B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

OBJECT ORIENTED JAVA PROGRAMMING

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

SANJANA SURESH

1BM22CS239

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019

INDEX

USN - 1BM22CS239

Name SANJANA SURESH Sub.

Std.: Div. Roll No.

Telephone No. E-mail ID.

Blood Group. Birth Day.

Sr.No.	Title	Page No.	Sign,/Remarks
01	Sample program 5/12/23		5-12
02	Program 1 - Quadratic 12/12/23		12-12
03	Program 2 - Student SGPA 19/12/23		19/12
04	Program 3 - Books 26/12/23		26-12-23
05	Program 4 - Shape 02/01/24		2/1/24
06	Program 5 - Bank 16/01/24		16-1-24
07	Program 6 - Student (Package) 23/01/24		24-1-24
08	Program 7 - Exception 30/01/24		30-1-24
09	Program 8 - Multithreads 06/02/24		6-2-24
10	Program 10 (a) Inter Process Communication 13/02/24 (b) Deadlock		13-2-24
11	Program - 9 Java Applet (Divisior) 20/02/24		20/2/2024

19/12/23

LAB PROGRAM - 01

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.

```
import java.util.Scanner;
class quadratic
{
    int a, b, c;
    double r1, r2, d;
    void getd()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the coefficients of a, b, c");
        a = s.nextInt();
        b = s.nextInt();
        c = s.nextInt();
    }
    void compute()
    {
        while (a==0)
        {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non zero value for a: ");
            Scanner s = new Scanner(System.in);
            a = s.nextInt();
        }
        d = b*b - 4*a*c;
        if (d==0)
        {
            r1 = (-b)/(2*a);
            System.out.println("Roots are real and equal");
            System.out.println("Root1 = Root2 = "+r1);
        }
```

```
else if (d > 0)
{
    r1 = ((-b) + (Math.sqrt(d))) / (double)(2*a);
    r2 = ((-b) - (Math.sqrt(d))) / (double)(2*a);
    System.out.println("Roots are real and distinct");
    System.out.println("Root 1 = " + r1 + "Root 2 = " + r2);
}
else if (d < 0)
{
    System.out.println("Roots are imaginary");
    r1 = (-b) / (2*a);
    r2 = Math.sqrt(-d) / (2*a);
    System.out.println("Root 1 = " + r1 + " + i" + r2);
    System.out.println("Root 1 = " + r1 + " - i" + r2);
}
}
}

class QuadraticMain
{
    public static void main(String args[])
    {
        Quadratic q = new Quadratic()
        q.getd();
        q.compute();
    }
}
```

OUTPUTS

① Enter the coefficients of a, b, c

3 2 1

Roots are imaginary

Root 1 = 0.0 + i0.471404520799

Root 2 = 0.0 - i0.471404520799

SANJANA SURESH
1BM22CS239

② Enter the coefficients of a, b, c

1 2 1

Roots are real and equal

Root 1 = Root 2 = -1.0

③ Enter the coefficients of a, b, c

1 4 1

Roots are real and distinct

Root 1 = -3.732050807568877

Root 2 = 0.0

12/12/2024

19/12

LAB - 02

Q2. 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.*;
class Subject
{
    int subjectMarks;
    int credits;
    int grade;
}

class Student
{
    Subject subject[];
    String name;
    String usn;
    double sgpa;
    Scanner s;
    Student()
    {
        int i;
        subject = new Subject[8];
        for(i=0; i<8; i++)
            subject[i] = new Subject();
        s = new Scanner(System.in);
    }
    void getStudentDetails()
    {
        System.out.println("Enter student name");
        name = s.next();
        System.out.println("Enter student USN");
        usn = s.next();
```

```
void getMarks()
{
    for (int i = 0; i<9; i++)
    {
        System.out.println("Enter marks");
        subject[i].subjectMarks = S.nextInt();
        System.out.println("Enter number of credits");
        subject[i].credits = S.nextInt();
        subject[i].grade = (subject[i].subjectMarks/10) + 1;
        if (subject[i].grade == 11)
            subject[i].grade = 10;
        if (subject[i].grade <= 4)
            subject[i].grade = 0;
    }
}

void computeSGPA()
{
    int effectiveScores = 0;
    int totalCredits = 0;
    for (int i=0; i<9; i++)
    {
        effectiveScores += subject[i].grade * subject[i].credits;
        totalCredits += subject[i].credits;
    }
    sgpa = (double) effectiveScores / (double) totalCredits;
}
}

class main
{
    public static void main(String args[])
    {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
```

```
        System.out.println(" Student name :  " + st.name);
        System.out.println(" Student USN :  " + st.usn);
        System.out.println(" Student SGPA :  " + st.sgpa);
    }
}
```

O/P

```
Enter student name
Sanjana
Enter student USN
1BM22CS219
Enter marks
98
Enter number of credits
4
Enter marks
89
Enter number of credits
3
Enter marks
91
Enter number of credits
3
Enter marks
87
Enter number of credits
3
Enter marks
78
Enter number of credits
2
Enter marks
87
Enter number of credits.
3
```

SANJANA SURESH
1BM22CS219

Enter marks
93
Enter number of credits:
3
Enter marks
89
Enter number of credits:
3
Enter marks
76
Enter number of credits:
3

Student name : Sanjana
Student usn : 1BM22CS239
Student SGPA : 9.185185185

SANJANA SURESH
1BM22CS239

19-12

26/12

LAB - 03

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

```
import java.util.Scanner ;

class Book
{
    String name ;
    String author ;
    int price ;
    int numPages ;

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

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

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

        System.out.println (" Enter the number of books ");
        n = s.nextInt ()
        Book [] books = new Book [n];

        for (int i = 0; i <n; i++)
        {
            System.out.println (" Enter the name of book ");
            String name = s.next ();

            System.out.println (" Enter the price of the book ");
            int price = s.nextInt ();
            System.out.println (" Enter the name of the author ");
            String author = s.next ();
            System.out.println (" Enter the pages of the book ");
            int numPages = s.nextInt() ;
            books [i] = new Book (name, author, price, numPages);
        }

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

OUTPUT

① Enter no. of books

2

Enter the name of the book

The Kite Runner

Enter the author of the book

Khaled Hosseini

Enter the price of the book

350

Enter the page of the book

540

Enter the name of the book

Harry Potter

Enter author of the book

J.K Rowling

Enter price of the book

200

Enter pages of the book

600

Book name : The Kite Runner
Author name : Khaled Hosseini
Price : 350
Number of pages : 540

Book name : Harry Potter
Author name : JK Rowling
Price : 200
Number of pages : 600

SANJANA SURESH
1BM21CS239

28-12-23

LAB - 04

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 extend the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

① Add class inputScanner

abstract class Shape extent inputScanner

class Rectangle extends Shape

class Triangle "

class Circle "

Add mainclass invoking main method with objects and

```
import java.util.*;
abstract class Shape
{
    double dim1;
    double dim2;
    double rad;
    Shape(double a, double b)
    {
        dim1 = a;
        dim2 = b;
    }
    Shape(double a)
    {
        rad = a;
    }
    abstract void area();
}
```

```
class Rectangle extends Shape
{
    Rectangle ( double a, double b)
    {
        super (a, b);
    }
    void area ()
    {
        System.out.println("Area of rectangle is : " + (dim1 * dim2));
    }
}

class Triangle extends Shape
{
    Triangle (double a, double b)
    {
        super (a, b);
    }
    void area()
    {
        System.out.println(" Area of triangle is " + (dim1 * dim2) / 2);
    }
}

class Circle extends Shape
{
    Circle (double a)
    {
        super(a);
    }
    void area();
    {
        System.out.println ("Area of circle is " + ((3.14 * rad * rad));
    }
}
```

```
class AbstractMain
{
    public static void main (String args[])
    {
        Scanner s = new Scanner (System.in);
        System.out.println (" Enter the dimensions of the rectangle ");
        double l = s.nextInt ();
        double b = s.nextInt ();
        System.out.println (" Enter the base and height of triangle ");
        double h1 = s.nextInt ();
        double h2 = s.nextInt ();
        System.out.println (" Enter radius of circle ");
        double r = s.nextInt ();
        Shapes sh;
        Rectangle rect = new Rectangle (l, b);
        Triangle tri = new Triangle (h1, h2);
        Circle cir = new Circle (r);
        sh = rect;
        sh.area ();
        sh = tri;
        sh.area ();
        sh = cir;
        sh.area ();
```

3

3

OUTPUT

Enter the dimensions of rectangle

5 6

Enter the base and height of triangle

4 4

Enter the radius of circle

6

Area of rectangle : 30.0

Area of triangle : 8.0

Area of circle : 113.039999

SANJANA SURESH
1BM22CS239

2/1/24

05/11/24

LAB - 05

Q. Develop a JAVA program to create a class Bank that maintains two kinds of account for 'n' customers, one called Savings account and the other current account. The saving account provide compound interest and withdrawal facilities but no cheque book facility. The current account provides check 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-act and sav-act 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.

```
import java.util.*;
class Account
{
    String name;
    int accno;
    String acc_type;
    double balance;

    Account(String name, int accno, String acc_type; double balance)
    {
        this.name = name;
        this.accno = accno;
        this.acc_type = acc_type;
        this.balance = balance;
    }
```

```
void deposit(double amount)
{
    balance += amount;
}

void withdraw(double amount)
{
    if ((balance-amount) >= 0)
    {
        balance -= amount;
    }
    else
    {
        System.out.println("Insufficient balance ! Cannot withdraw");
    }
}

void display()
{
    System.out.println("Name: " + name + " Accno: " + accno + "Type: " + type
        "Balance : " + balance);
}
}

class sav_Account extends Account
{
    private static double rate = 5;
    sav_Account(String name, int accno, double balance)
    {
        super(name, accno, "savings", balance);
    }
    void interest()
    {
        balance += balance * (rate) /100;
        System.out.println("Balance : " + balance);
    }
}
```

```
class Curr_account extends Account
{
    private double minBal = 500;
    private double serviceCharge = 50;
    Curr_account (String name, int accno, double balance)
    {
        super (name, accno, "current", balance);
    }

void checkMin ()
{
    if (balance < minBalance)
    {
        System.out.println ("Balance is less than minimum, service charge is
        imposed " + serviceCharge);
        balance -= serviceCharge;
        System.out.println ("Balance is : " + balance);
    }
}
}

class Bank
{
    public static void main (String args[])
    {
        Scanner s = new Scanner (System.in);
        System.out.println ("Enter customer name");
        String name = s.next();
        System.out.println ("Enter account number");
        int accno = s.nextInt();
        System.out.println ("Enter the type of account - current or saving");
        String acc_type = s.next();
        System.out.println ("Enter the initial balance");
        double balance = s.nextDouble();
        Account ob1 = new Account (name, accno, acc_type, balance);
```

```
Sav_account sa = new Sav_account (name, accno, balance);
Curr_account ca = new Curr_account (name, accno, balance);

while (true)
{
    if (acc_type.equals ("Savings"))
    {
        System.out.println ("Menu");
        System.out.println ("1. Deposit \n 2. Withdraw 3. Compute interest
        4. Display account details 5. Exit");
        System.out.println ("Enter your choice - ");
        int choice = s.nextInt();

        switch (choice)
        {
            case 1 :
            {
                System.out.println ("Enter the amount to be deposited");
                double amt1 = s.nextDouble ();
                sa.deposit (amt1);
                break;
            }
            case 2 :
            {
                System.out.println ("Enter the amount to be withdrawn");
                double amt2 = s.nextDouble ();
                sa.withdraw (amt2);
                break;
            }
            case 3:
            {
                sa.interest ();
                break;
            }
            case 4 :
            {
                sa.display ();
                break;
            }
```

```
    {
        break;
    }
    }
    default:
    {
        System.out.println("Enter valid input");
        break;
    }
    }
}
```

```
System.out.println("Menu");
System.out.println("1. Deposit 2. Withdraw 3. Display Account Details 4. Exit");
System.out.println("Enter your choice");
int choice = s.nextInt();
switch(choice)
{
    case 1:
    {
        System.out.println("Enter the amount to be deposited");
        double amt1 = s.nextDouble();
        cu.deposit(amt1);
        break;
    }
    case 2:
    {
        System.out.println("Enter the amount to be ~~deposited~~ withdrawn");
        double amt2 = s.nextDouble();
        cu.withdraw(amt2);
        break;
    }
    case 3:
    {
        cu.display();
        break;
    }
```

```
case 4:
{
    break;
}
3
default:
{
    System.out.println("Enter valid input");
    break;
}
3
3
3
3
3
3
```

OUTPUT

```
// SAVINGS
Enter customer name
Manjana
Enter account number
1
Enter the type of account - current or savings
Saving
Enter the initial balance 5000
Menu
1. Deposit 2. Withdraw 3. Compute Interest 4. Display Account Details 5. Exit
Enter your choice 1
Enter the amount to be deposited 3000
Menu
1. Deposit 2. Withdraw 3. Compute Interest 4. Display Account Details 5. Exit
2
Enter the amount to be withdrawn 9000
Insufficient balance! Cannot withdraw.
Menu
1. Deposit 2. Withdraw 3. Compute Interest 4. Display Account Details 5. Exit
```

4

Name : Sanjana Account number : 1 Account type : Saving Balance : 5000.0

// CURRENT.

Enter customer name

Sanjana

Enter account number

2

Enter the type - current or saving

current

Enter the initial balance

4000

Menu

1. ~~Withdraw~~ Deposit 2. Withdraw 3. Display Account Details 4. Exit.

Enter your choice

1

Enter the amount to be deposited 2000

Menu

1. Deposit 2. Withdraw 3. Display Account Details 4. Exit.

Enter your choice

3

Name : Sanjana Account number : 2 Account type : Current Balance : 6000.

SANJANA SURESH
1BM22CS239.

1-24

23/01/24

LAB - 06

Create a package called Maths having the class called Number (add and subtract methods). Implement a single class called MathDemo to use Maths (outside package Maths) that make use of classes provided by Maths.

Create a package CIE which has two classes — Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is derived class of Student. This class has an array that stores the SEE marks scored in 5 courses of the current semester of the student. Import the two packages in a file that declares final marks of a student in all 5 courses.

```
// Student.java
package CIE;
import java.util.*;
public class Student
{
    protected String usn = new String();
    protected String name = new String();
    protected int sem;

    public void inputStudentDetails()
    {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter student USN");
      usn = sc.next();
      System.out.println("Enter student name");
      name = sc.next();
```

```
System.out.println (" Enter semester ");
sem = sc.nextInt ();


public void displayStudentDetails ()
{
  System.out.println (" Student usn : " + usn);
  System.out.println (" Student name : " + name);
  System.out.println (" Student semester : " + sem);
}
```

// Internals.java

```
package CIE ;
import java.util.*;

public class Internals extends Student
{
   protected int marks [] = new int [5] ;

   public void inputCIEmarks ()
   {
      Scanner sc = new Scanner (System.in) ;
      for (int i = 0 ; i <5 ; i++)
      {
          System.out.println (" Enter 5 subject marks ");
          marks [i] = sc.nextInt ();
      }
   }
}
```

```
// External.java

package SEE;
import CIE.Internals;
import java.util.Scanner;

public class External extends Internals
{
    protected int marks[];
    protected int finalMarks[];

    public External()
    {
        marks = new int[5];
        finalMarks = new int[5];
    }

    public void inputSEEMarks()
    {
        Scanner sc = new Scanner(System.in);
        for (int i=0; i<5; i++)
        {
            System.out.println("Subject " + (i+1) + " marks: ");
            marks[i] = sc.nextInt();
        }
    }

    public void calculateFinalMarks()
    {
        for(int i=0; i<5; i++)
        {
            finalMarks[i] = marks[i]/2 + super.marks[i];
        }
    }

public void displayFinalMarks()
{
    displayStudentDetails();
    for (int i=0; i<5; i++)
```

```
        System.out.println("Subject " + (i+1) + " : " + finalMarks[i]);
    }
}
}
```

// main.java

```
import SEE.External
class Main
{
    public static void main(String args[])
    {
        int numOfStudents = 2;
        External finalMarks[] = new External[numOfStudents];
        for (int i=0; i < numOfStudents ; i++)
        {
            finalMarks[i].inputCIEmarks();
            finalMarks[i]= new External();
            finalMarks[i].inputStudentDetails();
            System.out.println("Enter CCE marks");
            finalMarks[i].inputCIEmarks();
            System.out.println("Enter SEE marks");
            finalMarks[i].inputSEEmarks();
        }
        System.out.println("Displaying data :\n");
        for(int i=0; i< numOfStudents; i++)
        {
            finalMark[i].calculateFinalMark();
            finalMarks[i].displayFinalMarks();
        }
    }
}
```

OUTPUT —

Enter student usn

1BM22CS239

Enter student name

Sanjana

Enter student semester

3

Enter CIE marks

Enter 5 marks : 39

Enter 5 marks : 38

Enter 5 marks : 26

Enter 5 marks : 40

Enter 5 marks : 36

Enter SEE marks

Subject 1 marks : 78

Subject 2 marks : 89

Subject 3 marks : 90

Subject 4 marks : 87

Subject 5 marks : 92

Enter student usn

1BM22CS240

Enter student name

Sanvi

Enter student semester

3

Enter CIE marks

Enter 5 marks : 39

Enter 5 marks : 38

Enter 5 marks : 30

Enter 5 marks : 30

SANJANA SURESH

1BM22CS239

Enter 5 marks : 25.

Enter SEE marks

Subject1 marks : 56

Subject 2 marks : 78

Subject 3 marks : 89

Subject 4 marks : 90

Subject 5 marks : 92

SANJANA SURESH
1BM22CS239

Displaying data —

Student name : Sanjana

Student usn : 1BM22CS239

Student semester : 3

Subject 1 : 78

Subject 2 : 82

Subject 3 : 71

Subject 4 : 83

Subject 5 : 82

Student name : Sanjana

Student usn : 1BM22CS240

Student semester : 3

Subject 1 : 58

Subject 2 : 77

Subject 3 : 80

Subject 4 : 70

Subject 5 : 80

24/1/24

30/01

LAB - 07

Write a program that demonstrates handling of exceptions in inheritance class. 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 an exception WrongAge when the input age < 0. In Son class, implement a constructor that takes both father and son's age and throws an exception if son's age >= father's age.

```
import java.util.*;
class WrongAge extends Exception
{
    public WrongAge(String s)
    {
        super(s);
    }
}

class Father
{
    Scanner sc= new Scanner(System.in);
    public int F_age;
    public Father() throws WrongAge
    {
        System.out.println("Enter Father's age");
        F_age = sc.nextInt();
        if (F_age <0)
        {
            throw new WrongAge("Age cannot be negative");
        }
    }
public vo
}
```

```
class Son extends Father
{
    private int S_age;
    public Son() throws WrongAge
    {
        System.out.println("Enter son's age");
        S_age = sc.nextInt();
        if (S_age > F_age)
        {
            throw new WrongAge("Son's age cannot be greater than father's age");
        }
        else if (S_age < 0)
        {
            throw new WrongAge("Age cannot be negative");
        }
        else if (S_age == F_age)
        {
            throw new WrongAge("Age cannot be same");
        }
        else
        {
            System.out.println("Valid age");
        }
    }
    public void display()
    {
        System.out.println("Father's age = " + F_age);
        System.out.println("Son's age = " + S_age);
    }
}
```

```
class Lab7
{
    public static void main (String args[])
    {
        try
        {
            Son son = new Son ();
            son.display ();
        }
        catch (WrongAge e)
        {
            System.out.println ("Exception caught");
            System.out.println (e.getMessage ());
        }
    }
}
```

O/P

① Enter Father's age 67
Enter Son's age 56
Valid age
Father's age = 67
Son's age = 56

SANJANA SURESH
1BM22CS279

② Enter Father's age = -10
Exception caught
Age cannot be negative.

③ Enter Father's age = 20
Enter Son's age = 20
Exception caught
Age cannot be same.

④ Enter father's age = 45
Enter son's age = 90

Exception caught
Son's age cannot be greater than father's age.

SANJANA SURESH
1DM22CS239

10/1/24

06/02

LAB - 08

Write a program which creates two thread, one thread displaying "BMS college of Engineering" once every 10 seconds and another displaying "CSE" once every 2 seconds —

```
class BMS extends Thread
{
    public void run ()
    {
        for (int i = 1; i <= 5; i++)
        {
            System.out.println ("bms college of engineering " + i);
            try
            {
                Thread.sleep (10000);
            }
            catch (InterruptedException e)
            {
                System.out.println ("The sleeping thread has woken up");
            }
        }
    }
}

class CSE extends Thread
{
    public void run ()
    {
        for (int i = 1; i <= 5; i++)
        {
            System.out.println ("CSE " + i);
            try
            {
                Thread.sleep (2000);
            }
            catch (InterruptedException e)
            {
                System.out.println ("The sleeping thread has woken up");
            }
        }
    }
}
```

```
class ThreadMain
{
    public static void main(String a[])
    {
        BMS obj1 = new BMS();
        CSE obj2 = new CSE();
        obj1.start();
        obj2.start();
    }
}
```

OUTPUT —

```
bms college of engineering 1
csc 1
csc 2
csc 3
csc 4
csc 5
bms college of engineering 2
csc 6
csc 7
csc 8
csc 9
csc 10
bmscollege of engineering 3
bms college of engineering 4
bms college of engineering 5
```

SANJANA SURESH
1BM22CS227

Lab
6-2-24

13/02

LAB - 10

Demonstrate Interprocess Communication and Deadlock

(i) Interprocess Communication

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

```
        notify ();
    }
}
class Producer implements Runnable
{
    Q q;
    Producer (Q q)
    {
        this.q = q;
        new Thread (this, "Producer").start ();
    }
    public void run()
    {
        int i = 0;
        while (i<15)
        {
            q.put (i++);
        }
    }
}

class Consumer implements Runnable
{
    Q q;
    Consumer (Q q)
    {
        this.q = q;
        new Thread (this, "Consumer").start ();
    }
    public void run ()
    {
        int i = 0;
        while (i < 15)
        {
            int r = q.get() ;
            System.out.println ("Consumed: "+r);
            i++;
        }
    }
}
```

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

OUTPUT

```
Put : 0
Intimate Consumer
Producer Waiting
Press Control-C to stop
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
```

SANJANA SURESH
1BM22CS239

Intimate Consumer

Producer Waiting

got : 3

Intimate ~~Consumer~~ Producer

~~Producer waiting~~ consumed : 3

~~got~~ Put : 4

Intimate Consumer

Producer waiting

got : 4

Intimate Producer

consumed : 4

Put : 5

Intimate Consumer

Producer waiting

got : 5

Intimate Producer

Consumed : 5

Put : 6

Intimate Consumer

Producer waiting

got : 6

Intimate Producer

Consumed : 6

Put : 7

Intimate Consumer

Producer waiting

got : 7

Intimate Producer

Consumed : 7

Put : 8

Intimate Consumer

SANJANA SURESH
1BM22CS239

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 : 11

Intimate Producer

Consumed : 11

Put : 12

Intimate Consumer

SANJANA SURESH
1AM22CS239

Producer waiting

qA : 12

Intimate Producer

consumed : 12

Put : 13

Intimate consumer

Producer waiting

SANTANA SURESH
1NM22CS279

13-2-24

(1) 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's 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 another thread");
    }
    public static void main(String args[])
    {
        new Deadlock();
    }
}
```

(1 o/p

Main Thread entered A.fro

Racing thread entered B.bar

Main Thread 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 ~~the~~ thread

SANJANA SURESH
1BM22CS231

13-2-24

30/01/24

LAB - 09

JAVA AWT AND APPLET

Write a program that creates a user interface to perform integer division. The user enters two numbers in the textfields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num2 were zero, the program would throw an exception in a message dialog box —

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

public class DivisionMain1 extends Frame implements ActionListener
{
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = "";
    double resultNum;
    int flag = 0;

    public DivisionMain1()
    {
        setLayout(new FlowLayout());
        dResult = new Button("RESULT");
        Label number1 = new Label("Number 1 :", Label.RIGHT);
        Label number2 = new Label("Number 2 :", Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
        outResult = new Label("Result :", Label.RIGHT);

        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);
```

```
num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this);
addWindowListener(new WindowAdapter())
{
    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }
});
}

public void actionPerformed(ActionEvent ae)
{
    int n1, n2;
    try
    {
        if(ae.getSource() == dResult)
        {
            n1 = Integer.parseInt(num1.getText());
            n2 = Integer.parseInt(num2.getText());
            out = n1 + " " + n2;
            resultNum = n1/n2;
            Out+ = String.valueOf(resultNum);
            repaint();
        }
    }
    catch(NumberFormatException e1)
    {
        flag = 1;
        Out = "Number Format Exception! " + e1;
        repaint();
    }
    catch(ArithmeticException e2)
    {
        flag = 1;
        out = "Divide by 0 exception! " + e2;
        repaint();
    }
}
```

```
public void paint (Graphics g)
{
    if (flag == 0)
    { g.drawString (out, outRect.getX() + outRect.getWidth(), outRect.getY() +
                    outRect.getHeight() - 8); }

    else
    { g.drawString (out, 100, 200);
      flag = 0; }
}

public static void main (String [] args)
{
    DivisionMain1 dm = new DivisionMain1();
    dm.setSize (new Dimension (500, 400));
    dm.setTitle ("Division of Integers");
    dm.setVisible (true);
}
}
```

SANTANA SURESH
1BM22CS239

OUTPUT —

Division of Integers

Number1 : 35 Number2 : 7 RESULT

Result : 35 7 5.0

Division of Integers

Number 1 : 20 Number2 : 0 RESULT

NumberFormatException ! java.lang.NumberFormatException : For input string : "20"

Functions used —

① FlowLayout () — The flow layout manager. Flow layout positions components left to right, top to bottom.

② Button — creates a push button control

③ Label — creates a label that displays a string

④ TextField — creates a single line edit control

⑤ WindowEvent — Generated when a window is activated, closed, deactivated, deconified, iconified, opened or quit.

⑥ WindowClosing — Their requested that the window is closed

⑦ ActionListener — Defines one method to receive action events

⑧ WindowListener — Defines seven methods to recognize when a window is activated, closed, deactivated ..

⑨ repaint () — Defined by Component. It causes the AWT run time system to execute a call to update () method.

⑩ setSize () — sets the size of this Dimension object to specified width and height

20/2/2024

PROGRAM-01

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

```
import java.util.Scanner;
class Quadratic
{
int a,b,c;
double r1,r2,d;
void getd()
{
	Scanner s= new Scanner(System.in);
	System.out.println("Enter the coefficients of a,b,c");
	a=s.nextInt();
	b=s.nextInt();
	c=s.nextInt();
}
void compute()
{
	while (a==0)
	{
		System.out.println("Not a quadratic equation");
		System.out.println("Enter a non zero value for a:");
		Scanner s=new Scanner(System.in);
		a=s.nextInt();
	}
	d=b*b-4*a*c;
	if (d==0)
	{
		r1=(-b)/(2*a);
		System.out.println("Roots are real and equal");
		System.out.println("Root1=Root2= "+r1);
	}
	else if(d>0)
	{
		r1=((-b)+(Math.sqrt(d)))/(double)(2*a);
		r1=((-b)-(Math.sqrt(d)))/(double)(2*a);
		System.out.println("Roots are real and distinct");
```

```
                System.out.println("Root1= "+r1+ "Root2= "+r2);
        }
        else if(d<0)
        {
                System.out.println("Roots are imaginary");
                r1=(-b)/(2*a);
                r2=Math.sqrt(-d)/(2*a);
                System.out.println("Root1= "+r1+ "+i"+r2);
                System.out.println("Root1= "+r1+ "-i"+r2);
        }
}
}

class QuadraticMain
{
        public static void main(String args[])
        {
                Quadratic q=new Quadratic();
                q.getd();
                q.compute();
        }
}
```

PROGRAM-02

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.*;
class Subject
{
        int subjectMarks;
        int credits;
        int grade;
}
class Student
```

```
{
	Subject subject[];
	String name;
	String usn;
	double sgpa;
	Scanner s;
	Student()
	{
		int i;
		subject =new Subject[9];
		for(i=0;i<9;i++)
			subject[i]=new Subject();
		s=new Scanner(System.in);
	}
	void getStudentDetails()
	{
		System.out.println("Enter student name");
		name=s.next();
		System.out.println("Enter student USN");
		usn=s.next();
	}
	void getMarks()
	{
		for(int i=0;i<9;i++)
		{
			System.out.println("Enter marks");
			subject[i].subjectMarks=s.nextInt();
			System.out.println("Enter number of +credits");
			subject[i].credits=s.nextInt();
			subject[i].grade=(subject[i].subjectMarks/10)+1;
			if(subject[i].grade==11)
				subject[i].grade=10;
			if(subject[i].grade<=4)
				subject[i].grade=0;

		}
	}
	void computeSGPA()
	{
		int effectiveScores=0;
		int totalCredits=0;
		for(int i=0;i<9;i++)
```

```
                {
                        effectiveScores+=subject[i].grade*subject[i].credits;
                        totalCredits+=subject[i].credits;
                }
                sgpa=(double)effectiveScores/(double)totalCredits;
        }
}



class Main
{
        public static void main(String args[])
        {
                Student s1=new Student();
                s1.getStudentDetails();
                s1.getMarks();
                s1.computeSGPA();
                System.out.println("Student name:"+s1.name);
                System.out.println("Student usn:"+s1.usn);
                System.out.println("Student SGPA:"+s1.sgpa);
        }
}
```

PROGRAM-03

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

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

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

```
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

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

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

        System.out.println("Enter the number of books: ");
        n = s.nextInt();
        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter the name of the book");
            String name = s.next();
            System.out.println("Enter the author of the book");
            String author = s.next();
            System.out.println("Enter the price of the book");
            int price = s.nextInt();
            System.out.println("Enter the pages of the book");
            int numPages = s.nextInt();
            books[i] = new Book(name, author, price, numPages);
        }

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

PROGRAM-04

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.

```
import java.util.*;


abstract class Shape{
        double dim1;
        double dim2;
        double rad;
Shape(double a,double b)
{
        dim1=a;
        dim2=b;
}


Shape(double a)
{
        rad=a;
}
abstract void area();
}

class Rectangle extends Shape
{
        Rectangle(double a,double b)
        {
                super(a,b);
        }
        void area()
        {
                System.out.println("Area of rectangle is:"+(dim1*dim2));
        }
}
```

```
class Triangle extends Shape
{
	Triangle(double a,double b)
	{
		super(a,b);
	}
	void area()
	{
		System.out.println("Area of triangle is"+(dim1*dim2)/2);
	}
}

class Circle extends Shape
{
	Circle(double a)
	{
		super(a);
	}
	void area()
	{
		System.out.println("Area of circle is:"+(3.14*rad*rad));
	}
}

class AbstractMain{
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		System.out.println("Enter the dimensions of rectangle");
		double l=s.nextInt();
		double b=s.nextInt();
		System.out.println("Enter the base and height of traingle");
		double h1=s.nextInt();
		double h2=s.nextDouble();
		System.out.println("Enter the radius of circle");
		double r=s.nextInt();
		Shape sh;
		Rectangle rect=new Rectangle(l,b);
		Triangle tri=new Triangle(h1,h2);
		Circle cir=new Circle(r);
		sh=rect;
```

```
                sh.area();
                sh=tri;
                sh.area();
                sh=cir;
                sh.area();
}
}
```

PROGRAM-05

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.

```
import java.util.*;
class Account{
String name;
int accno;
```

```
String acc_type;
double balance;

Account(String name,int accno,String acc_type,double balance)
{
	this.name=name;
	this.accno=accno;
	this.acc_type=acc_type;
	this.balance=balance;
}

void deposit(double amount)
{
	balance+=amount;
}

void withdraw(double amount)
{
	if((balance-amount)>=0)
	{
		balance-=amount;
	}
	else
	{
		System.out.println("Insufficient balance! Cannot withdraw");
	}
}

void display()
{
	System.out.println("Name:"+" "+name+"Account number:"+" "+accno+"Account type:"+"
"+acc_type+"Balance:"+" "+balance);
}
}

class Sav_acct extends Account
{
	static double rate=5.0;
	Sav_acct(String name,int accno,double balance)
{
	super(name,accno,"Savings",balance);
}
```

```
void interest()
{
	balance+=balance*(rate)/100;
	System.out.println("Balance:"+balance);
}
}

class Curr_account extends Account
{
	private double minBal=500;
	private double serviceCharges=50;
	Curr_account(String name,int accno,double balance)
	{
	super(name,accno,"Current",balance);
	}

void checkMin()
{
	if (balance<minBal)
	{
		System.out.println("Balance is less than minimum, penlaty is imposed"+serviceCharges);
		balance-=serviceCharges;
		System.out.println("Balance is:"+balance);
	}
}
}

class Bank
{
public static void main(String[] args)
	{
		Scanner s=new Scanner(System.in);
		System.out.println("Enter customer name");
		String name=s.next();
		System.out.println("Enter account number");
		int accno=s.nextInt();
		System.out.println("Enter the type of account-Current or Savings");
		String acc_type=s.next();
		System.out.println("Enter the initial balance");
		double balance=s.nextDouble();
		Account ob1=new Account(name,accno,acc_type,balance);
```

```
Sav_acct sa=new Sav_acct(name,accno,balance);
Curr_account cu=new Curr_account(name,accno,balance);
while (true)
{
        if(acc_type.equals("Savings"))
        {
                System.out.println("Menu");
                System.out.println("1.Deposit");
                System.out.println("2.Withdraw");
                System.out.println("3.Compute Interest for savings account");
                System.out.println("4.Display account details");
                System.out.println("5.Exit");
                System.out.println("Enter your choice:");
                int choice=s.nextInt();

switch(choice)
{
        case 1:
        {
                System.out.println("Enter the amount to be deposited");
                double amt1=s.nextDouble();
                sa.deposit(amt1);
                break;
        }
        case 2:
        {
                System.out.println("Enter the amount to be withdrawn");
                double amt2=s.nextDouble();
                sa.withdraw(amt2);
                break;
        }
        case 3:
        {
                sa.interest();
                break;
        }
        case 4:
        {
                sa.display();
                break;
        }
        case 5:
```

```
                        {
                                break;
                        }
                        default:
                        {
                                System.out.println("Enter valid input");
                                break;
                        }
                }

}
else
{
System.out.println("Menu");
System.out.println("1.Deposit");
System.out.println("2.Withdraw");
System.out.println("3.Display account details");
System.out.println("4.Exit");
System.out.println("Enter your choice:");
int choice=s.nextInt();
switch(choice)
                {
                        case 1:
                        {
                                System.out.println("Enter the amount to be deposited");
                                double amt1=s.nextDouble();
                                cu.deposit(amt1);
                                break;
                        }
                        case 2:
                        {
                                System.out.println("Enter the amount to be withdrawn");
                                double amt2=s.nextDouble();
                                cu.withdraw(amt2);
                                break;
                        }
                        case 3:
                        {
                                cu.display();
                                cu.checkMin();
                                break;
                        }
```

```
                    case 4:
                    {
                            break;
                    }
                    default:
                    {
                            System.out.println("Enter valid input");
                            break;
                    }
            }

}
}
}
}
```

PROGRAM-06

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

```
//Student.java
package CIE;

import java.util.Scanner;

public class Student {

protected String usn = new String();
protected String name = new String();
protected int sem;

public void inputStudentDetails()
{
Scanner sc=new Scanner(System.in);
```

```
System.out.println(&quot;Enter student usn&quot;);
usn=sc.next();
System.out.println(&quot;Enter student name&quot;);
name=sc.next();
System.out.println(&quot;Enter student semester&quot;);
sem=sc.nextInt();
}

public void displayStudentDetails() {
System.out.println(&quot;student usn:&quot;+ usn);
System.out.println(&quot;student name:&quot;+name);
System.out.println(&quot; student sem:&quot;+ sem);
}
}

//Internals.java
package CIE;
import java.util.Scanner;

public class Internals extends Student {

protected int marks[] = new int[5];

public void inputCIEmarks()
{
Scanner sc=new Scanner(System.in);
for (int i=0;i&lt;5;i++)
{
System.out.println(&quot;Enter 5 subject marks&quot;);
marks[i]=sc.nextInt();
}
}
}

//Externals.java
package SEE;

import CIE.Internals;

import java.util.Scanner;

public class Externals extends Internals {
```

```
protected int marks[];

protected int finalMarks[];

public Externals() {

marks = new int[5];
finalMarks = new int[5];
}

public void inputSEEmarks()
{

Scanner sc = new Scanner(System.in);

for(int i=0;i&amp;lt;5;i++)
{

System.out.print(&amp;quot;Subject &amp;quot;+(i+1)+&amp;quot; marks: &amp;quot;);

marks[i] = sc.nextInt();
}
}

public void calculateFinalMarks() {

for(int i=0;i&amp;lt;5;i++)

finalMarks[i] = marks[i]/2 + super.marks[i];

}

public void displayFinalMarks() {

displayStudentDetails();

for(int i=0;i&amp;lt;5;i++)

System.out.println(&amp;quot;Subject &amp;quot; + (i+1) + &amp;quot;: &amp;quot; + finalMarks[i]);

}
```

```
}

//Main1.java
import SEE.Externals;

class Main1 {

public static void main(String args[])

{

int numOfStudents = 2;

Externals finalMarks[] = new
Externals[numOfStudents];

for(int i=0;i&lt;numOfStudents;i++)

{

finalMarks[i] = new Externals();
finalMarks[i].inputStudentDetails();

System.out.println(&quot;Enter CIE marks&quot;);

finalMarks[i].inputCIEmarks();

System.out.println(&quot;Enter SEE marks&quot;);

finalMarks[i].inputSEEmarks();

}
System.out.println(&quot;Displaying data:\n&quot;);

for(int i=0;i&lt;numOfStudents;i++)

{

finalMarks[i].calculateFinalMarks();

finalMarks[i].displayFinalMarks();
```

```
} //end of for loop

} // end of public main

} //end of class main
```

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

```
import java.util.*;
class WrongAge extends Exception
{
	public WrongAge(String s)
	{
		super(s);
	}
}

class Father
{
	Scanner sc=new Scanner(System.in);
	public int F_age;
	public Father() throws WrongAge
	{

		System.out.println("Enter Father's age");
		F_age=sc.nextInt();
		if (F_age<0)
		{
			throw new WrongAge("Age cannot be negative");
		}
	}
	public void display()
	{
```

```
		System.out.println("Father's age="+F_age);
	}
}

class Son extends Father
{
	private int S_age;
	public Son() throws WrongAge
	{
		System.out.println("Enter son's age");
		S_age=sc.nextInt();
		if (S_age>F_age)
		{
			throw new WrongAge("Son's age cannot be greater than father's age");
		}
		else if (S_age<0)
		{
			throw new WrongAge("Age cannot be negative");
		}
		else if (S_age==F_age)
		{
			throw new WrongAge("Age cannot be same");
		}
		else
		{
			System.out.println("Valid age");
		}
	}
	public void display()
	{
		System.out.println("Father's age="+F_age);
		System.out.println("Son's age="+S_age);
	}
}


class Lab7
{
	public static void main(String args[])
	{

		try
		{
			Son son=new Son();
			son.display();

		}
		catch(WrongAge e)
```

```
                {
                        System.out.println("Exception caught");
                        System.out.println(e.getMessage());
                }
        }
}
```

PROGRAM-08

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

```
class BMS extends Thread{
        public void run(){
                for(int i=1;i<=5;i++){
                        System.out.println("bms college of engineering"+i);
                        try
                        {
                                Thread.sleep(10000);
                        }
                        catch(InterruptedException e){
                                e.printStackTrace();
                        }
                        }
                }
}
class CSE extends Thread{
        public void run(){
                for(int i=1;i<=10;i++){
                        System.out.println("Cse"+i);
                        try
                        {
                                Thread.sleep(2000);
                        }
                        catch(InterruptedException e){
                                e.printStackTrace();
                        }
                        }
                }
}
```

```
class threadMain{
	public static void main(String a[]){
		BMS obj1=new BMS();
		CSE obj2=new CSE();
		obj1.start();
		obj2.start();
	}
}
```

PROGRAM-09

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.

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

public class DivisionMain1 extends Frame implements ActionListener
{
	TextField num1,num2;
	Button dResult;
	Label outResult;
	String out="";
	double resultNum;
	int flag=0;

	public DivisionMain1()
	{
		setLayout(new FlowLayout());

		dResult = new Button("RESULT");
		Label number1 = new Label("Number 1:",Label.RIGHT);
```

```
        Label number2 = new Label("Number 2:",Label.RIGHT);
        num1=new TextField(5);
        num2=new TextField(5);
        outResult = new Label("Result:",Label.RIGHT);

        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);

        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);
        addWindowListener(new WindowAdapter()
        {
                public void windowClosing(WindowEvent we)
                {
                        System.exit(0);
                }
        });
}

public void actionPerformed(ActionEvent ae)
{
        int n1,n2;
        try
        {
                if (ae.getSource() == dResult)
                {
                        n1=Integer.parseInt(num1.getText());
                        n2=Integer.parseInt(num2.getText());

                        /*if(n2==0)
                                throw new ArithmeticException();*/
                        out=n1+" "+n2;
                        resultNum=n1/n2;
                        out+=String.valueOf(resultNum);
                        repaint();

                }
```

```
                }
                catch(NumberFormatException e1)
                {
                        flag=1;
                        out="Number Format Exception!  "+e1;
                        repaint();
                }
                catch(ArithmeticException e2)
                {
                        flag=1;
                        out="Divide by 0 Exception!  "+e2;
                        repaint();
                }

        }

        public void paint(Graphics g)
        {
                if(flag==0)

g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outResult.getHeight()-8);
                else
                g.drawString(out,100,200);
                flag=0;
        }

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

}
```

PROGRAM-10

10.A)Demonstrate Inter process Communication and deadlock

```
class Q {
	int n;
	boolean valueSet = false;
	synchronized int get() {
		while(!valueSet){
			try {
				System.out.println("Consumer waiting");
				wait();
			} catch(InterruptedException e) {
			System.out.println("InterruptedException caught");
			}

			}
		System.out.println("Got: " + n);
		valueSet = false;
		System.out.println("Intimate Producer\n");
		notify();
		return n;
	}
	synchronized void put(int n) {
		while(valueSet){
			try {
				System.out.println("Producer waiting");
				wait();
				} catch(InterruptedException e) {
					System.out.println("InterruptedException caught");
				}

			}
		this.n = n;
		valueSet = true;
		System.out.println("Put: " + n);
		System.out.println("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<5) {
		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<5) {
			int r=q.get();
			System.out.println("consumed:"+r);
			i++;
		}
	}
}
class PCFixed {
	public static void main(String args[]) {
	Q q = new Q();
	new Producer(q);
	new Consumer(q);
	System.out.println("Press Control-C to stop.");
	}
}
```

10B)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); // get lock on a in this thread.
System.out.println("Back in main thread");
}
public void run() {
b.bar(a); // get lock on b in other thread.
System.out.println("Back in other thread");
}
public static void main(String args[]) {
new Deadlock();

}

}
```