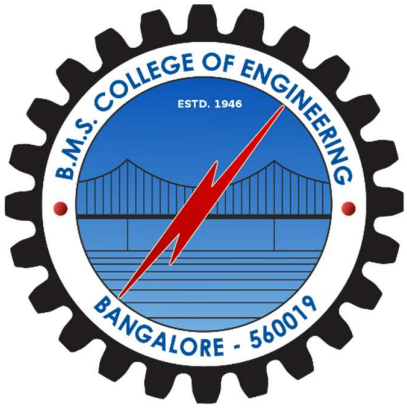
B.M.S. COLLEGE OF ENGINEERING BENGALURU

Autonomous Institution affiliated to VTU



LAB REPORT

23CS3PCOOJ

Submitted in partial fulfilment of the requirements for Lab

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

Shraddha

(1BM22CS357)

Department of Computer Science and Engineering,

B.M.S College of Engineering,

Bull Temple Road, Basavanagudi, Bangalore, 560019

2023-2024

LAB-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 -ve, display a msg stating that there are no real solutions.

```
import java.util.Scanner;
class Quadratic {
    int a,b,c;
    double r1, r2, d;
    void getCoefficients() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the coefficient of a,
        b, c:");
    a = scanner.nextInt();
    b = scanner.nextInt();
    c = scanner.nextInt();
    }
    void computeRoots() {
        while (a==0) {
            System.out.println("Not a quadratic eqn");
            System.out.println("Enter a non-zero value
                for 'a':");
```

```
        Scanner scanner = new Scanner(system.in);
        a = scanner.nextInt();
    }
    d = b*b - 4*a*c;
    if (d==0){
        r1 = -b/(2.0*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))/(2.0*a);
        r2 = (-b - Math.sqrt(d))/(2.0*a);
        System.out.println("Roots are real and distinct");
        System.out.println("Root 1 = "+r1+" Root 2 = "+r2);
    }else{
        System.out.println("Roots are imaginary");
        r1 = -b/(2.0*a);
        r2 = Math.sqrt(-d)/(2.0*a);
        System.out.println("Root 1 = "+r1+" "+i" +r2);
        System.out.println("Root 2 = "+r1+" "-i" +r2);
    }
  }
}
class QuadraticMain{
    public static void main(String[]args){
```

```
        Quadratic quadratic = new Quadratic();
        quadratic.getCoefficients();
        quadratic.computeRoots();
    }
}
```

O/P:-

```
javac Quadratic.java.
java QuadraticMain

Enter the coefficient of a, b and c:
1 2 3
Roots are imaginary.
Root1 = -1.0 + i 1.4142135623730951
Root2 = -1.0 - i 1.4142135623730951
```

```
import java.util.Scanner;

class Quadratic {
    int a, b, c;
    double r1, r2, d;

    void getCoefficients() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the coefficients of a, b, and c:");
        a = scanner.nextInt();
        b = scanner.nextInt();
        c = scanner.nextInt();
    }

    void computeRoots() {
        while (a == 0) {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non-zero value for 'a':");
            Scanner scanner = new Scanner(System.in);
            a = scanner.nextInt();
        }

        d = b * b - 4 * a * c;

        if (d == 0) {
            r1 = -b / (2.0 * a);
            System.out.println("Roots are real and equal");
            System.out.println("Root1 = Root2 = " + r1);
        } else if (d > 0) {
            r1 = (-b + Math.sqrt(d)) / (2.0 * a);
            r2 = (-b - Math.sqrt(d)) / (2.0 * a);
            System.out.println("Roots are real and distinct");
            System.out.println("Root1 = " + r1 + " Root2 = " + r2);
        } else {
            System.out.println("Roots are imaginary");
            r1 = -b / (2.0 * a);
            r2 = Math.sqrt(-d) / (2.0 * a);
            System.out.println("Root1 = " + r1 + " + i" + r2);
            System.out.println("Root2 = " + r1 + " - i" + r2);
        }
    }
}

class QuadraticMain {
    public static void main(String[] args) {
        Quadratic quadratic = new Quadratic();
        quadratic.getCoefficients();
        quadratic.computeRoots();
    }
}
```

LAB-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 marks;
    int credits;
    int grades;
}
class student
{
   string name;
   string usn;
   double SGPA;
   Scanner s;

Student ()
   int i;
   Subject s= new Subject[8];
   for (i=0, i<8; i+1)
```

```
        subjects[i] = new Subject();
    s = new Scanner(System.in);
}
void getStudentDetails(){
System.out.println("Enter Student Name:");
name = s.next();
System.out.println("Enter USN:");
USN = s.next();
{
void getMarks(){
for(int i=0; i<8, i++)
{
  System.out.println("Enter details of subject");
  System.out.println("Enter marks:");
  Subject[i].SubjectMarks = s.nextInt();
  System.out.println("Enter credits:");
  Subject[i].credits = s.nextInt();
  if(subject[i].subjectMarks >= 90)
  {
  Subjects[i].grade = 1.0;
  }
  else if(subjects[i].subjectMarks >= 80)
  {
  Subjects[i].grade = 9;
  }
```

```
else if (subjects [i] . subjectMarks >= 70)
{
  subjects[i]. grade = 8;
}
else if (subjects [i]. subject Marks >= 60)
{
 subjects[i]. grade = 7;
}
else if (subjects[i]. subject Marks >= 50)
{
 Subjects [i]. grade=6; }
else if (subjects[i]=subjectMarks >= 40)
{
 Subjects [i]. grade = 5;
}
else {
   sUbjects[i]. grade = 0;
}
}
}
void compute SGPA( )
{
  double total credits = 0;
  double weightedsum = 0;
```

```
    for (int i=0; i<8, i++)
{
  totalcredits t = subjects[i];
  weightedsum += subjects[i].grade * subjects[i].credits;
}
  SGPA = weightedsum / total credits
{
  void displayDetails() {
  System.out.println("Student details");
  System.out.println("Name:" + name);
  System.out.println("USN:" +USN);
  System.out.println("SGPA:" + SGPA);
  }
public class student Main{
    public static void main(String[]args)
    {
    Student S1 = new Student();
    S1.getStudentDetails();
    S1.get Marks();
    S1.compute SGPA();
    S1.displayDetails();
    }
}
```

Output :-

Enter name
Shraddha

Enter USN
232

Enter Marks
96
97
89
100
92
90
87
93

Enter credits
4
4
3
3
3
1
1
1

SGPA = 9.8

```
import java.util.Scanner;
class Subject
{
    int marks;
    int credits;
    int grades;
}
class Student{
    String name;
    String USN;
    double SGPA;
    Scanner s;
    Subject[] subjects;

    Student(){
        int i;
        subjects= new Subject[8];
        for(i=0;i<8;i++)
        subjects[i] = new Subject();
    s = new Scanner(System.in);
    }
    void getStudentDetails(){
        System.out.println("Enter Student Name:");
        name=s.next();
        System.out.println("Enter USN:");
        USN=s.next();
    }
    void getMarks(){
        for(int i=0;i<8;i++){
            System.out.println("Enter details of Subject:");
            System.out.println("Enter Marks:");
            subjects[i].marks=s.nextInt();
            System.out.println("Enter Credits:");
            subjects[i].credits=s.nextInt();
            if(subjects[i].marks>=90)
            {
                subjects[i].grades=10;
            }
            else if(subjects[i].marks>=80)
            {
                subjects[i].grades=9;
            }
            else if(subjects[i].marks>=70)
            {
                subjects[i].grades=8;
            }
            else if(subjects[i].marks>=60)
            {
                subjects[i].grades=7;
            }
            else if(subjects[i].marks>=50)
            {
                subjects[i].grades=6;
            }
            else if(subjects[i].marks>=40)
            {
                subjects[i].grades=5;
```

```
            }
            else
            {
                subjects[i].grades=0;
            }
        }
    }
    void computeSGPA(){
        double totalCredits=0;
        double weightedSum=0;
        for(int i=0;i<8;i++){
            totalCredits += subjects[i].credits;
            weightedSum += subjects[i].grades*subjects[i].credits;
        }
        SGPA=weightedSum/totalCredits;
    }
    void displayDetails(){
        System.out.println("Student Details");
        System.out.println("Name:"+name);
        System.out.println("USN:"+USN);
        System.out.println("SGPA:"+SGPA);
    }
    public static void main(String[]args){
        Student s1=new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.displayDetails();
    }
}
```

LAB 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.*;
class Books
{
  int price;
  int num_pages;
  string name;
  String author;
  Books (String name, String author, int price, int num_pages)
  {
     this.name = name;
     this.author = author;
     this.price = price;
     this.numpage = num_page;
  }
```

```
public String tostring ()
{
    String name;
    String author;
    String price;
    String num_page;
    name = "Book name:" + this name + "\n";
    author = "Author name:" + this author + "\n";
    price = "price" + this price + "\n";
    num_page = "Number of pages:" + this.num_page + "\n";
    return name + author + price + num_page;
    }
}
class Main
{
    public static void main (String args[])
    {
    Scanner s = new Scanner (Stystem.in);
    int n;
    String nam;
    String author;
    int price;
    int num_page;
    System.out.println ("Enter no. of books");
    n = s.nextInt();
```

```
Books b[];
b=new Books[n];
for (int i=0; i< n; i++)
{
System.out.println("Enter name, author, price,
                    num_page");
name =s.next();
author = s.next();
price = s.nextInt();
num_page = s.nextInt();
b[i]=new Books(name, author, price, num_page);
}
for(int i=0;i<n;i++)
{
  System.out.println(b[i])
}
}
}
```

Output:-

```
Enter no. of books
2
Enter name, author, price, number of pages
Jara Yen 100 80
C Reema 200 300
```

Book name: Java
Author : Yen
Price : 100
Number of pages: 80

Book name: C
Author : Reema
Price : 200
Number of pages: 300

```
import java.util.Scanner;

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

    Books(String name, String author, int price, int num_pages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.num_pages = num_pages;
    }
lo0o,o
    public String toString()
    {
        String name;
        String author;
        String price;
        String num_pages;
        name = "Book name: " + this.name + "\n";
        author = "Author name : " + this.author + "\n";
        price = "Price : " + this.price + "\n";
        num_pages = "Number of pages : " + this.num_pages + "\n";
        return name + author + price + num_pages;
    }
}

public class Books Main
 {
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        int n;
        String name;
        String author;
        int price;
        int num_pages;
        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:");
            num_pages = s.nextInt();
```

```
            b[i] = new Books(name, author, price, num_pages);
        }
        System.out.println("Book details:");
        for (i = 0; i < n; i++)
        {
            System.out.println(b[i]);
        }
    }
}
```

LAB-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.

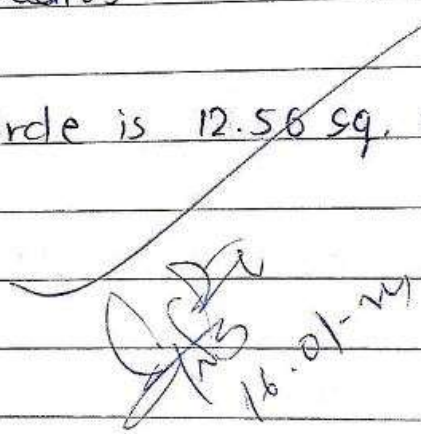
```
import java.util.Scanner;
abstract class shape
{
   int a,b;
   abstract void printarea();
   abstract void input();
   Scanner s = new Scanner(System.in)
}
class rectangle extends shape
{
   void input()
    {
    System.out.println("Enter l and b:");
   a = s.nextInt();
     b = s.nextInt();
```

```
    }
    void printarea()
    {
    System.out.println("Area of rectangle is:" +
                        (a*b)+"sq units");
    }
}
class Triangle extends shape;
{
    void input()
    {
    System.out.println("Enter b and h");
    a=s.nextInt();
    b=s.nextInt();
    }
    void printarea()
    {
    System.out.println("Area of Triangle is " + (a*b)/2)+"
                                    sq. units");
    }
}
class Circle extends shape;
{
    void input()
    {
```

```
        System.out.println("Enter the radius of circle");
        a=s.nextInt();
        }
        void printarea()
        {
        System.out.println("Area of circle is"+(3.14*a*a)+
                                    " sq units");

        }
}
class Area
{
    public static void main (String args[])
    {
        rectangle r = new rectangle();
        r.input();
        r.printarea();

        triangle t = new triangle();
        t.input();
        t.printarea();

        circle c = new circle();
        c.input();
        c.printarea();
    }
}
```

Output :-

```
Enter l and b
4 5
Area of rectangle is 20 sq. units.

Enter b and h
5 6
Area of triangle is 30 sq. units.

Enter the radius
2
Area of circle is 12.56 sq. units.
```



```
import java.util.Scanner;
abstract class Shape {
    int a, b;
    abstract void printArea();
    abstract void input();
}
class Rectangle extends Shape {
    Scanner s = new Scanner(System.in);
    void input() {
        System.out.println("Enter length and breadth:");
        a = s.nextInt();
        b = s.nextInt();
    }
    void printArea() {
        System.out.println("Area of rectangle is: " + (a * b) + " sq units");
    }
}

class Triangle extends Shape {
    Scanner s = new Scanner(System.in);
    void input() {
        System.out.println("Enter base and height:");
        a = s.nextInt();
        b = s.nextInt();
    }
    void printArea() {
        System.out.println("Area of Triangle is " + ((a * b) / 2) + " sq. units");
    }
}

class Circle extends Shape {
    Scanner s = new Scanner(System.in);
    void input() {
        System.out.println("Enter the radius of circle:");
        a = s.nextInt();
    }
    void printArea() {
```

```
        System.out.println("Area of circle is " + (3.14 * a * a) + " sq units");
    }
}
public class Area {
    public static void main(String args[]) {
        Rectangle r = new Rectangle();
        r.input();
        r.printArea();
        Triangle t = new Triangle();
        t.input();
        t.printArea();
        Circle c = new Circle();
        c.input();
        c.printArea();
      }
}
```

LAB-5

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

```
import java.util.Scanner
class account
{
    String name;
    int acc_no;
    String acc_type;
    double balance;
    account(String name, int acc_no, String acc_type,
            double balance)
    {
        this.name = name;
        this.acc_no = acc_no;
        this.acc_type = acc_type;
```

```
        this.balance = balance;
    }
    void deposit(double amount)
    {
        balance += amount;
    }
    void withdraw(double amount)
    {
        if ((balance - amount) >= 0)
        {
            balance -= amount;
        }
        else
        {
            System.out.println("Insufficient balance,
                                can't withdraw");
        }
    }
    void display()
    {
        System.out.println("Customer name:" +
        name + "\n acc.no:" + acc_no + "\n type of account" +
        acc_type + "\n balance" + balance);
    }
}
```

```
class sav_Acc extends account
{
    private static double rate = 5;
    sav_Acc (String name, int acc_no, double balance)
    {
        sUper (name, acc_no, "savings", balance);
    }
    void interest ()
    {
        balance += balance * (rate)/100;
        System.out.println ("balance:" + balance);
    }
}
class cur_Acc extends account
{
    private double min_Bal = 500;
    private double serrice_charge = 50;
    cur_Acc (String name, int acc_no, double balance)
    {
        super(name, acc_no, "current", balance);
    }
    void checkmin ()
    {
        if (balance < min_Bal)
        {
```

```
            System.out.println("Balance is less than
                min_balance : service charges imposed:"
                + service_charge);
            balance -= service_charge;
            System.out.println("Balance is":+balance);
        }
    }
}
class account_Main
{
    public static void Main(String a[])
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the customer name:");
        String name =s.next();
        System.out.println("Enter the type (current/savings):");
        String type =s.next();
        System.out.println("Enter the account no.:");
        int acc_no =s.nextInt();
        System.out.println("Enter the initial balance:");
        double balance = s.nextDouble();
        int ch;
        double amount1, amount2;
        account acc=new account(name, acc_no, type, balance);
        sav_Acc sa= new sav_Acc(name, acc_no, balance);
```

```
cur_Acc ca = new cur_Acc(name, acc_no, balance);
while (true)
{
    if (acc.type.equals("saving"))
    {
        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;
        }
```

```
        }
    }
  }
}
```

Output:-

```
Enter name : John
Enter type : Current
Enter acc-no: 10
Enter initial balance: 2000
Enter deposit amount: 500
Enter choice : 2
Enter amount : 600
Menu
1. deposit  2. withdraw  3. display.
```

```
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 Account Main
{
    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 intial 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("\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;
                }
            }
        }
    }
}
```

LAB-6

Create a package CIE which has two classes - Student & 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 sem. Create another package SEE. This class has array that stores SEE marks scored in five courses. Import two packages in a file that declares the final marks of a student.

```
package CIE;
import java.util.Scanner;
public class student
{
    protected String usn = new String();
    protected String name = new String();
    protected int sem;
    public void input Student details()
    {
        Scanner sc = new scanner (system.in);
        System.out.println("Enter usn");
        usn = sc.next();
        System.out.println("Enter name");
        name = sc.next();
```

```
        System.out.println("Enter sem");
        sem = sc.nextInt();
    }
    public void displayStudentdetails
    {
        System.out.println("USN is" + usn + "\n The name
            is" + name + "\n The sem is" + sem);
    }


public class Internals extends Student
{
    protected int marks[] = new int [5];
    public void inputciemarks()
    {
        Scanner s = new Scanner(System.in);
        int i;
        for (i=0; i<5; i++)
        {
            System.out.println("Enter marks of subjects"
                + (i++));
            marks [i] = s.nextInt();
        }
    }
```

External.java

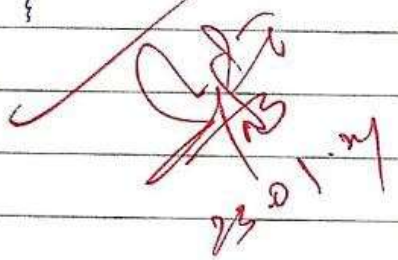
```
package SEE;
import CIE internals;
import java.util.Scanner;
public class External extends Internals
{
    protected int marks[];
    protected int final marks[];
    public external()
    {
        marks = new int [5];
        final Marks = new int [5];
    }
    public void input SEE marks()
    {
        Scanner scanner = new Scanner (System.in);
        System.out.println("Enter SEE marks for each
            course");
        for (int = 0; i < 5; i++)
        {
            final marks[i] = marks[i] / 2 + Super.internal
            Marks[i];
        }
    }
```

```
public void displayfinalmarks()
{
    displayStudentdetails();
    for(int i=0, i<5; i++)
    {
        System.out.println("Subject" + (i++)+ finalMarks[i]);
    }
}
```

Main.java.

```
import SEE.externals;
class main
{
    public static void main(String a[])
    {
        int numof_students = 2;
        externals finalmarks [] = new externals
                                  [num_of_students];
        for (int i=0;i<num_of_students; i++)
        {
            final marks[i]= new external();
            final marks[i]= inputStudentDetails();
            System.out.println("Enter CIE marks");
            System.out.println("Enter SEE marks");
```

```
        finalMarks[i]= input SEE marks();
    }
    System.out.println("Display data:\n");
        for(int i=0; i<num. of _students, i++)
        {
        finalMarks[i]. calculate Final Marks();
        finalMarks[i]. display Final Marks();
        }
    }
}
```



```
package CIE;
import java.util.Scanner;
public class Student {
    protected String usn = new String();
    protected String name = new String();
    protected int sem;
    public void inputStudentDetails() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter usn");
        usn = sc.next();
        System.out.println("Enter name");
        name = sc.next();
        System.out.println("Enter sem");
        sem = sc.nextInt();
    }
    public void displayStudentDetails() {
        System.out.println("USN is " + usn + "\nThe name is " + name + "\nThe sem is
" + sem);
    }
}
```

```
package CIE;
import java.util.Scanner;
public class Internals extends Student {
    protected int marks[];
    public void inputCIEMarks() {
        Scanner s = new Scanner(System.in);
        marks = new int[s];
        for (int i = 0; i < s; i++) {
            System.out.println("Enter marks of subjects " + (i + 1));
            marks[i] = s.nextInt();
        }
    }
}
```

```
package SEE;
import CIE.Internals;
import java.util.Scanner;
public class External extends Internals {
    protected int finalMarks[];
    public External() {
        finalMarks = new int[s];
    }
    public void inputSEEMarks() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter SEE marks for each Course");
        for (int i = 0; i < 5; i++) {
            finalMarks[i] = marks[i] / 2 + super.marks[i];
        }
    }
    public void displayFinalMarks() {
        displayStudentDetails();
        for (int i = 0; i < s; i++) {
            System.out.println("Subject " + (i + 1) + " Final Marks: " +
finalMarks[i]);
        }
    }
}



import SEE.External;


class Main {
    public static void main(String a[]) {
        int num_of_students = 2;
        External finalMarks[] = new External[num_of_students];
        for (int i = 0; i < num_of_students; i++) {
            finalMarks[i] = new External();
```

```
            finalMarks[i].inputStudentDetails();
            System.out.println("Enter CIE marks");
            finalMarks[i].inputCIEMarks();
            System.out.println("Enter SEE marks");
            finalMarks[i].inputSEEMarks();
        }
        System.out.println("Display data in");
        for (int i = 0; i < num_of_students; i++) {
            finalMarks[i].displayFinalMarks();
        }
    }
}
```

30/1/24

LAB-7

Write a program that demonstrates handling of exception in inheritance tree. Create a base class called "Father" and a 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;
class WrongAge extends Exception {
    WrongAge(String message){
        super(message);
    }
}
class InputScanner{
    static Scanner sc=new Scanner(System.in);
}
class father extends InputScanner {
    int fatherage;
```

```
Father() throws WrongAge {
    System.out.println("Enter Father's Age");
    fatherAge = sc.nextInt();
    if (fatherAge < 0) {
        throw new WrongAge("Age cannot be
                negative");
    }
}
void display() {
    System.out.println("Father's age is" + fatherAge);
}
}
class Son extends Father {
    int sonAge;

    Son() throws WrongAge {
        System.out.println("Enter Son's age");
        sonAge = sc.nextInt();
        if (sonAge > fatherAge){
            throw new Wrongage("Son's age
                cannot be greater than father's age");
        }
        else if (sonAge < 0) {
            throw new WrongAge("Age cannot
                be negative");
        }
```

```
    }
    void display(){
        System.out.println("Son's age is" +sonAge);
    }
}
public class ExceptionHandling{
    public static void main(String args[]){
        try{
            Son son = new Son();
            son.display();
        }
        catch(Wrongage e){
            System.out.println("Exception" + e.get
                                Message());
        }
    }
}
```

30/01/24

Output:

Enter father's age: 24

Enter son's age = 40

Error: ~~father's age~~ Son's age cannot be greater than father's age.

```
import java.util.Scanner;

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

}

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

class Father extends InputScanner {
    int fatherAge;

    Father() throws WrongAge {
        System.out.println("Enter Father's age");
        fatherAge = sc.nextInt();
        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");

        }
    }

    void display() {
        System.out.println("Father's age is " + fatherAge);
    }

}

class Son extends Father {
    int sonAge;

    Son() throws WrongAge {
        System.out.println("Enter son's age");
        sonAge = sc.nextInt();
        if (sonAge > fatherAge) {
            throw new WrongAge("Son's age cannot be greater than father's age");
        } else if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }

    }

    void display() {
        System.out.println("Son's age is " + sonAge);
    }

}

public class ExceptionHandling {
    public static void Main(String args[]) {
        try{
            Son son= new Son();
            son.display();
        }
```

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


    }
}
```

6/2/24

LAB-8

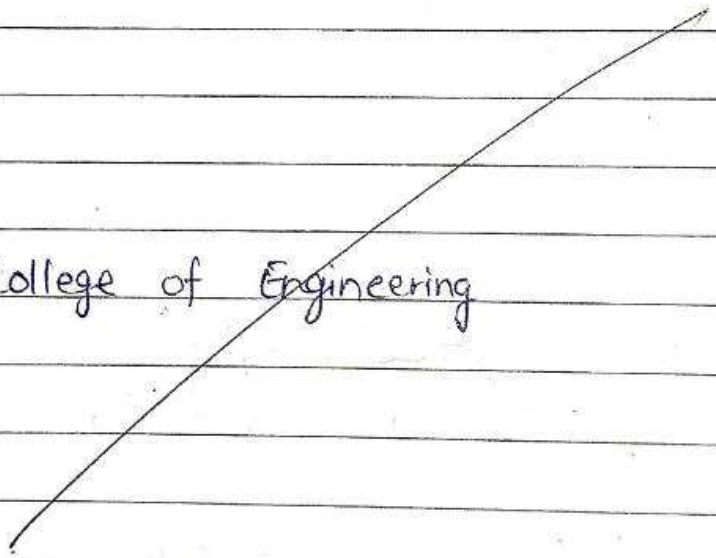
Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every 10 sec and another displaying "CSE" once every 2 sec.

```
class BMSThread extends Thread{
    @Override
    public void run(){
        while (true){
            System.out.println("BMS College of
                                Engineering");
            try{
                Thread.sleep(10000);
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

```
class CSEThred extends Thread{
    @override
    public void run(){
        while (true){
            System.out.println ("CSE");
            try{
                Thread.sleep(2000);
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    }
}
public class ThreadExample{
    public static void main(String[] args)
        {
        // Creating and starting the first thred.
        BMSThread bmsThread = new BMSThredd();
        bmsThread.start();
        CSEThread cscThread = new CSEThread();
        cseThread.start();
    }
}
```

OUTPUT :-

BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
:
:
:



```
class BMSThread extends Thread {
    @Override
    public void run() {
        while (true) {
            System.out.println("BMS College of Engineering");
            try {
                Thread.sleep(10000); // Sleep for 10 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class CSEThread extends Thread {
    @Override
    public void run() {
        while (true) {
            System.out.println("CSE");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        // Creating and starting the first thread
        BMSThread bmsThread = new BMSThread();
        bmsThread.start();

        // Creating and starting the second thread
        CSEThread cseThread = new CSEThread();
        cseThread.start();
    }
}
```

6/2/24

LAB-10

Demonstrate Interprocess Communication and deadlock.

→ An incorrect implementation

```
class Q{
int n;
Synchronised int get(){
System.out.println("Got:" +n);
return n;
}
Synchronised void put(int n){
    this.n = n;
    System.out.println("Put:" + 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++);
        }
    }
}
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 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:-

```
Put: 1
Got: 1
Got: 1
Got: 1
Got: 1
Got: 1
Put: 2
Put: 3
Put: 4
Put: 5
Put: 6
Put: 7
Got: 7
```

→ Correct implementation.

```
class Q {
    int n;
    boolean valueSet = false;
    synchronised int get(){
        while (!valueSet)
        try {
            System.out.println("\n Consumer waiting\n");
            wait();
```

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

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

OUTPUT:

```
Put : 1
Got : 1
Put : 2
Got : 2
Put : 3
Got : 3
Put : 4
Got : 4
Put : 5
Got : 5
```

```
class Q {
    int n;
    synchronized int get()
    {
        System.out.println("Got: " + n);
        return n;
    }
    synchronized void put(int n)
    {
    this.n = n;
    System.out.println("Put: " + 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++);
        }
    }
}
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 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.");
    }
```

```
}
```

```
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)
        {
            this.n = n;
            valueSet = true;
            System.out.println("Put: " + n);
            System.out.println("\nIntimate Consumer\n");
            notify();
        }
    }
    class Producer implements Runnable
    {
        Q q;
        Producer(Q q)
        {
            this.q = q;
            new Thread(this, "Producer").start();
        }
        public void run()
        {
```

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

13/2/24

LAB-10

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
        {
            System.out.println("A Interrupted");
        }
        System.out.println(name + " trying to call B.last()");
        b.last();
    }
    void last()
    {
        System.out.println(" Inside A.last ");
    }
}
```

```
class B
{
    synchronized void bar(A a)
    {
        String name = Thread.currentThread().getname();
        System.out.println(name + "entered B.bar");
        try
        {
            Thread.sleep(1000);
        }
        catch(Exception e)
        {
            System.out.println("B Interrupted");
        }
        System.out.println(name + "trying to call A.last()");
        a.last();
    }
    void last()
    {
        System.out.println("Inside A.last");
    }
}
class Deadlock implements Runnable
{
    A a = new A();
```

```
    B b=new B();
    Deadlock()
    {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this,"RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }
    public void run()
    {
        b.bar(a);
        System.out.println("Back in other Thread");
    }
    public static void main(String args[])
    {
        new Deadlock();
    }
}
```

O/P => Main Thread entered A.foo

RacingThread entered B.bar

MainThread trying to call B.last()

Inside A.last

Back in main thread

RacingThread trying to call A.last()

Inside A.last.

Back in other thread.

13/2/24

```
class A
{
    synchronized void foo(B b)
    {
        String name =Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");
        try
        {
            Thread.sleep(1000);
        }
        catch(Exception e)
        {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");
        b.last();

    }

    void last()
    {
        System.out.println("Inside A.last");
    }
}

class B
{
    synchronized void bar(A a)
    {
        String name =Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");

        try
        {
            Thread.sleep(1000);
        }
        catch(Exception e)
        {
            System.out.println("B Interrupted");
        }

        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    void last()
    {
        System.out.println("Inside A.last");
    }
}

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

```
    {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this,"RacingThread");
        t.start();
        a.foo(b); // get lock on a in this thread.
        System.out.println("Back in main thread");
    }

    public void run()
    {
        b.bar(a); // get lock on b in other thread.
        System.out.println("Back in other thread");
    }

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

}
```

20/2/24

LAB-9

WAP that creates a user interface to perform integer divisions.

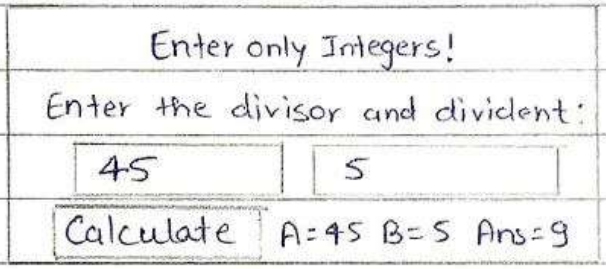
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
    SwingDemo() {
        Jframe jfrm = new Jframe("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.
                EXIT_ON_CLOSE);
        Jlabel jlab = new Jlabel("Enter the
                divider and divident:");
        JTextField ajtf = new JTextField(8);
        JText field bjtf = new JText field(8);

        JButton button = new JButton("Calculate");

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

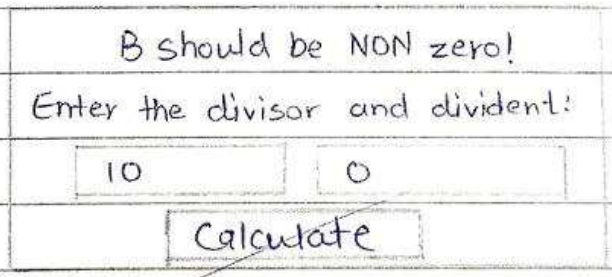
```
JLabel blab = new JLabel();
JLabel anslab = new JLabel();

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

ActionListener l = new ActionListener(){
    public void actionPerformed(ActionEvent 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)
    {
```

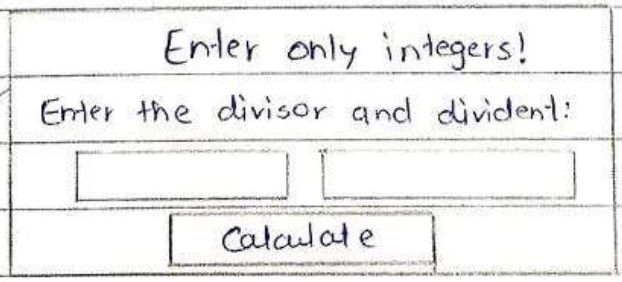
```
try{
    int a=Integer.parseInt(ajtf.getText());
    int b=Integer.parseInt(bjtf.getText());
    int ans = a/b;

    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!");
}
}
});
jfrm.setVisible(true);
}
```

```
public static void main (String Args[]) {
    SwingUtilities.invokeLater (new Runnable() {
        public void run() {
            new SwingDemo();
        }
    });
  }
}
```

O/P =>







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

Some definitions:-

① JFrame => It is an essential component of Java Swing. JFrame is a class that allows to create and manage a top-level window in a Java Application.

② Setsize => sets the size of this Dimension object to the specified width and height.

③ setLayout => allows you to set the layout of the container often a JPanel, to say FlowLayout, BorderLayout, GridLayout or whatever layout desired.

④ set default close operation => method provided by javax.swing, used to create GUI-based applications.

⑤ JLabel => is a built-in Java Swing class that lets you display information on a JFrame.

⑥ JTextfield => a lightweight component that allows

the editing of a single of text.

⑦ add action Listener ⇒ an interface in java.awt.event package that recieves a notification whenever any action is performed in the application.

⑧ setText ⇒ substitutes the characters sText for the text in the text field.

⑨ add frame ⇒ used to add a new frame inside the existing one.