VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SHAMARAO(1BM23CS308)

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-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**" carried out by **SHAMARAO(1BM23CS308),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

GITHUB LINK: https://github.com/1BM23CS308/1BM23CS308-OOJ-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;
class quadratic {
   float d;
   Scanner sc = new Scanner(System.in);

   void check()

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


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

```
            double r1 = Math.sqrt(-d)/(2*a);
            double r2= (-b)/(2*a);
            System.out.println(r2+"+i"+r1 + " "+r2+"-i"+r1 );
        }
    }
  }
}
public class qMain {
   public static void main(String[] args) {
      quadratic q1 = new quadratic();
       q1.check();
   }
}
```

OUTPUT:

```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

D:\JAVA LAB\LAB1>javac qMain.java

D:\JAVA LAB\LAB1>java qMain
enter the values of a,b, and c
2
4
2
0.0
the solutions are
roots are equal
-1.0

D:\JAVA LAB\LAB1>java qMain
enter the values of a,b, and c
2
5
2
9.0
the solutions are
roots are unique
-0.5 -2.0

D:\JAVA LAB\LAB1>java qMain
enter the values of a,b, and c
1
2
3
-8.0
the solutions are
roots are imaginary
-1.0+i1.4142135623730951 -1.0-i1.4142135623730951
```

Quadratic equation

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

    void check()

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

        if(a==0){
            system.out.println("invalid
    equation");
        }
        else {
            d = b*b - 4*a*c;
            system.out.println(d);
            system.out.println("the solutions
                are");

            if(d>0){
                system.out.println("roots are
    unique");
                double r1 = (-b + Math.sqrt(d))/
                double r2 = (-b - Math.sqrt(d))/
                                (2*a);
                system.out.println(r1+" "+r2
    }
```

```
if(d==0){
    system.out.println("roots are equal");
    double r = -b/(2*a);
    System.out.println(r);
}
if(d<0){
    system.out.println("roots are imaginary");
    double r1 = Math.sqrt(-d)/(2*a);
    double r2 = (-b)/(2*a);
    System.out.println("r1 = " + r1 + "i" + r1 + "r2 = " + r2 + "-i" + r1);
}
}
}
}
Public class Main{ Quadratic{
public static void main(String[] args){
    quadratic q1 = new quadratic();
    q1.check();
}
}
}
```

Output:

Enter the values of a, b, c
2
-6
-7
9 2 -0
roots are unique

Enter the values of a, b, c
10
5
-10
-375.0
roots are imaginary.

Enter the values of a, b, c
2
4
2
Roots are equeal

Enter the values of a, b, c
2
5
3
1.0
roots are Unique
-1.0 -1.5.

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.

CODE:

```
import java.util.Scanner;

public class cgpa {

    public static void main(String[] args) {

            Scanner scanner = new Scanner(System.in);

           System.out.print("Enter the number of subjects: ");

            int numSubjects = scanner.nextInt();


            double[] gradePoints = new double[numSubjects];

            int[] credits = new int[numSubjects];

            int totalCredits = 0;

            double total = 0;



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

                System.out.print("Enter the grade points for subject " + (i + 1) + ": ");

                gradePoints[i] = scanner.nextDouble();


                System.out.print("Enter the credits for subject " + (i + 1) + ": ");

                credits[i] = scanner.nextInt();
```

```
            total += gradePoints[i] * credits[i];
            totalCredits += credits[i];
        }
        double cgpa =  total / totalCredits;
        System.out.printf("Your CGPA is:" + cgpa);
    }
}
```

OUTPUT :

```
D:\JAVA LAB\LAB2>javac cgpa.java

D:\JAVA LAB\LAB2>java cgpa
Enter the number of subjects: 5
Enter the grade points for subject 1: 9
Enter the credits for subject 1: 4
Enter the grade points for subject 2: 9
Enter the credits for subject 2: 4
Enter the grade points for subject 3: 9
Enter the credits for subject 3: 4
Enter the grade points for subject 4: 9
Enter the credits for subject 4: 3
Enter the grade points for subject 5: 9
Enter the credits for subject 5: 3
Your CGPA is:9.0
```

Q> Calculate the Quadi cgpa of student.

```
import java.util.scanner;
public class cgpa{
    public static void main(string[] args){
        scanner scanner = new
        scanner(system.in);

        System.out.print("Enter the
number of subjects:");
        int numsubjects = scanner.
        nextInt();

        double[] grade points = new
double[num subjects];
        int[] credits = new int[num
    subjects];
        int[] credits = new int[numsubjects];
        int totalcredits = 0;
        double total = 0;

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

            System.out.print("Enter the grade
points for subject" + (i+1) + ":");
            gradepoints[i] = scanner.nextDouble()

            System.out.print("Enter the credits
for subject" + (i+1) + ":");
            credits[i] = scanner.nextInt();
            total += gradePoints[i] * credits[i];
```

```
totalcredits += credits[i];
}
double cgpa = total/totalcredits;
System.out.printf("Your CGPA is:"
                  +cgpa);
}
```

output:

Enter the number of subjects: 3
Enter the grade points for subject: 1 9
Enter the credits for subject 1: 4
Enter the grade points for subject 2: 9
Enter the credits for subject 2: 9
Enter the grade points for subject 3: 9
Enter the credit for subject 3: 4

Your CGPA is: 9.0

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.

CODE :

```
import java.util.Scanner;
class Book{
 private String name;
 private String author;
 private double price;
 private int num_pages;

 public Book(String name,String author,double price,int num_pages){
  this.name=name;
  this.author=author;
  this.price=price;
  this.num_pages=num_pages;}
 public String getName(){
  return name;}
 public void setName(String name){
  this.name=name;}
 public String getauthor(){
  return  author;}
 public void setauthor(String author){
  this.author=author;}
```

```
 public double getprice(){
  return price;}
 public void setprice(){
  this.price=price;}
 public int getnumpages(){
  return num_pages;}
 public void setnumpages(){
  this.num_pages=num_pages;}

 public String toString(){
  return("Book[name:"+name+" author:"+author+" price:"+price+"
pages:"+num_pages);}
}
public class Bookmain{
 public static void main(String[] args){
  Scanner sc=new Scanner(System.in);
  System.out.println("enter no of books");
  int n=sc.nextInt();
  sc.nextLine();
  Book[] books=new Book[n];
  for(int i=0;i<n;i++){
   System.out.println("enter the details of book"+(i+1)+":");
   System.out.print("enter name:");
   String name=sc.nextLine();
   sc.nextLine();
```

```
   System.out.print("enter author:");
   String author=sc.nextLine();
   System.out.print("enter price:");
   int price=sc.nextInt();
   System.out.print("enter the no of pages:");
   int num_pages=sc.nextInt();
   books[i]=new Book(name,author,price,num_pages);}
  System.out.print("\ndetails of books");
  for(int i=0;i<n;i++){
   System.out.println(books[i].toString());
  }
}
}
```

OUTPUT :

```
D:\JAVA LAB\LAB3>javac Bookmain.java

D:\JAVA LAB\LAB3>java Bookmain
enter no of books
2
enter the details of book1:
enter name:aa

enter author:bb
enter price:200
enter the no of pages:150
enter the details of book2:
enter name:cc
enter author:dd
enter price:400
enter the no of pages:300

details of booksBook[name:aa author:bb price:200.0 pages:150
Book[name: author:dd price:400.0 pages:300
```

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 & 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 an book objects.

→

```
import java.util.scanner;
class Book{
    private string name;
    private string author;
    private double price;
    private int numpages;

public Book (String name, string
    author, double price, int numpages)
{
    this.name = name;
    this.author = author;
    this.price = price;
    this.numpages = numpages;
}
public void setname (string name){
    this.name = name;

public void setauthor (string author){
    this.author = author
```

```
public void setPrice(double price){
        this.price = price;
}
public void setNumPages(int numPages){
        this.numPages = numPages;
}
public String getName(){
        return name;
}
public String getAuthor(){
        return author;
}
public double getPrice(){
        return price;
}
public int getNumPages(){
        return numPages;
}
@Override
public String toString(){
        return "Book Details: \n" +
                "Name: " + Name + "\n" +
                "author: " + author + "\n" +
                "price: $" + price + "\n" +
                "Num of pages: " + numPages
}
public class Main {
        public static void main(String[]
                                args)
        {
        Scanner sc = new Scanner(System.
```

```
System.out.println("Enter number of
                    Books: ");
int n = sc.nextInt();
    sc.nextLine();
Book[] books = new Book[n];

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

    String name = scanner.nextLine();
      sc.nextLine();
  System.out.println("Enter author
                    name:");
    String author = sc.nextLine();

  System.out.println("Enter price:");
    double price = sc.nextDouble();

  System.out.println("Enter numpages:");
    int numpages = scanner.nextInt();
      scanner.nextLine();
  book[i] = new Book[name, author,
                    price, numpages);
  System.out.println(books[i].getDetails());
}
  scanner.close();
}
```

Enter the number of books : 2
Enter the details of book 1 :
Enter Book name : The Night Circus
Enter author : Erin Morgenstern
Enter price : 400
Enter the number of pages : 200

Enter the details of book 2 :
Enter Book name : Atomic Habits
Enter author : James Clear
Enter price : 600
Enter the number of pages

Details of Books :

Book[name : The Night Circus ; author : Erin Morgenstern, price : 400.0, pages : 200]
Book[name : Atomic Habits ; author : James Clear, price : 600.0 pages : 300]

PROGRAM-4

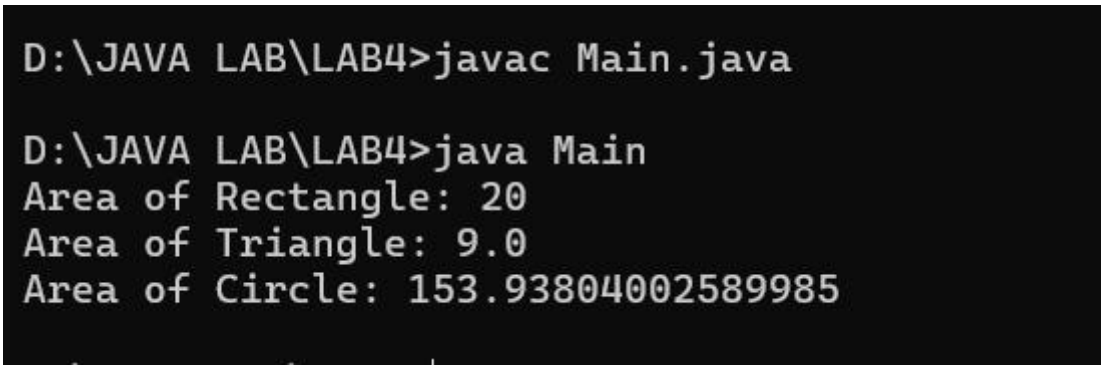
Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea( ). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea(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

CODE :

```
import java.util.Scanner;
abstract class Shape {
    int D1;
    int d2;


    abstract void printArea();
}


class Rectangle extends Shape {

    Rectangle(int length, int width) {
        this.d1 = length;
        this.d2 = width;
    }


    void printArea() {
        int area = d1 * d2;
        System.out.println("Area of Rectangle: " + area);
    }
}


class Triangle extends Shape {

    Triangle(int base, int height) {
        this.d1 = base;
        this.d2 = height;
    }
```

```
    void printArea() {
        double area = 0.5 * d1 * d2;
        System.out.println("Area of Triangle: " + area);
    }
}


class Circle extends Shape {

    Circle(int radius) {
        this.d1 = radius;
    }

    void printArea() {
        double area = Math.PI * d1 * d1;
        System.out.println("Area of Circle: " + area);
    }
}


public class Main {
    public static void main(String[] args) {

        Rectangle rectangle = new Rectangle(5, 4);
        Triangle triangle = new Triangle(3, 6);
        Circle circle = new Circle(7);


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

OUTPUT:


D:\JAVA LAB\LAB4>javac Main.java
D:\JAVA LAB\LAB4>java Main
Area of Rectangle: 20
Area of Triangle: 9.0
Area of Circle: 153.93804002589985

9] Develop a java program to create abstract class named shape that contains two integers & an empty method named print area(). provide three classes named rectangle, Triangle, circle such that each one of it extends class shape.

```
import java.util.scanner;

abstract class shape{
    int s1, s2;
    abstract void printarea();

class Rectangle extends shape{
    Rectangle (int s1, int s2){
    this.s1 = s1;
    this.s2 = s2;}
    void printarea(){
        system.out.println("area of
            rectangle is" + (s1*s2));}
}

class Triangle extends shape{
    Triangle (int s1, int s2)
    this.s1 = s1
    this.s2 = s2;}
    void printarea(){
        System.out.println(area of
    triangle is" + (0.5*s1*s2));}}
```

```
}
class Circle extends shape{
    Circle (int s1, int s2) {
     this.s1 = s1;
     this.s1 = s2;
     void printarea() {
      system.out.println("area of
        circle is " + ( 3.14*s1* s1))}
    }

class shapemain{

  public static void main(String [] args)
  {
      Rectangle rectangle = new Rectangle(5,4
      rectangle.printarea();
      Triangle triangle = new Triangle(5,4
      triangle.printarea();
      Circle Circle = new circle(5.f)
       circle.printarea();}}
```

area of rectangle is 20
area of triangle is 0
area of Circle is 5

14/11/24

PROGRAM-5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

a) Accept deposit from customer and update the balance.

b) Display the balance.

c) Compute and deposit interest

d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.

CODE:

```
Class Account {

   String customerName;

   String accountNumber;

   String accountType;

   double balance;



   public Account(String customerName, String accountNumber, String accountType, double initialBalance) {

      this.customerName = customerName;

      this.accountNumber = accountNumber;

      this.accountType = accountType;

      this.balance = initialBalance;
```

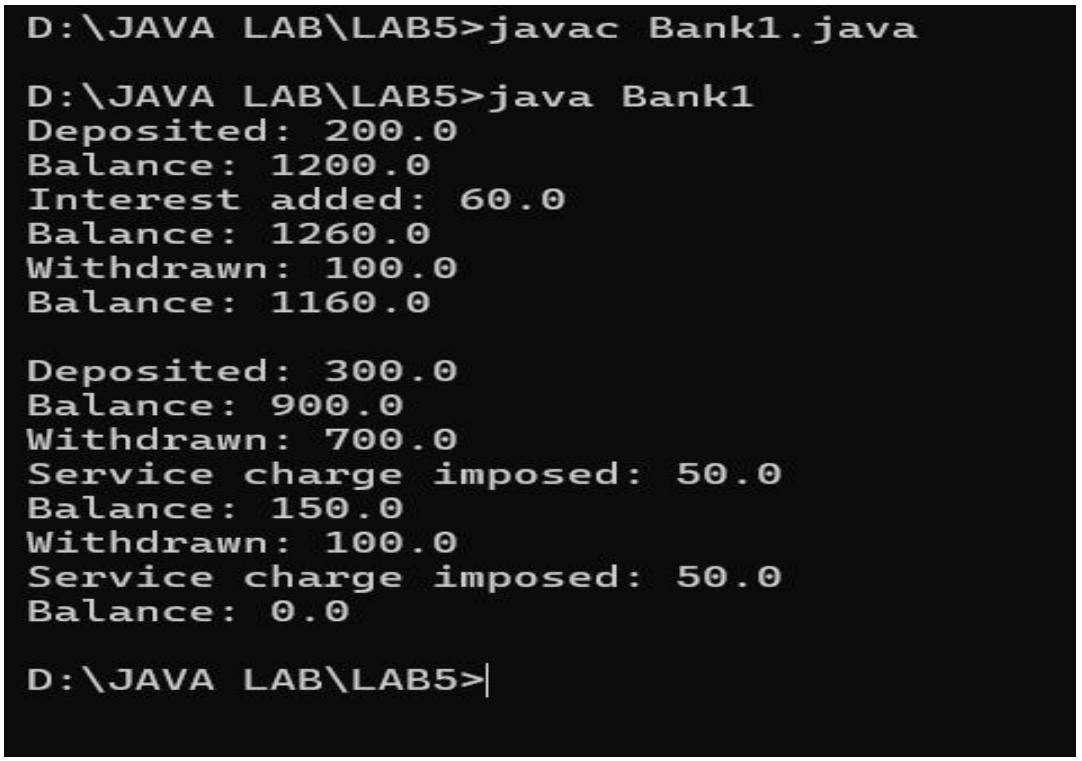
```
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

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

class SavAcct extends Account {
    private double interestRate;

    public SavAcct(String customerName, String accountNumber, double initialBalance,
double interestRate) {
```

```
        super(customerName, accountNumber, "Savings", initialBalance);
        this.interestRate = interestRate;
    }

    public void computeAndDepositInterest() {
        double interest = balance * (interestRate / 100);
        balance += interest;
        System.out.println("Interest added: " + interest);
    }

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

class CurAcct extends Account {
    private static final double MIN_BALANCE = 500.0;
```

```
    private static final double SERVICE_CHARGE = 50.0;



    public CurAcct(String customerName, String accountNumber, double initialBalance)
{
        super(customerName, accountNumber, "Current", initialBalance);
    }



    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
            if (balance < MIN_BALANCE) {
                balance -= SERVICE_CHARGE;
                System.out.println("Service charge imposed: " + SERVICE_CHARGE);
            }
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}



public class Bank1 {
```

```
    public static void main(String[] args) {

        SavAcct savings = new SavAcct("Alice", "SAV123", 1000.0, 5.0);
        savings.deposit(200);
        savings.displayBalance();
        savings.computeAndDepositInterest();
        savings.displayBalance();
        savings.withdraw(100);
        savings.displayBalance();

        System.out.println();


        CurAcct current = new CurAcct("Bob", "CUR456", 600.0);
        current.deposit(300);
        current.displayBalance();
        current.withdraw(700);
        current.displayBalance();
        current.withdraw(100);
        current.displayBalance();
    }
}
```

OUTPUT:


D:\JAVA LAB\LAB5>javac Bank1.java
D:\JAVA LAB\LAB5>java Bank1
Deposited: 200.0
Balance: 1200.0
Interest added: 60.0
Balance: 1260.0
Withdrawn: 100.0
Balance: 1160.0
Deposited: 300.0
Balance: 900.0
Withdrawn: 700.0
Service charge imposed: 50.0
Balance: 150.0
Withdrawn: 100.0
Service charge imposed: 50.0
Balance: 0.0
D:\JAVA LAB\LAB5>

Develop a java program to create a

5) class Bank that maintains two kinds of account for its customers, called savings account & the other current account. The savings account provides compound interest & 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 & if the balance falls below this level.

```
class Account {
    string customerName;
    string acc_no;
    string acc_Type;
    double balance;

    public Account (string CustomerName,
    string acc_no, string acc_Type,
    double balance) {
        this.customername = customer name;
        this.acc_no = acc_no;
        this.acc_Type = acc_Type;
        this.balance = initial balance;
    }
    public void deposit (double amount) {
        if (amount > 0) {
            balance += amount;
            S.o.p ("Deposited:" + amount);
        }
        else {
            S.o.p ("Invalid deposit
                    amount.");
        }
    }
```

```
public void displayBalance(){
    s.o.p("Balance:" + balance);
  }
}

class savAcct extends Account{
  private double interestRate;

  public savAcct(String customerName,
    String acc_no, double initialBalance,
      double interestRate){

  super(customerName, acc_no, "SAVINGS",
        initialBalance),
    this.interestRate = interestRate;
}

public void computeAndDepositInterest(){

  double interest = balance * (interestRate/100);
  balance += interest;
  s.o.p("interest added:" + interest);
}

Public void withdraw(double amount){
  if(amount <= balance){
    balance -= amount;
    s.o.p("withdrawn:" + amount);
  }else{
    s.o.p("Insufficient balance");
  }
}
```

```
class CurAcct extends Account {
    private static final double
    MIN_BALANCE = 500.0;
    private static final double
    SERVICE_CHARGE = 50.0;

    public CurAcct (String customername,
    String acc_no, double initialbalance) {

        super (customername, acc_no,
        "Current", initialbalance);
    }

    public void withdraw (double amount)
    { if (amount <= balance) {
        balance = balance - amount;
        S.o.p ("Withdrawn:" + amount);

        if (balance < MIN_BALANCE) {
        balance = balance - SERVICE_CHARGE;
        S.o.p ("Service charge imposed:"
                + SERVICE_CHARGE);
        } else
            S.o.p ("Insufficient balance");
        }
    }
}

public class Bank {
public static void main (String[]
                args) {
```

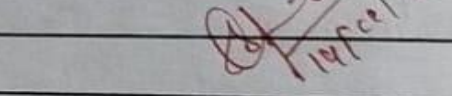
```
SavAcct savings = new SavAcct(
        "Alice", "SAV123", 1000.0, 5.0);
savings.displayBalance();

savings.computeAndDepositInterest();
savings.displayBalance();
savings.withdraw(100);
savings.displayBalance();

S.O.P();

CurAcct current = new CurAcct("Bob",
    "CUR456", 600.0);
current.deposit(300);
current.displayBalance();
current.withdraw(700);
current.displayBalance();
current.withdraw(100);
current.displayBalance();
}
}
```

output

Deposited : 200.0
Balance : 1200.0
Intrest added : 60.0
Balance : ~~1260.0~~
withdrawn : 300.0

Deposited : 300.0
Balance : 900.0
withdrawn : 700.0
Service charge imposed : 50.0
Balance : 150.0
withdrawn : 100.0
Service charge imposed : 50.0
Balance : 150.0

Read i/p from user



PROGRAM-6

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

CODE:

```
package CIE;


public class Student1 {
    protected String usn;
    protected String name;
    protected int sem;
       public Student1 (String usn, String name, int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }

       public String getUsn() {
        return usn;
    }

    public String getName() {
        return name;
```

```
    }

    public int getSem() {
        return sem;
    }
}
package CIE;

public class Internals extends Student1 {
    private 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;
    }

      public int getTotalInternalMarks() {
        int total = 0;
        for (int mark : internalMarks) {
            total += mark;
        }
```

```
        return total;
    }
}
package SEE;


import CIE.Internals;


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



    public int[] getExternalMarks() {
        return externalMarks;
    }



    public int getTotalExternalMarks() {
        int total = 0;
        for (int mark : externalMarks) {
            total += mark;
```

```
        }
        return total;
    }


    public int getFinalMarks() {
        return getTotalInternalMarks() + getTotalExternalMarks();
    }
}
import CIE.Internals;
import SEE.External;
import java.util.Scanner;

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


        System.out.print("Enter USN for student 1: ");
        String usn1 = scanner.nextLine();
        System.out.print("Enter name for student 1: ");
        String name1 = scanner.nextLine();
        System.out.println("Enter 5 internal marks for student 1:");
        int[] internalMarks1 = new int[5];
        for (int i = 0; i < 5; i++) {
            internalMarks1[i] = scanner.nextInt();
```

```
}
System.out.println("Enter 5 external marks for student 1:");
int[] externalMarks1 = new int[5];
for (int i = 0; i < 5; i++) {
    externalMarks1[i] = scanner.nextInt();
}



scanner.nextLine();
System.out.print("Enter USN for student 2: ");
String usn2 = scanner.nextLine();
System.out.print("Enter name for student 2: ");
String name2 = scanner.nextLine();
System.out.println("Enter 5 internal marks for student 2:");
int[] internalMarks2 = new int[5];
for (int i = 0; i < 5; i++) {
    internalMarks2[i] = scanner.nextInt();
}
System.out.println("Enter 5 external marks for student 2:");
int[] externalMarks2 = new int[5];
for (int i = 0; i < 5; i++) {
    externalMarks2[i] = scanner.nextInt();
}
```

```
        External student1 = new External(usn1, name1, 5, internalMarks1,
externalMarks1);

        External student2 = new External(usn2, name2, 5, internalMarks2,
externalMarks2);




        System.out.println("\nFinal Marks of " + student1.getName() + " (" +
student1.getUsn() + "):");

        System.out.println("Total Internal Marks: " + student1.getTotalInternalMarks());

        System.out.println("Total External Marks: " + student1.getTotalExternalMarks());

        System.out.println("Final Marks: " + student1.getFinalMarks());




        System.out.println("\nFinal Marks of " + student2.getName() + " (" +
student2.getUsn() + "):");

        System.out.println("Total Internal Marks: " + student2.getTotalInternalMarks());

        System.out.println("Total External Marks: " + student2.getTotalExternalMarks());

        System.out.println("Final Marks: " + student2.getFinalMarks());


        scanner.close();

    }

}
```

OUTPUT:

```
D:\JAVA LAB\LAB6>javac CIE/*.java SEE/*.java Main.java

D:\JAVA LAB\LAB6>java Main
Enter USN for student 1: 1BM22MI390
Enter name for student 1: RAKESH
Enter 5 internal marks for student 1:
44
45
46
47
48
Enter 5 external marks for student 1:
90
91
92
93
94
Enter USN for student 2: 1BM22AE044
Enter name for student 2: KHAN
Enter 5 internal marks for student 2:
42
43
44
45
46
Enter 5 external marks for student 2:
89
90
91
92
93

Final Marks of RAKESH (1BM22MI390):
Total Internal Marks: 230
Total External Marks: 460
Final Marks: 690

Final Marks of KHAN (1BM22AE044):
Total Internal Marks: 220
Total External Marks: 455
Final Marks: 675
```

CNB-06

Create a package CIE which has two classes-student and internals. The class personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of student. This class has array that stores the SEE Marks scored in five 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 courses.

```
package CIE;

public class student {
    protected string usn;
    protected string name;
    protected int sem;

    public student (string usn,
                    string name, int sem){
        this.usn = usn;
        this.name = name;
        this.sem = sem;
```

}

```
public String getUsn(){
    return usn;
}
public String getName(){
    return name;
}
public int getSem(){
    return sem;
}
}

package CIE;

public class Internals{
    private int[] internalMarks=
        new int[5];

    public Internals(int[] marks){
        if(marks.length==5){
            this.internalMarks = marks;
        }
        else{
            System.out.println("Error: Exactly 5 marks
                must be provided:");
        }
    }
    public int[] getInternalMarks(){
        return internalMarks;
```

```
    }
public int calculateInteralTotal(){
    int total = 0;
    for(int mark : internalMarks){
        total += marks;
    }
    return total;
}
}

package SEE;

import CIE.student;
import CIE.Internals;

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

    public External(String usn,
            String name, int sem,
            int[] external marks){
        super(name, usn, sem);
        if(ExternalMarks.length == 5){
            this.External marks = ExternalMarks;
        }else{
            S.o.p("Error. Exactly 5 marks
                must be provided!");
        }
    }
}
```

```
public int[] getExternalMarks(){
        return externalMarks;
}

public int calculateExternalTotal(){
        int total = 0;
        for(int mark : ExternalMarks){
            total += Marks;
        }
        return total;
}

public int calculateFinalMarks(
            Internals Internals){
    int internalTotal = internals.calculate
                        Internal Total();
    int ExternalTotal = External.calculateExternalTotal()

    return internalTotal + ExternalTotal;
}
}
```

```
import CIE.*;
import SEE.*;

import java.Util.Scanner;

public class Main{
    p s vm (String [] args){
        Scanner sc = new Scanner(System.in);
        s.o.p("Enter no. of students:");
        int n = sc.nextInt();
        sc.nextLine();

        s.o.p();
        External [] student = new External[n];

        for(int i=0; i<n; i++){
            s.o.p("Enter details for student"
                    +(i+1));
            s.o.p("Enter USN:");
            String USN = sc.nextLine();

            s.o.p("Enter Name:");
            String name = sc.nextLine();

            s.o.p("Enter semester:");
            int sem = sc.nextInt();

            int [] internalMarks = new int[5]
```

output:

Enter details for student 1
Enter USN: 1cs21cs001
Enter Name: kk
Enter semester: 5
Enter internal marks for 5 courses:
50
60
70
80
90
Enter External marks for 5 courses
60
70
80
90
100
final marks for student kk
(1cs21cs001) 600

dip not seen

PROGRAM-7

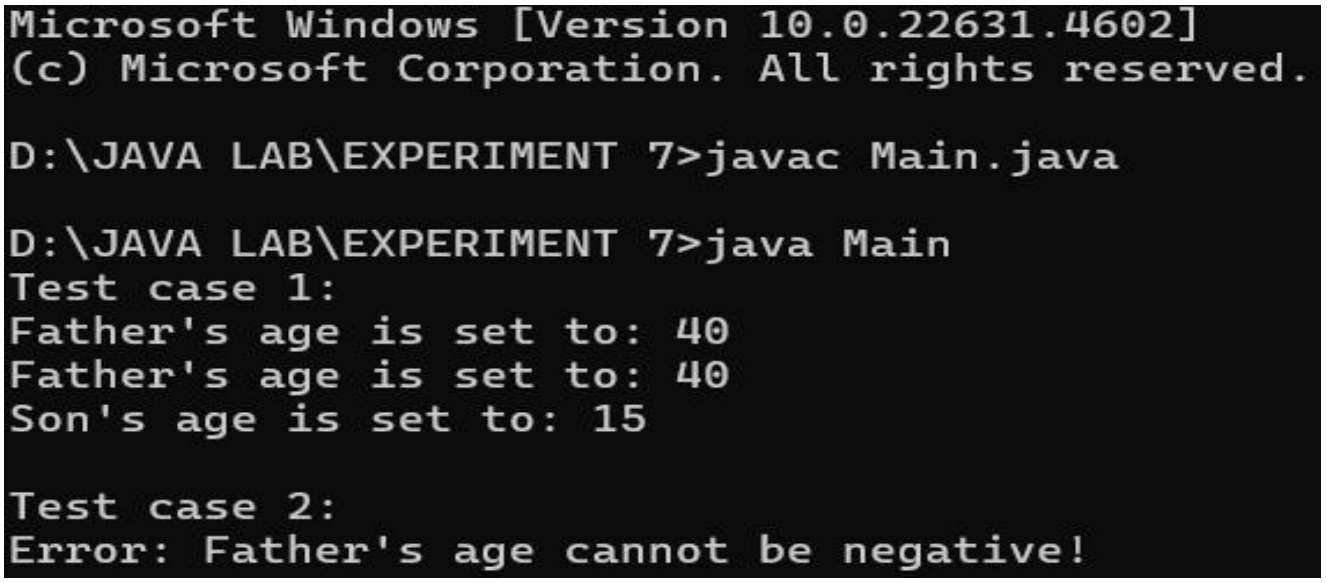
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son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

CODE:

```
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's age cannot be negative!");
        }
        this.age = age;
        System.out.println("Father's age is set to: " + this.age);
    }
}


class Son extends Father {
    int sonAge;


    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge); // Call Father class constructor for validation


        if (sonAge >= fatherAge) {
            System.out.println("Error: Son's age cannot be greater than or equal to Father's
age!");
        } else {
```

```
            this.sonAge = sonAge;
            System.out.println("Son's age is set to: " + this.sonAge);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        try {

            System.out.println("Test case 1:");
            Father father1 = new Father(40);
            Son son1 = new Son(40, 15);
            System.out.println();


            System.out.println("Test case 2:");
            Father father2 = new Father(-10);
            System.out.println();


            System.out.println("Test case 3:");
            Father father3 = new Father(30);
            Son son3 = new Son(30, 30);
            System.out.println();


            System.out.println("Test case 4:");
            Father father4 = new Father(50);
            Son son4 = new Son(50, 25);
        } catch (WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:


Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.
D:\JAVA LAB\EXPERIMENT 7>javac Main.java
D:\JAVA LAB\EXPERIMENT 7>java Main
Test case 1:
Father's age is set to: 40
Father's age is set to: 40
Son's age is set to: 15
Test case 2:
Error: Father's age cannot be negative!

LAB - PROGRAM - 07

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 takes both father and son's age. Constructor that uses both father & son's age. Constructor that uses both father, son's age & throws an exception if son's age > father's age.

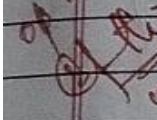
=>
```
class WrongAge Exception extends
        Exception {
    public WrongAge Exception (string
                    message) {
        super (message);
    }
}
class InvalidSonAge Exception
    Extends Exception {
    public InvalidSonAge Exception
        (string message) {
        super (message);
    }
}
```

```
class Father {
    int age;

    Public Father(int Age) throws
                    WrongAgeException {
        if(age<0)
            throw new WrongAgeException
            ("Father age cannot be -ve");
        this.age=age;
    }
}

class Son Extends Father{
    int sonAge;

    public Son(int fage, int sage) throws
    WrongAgeException, InvalidSonAgeException
    {
        super(fage);
        if(sage<0){
            throw new WrongAgeException("
            Son's age cannot be -ve");
        }
        if(sage>=fatherage){
            throw new InvalidSonAg.Exception
            ("Son's age cannot be greater than
            or equal to father age.");
        }
        this.sonAge=sage;
    }
}
```

```
public class ExceptionDemo {
    public static void main(String
            [] args){
try {
    Father f = new father (40);
    Son s = new Son (40,20);
     s.o.p("Father's Age:" +
                f.age);

    s.o.p("Son's age:" + s.age);
}
catch (WrongAgeException e){
      s.o.p(e);
}
try {
    Father f2 = new father(-5);
}
catch (WrongAgeException e){
    s.o.p(e);
}

try {
    Son s2 = new Son(30,35);
}
catch (InvalidSonAgeException
                e){
    s.o.p(e);
}
}
```

output:

Enter your father Age : 45
Enter son's Age = 12
Son's age cannot be less than zero
Would like to enter details (Yes/No)
Yes
Enter father age : 12
Enter son's age : 23
Wrong age
Would you like to



PROGRAM-8

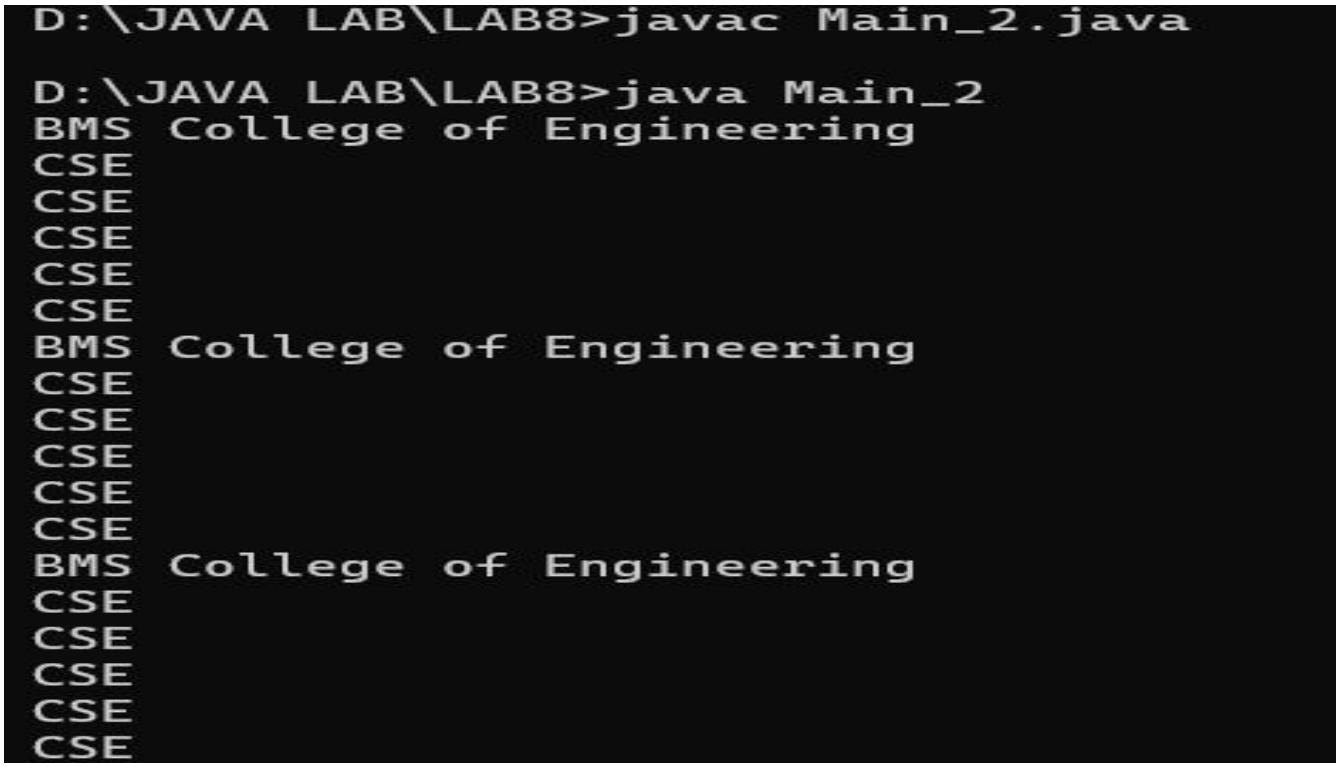
Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

CODE:

```
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(e);
        }
    }
}

class CseThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
```

```
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

public class Main_2 {
    public static void main(String[] args) {

        CollegeThread collegeThread = new CollegeThread();
        CseThread cseThread = new CseThread();



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

OUTPUT:


D:\JAVA LAB\LAB8>javac Main_2.java
D:\JAVA LAB\LAB8>java Main_2
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE

write a programm which creates two threads, one thread displaying "BMS COLLEGE Of ENGINEERING" once every ten seconds & another displaying "CSE" one every two seconds.

```
class ThreadBMS extends thread {
    public void RUN() {
        while (true) {
            try {
                S.O.P("BMS College of Engineering");
                Thread.sleep(10000);
            }
            catch (InterruptException e) {
                S.O.P(e);
            }
        }
    }
}
class ThreadCSE extends Thread {
    public void run() {
        while (true) {
            try {
                S.O.P("CSE");
                Thread.sleep(20000);
            }
            catch (InterruptedException e) {
                S.O.P(e);
            }
        }
    }
}
```

```
public class Main {
    p s v m (String [] args) {
        ThreadBMS threadBMS = new ThreadBMS();
        ThreadCSE threadCSE = new ThreadCSE();
        threadBMS.start();
        threadCSE.start();
    }
}
```

output

BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
. . .

o/p not same

PROGRAM-9

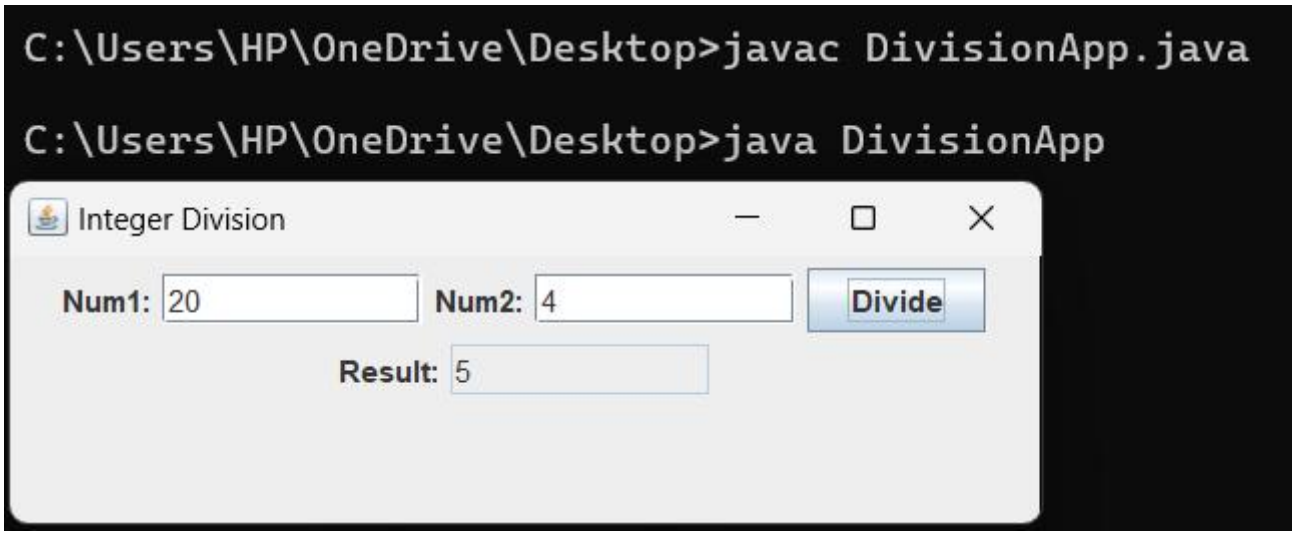
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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

CODE:

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

public class DivisionApp {
   public static void main(String[] args) {

       JFrame frame = new JFrame("Integer Division");
       frame.setSize(400, 300);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setLayout(new FlowLayout());


       JLabel label1 = new JLabel("Num1:");
       JTextField num1Field = new JTextField(10);
       JLabel label2 = new JLabel("Num2:");
       JTextField num2Field = new JTextField(10);
       JButton divideButton = new JButton("Divide");
       JTextField resultField = new JTextField(10);
       resultField.setEditable(false);

       frame.add(label1);
       frame.add(num1Field);
       frame.add(label2);
       frame.add(num2Field);
       frame.add(divideButton);
```

```
        frame.add(new JLabel("Result:"));
        frame.add(resultField);

        divideButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int num1 = Integer.parseInt(num1Field.getText());
                    int num2 = Integer.parseInt(num2Field.getText());


                    if (num2 == 0) {
                        throw new ArithmeticException("Division by zero is not allowed.");
                    }


                    int result = num1 / num2;


                    resultField.setText(String.valueOf(result));
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(frame, "Invalid input. Please enter
integers.", "Error", JOptionPane.ERROR_MESSAGE);
                } catch (ArithmeticException ex) {

                    JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        frame.setVisible(true);
    }
}
```

OUTPUT:


C:\Users\HP\OneDrive\Desktop>javac DivisionApp.java
C:\Users\HP\OneDrive\Desktop>java DivisionApp
Integer Division
Num1: 20
Num2: 4
Divide
Result: 5

lab-9

Write a program that creates a User interface to perform integer division. The user enters two numbers in the text fields, NUM1 & NUM2. The division of NUM1 & 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 ArithmeticException Display the exception in a message dialog box.

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

public class DivisionApp {
    Public static void main (String[] args) {

        JFrame frame = new JFrame("Integer
                                  Division");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.
            EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());
```

```
JLabel label1 = new JLabel("Num1:");
JTextField num1Field = new JTextField(10);
JLabel label2 = new JLabel("Num 2:");
JTextField num2Field = new JTextField(10);
JButton divideButton = new JButton("Divide");
JTextField resultField = new JTextField(10);
resultField.setEditable(false);

frame.add(label1);
frame.add(num1Field);
frame.add(label2);
frame.add(num2Field);
frame.add(divideButton);
frame.add(new JLabel("Result:"));
frame.add(resultField);

divideButton.addActionListener(new
        ActionListener() {
public void actionPerformed(ActionEvent e)
{
try {
int num1 = Integer.parseInt(num1Field.
                getText());
int num2 = Integer.parseInt(num2Field.
                getText());
if (num2==0) {
    throw new ArithmeticException(
            "Division by zero is not
                allowed.");
}
int result = num1/num2;
```

```
resultField.setText(String.valueOf(result));
}
catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog
    (frame, "Invalid input. Please enter integers.",
    "Error", JOptionPane.ERROR_MESSAGE);
} catch (ArithmeticException ex) {

    JOptionPane.showMessageDialog(frame,
    ex.getMessage(), "Error",
    JOptionPane.ERROR_MESSAGE);

    }
});

frame.setVisible(true);
}
}
```

Output

Integer Division.

Num 1: [20] Num 2: [4]

[Divide]

Result: [5]

PROGRAM-10

Demonstrate Inter process Communication and deadlock.

CODE: i) inter process communication

```
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<05) {
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<05) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}
class PC_Fixed {
```

```
public static void main(String args[]) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control-C to stop.");
}
}
```

```
D:\JAVA LAB\LAB10>javac PC_Fixed.java

D:\JAVA LAB\LAB10>java PC_Fixed
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

consumed:0
Got: 1

Intimate Producer

consumed:1
Put: 2

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

consumed:2
Put: 3

Intimate Consumer


Producer waiting

Got: 3

Intimate Producer

consumed:3
Put: 4

Intimate Consumer

Got: 4

Intimate Producer

consumed:4
```

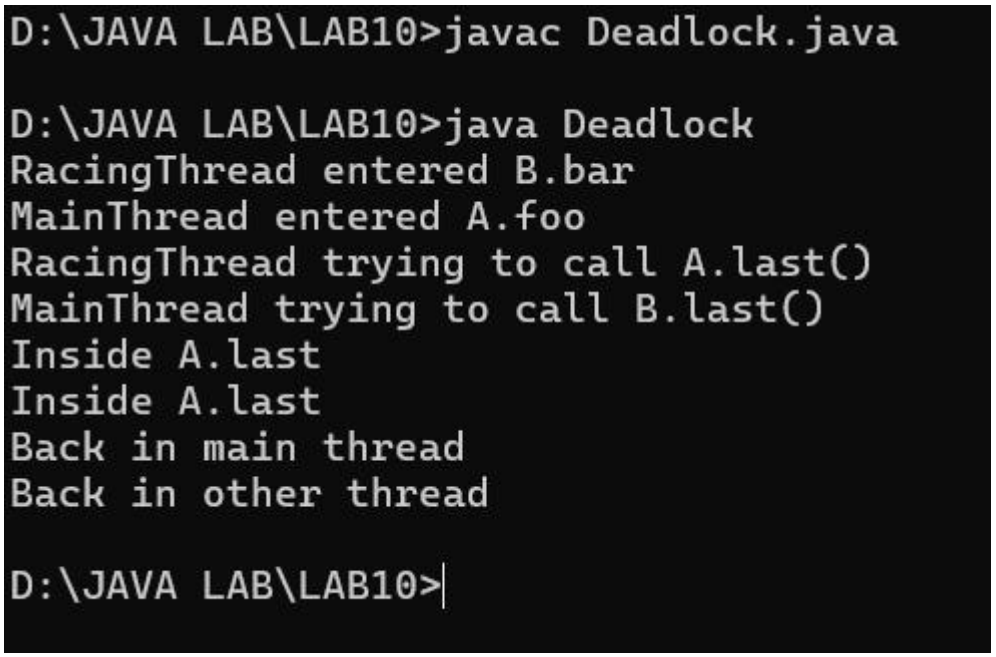
ii)Deadlock

```
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");

}

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);
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();

}

}
```

OUTPUT:



Inter process Communication &
Deadlock

i) Inter Process Communication

```
class Q {
int n;
boolean ValueSet = false;
Synchronized int get () {
while (!valueset)
try {
System.out.println("\n Consume waiting\n");
wait();
} catch (InterruptException e) {
s.o.p (" Interrupt Exception caught");
}
System.out.println ("Got: " + n);
ValueSet = false;
System.out.println ("\n Intimade producer\n");
notify();
return n;
}
synchronized void put(int n){
while (ValueSet)
try {
  s.o.p ("\n producer waiting\n");
  wait();
}
catch (InterruptException e){
  s.o.p("Interupted 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 < 05) {
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 < 05) {
int r = q.get();
s.o.p ("Consumed: " + r);
i++;
```

```
}
}
}
class PCFixed {
public static void main(String args[]) {
  Q q = new Q();
  new Producer(q);
  new Consumer(q);
  S.O.P("Press Control-C to stop.");
}
}
```

output

```
Press Control-C to stop.
Put : 0
Intimate Consumer
Producer Waiting.
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
.
Producer Waiting.
Consumed : 0
Got : 1
Intimate Producer
Consumed : 1
Put : 2
Intimate Consumer
Producer Waiting
```

ii) 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) {
  s.o.p("A Interupted");
}
s.o.p(name + "trying to call B.last()");
b.last();
}
void last() {
s.o.p("Inside A.last");
}
}
class B {
synchronized void bar(A a)
String name = Thread.currentThread().
                getName();
s.o.p(name + "entered B.bar");
try {
  Thread.sleep(1000);
} catch(Exception e) {
```

```
        s.o.p ("B.Interrupted");
    }
    s.o.p( name + "trying to call A.last()");
    a.last();
    }
    void last(){
      s.o.p("Inside A.Last");
    }
}

class Deadlock implements Runnable
{
    A a = new A();
    B b = new B();
    Deadlock(){
    Thread.currentThread().setName("Main Thread");
    Thread t = new Thread(this, "Racing
                                thread");
    t.start();
    a.foo(b);
    s.o.p("Back in main Thread");
    }
    public void run(){
     b.bar(a);
     s.o.p("Back in other thread");
    }
    public static void main(String args[]){
     new Deadlock();
    }
}
```

output:

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