B. M. S. COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELGAUM)
BANGALORE – 560019

2022-23

LAB RECORD

OBJECT ORIENTED JAVA PROGRAMMING (23CS3PCOOJ)

Submitted by :

NAME : Sohan A R

USN : 1BM22CS285

SEMESTER: III

SECTION : E

Submitted to

Dr. Seema Patil
Assistant Professor
Dept. of CSE, BMSCE

NAME : Sohan A R

SECTION : 3E

USN : 1BM22CS285

INDEX

SNO	NAME	DATE
1	Quadratic	12/12/23
2	SGPA Calculation	19/12/23
3	Book Program	26/12/23
4	Abstract class shape	2/1/24
5	Bank Account	9/1/24
6	Package	23/1/24
7	Exception Handling	30/1/24
8	Threads	6/2/24
10	Deadlock, ICP	13/2/24
9	Applet	20/2/24

LAB PROGRAM 12/12/23 RA-12-12-2-3

Q) Develop a java program that prints all real solutions to the quadratic equations $ax^2 + bx + c = 0$. 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("Root 1 = Root 2 = "+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();
        System.out.println("Sohan AR - 1BM22CS285");
    }
}
```

Output

1) Enter the coefficients of a, b, c

1
2
1

Roots are real and equal
Root 1 = Root 2 = -1
Sohan AR - 1BM22CS285

2) Enter the coefficients of a, b, c

1
4
1

Roots are Real and distinct
Root 1 = -0.2679491924311728
Root 2 = -3.7320508075688271
Sohan AR - 1BM22CS285

3) Enter the coefficients of a, b, c

4
1
1

Roots are Imaginary
Root 1 = 0.0 + i0.48412291827592711
Root 2 = 0.0 - i0.48412291827592711
Sohan AR - 1BM22CS285

LAB Program 2 Aug-12-23

Q Develop a java program to create a class students with members usn, name and array credits and 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;
    double grade;
}

class student {
    String name;
    String usn;
    double SGPA;
    Scanner s;
    Subject[] subjects;

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

    void getStudentDetails() {
        System.out.println("Enter the student name");
        name = s.nextLine();
```

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

    void getmarks() {
        for (int i = 0; i < 8; i++) {
            System.out.println("Enter marks for subject " + (i+1) + ":");
            Subjects[i].subjectMarks = s.nextInt();
            System.out.println("Enter credits for subject " + (i+1) + ":");
            Subjects[i].credits = s.nextInt();
            Subjects[i].credits
            Subjects[i].grade = subjects[i].SubjectMarks/10.0 + 1.0;
            if (subjects[i].grade == 11)
                SubjectS[i].grade = 10;
        }
    }

    Void computeSGPA() {
        double totalcredits = 0;
        double weightedSum = 0;

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

        SGPA = weightedSum / totalCredits;
    }

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

```
public class main {
    Public static void main ( string[ ] args ) {
        Student s1 = new Student ( );
        s1 . get StudentDetails ( );
        s1 . get Marks ( );
        s1 . compute SGPA ( );
        s1 . display Result ( );
    }
}
```

Output:

```
Enter Student Name :
Sohan
Enter USN :
1BM22CS285
Enter marks for Subject 1 :
90
Enter credits for Subject 1 :
4
Enter marks for Subject 2 :
90
Enter credits for Subject 2 :
4
Enter marks for Subject 3 :
85
Enter credits for Subject 3 :
3
Enter marks for Subject 4 :
89
Enter credits for Subject 4 :
4
```

Enter marks for Subject 5 :
85
Enter credits for Subject 5 :
3
Enter marks for Subject 6 :
90
Enter credits for Subject 6 :
2
Enter marks for Subject 7 :
90
Enter credits for Subject 7
1
Enter marks for Subject 8
90
Enter credits for Subject 8 :
1
Student Name : sohan
USN : 1BM22CS285
SGPA : 9.84545454545

LAB PROGRAM 3 26-12-23

VRA 26-12-23

Q Create 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 to string() 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, author;
    int price, 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 BookMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        String name, author;
        int price, numPages;
        System.out.println("Enter the number of books:");
        n = sc.nextInt();
        Books b[] = new Books[n];
        for(int i=0; i<n; i++){
            System.out.println("Enter Name, author, price and number of pages:");
            name = sc.next();
            author = sc.next();
            price = sc.nextInt();
            b[i] = new Books(name, author, price, numPages);
        }
        System.out.println("Book Details:");
        for(int i=0; i<n; i++){
            System.out.println(b[i].toString());
        }
    }
}
```

Output:

Name: Sohan A R

USN: 1BM22CS285

```
Enter the number of books:
3
Enter Name, author, price and number of pages:
DOS
Alex
1000
100
```

Enter Name, author, price and Number of pages :

Java for beginners

Harry

3999

499

Enter Name, author, price and number of pages :

Advanced Java

Kevin

2999

799

Book Details :

Book name s : OOJ

Author name : Alex

Price : 1000

Number of Pages = 100

Book name s : Java for Beginners

Author name : Harry

Price : 3999

Number of pages : 499

Book name s : Advanced Java

Author name : Kevin

Price : 2999

Number of pages : 799

LAB PROGRAM 4 7-1-24 LAB 2-1-24

* Develop a Java Program to create an abstract class name 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 {
    Scanner s;
    InputScanner() {
        s = new Scanner(System.in);
    }
}

abstract class Shape extends InputScanner {
    double a;
    double b;
    double r;
    abstract void getInput();
    abstract void DisplayArea();
}

class Rectangle extends Shape {
    void getInput() {
        System.out.println("Enter the sides of the rectangle");
        a = s.nextDouble();
        b = s.nextDouble();
    }

    void DisplayArea() {
        System.out.println("The area of the rectangle is" + a*b+ "");
    }
```

```
class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter the sides of the triangle");
        a = s.nextDouble();
        b = s.nextDouble();
    }

    void DisplayArea() {
        System.out.println("The area of the triangle is" + (0.5 * a * b) + "");
    }
}

    class Circle extends shape {
        void getInput() {
            System.out.println("Enter the radius of the circle");
            r = s.nextDouble();
        }

        void DisplayArea() {
            System.out.println("The area of the circle is" + 3.14 * r * r + "");
        }
    }

    class Shape main {
        public static void main(String args[]) {
            Rectangle r = new Rectangle();
            Triangle t = new Triangle();
            Circle c = new Circle();
            r.getInput();
            r.DisplayArea();
            t.getInput();
            t.DisplayArea();
            c.getInput();
            c.DisplayArea();
```

```
System.out.println("Sohan A R - 1BM22CS285");
    }
}
```

Output:

```
Enter the sides of the rectangle
2
2
The area of the rectangle is 4.0
Enter the sides of the triangle
2
6
The area of the triangle is 6.0
Enter the radius of the circle
4
The area of the circle is 50.24
Sohan A R - 1BM22CS285
```

LAB-5

9a) 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 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 the deposit from customer and update the balance

b) Display the balance

c) Compute and deposit interest

d) permit withdrawal and update the balance

* check for the minimum balance, impose penalty if necessary and update the balance

Code:

```
import java.util.Scanner;
Class account
  {
    String name;
    int acc no ;
    String type;
    double balance;
    account (String name, int acc no, String type, double balance)
```

```
{
  this.name = name;
  this.accno = accno;
  this.type = 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, cant withdraw");
  }
  }
void display ()
{
System.out.println ("name :" + name + "accno:" + accno + "type:" + type
  + "balance:" ) + balance);
  }
  }
  class savAcct extends account
  {
     private static double rate = 5;
      SavAcct (String name, int accno, double balance).
       {
```

```
super (name, accno, "savings", balance);
}
void interest()
{
balance += balance * (rate)/100;
System.out.println("balance :" + balance);
}
}
class curAcct extends account
{
private double minBal = 500;
private double serviceCharges = 50;
curAcct(String name, int accno, double balance)
{
super(name, accno, "current", balance);
}
void checkmin()
{
if (balance < minBal)
{
System.out.println("balance is less than min balance, service
charges imposed :" + serviceCharges);
System.out.println("balance is :" + balance);
}
}
}
class accountMain
{
```

```
public static void main(String a[])
{
Scanner s= new Scanner(System.in);
System.out.println("Enter the name:");
String name =s.next();
System.out.println("Enter the type (current/savings):");
String type=s.next();
System.out.println("Enter the account number:");
int accno=s.nextInt();
System.out.println("Enter the initial balance:");
double balance=s.nextDouble();
int ch;
double amount1, amount2;
account acc = new account(name,accno,type,balance)
SavAcct sa = new savAcct(name, accno, balance);
CurAcct ca = new curAcct(name, accno, balance);
while(true)
{
if(acc.type.equals("savings"))
{
System.out.println("\n Menu\n 1. deposit 2. withdraw 3. compute interest
4. display");
System.out.println("Enter the choice");
ch=s.nextInt();
switch(ch)
{
case 1 :System.out.println("Enter the amount:");
amount1=s.nextInt();
sa.deposit(amount1);
break;
```

```
case 2: System.out.println("Enter the amount");
amount2 = s.nextInt();
sa.withdraw(amount2);
break;
case 3 : sa.interest();
break;
case 4: sa.display();
break;
case 5 : System.exit(0);
default: System.out.println("Invalid input");
break;
}
}
else
{
System.out.println("\n Menu\n 1.deposit 2.withdraw 3.display");
System.out.println("Enter the choice");
ch = s.nextInt();
switch(ch)
{
case 1: System.out.println("Enter the amount:");
amount1 = s.nextInt();
ca.deposit(amount1);
break;
case 2 : System.out.println("Enter the amount:");
amount2=s.nextInt();
ca.withdraw(amount2);
ca.checkmin();
break;
```

```
case 3: ca.display();
break;
case 4: System.exit(0);
default: System.out.println("invalid input");
break
}
System.out.println("Sohan A.R - 1BM22CS285");
}
}
}
```

Output: Sohan AR - 1BM22CS285

Enter the name: sohan

Enter the type (current/savings):

current

Enter the account number:

1217776

Enter the initial balance:

10000

Menu

1. deposit 2. withdraw 3 display

enter the choice:

3

name: sohan accno: 1217776 type: current balance: 9800.0 menu

1. deposit 2. withdraw 3. display

enter the choice:

Enter the name: sohan

Enter the type (current / savings): savings

Enter the account number: 122222

Enter the initial balance: 7800

Menu

1. deposit 2. withdraw 3. compute interest 4. display

Enter the choice

2

Enter the amount:

6000

Menu

1. deposit 2. withdraw 3. compute interest 4. display

Enter the choice:

3

balance: 1890

Menu

1 - deposit 2. withdraw 3 - compute interest 4. display enter the choice:

name: sohan accno: 122222 type: savings balance: 1890.0

9/11/24

LAB-6

1) Demonstrate String length, String literal, string Concat;

```
public class Main{
  pubic static void main (string [] args) {
    // String length demonstration
    string str1 = "Hello world";
    int length = str1.length();
    System.out.println("string :" + str1);
    System.out.println("length of the string :" + length);
    string str2 = "Java";
    string str3 = "Java";
    boolean areEqual = (str2 == str3);
    System.out.println("Are str2 and str3 equal? " + areEqual)
    String firstname = "John";
    String lastname = "Doe";
    String fullname = firstname + " " + lastname;
    System.out.println("Full name :" + fullname);
    System.out.println("Sohan AR - 1BM22CS285");
  }
}
```

O/P: Sohan AR - 1BM22CS285

String : Hello, world!

length of the string : 13

Are str2 and str3 equal? true

Full name : John Doe

8) write a Java program to perform sorting of numbers from 10 to 1 using comparator to 1)

```
import java.util.Arrays;

public class NumberSorting {
    public static void main(String[] args) {
        Integer[] numbers = new Integer[]{10,9,8,7,6,5,4,3,2,1};

        Array.sort(numbers);
        System.out.println("Sorted Numbers(Ascending order): ");
        for(Integer number : numbers){
            System.out.print(number + " ");
        }

        System.out.println();
        Array.sort(numbers, (a,b) -> b.compareTo(a));

        System.out.println("Sorted nos in descending order : ");
        for(Integer number : number){
            System.out.println(number + " ");
        }
        System.out.println("Sohan AR - 1BM22CS285");
    }
}
```

Output: Sohan AR - 1BM22CS285

Sorted Numbers (Ascending order) :

1 2 3 4 5 6 7 8 9 10

Sorted nos in descending order :

10 9 8 7 6 5 4 3 2 1

8) Write a java program to create an abstract class shape with abstract methods calculate Area() and calculate Perimeter(). Create subclasses Circle and Triangle that extend the shape class and implement the respective methods to calculate the area and perimeter of each shape

```
import java.util.Scanner;

abstract class Shape{
    abstract double calculateArea();
    abstract double calculatePerimeter();
}
class Circle extends Shape{
    private double radius;
    public Circle(double radius){
        this.radius=radius;
    }
@Override
double calculateArea(){
    return Math.PI* Math.pow(radius,b:2);
}
@Override
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;
    }

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

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

public class Hi {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the radius of the Circle:");
        double circleRadius = scanner.nextDouble();
        Circle circle = new Circle(circleRadius);
        System.out.println("Enter the length of side1 of the triangle");
        double triangleSide1 = scanner.nextDouble();
        System.out.println("Enter the length of side2 of the triangle");
        double triangleSide2 = scanner.nextDouble();
```

```
System.out.println("Enter the length of side 3 of the triangle : ");
double triangleSide3 = scanner.nextDouble();
Triangle triangle = new Triangle(triangleSide1, triangleSide2, triangleSide3);
scanner.close();

System.out.println("\n Circle Area :" + circle.calculateArea());
System.out.println("Circle Perimeter :" + circle.calculatePerimeter());
System.out.println("\n Triangle Area :" + triangle.calculateArea());
System.out.println("Triangle Perimeter :" + triangle.calculatePerimeter());
System.out.println("Sohan AR - 1BM22CS285");
}
}
```

Output : Sohan AR - 1BM22CS285

Enter the radius of circle : 2

Enter the length of side1 of the triangle : 3

Enter the length of side2 of the triangle : 3

Enter the length of side 3 of the triangle : 2

Circle Area : 12.56

Circle Perimeter : 12.56

Triangle Area = NaN

Triangle Perimeter : 38.0

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

```
class Generic stack<T>{
    private Object[] stackArray;
    private int top = -1;
    private static final int MAX_SIZE = 5;

    public Generic stack(){
        stackArray = new Object[MAX_SIZE];
    }

    public void push(T value){
        if(top < MAX_SIZE - 1) stackArray[++top] = value;
        else System.out.println("Stack is full. Cannot push more elements");
    }

    @SuppressWarnings("unchecked")
    public T pop(){
        if(top >= 0)
            return (T) stackArray[top--];
            else{
                System.out.println("Stack is empty, Cannot pop more
                elements");
                return null;
            }
    }

    public boolean isFull(){
        return top == MAX_SIZE - 1;
    }
```

```
class generics {
public static void main (String [] args) {
    Generic Stack <Integer> integer stack = new Generic Stack <> ();
    Generic Stack <Double> double stack = new Generic Stack <> ();

for (int i = 1; i <= 5 ; i++ ) {
        integer Stack . push (i);
}

for (double i = 1.0; i <= 5.0; i++) {
        double Stack . push (i);
}

System . out . println ("popped integers from the Stack :");
while (Integer Stack . is Empty ()) {
  System . out . println ( integer Stack . pop() );
}

System . out . println ("Popped doubles from the stack");
while (! double Stack . is Empty()) {
        System . out . println ( double Stack . pop());
} System . out . println ("Sohan A R - 1BM22CS285 ");

}
}
```

16/1/24

Output : Sohan A R - [redacted] 1BM22CS285
Popped integers from the stack :

5
4
3
2
1

Popped doubles from the stack

5.0
4.1
3.0
2.0
1.0

* 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.

PROGRAM:
Inside CIE folder
Internals.java:

```
Package CIE;
import java.util.Scanner;
public class Internals extends Student {
    protected int marks[ ] = new int [5];
    public Internals ( ) {
        // constructor for Internals
    }

    public void input CIEmarks ( ) {
```

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter Internal Marks for " + name);
for (int i=0; i<5; i++){
    System.out.print("Subject " + (i+1) + " marks: ");
    marks[i] = scanner.nextInt();
}
}
}
```

student.java

```
import java.util.Scanner;
public class Student {
    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.next();
    System.out.println("Enter Name: ");
    name = scanner.next();
    System.out.print("Enter Semester: ");
    sem = scanner.nextInt();
}

public void displayStudentDetails(){
    System.out.println("USN: " + usn);
    System.out.println("Name: " + name);
```

```
        System.out.println("Semester: " + sem);
    }
}
```

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 scanner = new Scanner(System.in);
        System.out.println("Enter SEE marks for " + name);
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + " marks: ");
            marks[i] = scanner.nextInt();
        }
    }

    public void calculateFinalMarks() {
        for (int i = 0; i < 5; i++)
            finalMarks[i] = marks[i] / 2 + subject.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.Externals;
public class Main {
    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");
            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++) {
            finalMarks[i].calculateFinalMarks();
            finalMarks(i).displayFinalMarks();
        }
        System.out.println("Sohan AR-1BM22CS285");
    }
}
```

Output :- Sohan A R - 1BM22CS285

Enter USN : 111

Enter Name : Soh

Enter Semester : 2

Enter CIE marks

Enter Internal Marks for Soh

Subject 1 marks : 222

Subject 2 marks : 211

Subject 3 marks : 44

Subject 4 marks : 33

Subject 5 marks : 2

Enter SEE marks

Enter SEE marks for Soh

Subject 1 marks : 111

Subject 2 marks : 3

Subject 3 marks : 3

Subject 4 marks : 3

Subject 5 marks : 3

30/1/24

LAB PROGRAM - 7

Write a program that demonstrates handling of exceptions in Inheritance tre

· 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 inpu age < 0. In son class, implement a constructor that cases both fat and son's age and throws an exception if son's age is >= father's ag

code:

```
import java.util.Scanner;
public class ExceptionInheritance {
    static class Father {
        int age;

        Father(int age) throws WrongAge {
            if (age < 0) {
                throw new WrongAge("Father's age cannot be negative");
            }
            this.age = age;
        }
    }

    static class Son {
        int age;
        Father father;

        Son(int fatherAge, int sonAge) throws WrongAge {
            father = new Father(fatherAge);
            if (sonAge >= father.age) {
                throw new WrongAge("Son's age cannot be equal or gr
                than father's age");
```

```
    }
    this.age = sonAge;
}

static class WrongAge extends Exception {
    WrongAge(String message) {
        super(message);
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter father's age:");
    int fatherAge = scanner.nextInt();
    System.out.println("Enter son's age:");
    int sonAge = scanner.nextInt();

    try {
        Son son = new Son(fatherAge, sonAge);
        System.out.println("Son's age: " + son.age);
    } catch (WrongAge e) {
        System.out.println(e.getMessage());
        System.out.println("Sohan A R - 1BM22CS285");
    }
}
```

30/1/24

Output:

Enter father's age:
29

Enter son's age
39

Son's age cannot be equal or greater than father's age

LAB – 8

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

```
class 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(InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + "interr
        }
    }
}

public class TwoThreadDemo {
    public static void main(String[] args) {
        DisplayMessageThread thread1 = new DisplayMessageThread("BMS Col
            "of Engineering", 10000);
        DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2
```

```
thread1.setName("Thread1");
thread2.setName("Thread2");
thread1.start();
thread2.start();

try {
    Thread.sleep(millis: 30000);
} catch (InterruptedException e) {
    System.out.println("Main thread interrupted.");
}

thread1.interrupt();
thread2.interrupt();

System.out.println("Main thread exiting");
System.out.println("Sohan AR - 1BM22CS285");
```

Output

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

13.02.24

LAB - 10 :

Demonstrate Inter process communication and Deadlock

```
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("Interrupted Exception 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("Interrupted Exception caught");
  }
```

```
this.n = n;
valueSet = true;
System.out.println("Put :" + n);
System.out.println("\n Intimate Consumer\n");
}
}
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++);
System.out.println("Sohan AR - 1BM22CS285");
}
}
}
class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
int i = 0;
while(i < 15) {
int r = q.get();
System.out.println("Consumed : " + r);
i++;
}
}
}
class PC {
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: Sohan AR - 1BM22CS285
Press Control C to stop
Put : 0
Intimate Customer
Producer waiting
Got : 0
Intimate producer
Put : 1

Program to Demonstrate 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); // 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();
  System.out.println("Sohan AR - 1BM22CS285");
}
```

Output: Sohan AR - 1BM22CS285

Main Thread entered A.foo

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 thread.

CAB-9

WAP 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 javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo {
    SwingDemo() {
    // create jframe container
    JFrame jfrm = new JFrame("Divider App");
    jfrm.setSize(275, 150);
    jfrm.setLayout(new FlowLayout());
    // to terminate onclose
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// text label
    JLabel jlab = new JLabel("Enter divisor and dividend");
// add text field for both numbers
    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);
```

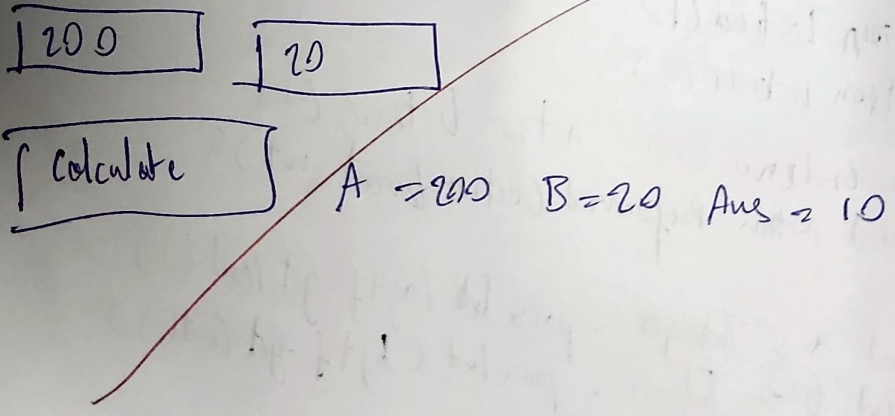
```
// calc button
JButton button = new JButton("Calculate");
// Labels
Jlabel err = new Jlabel();
Jlabel alab = new Jlabel();
Jlabel blab = new Jlabel();
Jlabel anslab = new Jlabel();
// add in order :)
jfram.add(err); // to display error
jfram.add(jlab);
jfram.add(ajtf);
jfram.add(bjtf);
jfram.add(button);
jfram.add(alab);
jfram.add(blab);
jfram.add(anslab)
ActionListner l = new ActionListner(){
    public void actionPerformed(ActionEvent evt){
        System.out.println("Action event from a text field");
    }};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent evt){
        try{
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            int ans = a/b;

            alab.setText("InA = "+a);
            blab.setText("InB = "+b);
            anslab.setText("In Ans = "+ans);
        }
```

```
catch (ArithmeticException e) {
        alab.setText("");
        blab.setText("");
        anslab.setText("");
        errsetText("B should be NON zero");
    }
});
    //display frame
    jframe.setVisible(true)
}
public static void main(String args[]) {
    //create frame on event dispatching thread
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new SwingDemo();
        }
    });
}
```

Output :

Divider App

Enter the Divider and dividend



Explanation:

JFrame : represent main windows of application

JLabel : used to display text labels on GUI

add(): This is used to add a Swing component (button / label, textfield) to a container (JFrame)

setText(text): It is used to set the text of text based components (such as JLabel) to specified string

getText(): It retrieves text from text based on component

setLayout(LayoutManager layout): It sets layout manager for container responsible for arranging elements inside it.

setVisible(boolean visible): It sets visibility of component. when true component and can set become visible on screen hidden when set to false

20/2/24

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("Roo1 = " + 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();
        System.out.println("Sohan A R -1BM22CS285");
    }
}
```

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

```
import java.util.Scanner;


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


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

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

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

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

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



        subjects[i].grade = subjects[i].subjectMarks / 10.0 + 1.0;
    }
}

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

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

    SGPA = weightedSum / totalCredits;
}

void displayResult() {
```

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

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

System.out.println("Sohan A R -1BM22CS285");


    }
}
```

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;
	}
}
```

```
public class Mainbook
{
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		int n;
		int i;
		String name;
		String author;
		int price;
		int numPages;

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

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

		for(i=0;i<n;i++)
		{
			System.out.println("Enter the details of book" + (i+1) + ":");
			System.out.println("Enter the name of the book:");
			name=s.next();
			System.out.println("Enter the author name:");
			author=s.next();
			System.out.println("Enter the price:");
			price=s.nextInt();
			System.out.println("Enter the number of pages:");
			numPages=s.nextInt();
```

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

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

            }
        }
          System.out.println("Sohan A R -1BM22CS285");

}
```

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

```
import java.util.*;
import java.math.*;

class InputScanner {

   Scanner sc;

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

}

abstract class Shape extends InputScanner {

   double a;

   double b;

   abstract void getInput();

   abstract void displayArea();

}

class Rectangle extends Shape {
   void getInput() {
```

```
        System.out.println("Enter the length and breadth:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of rectangle is :" + (a * b));
    }
}

class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter the length and height:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of triangle is :" + (a * b * 0.5));
    }
}

class Circle extends Shape {
    void getInput() {
        System.out.println("Enter the radius:");
        a = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of circle is :" + (a * a * Math.PI));
    }
```

```
}


class ShapeMain {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getInput();
        r.displayArea();
        t.getInput();
        t.displayArea();
        c.getInput();
        c.displayArea();

        System.out.println("Sohan A R -1BM22CS285");

    }
}
```

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.Scanner;

class account {
   String name;
   int accno;
   String type;
   double balance;

   account(String name, int accno, String type, double balance) {
      this.name = name;
      this.accno = accno;
      this.type = 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,cant  withdraw");
    }
  }

  void display() {
    System.out.println("Name:" + name + "\nAccno:" + accno + "\nType:" + type + "\nBalance:" +
balance);
  }
}

class savAcct extends account {

  private static double rate = 5;

  savAcct(String name, int accno, double balance) {
    super(name, accno, "Savings", balance);

  }

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

```
}

class curAcct extends account {

   private double minBal = 500;
   private double serviceCharges = 50;

   curAcct(String name, int accno, double balance) {
      super(name, accno, "Current", balance);

   }

   void checkmin() {

      if (balance < minBal) {
         System.out.println("Balance is less than min balance,service charges imposed:" +
serviceCharges);
         balance -= serviceCharges;
         System.out.println("Balance is:" + balance);
      }

   }

}

class Bank {
   public static void main(String a[]) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the name,type(current/savings),account number,initial balance:");
      String name = s.next();
      String type = s.next();
```

```
int accno = s.nextInt();
double balance = s.nextDouble();
int ch;
double amount1, amount2;
account acc = new account(name, accno, type, balance);
savAcct sa = new savAcct(name, accno, balance);
curAcct ca = new curAcct(name, accno, balance);
while (true) {
   if (acc.type.equals("savings")) {
      System.out.println("\nMenu\n1.deposit 2.withdraw 3.compute interest 4.display");
      System.out.println("Enter the choice:");
      ch = s.nextInt();
      switch (ch) {
         case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            sa.deposit(amount1);
            break;
         case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            sa.withdraw(amount2);
            break;
         case 3:
            sa.interest();
            break;
         case 4:
            sa.display();
            break;
         case 5:
            System.exit(0);
```

```
        default:
            System.out.println("invalid input");
            break;
    }
} else {
    System.out.println("\nMenu\n1.deposit 2.withdraw  3.display");
    System.out.println("Enter the choice:");
    ch = s.nextInt();
    switch (ch) {
        case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            ca.deposit(amount1);
            break;
        case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            ca.withdraw(amount2);
            ca.checkmin();
            break;

        case 3:
            ca.display();
            break;
        case 4:
            System.exit(0);
        default:
            System.out.println("Invalid input");
            break;

          System.out.println("Sohan A R -1BM22CS285");

    }
}
```

```
    }


  }
}
```

NUMBER SORTING

```
import java.util.Arrays;

public class NumberSorting {
    public static void main(String[] args) {
        // Create an array of Integer objects with numbers from 10 to 1
        Integer[] numbers = new Integer[] { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };

        // Sorting the array in ascending order using Arrays.sort()
        Arrays.sort(numbers);

        // Displaying the sorted numbers
        System.out.println("Sorted Numbers (Ascending Order):");
        for (Integer number : numbers) {
            System.out.print(number + " ");
        }

        System.out.println(); // New line for better readability

        // Sorting the array in descending order using a custom comparator
        Arrays.sort(numbers, (a, b) -> b.compareTo(a));

        // Displaying the sorted numbers in descending order
        System.out.println("Sorted Numbers (Descending Order):");
        for (Integer number : numbers) {
            System.out.print(number + " ");
        }
    }
}
```

GENERICS

```
class GenericStack<T> {
    private Object[] stackArray;
    private int top = -1;
    private static final int MAX_SIZE = 5;

    public GenericStack() {
        stackArray = new Object[MAX_SIZE];
    }

    public void push(T value) {
        if (top < MAX_SIZE - 1) stackArray[++top] = value;
        else System.out.println("Stack is full. Cannot push more elements.");
    }

    @SuppressWarnings("unchecked")
    public T pop() {
        if (top >= 0)
            return (T) stackArray[top--];
        else {
            System.out.println("Stack is empty. Cannot pop more elements.");
            return null;
        }
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == MAX_SIZE - 1;
```

```
    }
}
class Main{
public static void main(String[] args) {
        GenericStack<Integer> integerStack = new GenericStack<>();
        GenericStack<Double> doubleStack = new GenericStack<>();

        // Push integers to the integer stack
        for (int i = 1; i <= 5; i++) {
            integerStack.push(i);
        }

        // Push doubles to the double stack
        for (double i = 1.0; i <= 5.0; i++) {
            doubleStack.push(i);
        }

        // Pop and print integers from the integer stack
        System.out.println("Popped integers from the stack:");
        while (!integerStack.isEmpty()) {
            System.out.println(integerStack.pop());
        }

        // Pop and print doubles from the double stack
        System.out.println("Popped doubles from the stack:");
        while (!doubleStack.isEmpty()) {
            System.out.println(doubleStack.pop());
        }
    }
}
```

Abstract class prog

```
import java.lang.Math;

abstract class Shape {
   double a;
   double b;
   double c;

   abstract void calculateArea();

   abstract void calculatePerimeter();
}

class Triangle extends Shape {
   Triangle(double x, double y, double z) {
      a = x;
      b = y;
      c = z;
   }

   void calculateArea() {
      double s = (a + b + c) / 2;
      System.out.println("Area=" + (Math.sqrt(s * (s - a) * (s - b) * (s - c))));
   }

   void calculatePerimeter() {
      System.out.println("Perimeter=" + (a + b + c));
   }
}

class Circle extends Shape {
```

```
    Circle(double r) {
        a = r;
    }

    void calculateArea() {
        System.out.println("Area=" + (Math.PI * a * a));
    }

    void calculatePerimeter() {
        System.out.println("Perimeter=" + (2 * Math.PI * a));
    }
}

class ShapeM {
    public static void main(String[] args) {
        Triangle t = new Triangle(2.0, 3.0, 5.0);
        Circle c = new Circle(5.0);
        t.calculateArea();
        t.calculatePerimeter();
        c.calculateArea();
        c.calculatePerimeter();

        System.out.println("Sohan A R -1BM22CS285");

    }
}
```

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.Scanner;

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

   public void inputCIEmarks() {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Internal Marks for " + name);
      for (int i = 0; i < 5; i++) {
         System.out.print("Subject " + (i + 1) + " marks: ");
         marks[i] = s.nextInt();
      }
   }
}
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 s = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = s.next();
        System.out.print("Enter Name: ");
        name = s.next();
        System.out.print("Enter Semester: ");
        sem = s.nextInt();
    }

    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}
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);
        System.out.println("Enter SEE Marks for " + name);
        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.Externals;

public class Pkgmain {
    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");
            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++) {
            finalMarks[i].calculateFinalMarks();
            finalMarks[i].displayFinalMarks();

          System.out.println("Sohan A R -1BM22CS285");

        }
    }
}
```

7. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called

"Father" and derived class called "Son" which extends the base class. In Father class, implement a

constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class,

implement a constructor that cases both father and son's age and throws an exception if son's age is

>=father's age.

```
import java.util.Scanner;

public class ExceptionInheritance {


   static class Father {

      int age;


      Father(int age) throws WrongAge {

         if (age < 0) {

            throw new WrongAge("Father's age cannot be negative.");

         }

         this.age = age;

      }

   }


   static class Son {

      int age;

      Father father;


      Son(int fatherAge, int sonAge) throws WrongAge {

         father = new Father(fatherAge);

         if (sonAge >= father.age) {

            throw new WrongAge("Son's age cannot be equal or greater than father's age.");
```

```
            }
            this.age = sonAge;
        }
    }

    static class WrongAge extends Exception {
        WrongAge(String message) {
            super(message);
        }
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter father's age: ");
        int fatherAge = scanner.nextInt();
        System.out.println("Enter son's age: ");
        int sonAge = scanner.nextInt();

        try {
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Son's age: " + son.age);
        } catch (WrongAge e) {
            System.out.println(e.getMessage());
           System.out.println("Sohan A R -1BM22CS285");

        }
    }
}
```

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

```
class 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 (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);
      DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2000);

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

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

        try {
            Thread.sleep(30000);
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        thread1.interrupt();
        thread2.interrupt();

        System.out.println("Main thread exiting.");
        System.out.println("Sohan A R -1BM22CS285");

    }
}
```

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

class SwingDemo {
   SwingDemo() {
      // create jframe container
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 150);
      jfrm.setLayout(new FlowLayout());
      // to terminate on close
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

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

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

      // labels
      JLabel err = new JLabel();
      JLabel alab = new JLabel();
```

```
JLabel blab = new JLabel();
JLabel anslab = new JLabel();

// add in order :)
jfrm.add(err); // to display error bois
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 evt) {
      System.out.println("Action event from a text field");
   }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);

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

         alab.setText("\nA = " + a);
         blab.setText("\nB = " + b);
         anslab.setText("\nAns = " + 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!");
            }
          }
        });

        // display frame
        jfrm.setVisible(true);
    }

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

         System.out.println("Sohan A R -1BM22CS285");

            }
        });
    }
}
```

10a) Interprocess communication using consumer and producer

```
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 < 6) {
            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 < 6) {
            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.");
        System.out.println("Sohan A R -1BM22CS285");

    }
}
```

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
        B.last");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() { Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread"); t.start();
        a.foo(b); // 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();

        System.out.println("Sohan A R -1BM22CS285");

    }
}
```