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:

Shuvam Rajbanshi
1BM22CS275

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

INDEX

NAME : Shriram Rajbanshi STD : _____ SEC : E ROLL NO : _____

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	12/12/23	Lab Program-1 (Quadratic)		
2	19/12/23	Lab Program 2 (SGPA Calculator)		
3.	26/12/23	Lab Program 3 (Book Objects)		
04.	02/01/24	Area of Rectangle, Triangle, Circle		
05	16/01/24	Bank Account Class		
06	23/01/24	Strings and Generics		
07	23/01/24	CIE and SEE package		
08	30/01/24	Wrong Age		
09	06/02/24	Thread		
10	13/02/24	IPC and deadlock		
11	20/02/24	Division of Integers		

Date 12/12/2023
Page

Lab Program - 1

Q. 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.

Program:

```
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 of 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("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("Root2 = "+r1+"-i"+r2);
}
    }
}

Class Quadratic Main
{
    Public Static void main (String args[])
    {
        Quadratic q = new Quadratic();
        q.getd();
        q.compute();
    }
}
```

Output:

Enter the coefficients of a, b, c
2.3 4 5.6
Roots are Imaginary
Root 1 = -0.87 + 1.30i
Root 2 = -0.87 - 1.30i

Shuvam Rajbanshi
1BM22CS275

Enter the coefficients of a, b, c
1 3 2
Roots are real and distinct
Root 1 = -1.0 Root 2 = -2.0

Enter the coefficients of a, b, c
1 2 1
Roots are real and equal
Root 1 = Root 2 = -10

Shuvam Rajbanshi
1BM22CS275

Additional programs:

Q. To write a program in Java to find the area of a rectangle and verify same with various inputs (length / breadth).

```
class Rectangle Area{
public static void main (String args[]){
int length, breath;
length = ~~new~~ Integer.parseInt(args[0]);
breath = Integer.parseInt(args[1]);
int area = length * breath;
System.out.println("length of rectangle =" + length);
System.out.println("breath of rectangle =" + breath);
System.out.println("area of rectangle =" + area);
}
}
```

Output:

```
10 8
length of rectangle =10
breath of rectangle =8
area of rectangle =80
```

Date
Page

d. Scanner Class

```
import java.util.Scanner;
Class HelloWorld {
public static void main (String args[])
{
int a; float b; String s;
Scanner in = new Scanner (System.in);
System.out.println ("Enter a string");
s = in.nextLine();
System.out.println ("You entered string " +s);
System.out.println ("Enter an integer");
a = in.nextInt();
System.out.println ("You entered integer " +a);
System.out.println ("Enter a float");
b = in.nextFloat();
System.out.println ("You entered a float " +b);
}
}
```

Output:

```
Enter a string
java
You entered string java
Enter an integer
67
You entered integer 67
Enter a float
7.6
You entered float 7.6
```

10

Sun
12/12/23

19/12/23

LAB-2

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

```
import java.util.Scanner;

class Subject {
    int subjectMarks;
    int credits;
    int grade;
}

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

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

```
public void getStudentDetails () {
    System.out.println ("Enter Student name : ");
    name = s.nextLine ();

    System.out.println ("Enter student USN : ");
    usn = s.nextLine();
}

public void getMarks () {
  for (int i=0 ; i<8; i++) {
    System.out.println ("Enter marks for Subject " +(i+1));
    Subject[i].credits = s.nextInt();
    subject[i].subjectMarks = s.nextInt ();

    while ( subject[i].subjectMarks < 0 || subject[i].subjectMarks
        > 100) {
      System.out.println ("Invalid marks. Please enter
        marks between 0 and 100.");
      subject[i].subjectMarks = s.nextInt();
    }

    System.out.println ("Enter credits for Subjects " + (i+1));
    subject[i].credits = s.nextInt();

    if ( subject[i].subjectMarks >= 90 ) {
      subject[i].grade = 10;
    }
    else if ( subject[i].subjectMarks >= 80) {
      subject[i].grade = 9;
    }
    else if ( subject[i].subjectMarks >= 70 ) {
      subject[i].grade = 8;
    }
```

```
        else if (subject[i].subjectMarks >= 60.){
            subject[i].grade = 7;
        }
        else if (subject[i].subjectMarks >= 50){
            subject[i].grade = 6;
        }
        else if (subject[i].subjectMarks >= 40.){
            subject[i].grade = 5;
        }
        else {
            subject[i].grade = 0;
        }
    }
}

public void computeSGPA(){
    double totalCredits = 0;
    double weightedSum = 0;

    for (int i=0; i<8; i++){
        totalCredits += subject[i].credits;
        weightedSum += subject[i].grade * subject[i].credits;
    }

    if (totalCredits == 0){
        SGPA = 0;
    }
    else{
        SGPA = weightedSum / totalCredits;
    }
}
}
```

```
public class StudentTest {
    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 ("SGPA : "+ s1.SGPA);
    }
}
```

Output :

```
Enter name : shivam
Enter usn : 1bm22cs275
Enter marks and credits for subject 1 :
Marks : 80
Credits : 9
Enter marks and credits for subject 2:
Marks : 80
Credits : 9
Enter marks and credits for subject 3:
Marks : 80
Credits : 9
Enter marks and credits for subject 4:
Marks : 60
Credits : 7
Enter marks and credits for subject 5:
```

Marks : 60
Credits : 7
Enter marks and credits for subject 6:
Marks : 70
Credits : 8
Enter marks and credits for subject 7:
Marks : 70
Credits : 8
Enter marks and credits for subject 8:
Marks : 60
Credits : 7

Result:

Name : Shivam
USN : 1bm22cs275
SGPA : 8.421875

(10)

19/12/23

Date 26/12/23
Page

LAB -3

Q. 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 Books {
    String name;
    String author;
    int price;
    int numPages;
}

public String toString() {
Books (String name, String author, int price, int numPages) {
    this.name = name;
    this.author = author;
    this.price = price;
```

```
import java.util.Scanner;

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

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

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

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

        int n;
        String name;
        String author;
        int price;
        int numPages;
```

```
System.out.print(s: "Enter the number of books:");
n = s.nextInt();
Books b[];
b = new Books[n];

for(int i=0; i<n; i++){
System.out.println("\n Enter details for Book" +
  (i+1) + ":");
System.out.print(s: "Name:");
name = s.next();
System.out.print(s: "Author:");
author = s.next();
System.out.print(s: "Price:");
price = s.nextInt();
System.out.print(s: "Number of pages:");
numPages = s.nextInt();

b[i] = new Books(name, author, price, numPages);
}

System.out.println(x: "\n Details of the Books:");
for(int i=0; i<n; i++){
   System.out.println("\n Book" + (i+1) + ":\n" + b[i]);
}

s.close();
}
}
```

Output:

Enter the number of books : 2
Enter details for Book 1 :
Name : Java Programming
Author : Sumit
Price : 20
Number of Pages : 300

Enter details for Book 2 :
Name : Data Science
Author : Sidhwin
Price : 25
Number of Pages : 350

Details of the Books :
Book 1 :
Book name : Java Programming
Author name : Sumit
Price : 25
Number of Pages : 300
Book 2 :
Book name : Data Science
Author name : Sidhwin
Price : 25
Number of pages : 350

26/12/23

Date 02/01/24
Page

LAB - 4

Q. 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 contains only the method printArea() that prints the area of the given shape.

```
import java.util.Scanner;

Class InputScanner {
    public static Scanner scanner = new Scanner(System.in);
}

abstract class shape extends InputScanner {
    protected int side1;
    protected int side2;

    abstract void printArea();
}

Class Rectangle extends shape {
    @ Override
    void printArea(){
        System.out.println("Enter length and width of the rectangle:");
        side1 = scanner.nextInt();
        side2 = scanner.nextInt();
        System.out.println("Area of Rectangle:" + (side1 * side2));
    }
}
```

```
class Triangle extends Shape {
    @Override
    void printArea() {
        System.out.println("Enter base and height of the triangle:");
        side1 = scanner.nextInt();
        side2 = scanner.nextInt();
        System.out.println("Area of Triangle: " + 0.5 * side1 * side2);
    }
}

class Circle extends Shape {
    @Override
    void printArea() {
        System.out.println("Enter radius of the circle:");
        side1 = scanner.nextInt();
        System.out.println("Area of Circle: " + (Math.PI * side1 * side1));
    }
}

public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        Triangle triangle = new Triangle();
        Circle circle = new Circle();

        rectangle.printArea();
        System.out.println();
        triangle.printArea();
        System.out.println();
        circle.printArea();
    }
}
```

Date
Page

Output:

Enter length and width of the rectangle:
20
40
Area of Rectangle : 800

Enter base and height of the triangle:
20
25
Area of Triangle : 250.0

Enter radius of the circle:
200
Area of circle : 125663.7061

LAB - 5

Q. Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called saving account and the other current account. The saving account provides compound interest and withdrawal facilities but no check 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.

```
import java.util.Scanner;

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

        SavAcct savingsAccount = new SavAcct("John Doe", 1001);
        processAccount(savingsAccount, scanner);

        CurAcct currentAccount = new CurAcct("Jane Doe", 2001);
        processAccount(currentAccount, scanner);

        scanner.close();
    }

    private static void processAccount(Account account, Scanner scanner) {
        account.displayBalance();

        System.out.println("Enter the amount to deposit:");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);
```

```
if (account instanceof SavAcct) {
    ((SavAcct) account).computeInterest();
}

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

if (account instanceof CurAcct) {
    ((CurAcct) account).checkMinimumBalance();
}

account.displayBalance();
}
}

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

public Account(String customerName, int accountNumber) {
this.customerName = customerName;
this.accountNumber = accountNumber;
}

public void deposit(double amount) {
balance += amount;
System.out.println("Deposit of " + amount + " successful.");
}
```

```
public void withdraw (double amount) {
    if (amount <= balance) {
        balance -= amount;
        System.out.println("Withdraw of amount" + amount);
    } else {
        System.out.println("Insufficient funds for withdrawal");
    }
}

public void displayBalance() {
    System.out.println("Account Number:" + accountNumber);
    System.out.println("Customer Name:" + customerName);
    System.out.println("Balance:" + balance);
}
}

class SavAct extends Account{
    private double interestRate = 0.05;

    public SavAct(String customerName, int accountNumber) {
        super(customerName, accountNumber);
    }

    public void computeInterest() {
        double interest = balance * interestRate;
        deposit(interest);
        System.out.println("Interest of computed and deposited");
    }
}
```

Date
Page

```
Class CurAcct extends Account {
    private double minimumBalance = 1000.0;
    private double serviceCharge = 10.0;

    public CurAcct (String customerName, int accountNumber) {
        super (customerName, accountNumber);
    }

    public void checkMinimumBalance () {
        if (balance < minimumBalance) {
            balance -= serviceCharge;
            System.out.println("Minimum balance not maintained.
            service charge imposed.");
        } else {
            System.out.println("Minimum balance maintained.");
        }
    }
}
```

Output:

```
AccountNumber : 1001
Customer Name : John Doe
Balance : 0.0
Enter the amount to deposit;
5000
Deposit of 5000.0 successful.
Interest of 200.0 computed and deposited
Enter the amount to withdrawal:
2000
Withdrawal of 2000.0 successful
```

Account Number : 1001
Customer Name : John Doe
Balance : 3250.0

Date 16/01/2024
Page

LAB-6

Q. Demonstrate various string constructor with proper java programs.

- String Literal:

```
String str1 = "Hello, World!";
System.out.println("String 1: " + str1);
```

Output: String 1: Hello, World!

- Using new keyword:

```
char[] charArray = {'H', 'e', 'l', 'l', 'o'};
String str2 = new String(charArray);
System.out.println("String 2: " + str2);
```

Output: String 2: Hello

- Using char array and specifying the range:

```
char[] charArray2 = {'W', 'o', 'r', 'l', 'd'};
String str3 = new String(charArray2, 1, 3);
System.out.println("String 3: " + str3);
```

Output: String 3: orl

- Using 'byte' array:

```
byte[] byteArray = {72, 101, 108, 108, 111};
String str4 = new String(byteArray);
System.out.println("String 4: " + str4);
```

Output: String 4: Hello

- Using 'StringBuilder'

```
StringBuilder stringBuilder = new StringBuilder("Hello");
stringBuilder.append(", ").append("Java");
String str5 = stringBuilder.toString();
System.out.println("String 5: " + str5);
```

Output: String 5: Hello, Java

- Using 'StringBuffer'

```
StringBuffer stringBuffer = new StringBuffer("Hello");
stringBuffer.append(", ").append("World");
String str6 = stringBuffer.toString();
System.out.println("String 6: " + str6);
```

Output: String 6: Hello, World

d. Demonstrate string length, string literal, string concat

- ~~String~~ String length:

```
public class StringLengthDemo {
    public static void main(String[] args) {
        String str = "Hello, World!";
        int length = str.length();
        System.out.println("Length of the string: " + length);
    }
}
```

Output:

Length of the string: 13

Date
Page

- String ~~literals~~ Concat

```
public class StringLiteralDemo {
    public static void main (String [] ~~args~~ args) {
        String str1 = "Hello";
        String str2 = "World";
        String result = str1 + ", " + str2;
        System.out.println("Concatenated String : "+ result);
    }
}
```

Output: Concatenated String : Hello, World

- String literal:

```
public class StringLiteralDemo {
    public static void main (String [] args) {
        String str = "Hello, World!";
        System.out.println (str);
    }
}
```

Output : Hello ~~world~~ World!

Q. Write a Java program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

⇒

```
abstract class Shape {
    abstract double calculateArea();
    abstract double calculatePerimeter();
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    double calculateArea() {
        return Math.PI * radius * radius;
    }

    double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}

class Triangle extends Shape {
    private double side1, side2, side3;
```

```
public Triangle (double side1, double side2, double side3){
    this. side1 = side1;
    this. side2 = side2;
    this. side3 = side3;
}

double calculateArea () {
    double s = (side1 + side2 + side3)/2;
    return Math. sqrt (s * (s - side1) * (s - side2) * (s - side3));
}

double calculatePerimeter () {
    return side1 + side2 + side3;
}
}

publer class ShapeTest {
    publer static void main (String [] args) {
    Circle circle = new Circle (5);
    Triangle triangle = new Triangle (3, 4, 5);

    System. out. println (" Circle Area : " + circle. calculateArea()
    + ", Perimeter : " + circle. calculatePerimeter ());
    System. out. println (" Triangle Area : " + triangle. calculateArea()
    + ", Perimeter : " + triangle. calculatePerimeter ());
    }
}
```

Output:

Circle Area : 78.53981 Perimeter : 31.415926

Triangle Area : 6.0 , Perimeter : 12.0

Q write a Java program to create a generic class Stack which hold 5 integers and 5 double values.

```
class Stack <T> {
    private Object [] stackArray;
    private int top;
    private int maxSize;

    public Stack (int maxSize) {
        this.maxSize = maxSize;
        this.stackArray = new Object [maxSize];
        this.top = -1;
    }

    public void push (T element) {
        if (top < maxSize - 1) {
            stackArray [++top] = element;
            System.out.println ("Pushed : " + element);
        } else {
            System.out.println ("Stack is Full. Cannot push : " + element);
        }
    }

    public T pop () {
        if (top >= 0) {
            T element = (T) stackArray [top--];
            System.out.println ("Popped : " + element);
            return element;
        } else {
            System.out.println ("Stack is empty. Cannot pop.");
            return null;
        }
    }
}
```

```
public class GenericStackTest {
  public static void main(String[] args) {
    Stack<Integer> integerStack = new Stack<>(5);

    integerStack.push(10);
    integerStack.push(20);
    integerStack.push(30);

    integerStack.pop();
    integerStack.pop();
    integerStack.pop();
    integerStack.pop();

    Stack<Double> doubleStack = new Stack<>(5);

    doubleStack.push(10.5);
    doubleStack.push(20.7);
    doubleStack.push(30.2);

    doubleStack.pop();
    doubleStack.pop();
    doubleStack.pop();
    doubleStack.pop();
  }
}
```

Output:

```
Pushed : 10
Pushed : 20
Pushed : 30
Popped : 30
Popped : 20
Popped : 10
Stack is empty - Cannot pop.
```

Pushed : 10.5
Pushed : 20.7
Pushed : 30.2
Pushed : 30.2
Pushed : 20.7
Popped : 10.5
Stack empty. Cannot pop.

Thu 16/01/24

Date 23/01/2024
Page

LAB-6

Q. 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.

File ~~CIE~~ / Students.java

```
Package CIE;
import java.util.Scanner;

public class Students {
    protected String usn = new String();
    protected String name = new String();
    protected int sem;

public void inputStudentDetails() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter USN: ");
    usn = scanner.nextLine();
    System.out.print("Enter Name: ");
    name = scanner.nextLine();
    System.out.print("Enter Semester: ");
    sem = scanner.nextInt();
}
```

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

File: CIE/Internals.java

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

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

    public void inputCIEMarks(){
        Scanner scanner = new Scanner(System.in);

        for(int i=0; i<5; i++){
            System.out.print("Enter CIE marks for subject " + (i+1));
            marks[i] = scanner.nextInt();
        }
    }
}
```

File : SEE / 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 input() {
        Scanner scanner = new Scanner(System.in);

        for (int i=0; i<5; i++) {
            System.out.print("Enter Marks " + (i+1) + ":");
            marks[i] = scanner.nextInt();
        }
    }

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

    public void display() {
        displayStudentDetails();
```

```
        for (int i = 0; i<5; i++)
        System.out.println("Subject" + (i+1) + ": " + finalMarks[i]);
    }
}
```

File: Main.java

```
import see.Externals;

class markmain {
    public static void main(String args[])
    {
        int numofStudents = 2;
        Externals finalMarks[] = new Externals[numofStudents];

        for (int i=0; i<numofStudents; i++){
            finalMarks[i] = new Externals();
            finalMarks[i].inputStudentDetails();
            System.out.println("Enter CIE marks");
            finalMark[i].inputCIEMarks();
            System.out.println("Enter SEE marks:");
            finalMarks[i].inputSEEmarks();
        }
        System.out.println("Displaying data:\n");
        for (int I=0; i<numberofStudents; i++){
            finalMarks[i].calculateFinalMarks();
            finalMark[i].displayFinalMarks();
        }
    }
}
```

Date
Page

Output:

Enter USN: 12
Enter name: Sumit
Enter sem: 3
Enter CIE marks
Enter internal marks for Sumit
Subject 1 marks: 33
Subject 2 marks: 23
Subject 3 marks: 22
Subject 4 marks: 33
Subject 5 marks: 11
Enter SEE marks for Sumit
Subject 1 marks: 89
Subject 2 marks: 90
Subject 3 marks: 99
Subject 4 marks: 88
Subject 5 mark: 89

Displaying Data:

USN: 12
Name: Sumit
Semester: 3
Subject 1: 67
Subject 2: 78
Subject 3: 71
Subject 4: 77
Subject 5: 55

30/01/2024

LAB-7

Q. Write a program that demonstrates handling of ~~ext~~ 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.

File : Father.java

```
import java.util.Scanner;

Class WrongAge extends Exception {
  public WrongAge (String e) {
      super(e);
  }
}

Class InputScanner {
   Scanner s = new Scanner(System.in);
}

Class Father extends InputScanner {
   int fatherAge;

   public Father() throws WrongAge {
      System.out.println(x: "Enter Father's age:");
      fatherAge = s.nextInt();
```

```
        if (fatherAge < 0) {
            throw new WrongAge(e:"Age cannot be negative");
        }
    }
    public void display() {
        System.out.println("Father's age: " + fatherAge);
    }
}

Class Son extends Father {
    int sonAge;
    public Son() throws WrongAge {
        super();
        System.out.println(x:"Enter Son's age:");
        sonAge = s.nextInt();
        if (sonAge >= fatherAge) {
            throw new WrongAge(e:"Son's age cannot be greater than
            father");
        } else if (sonAge < 0) {
            throw new WrongAge(e:"Age cannot be negative");
        }
    }
}
```

File: Main.java

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

Output:

1. Enter Father's age:
45
Enter Son's age:
21
Father's age: 45
Son's age: 21

2. Enter Father's age:
21
Enter Son's age:
33
Error: "Son's age cannot be greater than father's age

3. Enter Father's age:
-21
Error: Age cannot be negative

4. Enter father's age:
20
Enter Son's age:
20
Error: Son's age cannot be equal to father's age.

30.01.24

Date
Page

LAB8:

WAP which creates 2 threads, one thread displaying "BMS College of Engineering" once every 10 seconds & another displaying "CSE" once every 2 seconds.

```
import java.lang.*;

Class DisplayMessageThread extends Thread {
    private final String message;
    private final long interval;

    DisplayMessageThread (String message, long interval){
        this. message = message;
        this. interval = interval;
    }.

    public void run (){
        Try {
            while (true) {
                System. out. println (message);
                Thread. sleep (interval);
            }
        } catch (InterruptException e) {
            System. out. println (Thread. currentThread().
                getName() + " Interrupted ");
        }
    }
}
public class Main {
    public static void Main (String [] args) {
        DisplayMessageThread t1 = new DisplayMessage;
                Thread ("BMS College of Engg", 10000);
```

```
DisplayMessageThread t2 = new DisplayMessageThread("CSE", 2000);
t1.setName("FirstThread");
t2.setName("SecondThread");
t1.start();
t2.start();

try{
    Thread.sleep(30000);
}
catch(InterruptException e){
    System.out.println("Main Thread interrupted");
}
t1.interrupt();
t2.interrupt();

System.out.println("Main Thread exiting!");
}
}
```

Output:

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
CSE
CSE
BMSCE
```

CSE
CSE
CSE
CSE
CSE
Main Thread exiting
Second Thread interrupted
First Thread interrupted.

LAB - 10

Demonstrate inter process communications & deadlock

```
Class Q {
    int n;
    boolean valueset = false;

    while(! Valueset ){
     Try {
        System. out. println ("Consumer waiting");
        wait ();
     }
     Catch ( InterruptException e) {
        System. out. println ("Interrupt Exception caught");
     }
  }
    System. out. println (" Got:" +n);
    Valueset = false;
    System. out. println ("Intimate producer");
     notify();
    return n;
}
Synchronized void put (int n) {
    while ( Valueset){
      Try {
        System. out. println ("Producer waiting");
        wait();
      }
      Catch( InterruptException e){
        System. out. println (" InterruptException caught");
      }
}
```

```
this.n = n;
valueSet = true;
System.out.println("Put": +n);
System.out.println("Intimate consumer");
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();
            i++;
        }
    }
}
```

```
this.n = n;
valueSet = true;
System.out.println("Put": +n);
System.out.println("Intimate 'consumer");
notify();
}
}

Class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        newThread(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;
    newThread(this, "Consumer").start();
    }
    Public void run() {
        int i = 0;
        while(i < 15) {
            int r = q.get();
            i++;
        }
    }
}
```

```
class PCfixed {
    public static void main (String[] args) {
        Q q = new Q(1);
        new Producer(q);
        new Consumer(q);
        System.out.println("Press control to stop");
    }
}
```

Output:

```
Put : 0
Got : 0
Put : 1
Got : 1
Put : 2
Got : 2
```

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("Main Thread");
        Thread t = new Thread(this, "Racing Thread");
        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();
}
}
```

Date
Page

Output.

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

LAB - 10:

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 Division button is clicked. If Num1 and Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw and ArithmeticException Display the exception in a message dialog box.

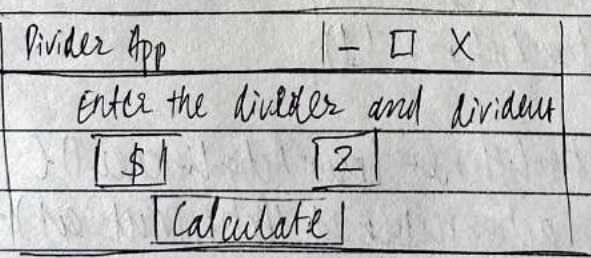
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel jlab = new JLabel("Enter the divisor and dividend:");

        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        JButton button = new JButton("Calculate");
```

```
JTable err = new JTable();
JTable alab = new JTable();
JTable blab = new JTable();
JTable anslab = new JTable();

jfrm.add(err);
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener(){
  public void actionPerformed(ActionEvent ae){
    System.out.println("Action event from a text field");
  }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);

button.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent ae){
    try{
      int a = Integer.parseInt(ajtf.getText());
      int b = Integer.parseInt(bjtf.getText());
      int ans = a/b;

      alab.setText("\n A = "+a);
      blab.setText("\n B = "+b);
      anslab.setText("\n Ans = "+ans);
    }
```

```
        catch (NumberFormatException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("Enter Only Integers!");
        }
        catch (ArithmeticException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("B should be NON zero!");
        }
    }
});

jfrm.setVisible(true);
}
```

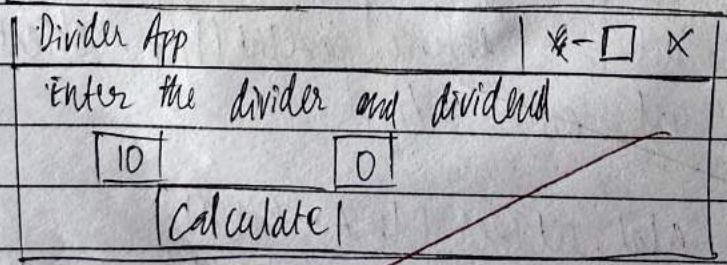
Output:

:)



→ Enter Only Integers!

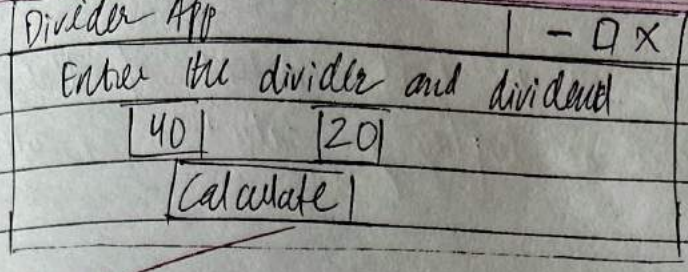
ii)



→ B should be NON zero!

Date
Page

iii)



→ A = 40 B = 20 Ans = 2

20.03.24

The main function and components used in the program.

1. JFrame:
Represents the main window of the application where all the GUI components are contained.

2. JLabel:
Used to display text label in the GUI, such as instruction and error message.

3. JTextField:
Provide an input field where users can enter numeric values for calculations.

4. JButton:
Represent a clickable button that triggers an action when pressed.

5. ActionListener:
Interface used to handle action events generated by user interactions, such as button clicks or text field input.

6. addActionListener():
Method used to add action listeners to swing components, such as buttons or text fields, to listen for user actions.

7. setText();
Method used to set the text content of JLabel components dynamically, such as displaying calculation results or error message.

8. ParseInt();
Static method of the Integer class used to parse a string into an integer value. It's used to convert the text entered in the text fields into integer values for calculation.

9. Arithmetic Exception:
Exception thrown when an arithmetic operation fails, such as division by zero.

10. setVisible();
Method used to make the JFrame or any other Swing component visible on the screen. It takes a boolean parameter where true makes the component visible and false hides it.

20.02.24

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

```
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("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();
 }
}
```

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

```
import java.util.Scanner;

class Subject
{
	int subMarks;
	int credits;
	int grade;
}

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

	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 your name: ");
		name = s.next();
		System.out.println("Enter your USN: ");
		usn = s.next();
	}

	void getMarks()
	{
		for(i=0;i<8;i++)
		{
			System.out.println("Enter marks of subject " + (i+1) + ": ");
			subject[i].subMarks = s.nextInt();
			System.out.println("Enter credits of subject " + (i+1) + ": ");
			subject[i].credits = s.nextInt();
			subject[i].grade = (subject[i].subMarks/10) + 1;
		}
	}
	void computeSGPA()
	{
		int effscore = 0;
		int totalcreds = 0;
		for(i=0;i<8;i++)
		{
			effscore += subject[i].grade * subject[i].credits;
			totalcreds += subject[i].credits;
		}
		SGPA = (double)effscore/(double)totalcreds;
	}
}

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

```
                System.out.println("SGPA: " + s1.SGPA);
        }
}
```

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

```
import java.util.Scanner;

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

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

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

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

```
                String name;
                String author;
                int price;
                int numPages;

                System.out.println("Enter number of books: ");
                n = s.nextInt();

                Books b[];
                b = new Books[n];

                for(i = 0; i < n; i++)
                {
                        System.out.println("Enter name of book: ");
                        name = s.next();
                        System.out.println("Enter author of book: ");
                        author = s.next();
                        System.out.println("Enter price of book: ");
                        price = s.nextInt();
                        System.out.println("Enter number of pages: ");
                        numPages = s.nextInt();
                        b[i] = new Books(name,author,price,numPages);
                }

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

        }
}
```

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 ofthe given shape.

```
import java.util.Scanner;

class InputScanner{
int d1, d2;
Scanner sc = new Scanner(System.in);
InputScanner(){
```

```
if(this.getClass() == Circle.class){
System.out.println("Enter radius of circle: ");
d1 = sc.nextInt();
}
else{
System.out.println("Enter height and weight: ");
d1 = sc.nextInt();
d2 = sc.nextInt();
}
}
}

abstract class Shape extends InputScanner{
abstract void printArea();
}

class Triangle extends Shape{
void printArea(){
System.out.println("Area of triangle is: " + (double)(d1*d2)/2);
}
}

class Rectangle extends Shape{
void printArea(){
System.out.println("Area of rectangle is: " + (double)(d1*d2));
}
}


class Circle extends Shape{
void printArea(){
System.out.println("Area of circle: " + (double)(3.14*d1*d1));
}
}

class AreaMain{
public static void main(String args[]){
Rectangle r = new Rectangle();
Triangle tr = new Triangle();
Circle c = new Circle();
r.printArea();
tr.printArea();
c.printArea();
}
```

}

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.

```
import java.util.*;
import java.lang.Math;
abstract class Bank{
	abstract void withdraw(double amt);
	abstract void deposit(double amt);
	abstract void display();
	abstract void menudisp();
}

class Account extends Bank{
	String name;
	int acc_num;
	String type;
	double bal;
	String menu = " ";
	Account(String name, int acc_num, String type, double bal, String menu){
		this.name = name;
		this.acc_num = acc_num;
		this.type = type;
		this.bal = bal;
		this.menu = menu;
	}
	public void withdraw(double amt){
		if(amt>bal){
			System.out.println("Withdraw declined! Max amount you can withdraw is:
" + bal);
		}
```

```
                else{
                        bal -= amt;
                        System.out.println("Updated balance is: " + bal);
                }
        }

        public void deposit(double amt){
                bal += amt;
                System.out.println("Updated balance is: " + bal);
        }


        public void display(){
                System.out.println("Account number: " + name);
                System.out.println("Account name: " + acc_num);
                System.out.println("Account type: " + type);
                System.out.println("Balance: " + bal);
        }
        public void menudisp(){
                menu = "---------MENU---------\n1. Deposit\n2. Withdraw\n3. Display";
        }
}

class Savings extends Account{
        double interest;
        Savings(String name, int acc_num, String type, double bal,String menu, double interest){
                super(name, acc_num, type, bal,menu);
                this.interest = interest;
        }
        public double interest(int time){
                double comp;
                comp = bal + Math.pow((bal*(1+(interest/100))),time);
                return comp;
        }
        public void menudisp(){
                super.menudisp();
                menu += "\n4. Compute Interest\n5. Exit";
        }
}

class Current extends Account{
        double minbal = 10000;
        Current(String name, int acc_num, String type, double bal, String menu, double minbal){
                super(name, acc_num, type, bal,menu);
```

```
		this.minbal = minbal;
	}

	public void menudisp(){
		super.menudisp();
		menu += "\n4. Cheque Book\n5. Exit";
	}

	public void withdraw(double amt){
		if(amt>bal){
			System.out.println("Withdraw declined! Max amount you can withdraw is:
" + bal);
		}
		else{
			bal -= amt;
			System.out.println("Updated balance is: " + bal);
		}
	}

}

class BankMain{
	public static void main(String args[]){
		Scanner s = new Scanner(System.in);
		String name;
		String menu = " ";
		int acc_num;
		String type;
		double bal = 0;
		double interest;
		int choice;
		int time;
		double money;
		System.out.println("Enter customer name: ");
		name = s.next();
		System.out.println("Enter account number: ");
		acc_num = s.nextInt();
		System.out.println("---------ACCOUNT TYPE---------\n1.Savings
Account\n2.Current Account\nPlease select account type: ");
		type = s.next();

		if(type.equals("savings")){
			System.out.println("Enter interest amount: ");
			interest = s.nextDouble();
```

```
                        Savings accs = new Savings(name, acc_num, type, bal, menu, interest);
                        do{
                                accs.menudisp();
                                System.out.println(accs.menu);
                                System.out.println("Enter choice: ");
                                choice = s.nextInt();
                                switch(choice){
                                        case 1:
                                                System.out.println("Enter amount to be deposited:
");
                                                money = s.nextDouble();
                                                accs.deposit(money);
                                                break;
                                        case 2:
                                                System.out.println("Enter amount to be withdrawn:
");
                                                money = s.nextDouble();
                                                accs.withdraw(money);
                                                break;
                                        case 3:
                                                accs.display();
                                                break;
                                        case 4:
                                                System.out.println("Enter time to calculate interest
in years: ");
                                                time = s.nextInt();
                                                money = accs.interest(time);
                                                System.out.println("Compund interest is: " +
money);
                                                break;
                                        case 5:
                                                break;
                                }
                        }while(choice!=5);
                }

        }
}
```

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

student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
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 s=new Scanner(System.in);
  this.usn=s.nextLine();
  this.name=s.nextLine();
  this.sem=s.nextInt();
 }

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

```
package CIE;
import java.util.Scanner;
public class Internals extends Student{
 protected int marks[]=new int[5];
 public void inputCIEmarks(){
  Scanner s=new Scanner(System.in);
  for(int i=0;i<5;i++){
   marks[i]=s.nextInt();
  }
 }
}
```

```
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 s = new Scanner(System.in);
  for(int i=0; i<5;i++){
   System.out.print("Subject "+(i+1)+" marks: ");
   marks[i] = s.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]);
 }

}

import SEE.*;

class Main1{
 public static void main(String args[]){
  int num=2;
  Externals finalMarks[]=new Externals[num];
  for(int i=0;i<num;i++){
   finalMarks[i]=new Externals();
   finalMarks[i].inputStudentDetails();
   System.out.println("Enter CIE marks");
   finalMarks[i].inputCIEmarks();
   System.out.println("Enter SEE marks");
   finalMarks[i].inputSEEmarks();
  }
  System.out.println("Displaying Data:\n");
  for(int i=0;i<num;i++){
   finalMarks[i].calculateFinalMarks();
```

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


 }
}
```

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

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

class InputScanner{
    Scanner s = new Scanner(System.in);
}

class Father extends InputScanner{
    int fatherAge;
    Father() throws WrongAge {
        System.out.println("Enter father's age: ");
        fatherAge = s.nextInt();
        if (fatherAge < 0)
            throw new WrongAge("Age cannot be negative");
    }

    void displayf(){
        System.out.println("\nFather's  age: " + fatherAge);
    }
}

class Son extends Father{
    int sonAge;
    Son() throws WrongAge{
        super();
        System.out.println("Enter son's age: ");
```

```
        sonAge = s.nextInt();
        if (sonAge < 0)
            throw new WrongAge("Age cannot be negative.");
        else if(sonAge > fatherAge)
            throw new WrongAge("Son's age cannot be greater than father's age.");
        else{
            displayf();
            displaySon();
            throw new WrongAge("Valid age.");
        }
    }

    void displaySon(){
        System.out.println("\nSon's age : "+ sonAge);
    }
}

class exceptionsmain {
    public static void main(String[] args) {
        try{
            Son son = new Son();
        } catch(WrongAge e) {
            System.err.println(e.getMessage());
        }
    }

}
```

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.

```
import java.lang.*;
class DisplayMessageThread extends Thread {
    private final String message;
    private final long interval; // in milliseconds

    DisplayMessageThread(String message, long interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        try {
```

```
            while (true) {
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + " interrupted.");
        }
    }
}

public class TwoThreadDemo {
    public static void main(String[] args) {
        DisplayMessageThread thread1 = new DisplayMessageThread("BMS College of
Engineering", 10000); // 10 seconds
        DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2000); // 2 seconds

        thread1.setName("Thread 1");
        thread2.setName("Thread 2");

        thread1.start();
        thread2.start();

        try {
            // Let the threads run for a while
            Thread.sleep(30000); // Let the program run for 30 seconds
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        // Interrupt both threads to stop them
        thread1.interrupt();
        thread2.interrupt();

        System.out.println("Main thread exiting.");
    }
}
```

10)Demonstrate Inter process Communication and deadlock
a)Inter process Communication
b)Deadlock

```
import java.lang.*;
class Q {
    int n;
```

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

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

```
    }
}

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

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 otherthread.

System.out.println("Back in otherthread");

}

public static void main(String args[]) {

new Deadlock();

}
```

```
  }
```

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.

```
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);
        }

}
```