Lab Program - 1 : Quadratic Equation

```
import java.util.Scanner
public class quadratic
{
  Public static void main (String [] args)
  {
    float a, b, c, r1, r2, d;
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter coefficient of x^2:");
    a = sc.nextFloat ();
    System.out.println ("Enter coefficient of x:");
    b = sc.nextFloat ();
    System.out.println ("Enter constant:");
    c = sc.nextFloat ();
    if (a==0)
      {
        System.out.println ("Invalid input");
      }
    else
      {
        d = (float) Math.pow (b,2) - 4*a*c;
        if (d>0)
          {
            r1 = (float) (-b + Math.sqrt (d)) / (2*a);
            r2 = (float) (-b - Math.sqrt (d)) / (2*a);
            System.out.println ("The roots are:" +
                                r1 + "and" + r2);
          }
        else if (d==0)
          {
            r1 = (float) -b / (2*a);
            System.out.println ("root is" + r1);
          }
```

```
        else
        {
            System.out.println ("No real sol");
        }
    }
}
```

Output:

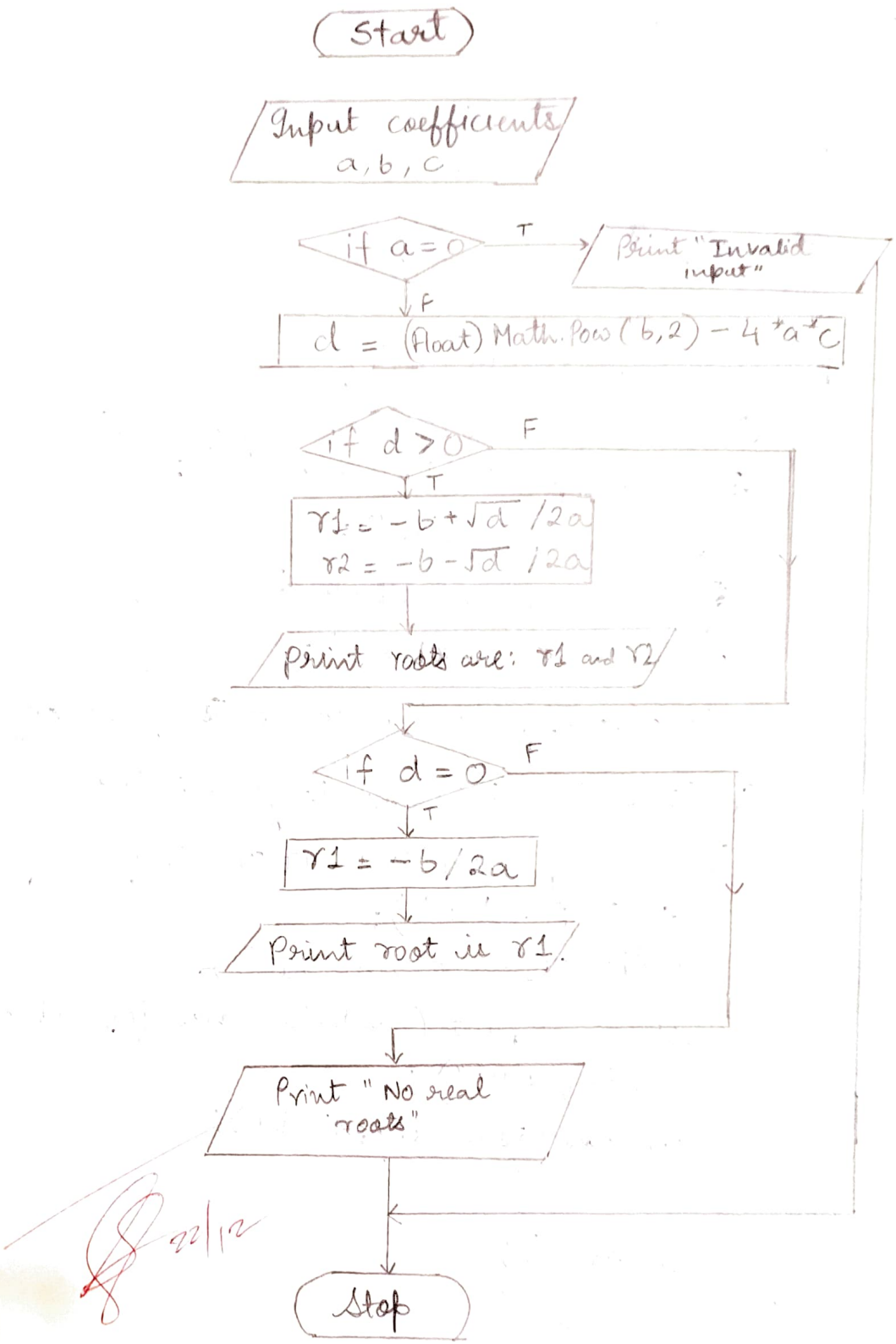
Enter coefficient of x^2:
4
Enter coefficient of x:
6
Enter constant:
2
The roots are : -0.5 and -1.0.

Enter coefficient of x^2
1
Enter coefficient of x
1
Enter constant:
1
No real solution

Algorithm:

Step 1: Start
Step 2: Input the variables (a, b, c)
Step 3: If a = 0, Print invalid input, else
d = $b^2 - 4ac$. If d > 0 r1 = $-b+\sqrt{d}/2a$; r2 = $\frac{-b+\sqrt{d}}{2a}$.
If d = 0, r1 = $-b/2a$. Else, print no solution
Step 4: Print the output
Step 5: Stop

Flowchart:



22/12

LAB PROGRAM - 2

```
import java.util.Scanner;
class student {
    string usn;
    String name;
    int [] credits = new int [8];
    int [] marks = new int [8];

    Public void acceptdetails () {
        Scanner sc = new Scanner (System.in);
        System.out.print ("Enter usn: ");
        usn = sc.nextLine();
        System.out.println ("Enter Name: ");
        name = sc.nextLine();
        System.out.print ("Enter details of each subject:")

        for (int i=0; i< credits.length; i++) {
        System.out.print ("\n Enter credits for subject"
            + (i+1) + " : ");
        credits [i] = sc.nextInt();
        System.out.print ("\n Enter marks for subject"
            + (i+1) + ":");
            marks [i] = sc.nextInt();
        }
        sc.close();
    }

    Public double calcSGPA() {
        int totalCredits = 0;
        int weightedsum = 0;
        double ans;
        for (int i = 0; i < credits.length; i++) {
```

```
        totalCredits += credits[i];
        int gradePoints;
        gradePoints = (marks[i]/10) + 1;
        if (gradePoints == 11) {
            gradePoints = 10;
        }
        else if (gradePoints <= 4) {
            gradePoints = 0;
        }
        weightedSum += gradePoints * credits[i];
    }
    ans = (double) weightedSum / (double) totalCredits;
    return ans;
}

Public class sgpa {
    Public static void main (String [] args) {
        Scanner sc = new Scanner (System.in);
        Student student = new Student ();
        student.acceptDetails();
        System.out.println ("Student Details:");
        System.out.println ("USN:" + student.usn);
        System.out.println ("Name:" + student.name);
        double sgpa = student.calcSGPA ();
        System.out.println ("\nSGPA:" + sgpa);
        scanner.close ();
    }
```

Algorithm:

Step 1: Start

Step 2: Initialise usn, name as string, credits and marks as array.

Step 3: Create method accept details for accepting input.

Step 4: Create another method for calculating sgpa.

Initialise total credits and weighted sum, ans.

total credits = credits [i] + total credits

if gradepoints = 11, set it to 10.

If gradepoints < 4, set it to 0.

weighted sum = gradepoints * credits [i]

ans = weighted sum / total credits.

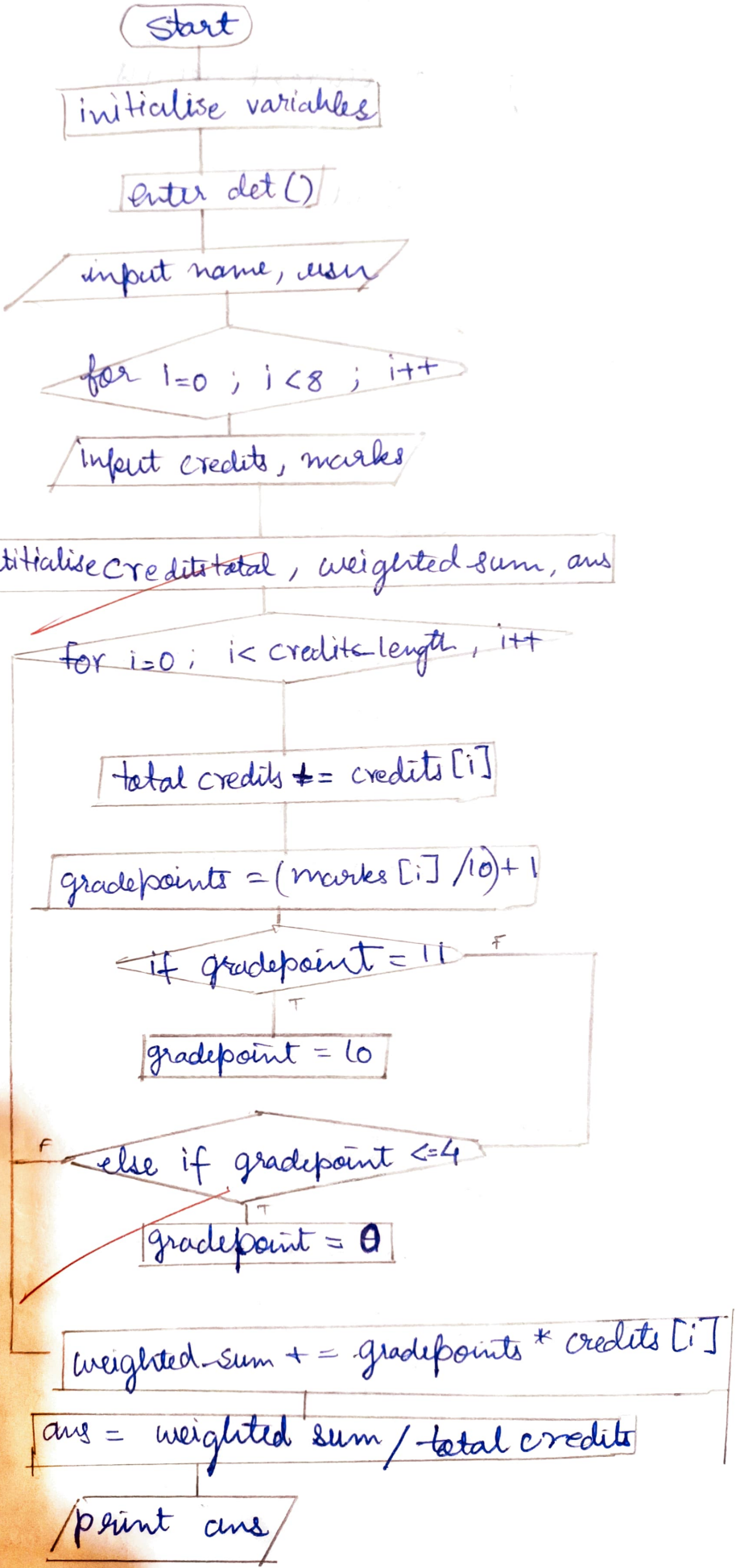
Return ans.

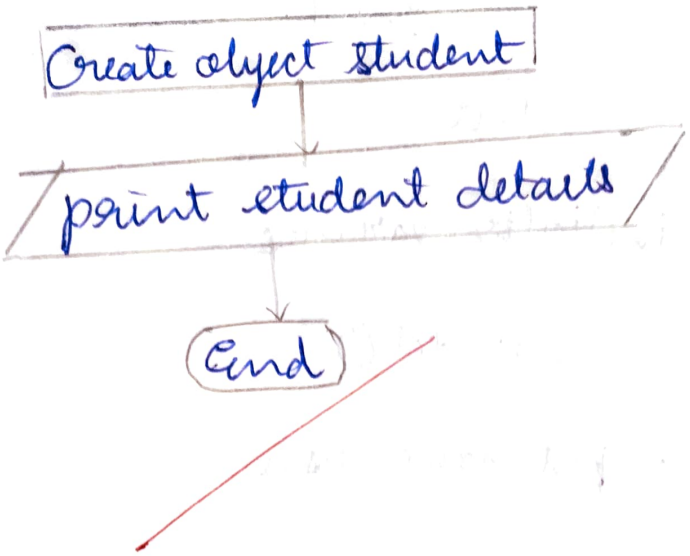
Step 5: Create public class sgpa to access all the methods declared above.

Step 6: Print final sgpa.

Step 7: Stop

Flowchart:


Start
initialise variables
Enter det()
input name, usn
for i=0 ; i<8 ; i++
input credits, marks
intitialise credits total , weighted_sum, ans
for i=0 ; i< credits_length , i++
total credits += credits[i]
gradepoints = (marks[i]/10)+1
if gradepoint = 11
T
F
gradepoint = 10
else if gradepoint <=4
F
T
gradepoint = 0
weighted_sum += gradepoints * credits[i]
ans = weighted sum / total credits
print ans


Create object student
print student details
End

LAB PROGRAM - 3

```
import java.util.scanner
class book {
  String name,
  String author;
  float price;
  int pages;
Void set_details() {
  Scanner sc = new scanner (System.in);
  System.out.println ("Enter book Name");
    name = sc.nextLine();
  System.out.println ("Enter author");
    author = sc.nextLine();
  System.out.println ("Enter price");
    price = sc.nextFloat();
  System.out.println ("Enter number of pages");
    pages = sc.nextInt();
  }
Void get_details()
{
    String details = toString();
    System.Out.println (details),
  }
Public string tostring ()
{
  return "The book" + name + " was written by" +
  author + " and it consists of " + pages + "pages
  and costs" + price;
}
```

```
public static void main (String[] args) {
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter number of books you
                         want to generate");
    int n = sc.nextInt ();
    book b[] = new book [n];
    for(int i=0 ; i<n ; i++)
    {
        b[i] = new book();
        b[i].setdetails ();
    }
    System.out.println ("Book details");
    System.out.println ();
    for (int i=0; i<n; i++)
    {
        b[i] = get.details ();
    }
}
}
```

Algorithm

Step 1: Start

Step 2: Create class book, initialise

Step 3: Create method set_details to get input for name, author, price and number of pages.

Step 4: Create method get details

Step 5: Define to string within get details,

Step 7: Input the number of books to be generated

Step 8: Create an array of books, assign objects and execute set details method.

Step 9: Print book details

Step 10: Stop

LAB PROGRAM-4

```
import java.util.scanner
abstract class Shape {
 int x, y;
 abstract void area();
 Public static void main ( String [] args)
 {
    Shape obj1 = new circle();
     obj1.area();
     Shape obj2 = new Rectangle();
     obj2.area();
     Shape obj3 = new Triangle();
     obj3.area();
  }
}
Class circle extends Shape {
 circle() {
    scanner sc = new Scanner (System.in)
    System.out.println ("Enter radius:")
        x = sc.nextInt();
        y = x; }
    Void area() {
        return 3.14 * x * y;
        System.out.println (" Area of the circle");
    }
}
```

```
Class rectangle extends shape {
 rectangle () {
    Scanner sc = new scanner (System.in)
    System.out.println ("Enter the length and bredth");
     x = sc.nextint ();
     y = sc.nextint ();
    }
     Void area () {

      System.out.println ("Area of rectangle is"
                                + x * y );

      }
}

Class Triangle extends Shape {
 Triangle () {
    Scanner sc = new Scanner (System.in)
    System.out.println ("Enter the base and height
      of the triangle");
     x = sc.nextInt ();
     y = sc.nextInt ();
  }
  void area ()
{

   System.out.println ("Area of triangle" + 0.5 * x * y );
  }
}
```

Algorithm

Step 1: Start

Step 2: Declare abstract class shape

Step 3: Initialise objects circle, rectangle, triangle

Step 4: Extend ~~method~~ class shape to ~~Drica~~ circle, Use method area to print area.

Step 5: Extend ~~method~~ class shape to rectangle, Use method area to print area

Step 6: Extend ~~metho~~ Class to triangle, Use method area to print area.

Step 7: Stop

Tu 18/01/24

Execute it

LAB PROGRAM - 5

```
import java.util.scanner;
class Account {
  String customerName;
  long accno;
  String accountType;
  double balance;
 public Account ( String customerName, String
  accountType, long accno) {
    this. customerName = customerName;
    this. accno = accno ;
    this. accountType = accountType;
    this. balance = 0.0 ;
  }
 public void display_balance () {
   System.out.println (" Account number " +accno) ;
   System.out.println (" Customer name :" + customerName);
   System.out.println (" Account type :" + accountType);
   System.out.println (" Balance : $" + balance) ;
  }
}
 class currentacc extends Account {
   double minBalance ;
   double serviceCharge ;
  public currentacc ( String customerName, long
                      accno)
  {
    super (customerName , accno, "current") ;
  }
```

```
this.minBalance = 500.0;
this.serviceCharge = 50.0;
}
Public void withdraw (double amount) {
 if ( balance - amount >= minBalance)
   {
      balance- = amount;
  System.out.println ("Withdrawl successful. Curren
   balance :" + balance);
   }
  else {
   System.out.println ("Insufficient Balance,
   withdrawl unseccessful");
   }
}
 Public imposeservicecharge () {
  if ( Balance < minBalance) {
     Balance -= Service Charge;
  system.out.println ("Service charge imposed
   current balance :" + balance);
   }
  }
}

Class Savingsacc extends Account {
    double interestrate;

Public Savingsacc ( String customername, long accno
 {
     super (customername, accno, "savings");
     this. interestrate = 0.05;
  }
```

```
public void depositInterest () {
    double interest = balance * interestrate;
      balance += interest;
    System.out.println ("Interest deposited. Current
     balance : "+ balance);
}

public void compoundInterest (double initialAmount,
 int term) {
    double compoundinterest = initialAmount *
      Math.Pow ((1 + interestrate), term) - initialAmount;
     balance += compoundInterest;
    System.out.println (" compond interest deposited"
      Balance " + balance);
  }
}

Public class Bank {
  Public static void main (String [] args) {
      Scanner sc = new scanner (System.in)
    System.out.println (" Choose amount type : "),
    System.out.println ("1. Current");
    System.out.println ("2. Savings");
    System.out.println ("Enter choice (1 or 2) : ");
    int choice = scanner.nextInt ();
    System.out.println ("Enter customer name : ");
    String customaNamer = sc.nextLine ();
    System.out.print ("Enter account Number : ");
```

```
long accno = sc.nextLong();
if (choice == 1) {
    currentacc currentaccount = new current acc
                                (customerName, accno);
    System.out.println("Enter initial balance:");
    double initialBalance = sc.nextDouble();
    currentaccount.balance = initialBalance;
    System.out.println("Enter withdrawal amount");
    double withdrawAmount = sc.nextDouble();
    currentaccount.withdraw(withdrawalAmount);
    currentaccount.imposeServiceCharge();
    currentaccount.displaybalance()
}
else if (choice == 2) {
    Scanningsacc savingsaccount = new savingsacc(
                                customername, accno);
    System.out.println("Enter intial balance");
    double initialbalance = sc.nextDouble();
    Savingsaccount.balance = initialbalance;
    System.out.println("Enter withdraw amount:");
    double withdrawal amount = sc.next double();
    savingsaccount.withdraw(withdrawalAmount);
    System.out.println("withdrawal successful.
        Balance:" + savingsaccount.balance);
```

```
System.out.println ("Enter interest rate:");
double interestrate = SC.nextDouble();
Savingsaccount.interestrate= interestrate;
Savingsaccount.displaybalance();
System.out.println ("Enter no of years");
int year = SC.nextInt();
Savingsaccount.compoundinterest (Initialbalance
                                 , year);
Savingsaccount.displayBalance();
}
else
{
System.out.println ("Invalid choice");
}
}
}
```

Algorithm:

Step 1: Start

Step 2: Create class account, initialize constructor accordingly

Step 3: Create a method display balance within the class account

Step 4: Extend class Account to currentacc

Step 5: Create method withdraw below inside current account.

Step 6: Create method imposeservicecharge.

Step 7: Create class Savings which extends Account

Step 8: Create method depositInterest inside this class.

Step 9: Create method compound intrest for calculating compound interest

Step 10: Create class bank

Step 11: Take inputs from the user like name, account number, amount, balance, withdrawl amount and execute the methods

Step 12: Stop

19/01/24

LAB PROGRAM - 6 Date - 2/2/24

8) Creation of a package CIE

```
Package CIE;
import java.util *
Public class Student {
  Public String usn, name;
  Public int sem;
  Public void details () {
    Scanner sc= new Scanner(System.in);
    System.out.println ("Enter name, usn and
                         semester:");
    name = sc.nextLine();
    usn = sc.nextLine();
    sem = sc.nextInt();
  }
}
Package CIE;
Public class internals {
  Public int imarks[]= new int [5];
  }
  Package SEE;
  import CIE.student;
  Public class external extends Student {
   Public int smarks [] = new int [5];
  }
```

```
import java.util.*;
import CIE *;
import SEE *;
Public class Finalmarks {
  Public static void main (String args [])
  {
    int fmarks= new int [5];
      System.out.println("Enter the number
          of students : ");
    int n = sc.nextInt ();
  SEE.external st = new SEE.external [n];
  CIE.internal ci = new CIE.internal [n];
 for (int i=0 ; i<n ; i++)
  {
      st [i] = new SEE.Externals ();
      ci [i] = new CIE.internals ();
    System.out.println (" Enter details :");
    st [i].details ();
for (int j=0 ; j<5 ; j++)
{
  System.out.println (" Enter internal marks
  of subjects" + (j+1));
      ci [i].imarks [j] = sc.nextInt ();
  fmarks [j] = ci [i].imarks [j] + st [i].
                                  smarks [j];
}
```

```
System.out.println("Final marks of"+ st[i].name);
    for (int k=0, k<5; k++)
    {
      System.out.println("course"+(k+1)+"="
                         + fmarks[k]);
    }
  }
}
```

Algorithm

Step 1: Start

Step 2: Create package CIE

Step 3: Include class Student in the package, store details like usn, name and semester.

Step 4: Define method for accepting details from the user.

Step 5: From package CIE, create class internals.

Step 6: Create package SEE and import student from CIE.

Step 7: Create class external which extends students

Step 8: Create class finalmarks for displaying the final details.

Step 9: Input the number of students you want to store the data for.

Step 10: Declare objects for accepting the marks and take input.

Step 11: Display final marks along with names of students.

Step 12: Stop.

02.02.24

LAB PROGRAM - 7

Date : 16/2/24

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

class Father {
    private int age;

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

    public int getAge() {
        return age;
    }
}

class Son extends Father {
    private int sonAge;

    public Son (int fatherage, int sonage) throws
                WrongAge {
        super (fatherAge);
        if (sonAge >= fatherage) {
            throw new WrongAge (" Son's age cannot be greater
                                than or equal to father's age");
        }
        this.sonAge = sonage;
    }

    public int getsonAge () {
        return sonage;
    }
```

17/2/24

```
}
Public class inheritanceDemo {
  Public static void main (String[] args) {
    try {
        Father father = new Father(45);
        Son son = new Son(45, 30);
        System.out.println("Father's age:" + father.getAge());
        System.out.println("Son's age:" + son.getSonAge());
      }
    catch (WrongAge e) {
      System.out.println("Exception caught " + e.get
                                    Message());
      }
  }
}
```

LAB PROGRAM-8

Date: 16/2/23

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

→ class Message extends Thread {

String message;
int interval;

Public Message (string message, int interval) {

this.message = message;
this.interval = interval;

}

Algorithm:

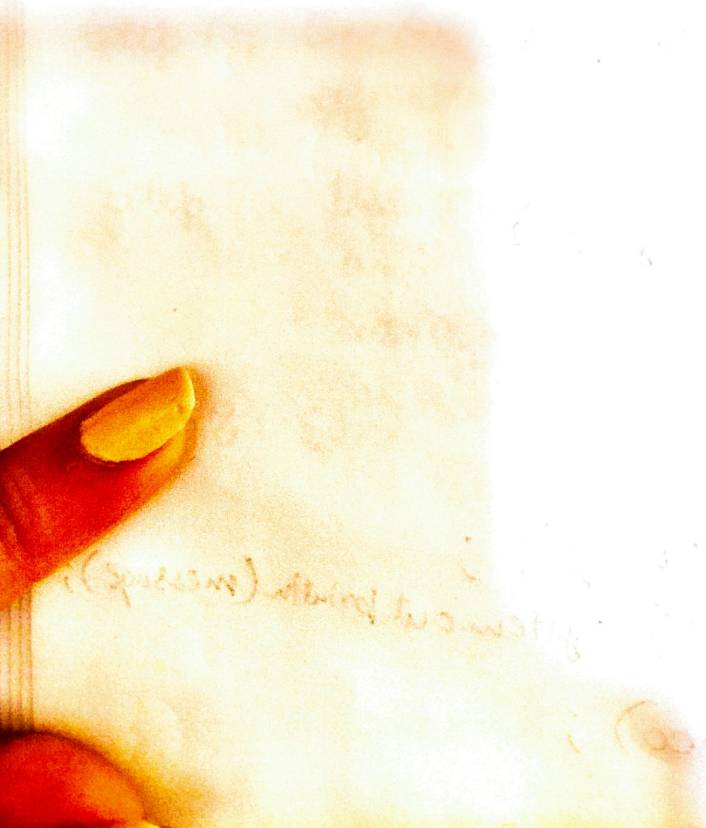
Step 1: Start
Step 2: create a class Message, which extends Thread.
Step 3: Create a constructor within the class to display the message and specify the time interval.
Step 4: Create the method run() for Message, in the try block, print out the message and specify the time interval after which the message is to be displayed.
Step 5: In the catch block, try catching interrupted exceptions.
Step 6: Create a main class, create two thread objects, declare message and interval time.
Step 7: Start the threads by .start() method.
Step 8: Stop

PROGRAM-8

```
class FirstThread extends Thread {
    String message;
FirstThread ( String message) {
    this. message = message;
{
public void run () {
    system.out.println (message);
    try { for (int i=0; i<10; i++) { System.out.println (message);
        Thread. Sleep (10000);
    }}
    catch ( Interrupted Exception e)
    {
        system.out.println ("Exception caught");
    }
}
class SecondThread extends Thread {
    String message;
    SecondThread ( String message) {
        this. message = message;
    }
public void run () {
    System. out.println (message);
    try { for(int i=0; i<10; i++) { system.out.println (message);
        Thread. sleep (2000);
    }}
    catch (Interrupted exception e)
    {
        System.out.println ("Exception caught");
    }
}
}
```

```
Public class DemoThread {
  public static void main (String[] args) {
    FirstThread t1 = new FirstThread ("BMS
                                college of Engineering");
    SecondThread t2 = new SecondThread ("CSE");
    t1.start();
    t2.start();
  }
}
```

16.02.24



LAB PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new JFrame("Divider Apb");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel jlab = new JLabel("Enter Divider and dividend:");
        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(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 ae) {
                System.out.println("Action event from text
                    field");
            }
        };
```

```
ajtf.addActionListener(1);
bjtf.addActionListener(1);

button.addActionListener(new ActionListener() {

 public void actionPerformed(ActionEvent ent) {

  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("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[]) {
   String Utilities.invokeLater(new Runnable() {
   public void run() {
          new StringDemo();
   }
```

```
});
System.out.println("Aastha Priya" "1BM22CS003");
}
}
```

Functions:

1. Label: Object that can display either text as an integer.
2. ActionListener: The listener interface for receiving action events
3. ActionEvent: A semantic event which indicates that a component defined action occured.
4. SetText() - Defines the single line of text
5. setVisible() - Shows or hides this window depending on the values as parameters.
6. InvokeLater: To operate or perform any task on event dispatcher thread.

23.02.24
Execute it