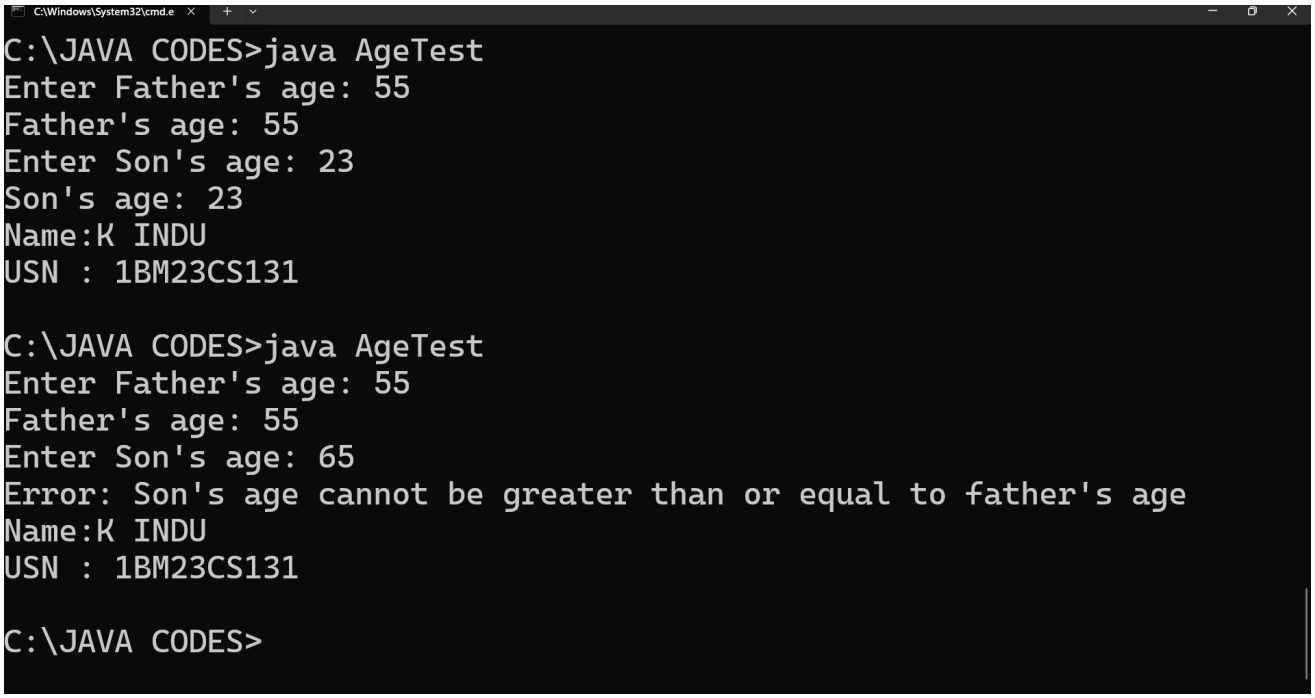
```
Enter two parameter : 3 3
Area of Rectangle is : 9
Enter two parameter : 3 3
Area of Triangle is : 4.5
Enter radius : 5
Area of circle is 78.55.
```

Ans to

Code:

```
import java.util.Scanner;
abstract class Shape{
double l,b,result;
void PrintArea() {
Scanner in=new Scanner(System.in);
System.out.println("enter 1st dimension(length for rectangle, base for triangle, radius for circle):");
l=in.nextDouble();
System.out.println("enter 2nd dimension(breadth for rectangle, height for triangle, zero for circle):");
b=in.nextDouble();
}
abstract void Calc();
}

class Rectangle extends Shape{
void Calc() {
result=l*b;
System.out.println(" area of rectangle is=" + result);
}
}

class Triangle extends Shape{
void Calc() {
result=0.5*l*b;
System.out.println(" area of triangle is=" + result);
}
```

```
}

class Circle extends Shape{

void Calc() {

result=3.14*l*l;

System.out.println(" area of circle is=" + result);

}

}

class Main_Shape{

public static void main(String args[]) {

Rectangle r=new Rectangle();

r.PrintArea();

r.Calc();

Triangle t=new Triangle();

t.PrintArea();

t.Calc();

Circle c=new Circle();

c.PrintArea();

c.Calc();
System.out.println("Name:K Indu");
System.out.println("USN:1BM23CS131");


}

}
```



Program 5

Bank

Algorithm:

⑤ Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings & current. Saving provides compound interest & withdrawal current account provide cheque book facility but no interest.

```
import java.util.Scanning
class Account{
    String customerName;
    int accountNumber;
    String accountType;
    double balance;
```

```
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");
 }
 void last() {
  System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable {
        A a=new A();
        B b= new B();
        Deadlock() {
        public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
        }
```

```
case 4:
    System.out.println(savingsAccount.calculateInterest());
    savingsAccount.displayBalance();
    break;

Case 5:
    System.exit(0);
    break;

default:
    System.out.println("Invalid choice");
}
else if (accType.equals("current"))
{
switch (choice)
case 1: System.out.println("Enter
deposit name");
    currentAccount.deposit(depositAmount);
```

case 5: System.exit(0);
break;
default:
System.out.println("Invalid");
}
}
}

Output: Enter Customer name: John
Enter acc number: 1
— Menu —
1. Deposit 2. withdraw
3. Compute Interest for SavingsAccount
4. Display Account details
5. Exit. 1
Enter the amount for deposit: 3000
3000 is the current balance.

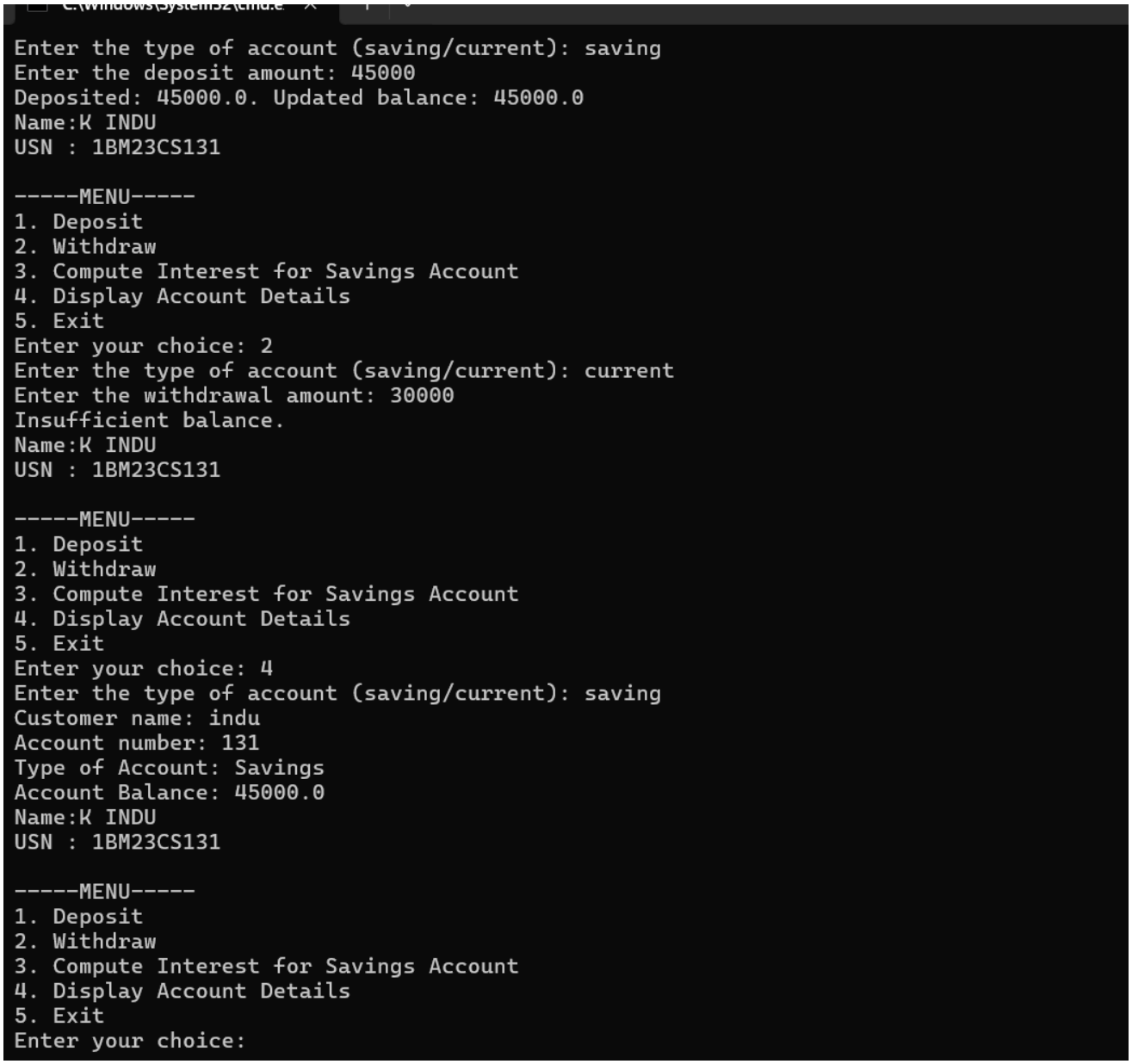
Code:

```
import java.util.Scanner;

class Account {
    String customerName;
    int accountNumber;
    String accountType;
    double balance;


    Account(String name, int accNumber, String accType) {
        customerName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = 0;
    }


    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ". Updated balance: " + balance);
    }

    public void displayBalance() {
        System.out.println("Account Balance: " + balance);
    }


    public void withdraw(double amount) {
        System.out.println("This operation is specific to account type.");
    }
}


class SavAccount extends Account {
    double interestRate = 0.04;  // 4% annual interest rate

    SavAccount(String name, int accNumber) {
        super(name, accNumber, "Savings");
    }


    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added: " + interest + ". Updated balance: " + balance);
    }
```

```
    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}
class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

    CurAccount(String name, int accNumber) {
        super(name, accNumber, "Current");
    }

    public void checkMinBalance() {
        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Balance below minimum. Service charge imposed: " + serviceCharge + ".
Updated balance: " + balance);
        }
    }


    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
            checkMinBalance();
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

public class Bank {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter customer name:");
        String name=sc.next();
        System.out.println("Enter account number:");
        int accountnumber=sc.nextInt();
```

```
        SavAccount savingsAccount = new SavAccount(name, accountnumber);
System.out.println("Enter customer name:");
        String name1=sc.next();
        System.out.println("Enter account number:");
        int accountnumber1=sc.nextInt();
        CurAccount currentAccount = new CurAccount(name1, accountnumber1);

        while (true) {
            System.out.println("\n-----MENU-----");
            System.out.println("1. Deposit\n2. Withdraw\n3. Compute Interest for Savings Account\n4.
Display Account Details\n5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();

            System.out.print("Enter the type of account (saving/current): ");
            String accType = sc.next();


            if (accType.equals("saving")) {
                switch (choice) {
                    case 1:
                        System.out.print("Enter the deposit amount: ");
                        double depositAmount = sc.nextDouble();
                        savingsAccount.deposit(depositAmount);
                        break;
                    case 2:
                        System.out.print("Enter the withdrawal amount: ");
                        double withdrawalAmount = sc.nextDouble();
                        savingsAccount.withdraw(withdrawalAmount);
                        break;
                    case 3:
                        savingsAccount.computeInterest();
                        break;
                    case 4:
                        System.out.println("Customer name: " + savingsAccount.customerName);
                        System.out.println("Account number: " + savingsAccount.accountNumber);
                        System.out.println("Type of Account: " + savingsAccount.accountType);
                        savingsAccount.displayBalance();
                        break;
                    case 5:
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Invalid choice.");
                }
            } else if (accType.equals("current")) {
                switch (choice) {
```

```
                case 1:
                    System.out.print("Enter the deposit amount: ");
                    double depositAmount = sc.nextDouble();
                    currentAccount.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter the withdrawal amount: ");
                    double withdrawalAmount = sc.nextDouble();
                    currentAccount.withdraw(withdrawalAmount);
                    break;
                case 3:
                    System.out.println("Current accounts do not earn interest.");
                    break;
                case 4:
                    System.out.println("Customer name: " + currentAccount.customerName);
                    System.out.println("Account number: " + currentAccount.accountNumber);
                    System.out.println("Type of Account: " + currentAccount.accountType);
                    currentAccount.displayBalance();
                    break;
                case 5:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } else {
            System.out.println("Invalid account type.");
        }
    }
  }
}
```



Program 6

Packages

Algorithm:

- 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 internals marks scored in five courses of the current sem 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 5 courses of the current sem of the student. Import the two packages in a file that declares the final marks of n students in all five(5) courses.

```
package CIE;
import java.util.Scanner;
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;
    }
```

```
package CIE;
public class Internals extends Student
{
    protected int[] Internal_marks = new int[5];
    public Internals (String usn, String name,
int sem, int[] Internal_marks) {
```

```
        super(usn, name, sem);
        this.internal_marks = internal_marks;
    }

    public int[] getInternal_marks()
        return internal_marks;
    }
}

package SEE;

import CIE.Student;

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

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

output

Enter no of students: 1

Enter details:

USN: 2

name: ABA

Sem: 3

Enter internal marks:

15

28

32

24

35

Enter SEE marks

87

89

90

100

91

final marks

Student name: ABA

usn: 2

sem: 3

final marks:

81

87

80

65

66.

Code:
```
package CIE;
import java.util.Scanner;
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;
}
}
package CIE;
public class Internals extends Student
{

 int internalmarks[]=new int[5];
public Internals( String usn,String name,int sem,int internalmarks[]){
super(usn,name,sem);
this.internalmarks=internalmarks;
}
public int[] getInternalmarks(){
return internalmarks;}
}
package SEE;
import CIE.Student;
public class External extends Student
{
 int[] externalmarks=new int[5];
public External ( String usn,
 String name,
 int sem,int externalmarks[])
{
super(usn,name,sem);
this.externalmarks=externalmarks;
}
public int[] getExternalMarks(){
return externalmarks;
}
}

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

```
public class Main1{
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);


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

         for (int i = 0; i < n; i++) {
      System.out.println("\nEnter 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();
      scanner.nextLine();
               int[] internalMarks = new int[5];
      System.out.println("Enter internal marks for 5 courses: ");
      for (int j = 0; j < 5; j++) {
         System.out.print("Course " + (j + 1) + ": ");
         internalMarks[j] = scanner.nextInt();
      }
      scanner.nextLine();

      int[] externalMarks = new int[5];
      System.out.println("Enter external marks for 5 courses: ");
      for (int j = 0; j < 5; j++) {
         System.out.print("Course " + (j + 1) + ": ");
         externalMarks[j] = scanner.nextInt();
      }
      scanner.nextLine();
      Internals internStudent = new Internals(usn, name, sem, internalMarks);
      External externalStudent = new External(usn, name, sem, externalMarks);

      System.out.println("\nFinal marks for " + internStudent.name + ":");
      for (int j = 0; j < 5; j++) {
         int finalMark = internStudent.getInternalmarks()[j] + externalMarks[j];
        System.out.println("Course " + (j + 1) + ": " + finalMark);
      }

            }
```

```
        scanner.close();
    }
}
```

```
D:\131>javac CIE/Student.java CIE/Internals.java SEE/External.java Main1.java

D:\131>java Main1
Enter the number of students: 1

Enter details for student 1
Enter USN: 2
Enter Name: l
Enter Semester: 1
Enter internal marks for 5 courses:
Course 1: 23
Course 2: 23
Course 3: 23
Course 4: 23
Course 5: 23
Enter external marks for 5 courses:
Course 1: 56
Course 2: 56
Course 3: 56
Course 4: 56
Course 5: 56

Final marks for l:
Course 1: 79
Course 2: 79
Course 3: 79
Course 4: 79
Course 5: 79

D:\131>
```

Program 7

Exceptions

Algorithm:

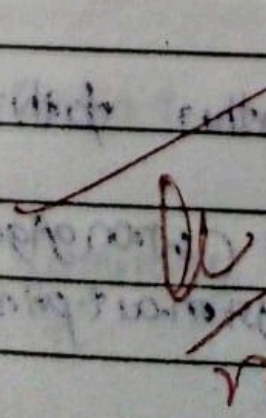
PROGRAM-4

```
class WrongAge extends Exception {
public wrongage (String message) {
super(message);
}}
class Father {
    int age;
    public Father(int age) throws WrongAge {
    if(age<0){
    throw new wrongAge("Father, age can't be
        negative");
    }
this.age = age;
System.out.println("Father's age:" +this.age);
}}
class son extends Father {
    int sonage;
    public son(int fatherage, int sonage)
    throws wrongage {
    super(fatherage);
if(sonage>=fatherage){
throw new wrongage("Son's age can't be
```

```
    greater than or equal to fathur's age:");
}
    this.sonage = sonage;
    System.out.println("Sons age:" this.sonage);
}
public class main {
    public static void main(String[] args)
    {
    try {
        son son1 = new son(30, 35);
    }
    catch (wrongAge e) {
        System.out.println("Exception:" +
            e.getmessage());
    }
try {
        Fadher fathur1 = new Fathur(-5);
}
    catch (wrongAge e) {
        System.out.println("Exception:" + e.getmessage());
}
try {
    son son2 = new son(40, 22);
}
```

```
catch (WrongAge e) {
    System.out.println("Exception!" + e.getMessage());
}
}
```

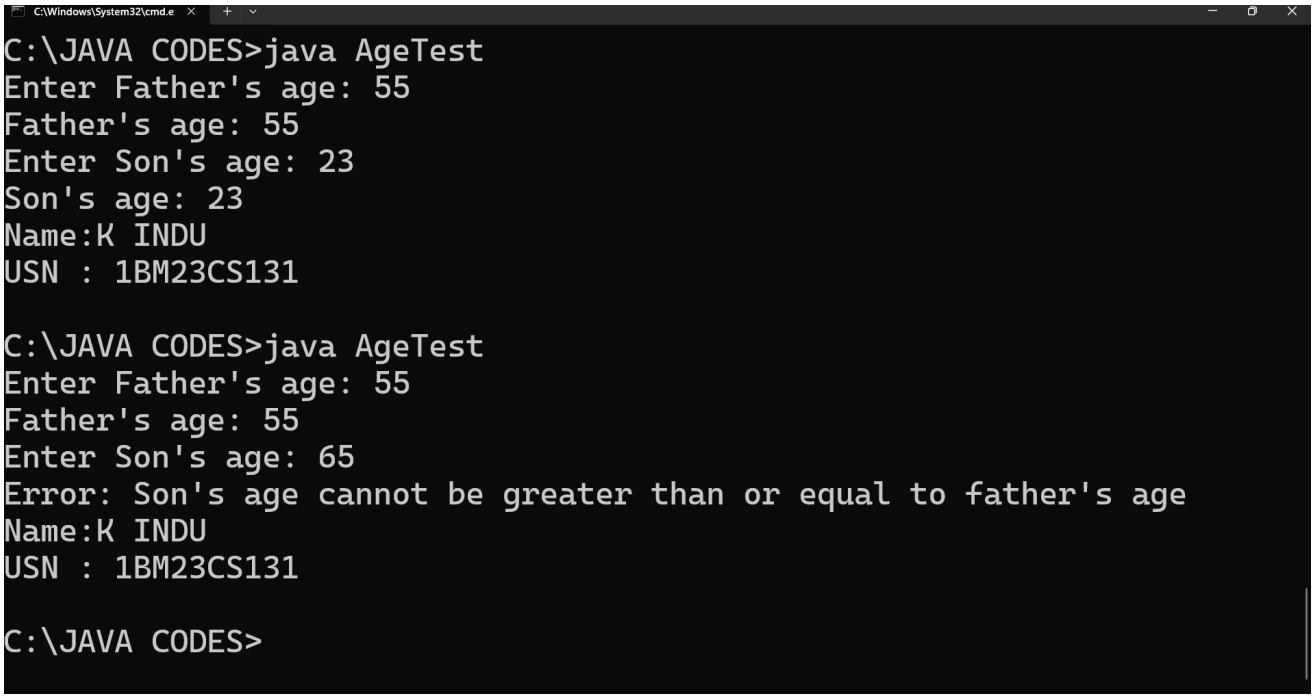
Father's age: 30
Exception: son's age cannot be greater than or equal to father's age
Exception: Father's age cannot be negative
Father's age: 40
son's age: 22



Code:import java.util.Scanner;

```
// Custom Exception Class
class WrongAge extends Exception {
    // Constructor for custom message
    public WrongAge(String message) {
        super(message);
    }
}

// InputScanner class to handle user input
class InputScanner {
    static Scanner s = new Scanner(System.in);
}

// Father class (base class)
class Father {
    int fatherAge;

    // Constructor for Father class
    public Father(int fatherAge) throws WrongAge {
        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
        this.fatherAge = fatherAge;
    }

    // Method to display father's age
    public void display() {
        System.out.println("Father's age: " + fatherAge);
    }
}

// Son class (derived class)
class Son extends Father {
    int sonAge;

    // Constructor for Son class
    public Son(int fatherAge) throws WrongAge {
        super(fatherAge);  // Pass the father's age to the Father constructor

        System.out.print("Enter Son's age: ");
        sonAge = InputScanner.s.nextInt();

        // Check if son's age is valid
        if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
```

```
        } else if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to father's age");
        }
    }

    // Method to display son's age
    public void display() {
        System.out.println("Son's age: " + sonAge);
    }
}

// Main class to test the program
public class AgeTest {
    public static void main(String[] args) {
        try {
            // Prompt for the father's age
            System.out.print("Enter Father's age: ");
            int fatherAge = InputScanner.s.nextInt();

            // Create Father object and display age
            Father father = new Father(fatherAge);
            father.display();

            // Create Son object and display age
            Son son = new Son(fatherAge);
            son.display();

        } catch (WrongAge e) {
            // Catch the WrongAge exception and print the error message
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

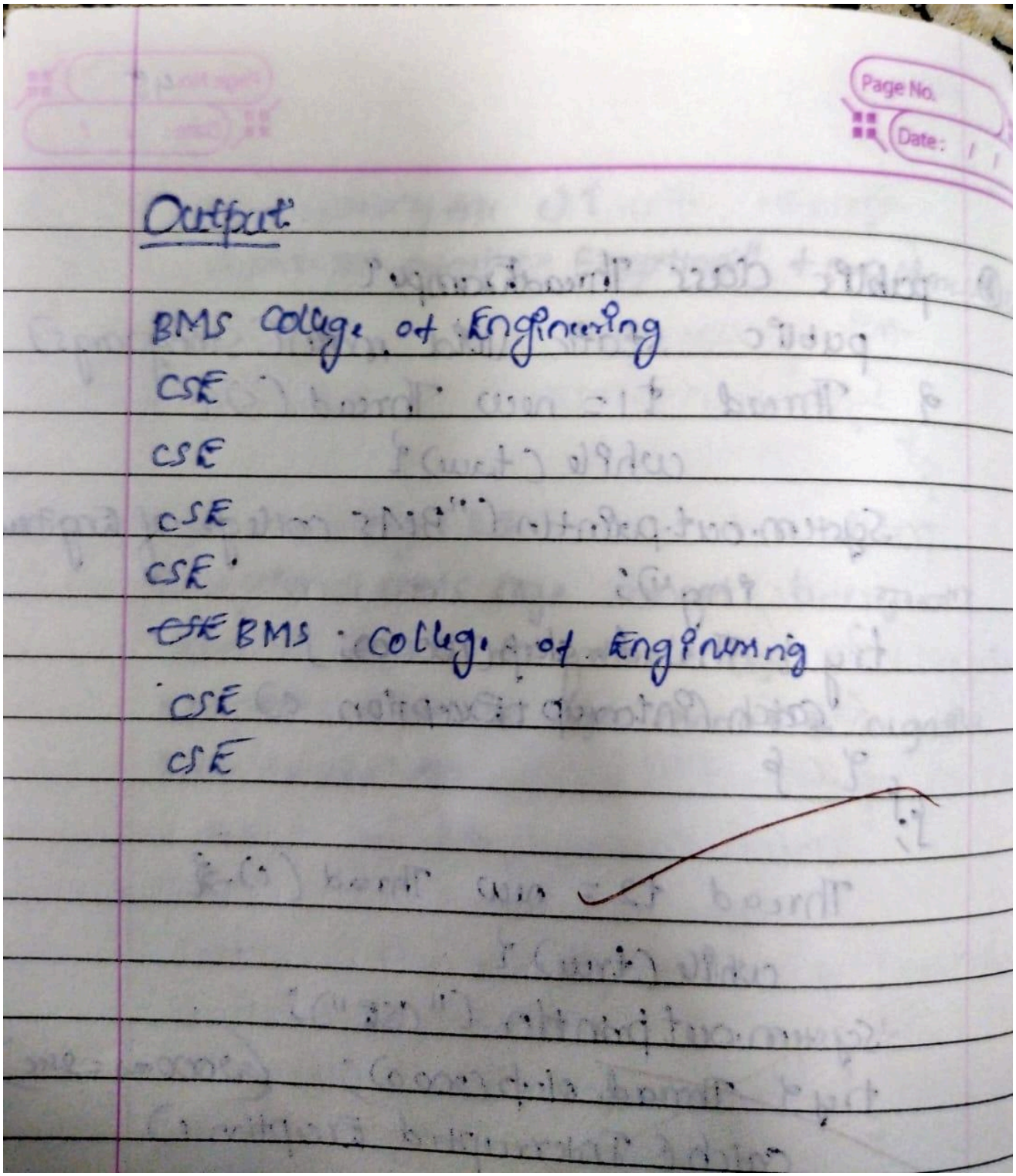


Program 8

Threads

Algorithm:

(8)
```
public class ThreadExample {
    public static void main(String args)
{ Thread t1 = new Thread (){
        while (true) {
System.out.println ("BMS college of Engineering");
try { Thread.sleep(10000); }
catch (InterruptException e)
{ }
};

    Thread t2 = new Thread (){
        while (true) {
System.out.println ("CSE");
try { Thread.sleep(2000); } (2000ms = 2sec)
    catch ( InterruptedException e)
    { }
});
t1.start();
t2.start();
}}
```



Code:public class ThreadExample {
    public static void main(String[] args) {
        Thread t1 = new Thread(() -> {
            while (true) {
                System.out.println("BMS College of Engineering");

```
            try { Thread.sleep(10000); } catch (InterruptedException e) {}
        }
    });

    Thread t2 = new Thread(() -> {
        while (true) {
            System.out.println("CSE");
            try { Thread.sleep(2000); } catch (InterruptedException e) {}
        }
    });

    t1.start();
    t2.start();
  }
}
```

```
C:\JAVA CODES>javac ThreadExample.java

C:\JAVA CODES>java ThreadExample
Name:K INDU
CSE
BMS College of Engineering
USN : 1BM23CS131
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
```

Program 9

Swing Demo

Algorithm:

④
```
import javax.swing;
import java.awt;
import java.awt.event;
class SwingDemo {
    SwingDemo() {
    JFrame jfrm = new JFrame ("Divider App");
    jFrm.setSize (275, 150);
    jfrm.setLayout (new FlowLayout());
    jfrm.setDefaultCloseOperation (JFrame.
        Exit_ON_CLOSE);
    JLabel jlab = new JLabel ("Enter the
        divider & dividend");
    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);
    JButton button = new JButton ("Calculate");
    JLabel err = new JLabel()
    JLabel alab = new JLabel();
    JLabel blab = new JLabel();
    JLabel blab = new JLabel();
    jfrm.add(err);
    jfrm.add (jlab);
```

```
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
ActionListener l = new ActionListener() {
public void actionPerformed(ActionEvent ae) {
System.out.println("Action event from a
text field");
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ae)
{
try {
int a = Integer.parseInt(ajtf.getText());
int b = Integer.parseInt(bjtf.getText());
int ans = a/b;
alab.setText("The A:" + a);
```

```
catch(ArithmeticException e) {
    alab.setText("");
    blab.setText("");
    anslab.setText("");
    err.setText("B should be NON zero!");
}
});
frm.setVisible(true);
}
public static void main(String args[]) {
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
new SwingDemo();
}
});
}
}
```

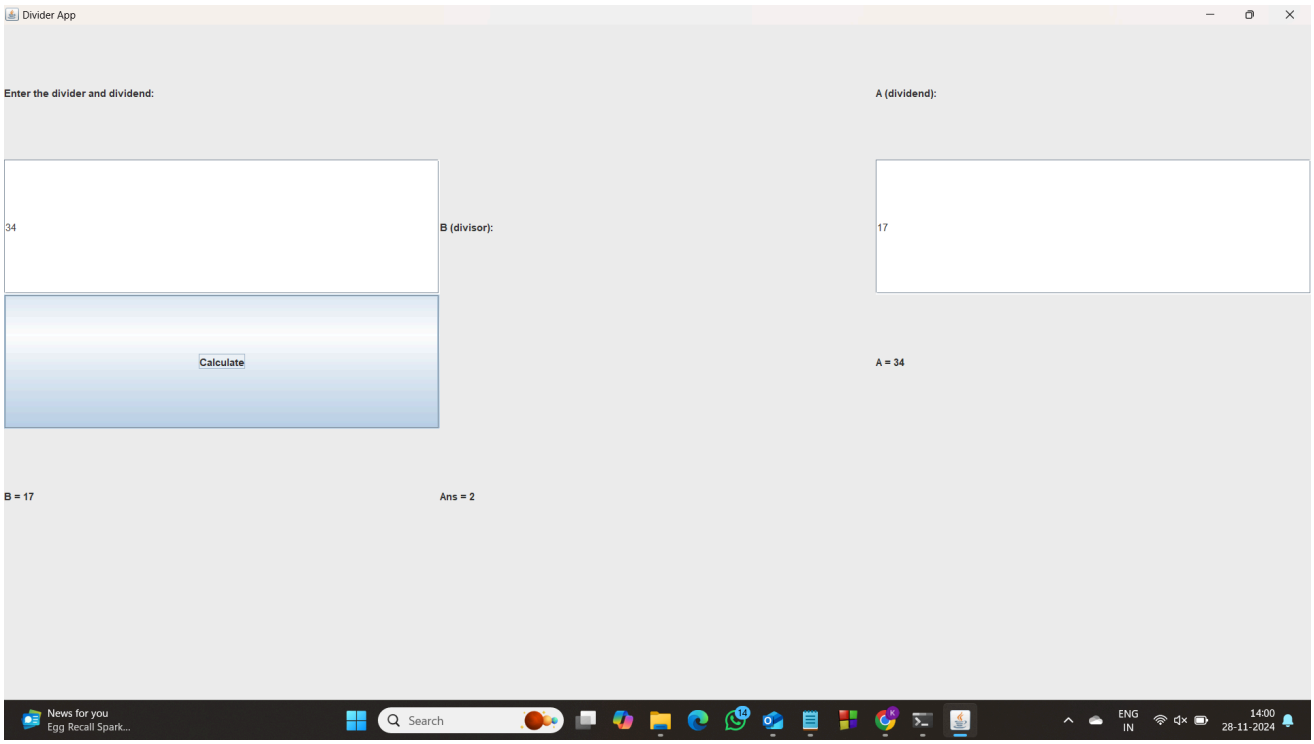
Output: Enter dividend and divisor

dividend	divisor
17	34

Calculate: 2

Code:
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
   SwingDemo() {
      // create JFrame container
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 150);
      jfrm.setLayout(new GridLayout(5, 2)); // Improved layout for clarity

      // to terminate on close
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // text label
      JLabel jlab = new JLabel("Enter the divider and dividend:");

      // add text field for both numbers
      JTextField ajtf = new JTextField(8);
      JTextField bjtf = new JTextField(8);

      // calc button
      JButton button = new JButton("Calculate");

      // labels to display inputs and result
      JLabel err = new JLabel();
      JLabel alab = new JLabel();
      JLabel blab = new JLabel();
      JLabel anslab = new JLabel();

      // add components in order
      jfrm.add(jlab);
      jfrm.add(new JLabel()); // empty space
      jfrm.add(new JLabel("A (dividend):"));
      jfrm.add(ajtf);
      jfrm.add(new JLabel("B (divisor):"));
      jfrm.add(bjtf);
      jfrm.add(button);
      jfrm.add(err);
      jfrm.add(alab);
      jfrm.add(blab);
      jfrm.add(anslab);

      // ActionListener for the calculate button
      button.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent evt) {
            try {
```

```
                // Parse integers from text fields
                int a = Integer.parseInt(ajtf.getText());
                int b = Integer.parseInt(bjtf.getText());

                // Check if division is valid
                if (b == 0) {
                    throw new ArithmeticException("B should be NON zero!");
                }

                // Perform division and update labels
                int ans = a / b;
                alab.setText("A = " + a);
                blab.setText("B = " + b);
                anslab.setText("Ans = " + ans);
                err.setText(""); // Clear any previous error message
            } catch (NumberFormatException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Enter Only Integers!");
            } catch (ArithmeticException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText(e.getMessage()); // Display arithmetic error message
            }
        }
    });

    // Display the frame
    jfrm.setVisible(true);
  }

  public static void main(String args[]) {
    // Create frame on event dispatching thread
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new SwingDemo();
      }
    });
  }
}
```



Program 10.a

InternalProcess Communication

Algorithm:

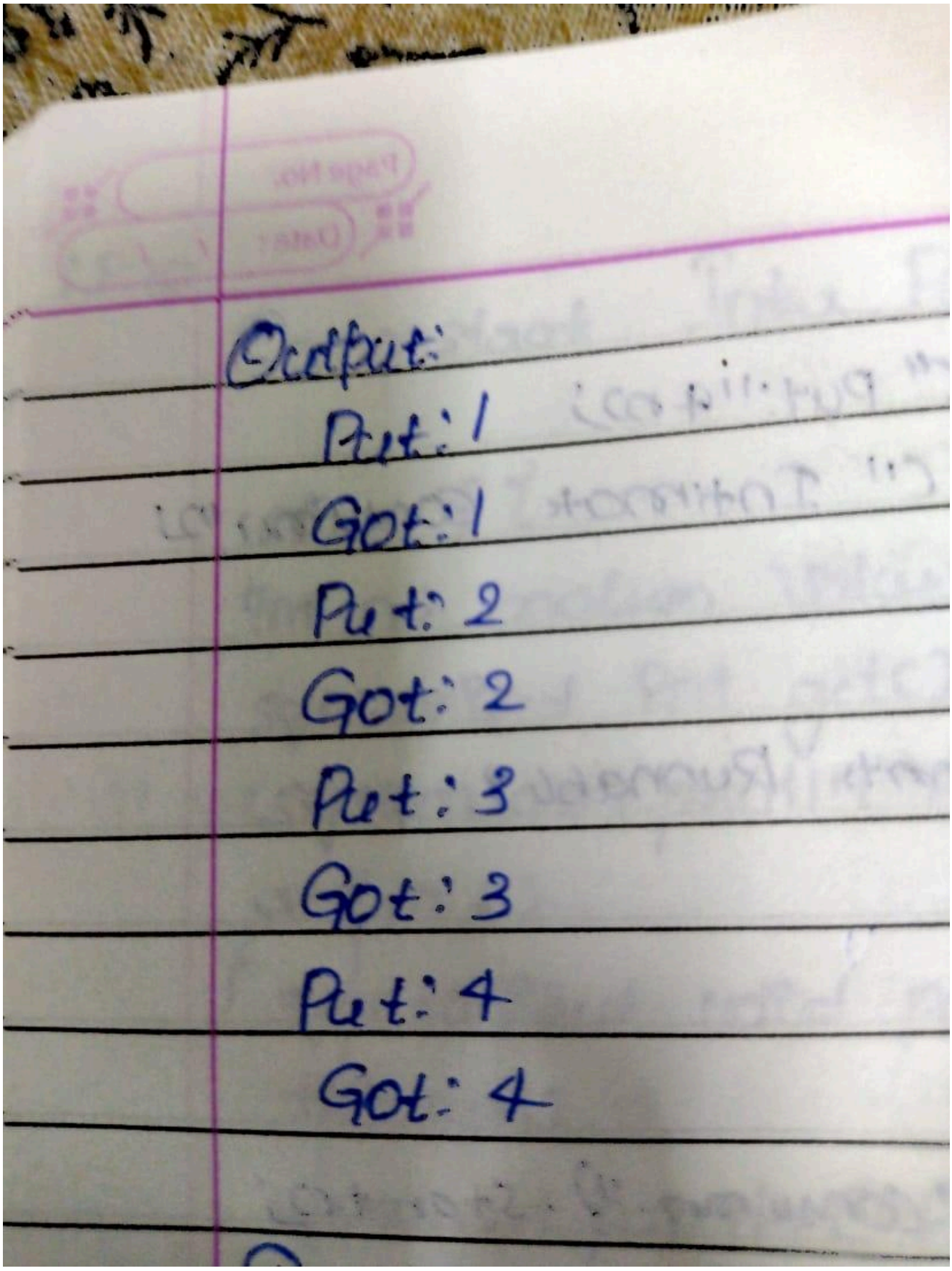
LP:10

Page No. 50
Date: / /

Demonstrate Inter Process Communication:

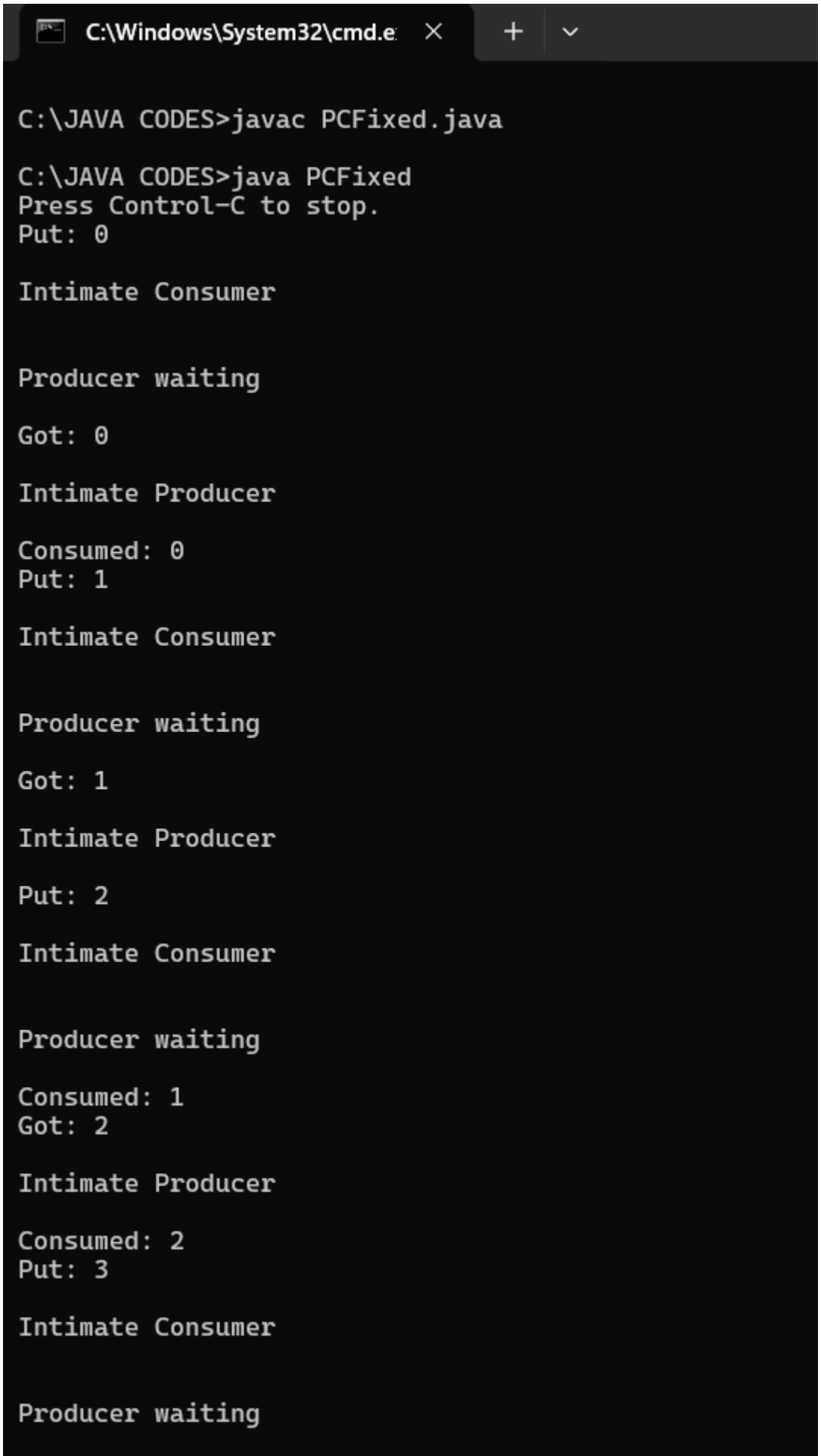
```
class Q{
int n; boolean valueSet=false;
synchronized int get(){ while(!valueSet) try{
System.out.println("Got: "+n); Consumer waiting;
action n;
}synchronized void put(int n){
this.n=n;
System.out.println("Put:"+n);
}}
wait();
}catch(InterruptedException e){
System.out.println("InterruptedException caught");
}
synchronized void put(int n){
while(valueSet)
try{
System.out.println("InterruptedException caught");
}
```

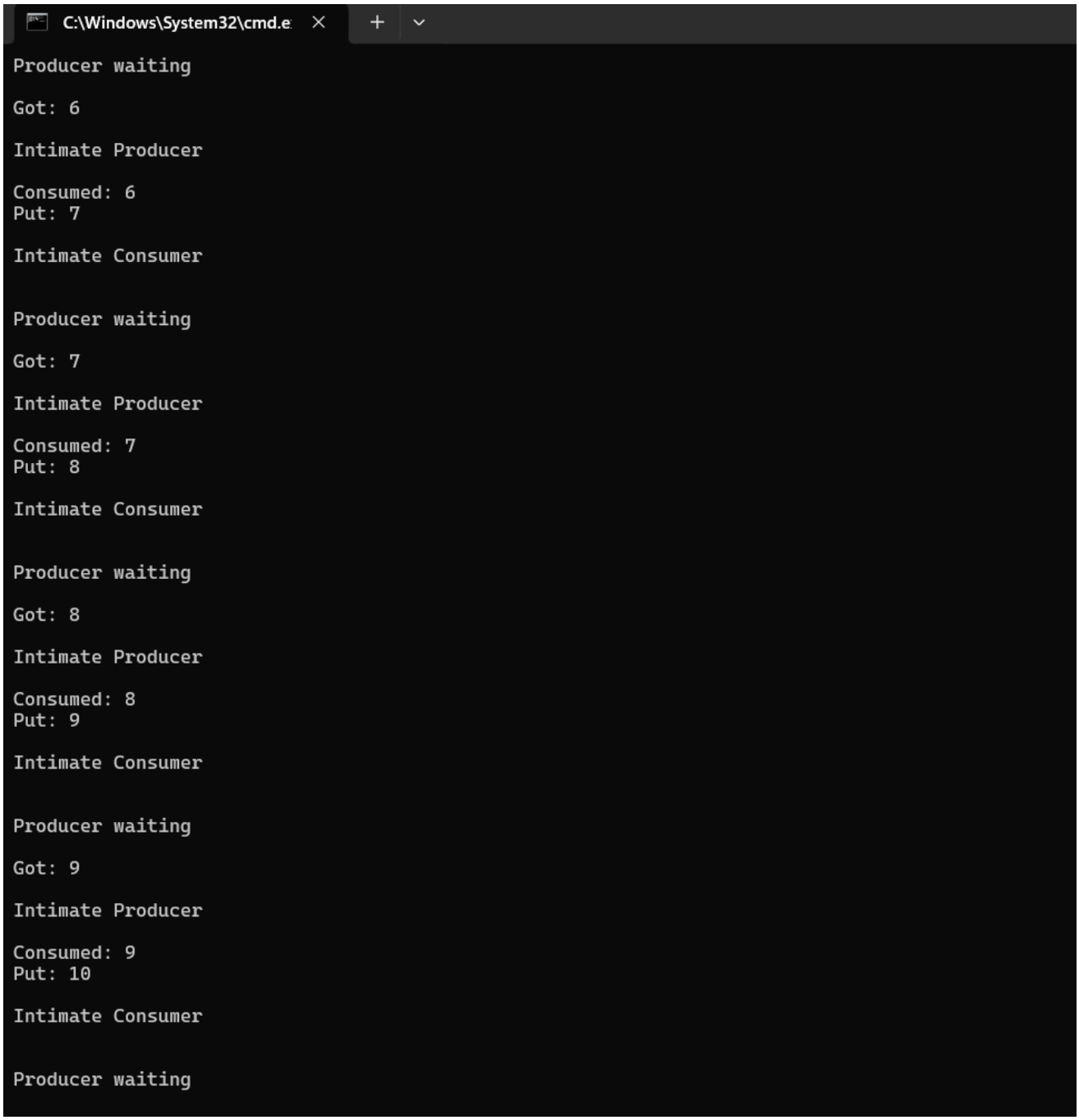
```
Value Set= True;
System.out.println("Put:"+n);
System.out.println("Intimate Consumer");
notifyAll();
}
class Consumer implements Runnable{
Q q;
Consumer(Q q){
this.q=q;
new Thread(this,"consumer").start();
}
Public void run(){
int i=0;
while(i<15){q.get();i++;}
int x=q.get();
System.out.println("Consumed:"+x);
i++;
}
}
```

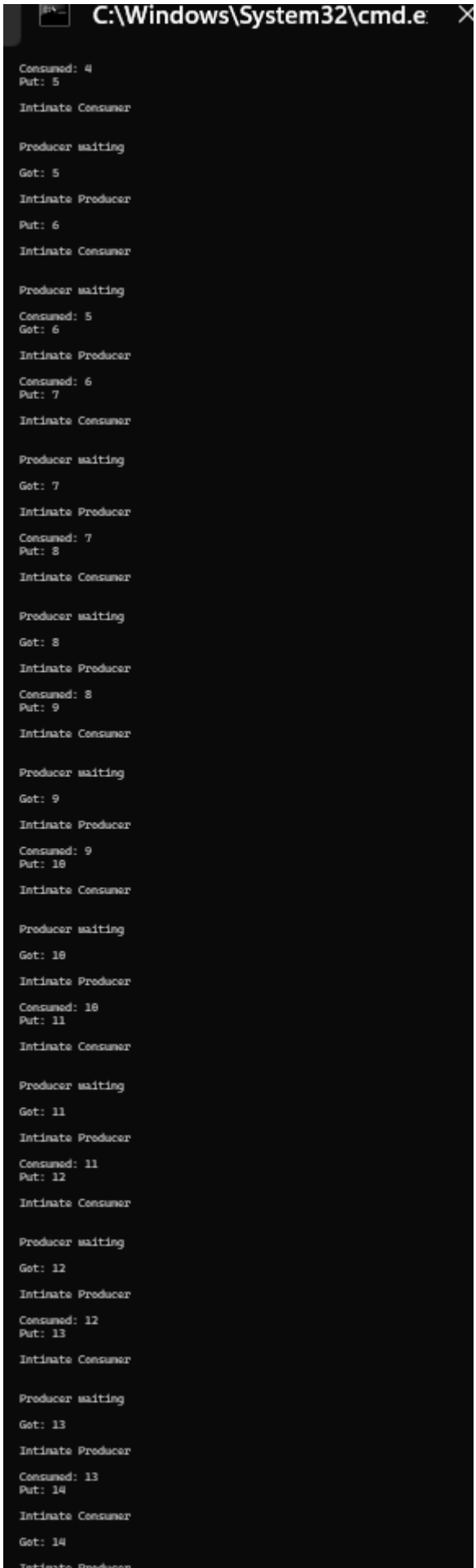

Output:
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4

Code:
```
// Lab_no_10
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++;
        }
    }
}

public 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.");
    }
}
```


C:\Windows\System32\cmd.e
C:\JAVA CODES>javac PCFixed.java
C:\JAVA CODES>java PCFixed
Press Control-C to stop.
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate Producer
Consumed: 0
Put: 1
Intimate Consumer
Producer waiting
Got: 1
Intimate Producer
Put: 2
Intimate Consumer
Producer waiting
Consumed: 1
Got: 2
Intimate Producer
Consumed: 2
Put: 3
Intimate Consumer
Producer waiting


C:\Windows\System32\cmd.e
Producer waiting
Got: 6
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
Intimate Consumer
Producer waiting

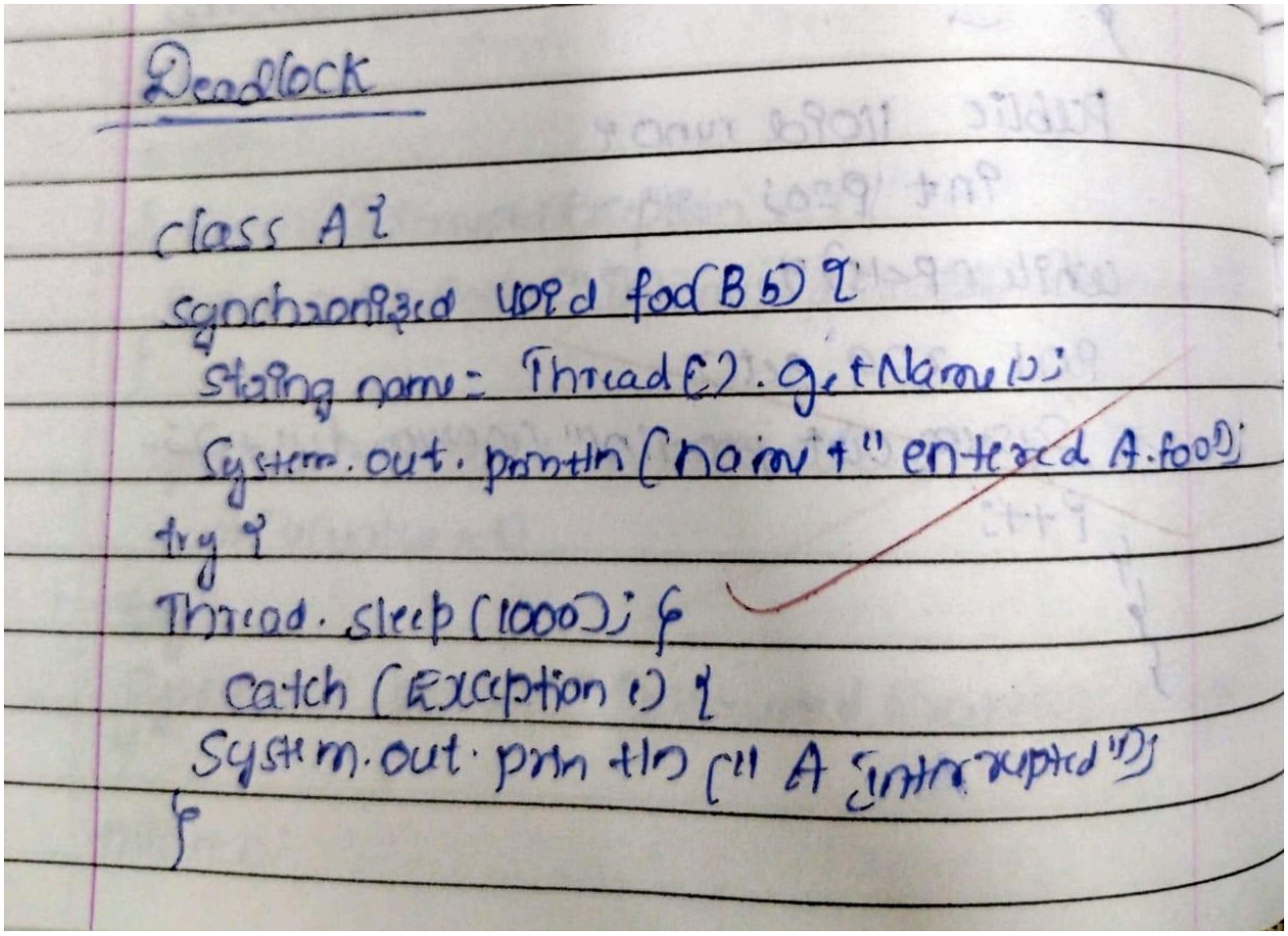

C:\Windows\System32\cmd.e
Consumed: 4
Put: 5
Intimate Consumer
Producer waiting
Got: 5
Intimate Producer
Put: 6
Intimate Consumer
Producer waiting
Consumed: 5
Got: 6
Intimate Producer
Consumed: 6
Put: 7
Intimate Consumer
Producer waiting
Got: 7
Intimate Producer
Consumed: 7
Put: 8
Intimate Consumer
Producer waiting
Got: 8
Intimate Producer
Consumed: 8
Put: 9
Intimate Consumer
Producer waiting
Got: 9
Intimate Producer
Consumed: 9
Put: 10
Intimate Consumer
Producer waiting
Got: 10
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
Got: 14

```
C:\JAVA CODES>java PCFixed
Press Control-C to stop.
Name:K INDU
USN : 1BM23CS131
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Consumed: 0
Put: 1

Intimate Consumer


Producer waiting

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
```

Program 10.b

Swing Demo

Algorithm:


Deadlock
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");
}

```
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");
 }
void last() {
  System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable {
      A a = new A();
      B b = new B();
      Deadlock() {
      public void run() {
      b.bar(a);
      System.out.println("Back in other thread");
      }
```

public static void main (String args[])
new Deadlock();
{
}

Output: MainThread entered A.foo
Racing Thread entered B.bar
MainThread Trying to Call B.last()
Inside A.last
Back in main Thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread

Code:
```
// lab_no_10_b
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();
   }

   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 B.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 args[]) {
      new Deadlock();
   }
}
```