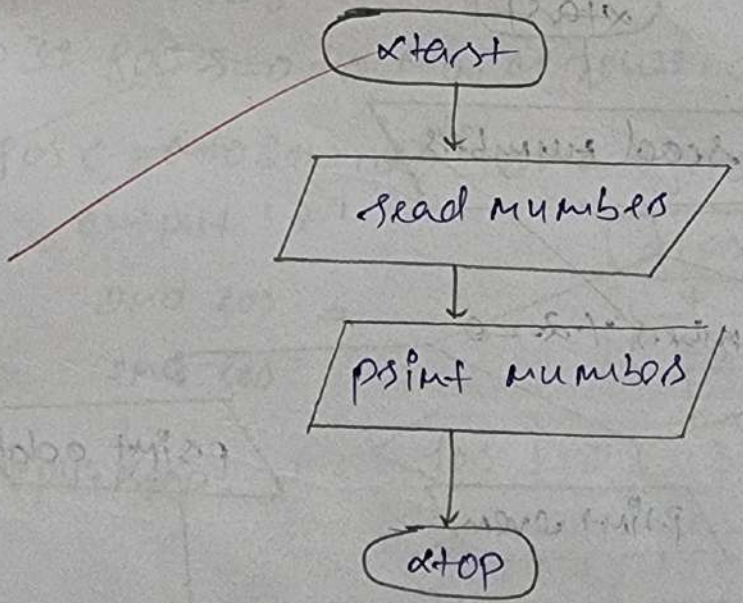
1. To print an int (entered by the user)

```
import java.util.scanner;
Public class Helloworld
{ public static void main (string[] args)
  { scanner reader = new scanner (system.in);
    system.out.print("enter a number");
     int number = reader.nextInt();
    system.out.println("you entered:"+number);
  }
}
```

Algorithm:

step 1: start
step 2: read numbers from user
step 3: display numbers
step 4: stop

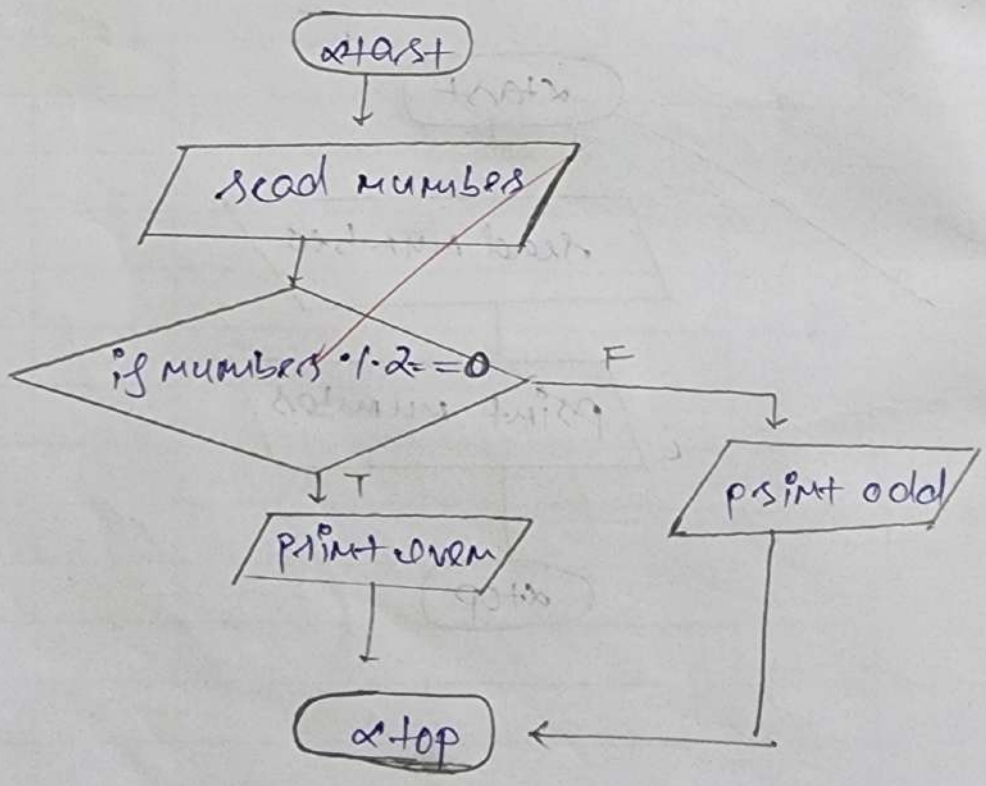


2. To check if a numbers is even or odd.

```
import java.util.scanner;
public class odd
{ public static void main(String [] arg)
  { int num = 5;
    if (num%2 == 0)
  System.out.println("number is even");
    else
      System.out.println("number is odd");
  }
}
```

Algorithm

step1: start → step2: read number from user.
step3: if number %2 == 0
        print number is even
       else
        print number is odd
step4: END



3. import java.util.Scanner;

```
public class demo{
{ public static void main(String [] args)
{
    int i,s;
    for (s=0; s<i; s++;)
    {
      for(i=0; i<=s; i++)
        {
          System.out.print(" * ");
        }
        System.out.println();
    }
}
}
```

Algorithm:

step1: start

step2: let s=8

step3: for s=0 to NO. of rows, do

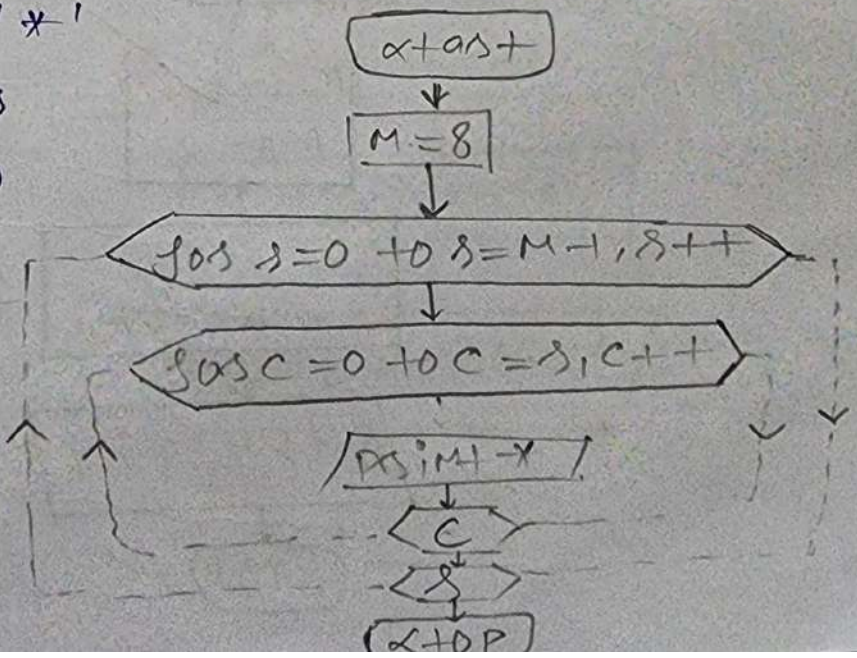
for c=0 to row, do ~~do no of rows, do~~

output '*'

end for

end for

step4: stop.



(4) find quotient & remainder

```
import java.util.Scanner;
public class ADITYA
{
public static void main (String[] args)
  {
   int num1 = 15; num2 = 2;
  int quot = num1/num2;
  int rem = num1 % num2;

System.out.println("Quotient = " + quot);
System.out.println("Remainder =" + rem);
  }
}
```

Algorithm:

step1: start

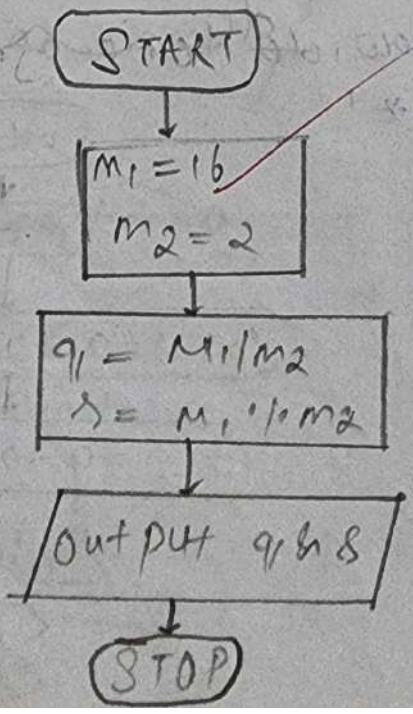
step2: num1 = 15, num2 = 2;

step3: quot = num1/num2

step4: rem = num1 % num2

step5: print quot, rem

step6: stop.

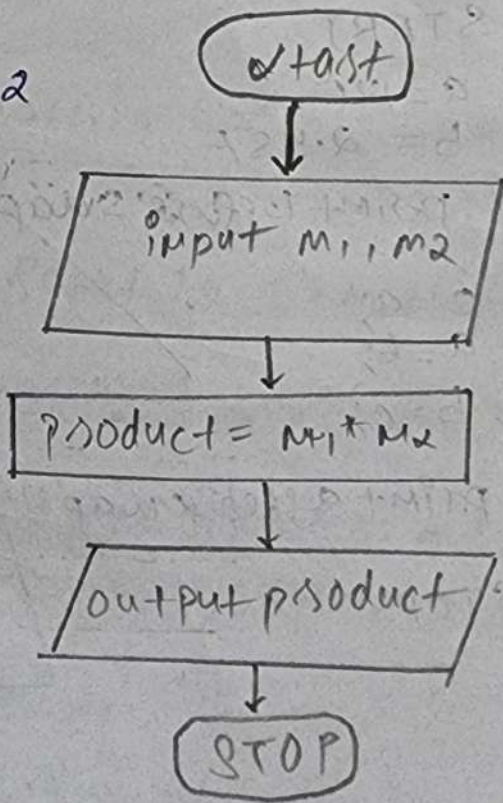


(5) Multiply 2 no.

```
→ import java.util.Scanner;
  public class demo
  {
    public static void main(String[]args)
    {
      Scanner input = new Scanner(System.in)
    System.out.println("Enter no.");
     int n1 = input.nextInt();
    System.out.println("enter no.");
     int n2 = input.nextInt();
     int p = n1*n2;
    System.out.println("PRODUCT"+p);
    }
  }
```

Algorithm:

Step1: START
Step2: input n1, n2
Step3: p = n1 * n2
Step4: output p
Step5: END



6. Swap two no. using temp variable

```
import java.util.Scanner;
public class swap
{
    public static void main(String[] args)
    {
        float a=1.2F, b=2.45F, c;
        System.out.println("before swap");
        System.out.println(a+" "+b);
        c=a;
        a=b;
        b=c;
        System.out.println("After Swap");
        System.out.println(a+" "+b);
    }
}
```

Algorithm:

Step1: START

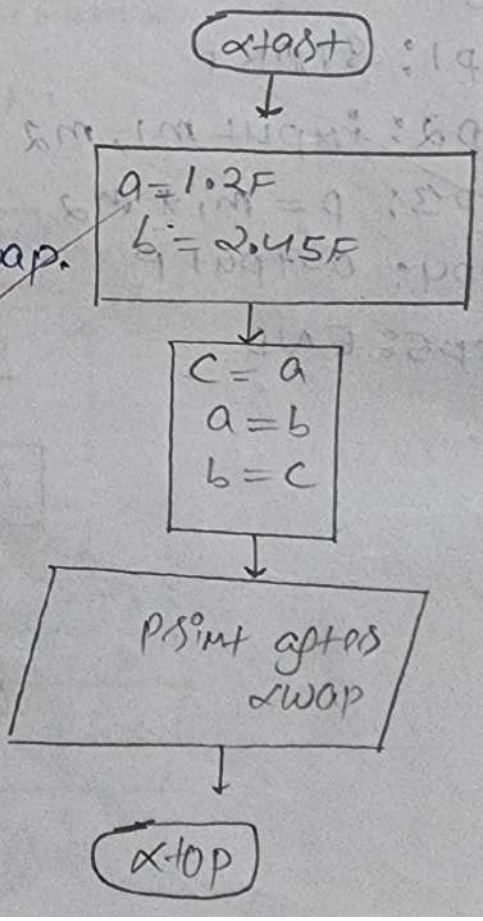
Step2: a=1.2F
b=2.45F

Step3: print before swap.

Step4: c=a;
a=b;
b=c;

Step5; print after swap

Step6: STOP



LAB Program-1 (Quadratic):

```
import java.util.*;
public class quadratic
{ public static void main (String[] args)
{
    float a,b,c,s1,s2,det;
Scanner sc = new Scanner (System.in);
System.out.println("Enter coeff of x^2");
a = sc.nextfloat();
System.out.println("Enter co-eff of x:");
b = sc.nextfloat();
System.out.println("Enter constant:");
c = sc.nextfloat();
if (a==0)
  {
    System.out.println("Invalid Input");
  }
else
  {
    det = (float)Math.pow(b,2)-4*a*c);
    if (det>0)
        { s1 = (float)(-b+Math.sqrt(det)/(2*a);
    s2 = (float)(-b-Math.sqrt(det))/(2b);
System.out.println("The roots are:
                    +s1+"and"+s2);
        }
else if (det==0)
        {s1 = (float) -b/(2*a);
System.out.println("root is = "+s1);
```

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

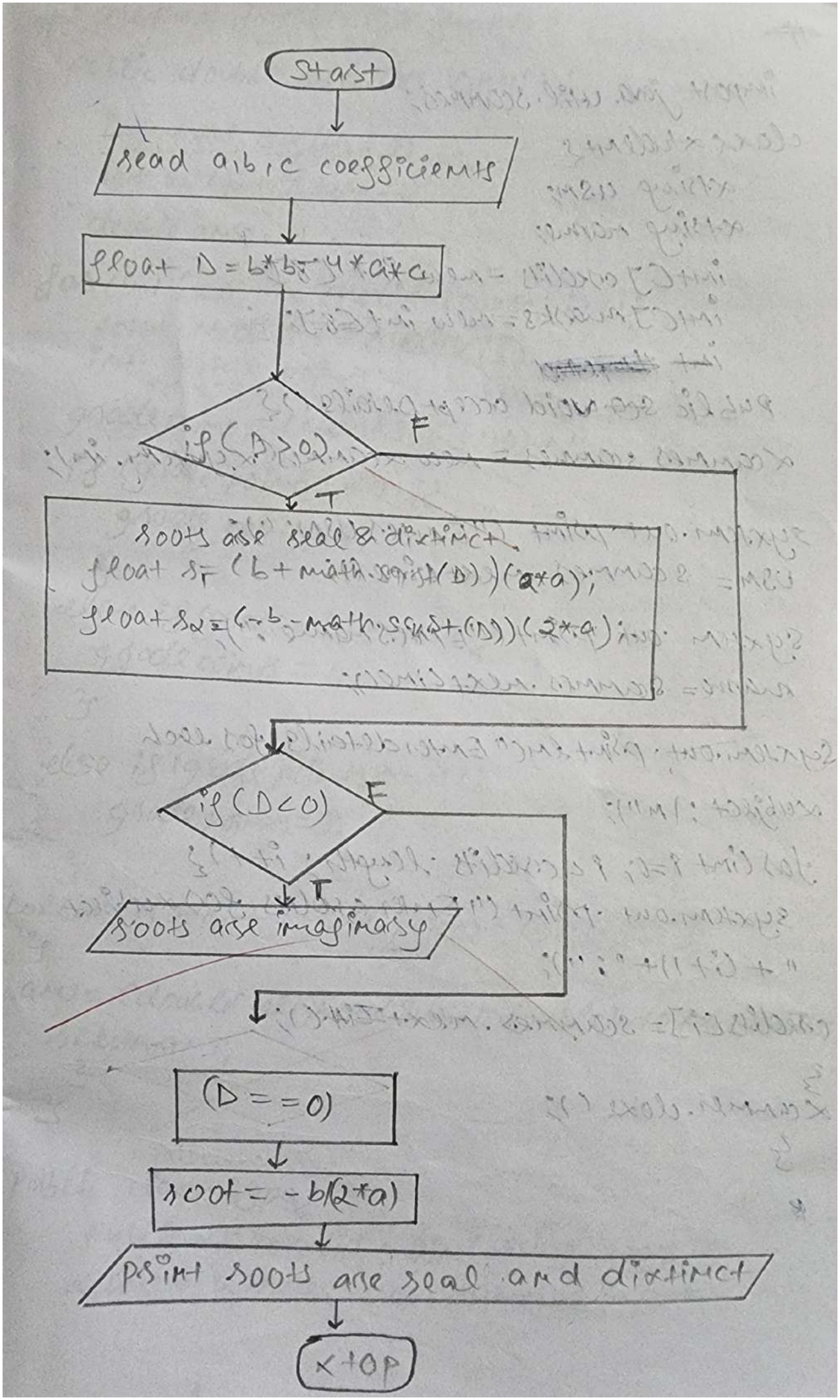
OUTPUT:

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.

22/12


Start
read a,b,c coefficients
float D = b*b - 4*a*c
if (D>=0)
F
T
roots are real & distinct
float r1 = (-b + math.sqrt(D))/(2*a);
float r2 = (-b - math.sqrt(D))/(2*a);
if (D<0)
F
T
roots are imaginary
(D==0)
root = -b/(2*a)
print roots are real and distinct
stop

#

```
import java.util.Scanner;
class student{
    String USN;
    String name;
    int[] credits = new int[8];
    int[] marks = new int[8];
    int [redacted]
public void acceptDetails(){
Scanner scanner = new Scanner(System.in);

System.out.print("Enter USN: ");
USN = scanner.nextLine();

System.out.print("Enter name :");
name = scanner.nextLine();

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

#

method to calculate SGPA →

```
public double calculateSGPA(){
    int totalCredits = 0;
    int weightedSum = 0;
    double gpa;
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 = 10;
}
else if (gradePoints <= 4){
    gradePoints = 0;
}
weightedSum += gradePoints * credits[i];
}
gpa = (double) weightedSum / (double) totalCredits;
return gpa;
}
}
public class Sgpa{
    public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
```

```
Student student = new student();
xtudent.acceptDetails();
System.out.println("Student Details:");
System.out.println("USN:" + student.usn);
system.out.println("name: " + student.name);
doubl sgpa = student.calculateSGPA();
System.out.println("SGPA:" + sgpa);
xcanner.close();
}
}
```

* Algorithm:

step1: start

step2: initialized USN, name as string, credits and marks as array.

step3: create method accept details for accepting input.

step4: create another method for calculating sgpa, initialize total credits and weighted sums, ams

step5: total credits = cre

Algorithm:

step(1) start

step(2) define a class named student declare instance variables: string usn; string name; int[] credits, int[] marks.

step(3) ~~define~~ initialize scanner to read inputs. prompt user to enter USN and name.
using a loop for each subject:
prompt user for credits and marks.
close scanner.

step(4) define method calculateSGPA()
initialize variables total credits and weighted sum.

total credits = Σ(credits)
SGPA = weighted sum / total credits
loop through each subjects credits and marks.

step(6) update weighted sum with grade points * credits.
update total credits with the sum of credits.
Return SGPA.

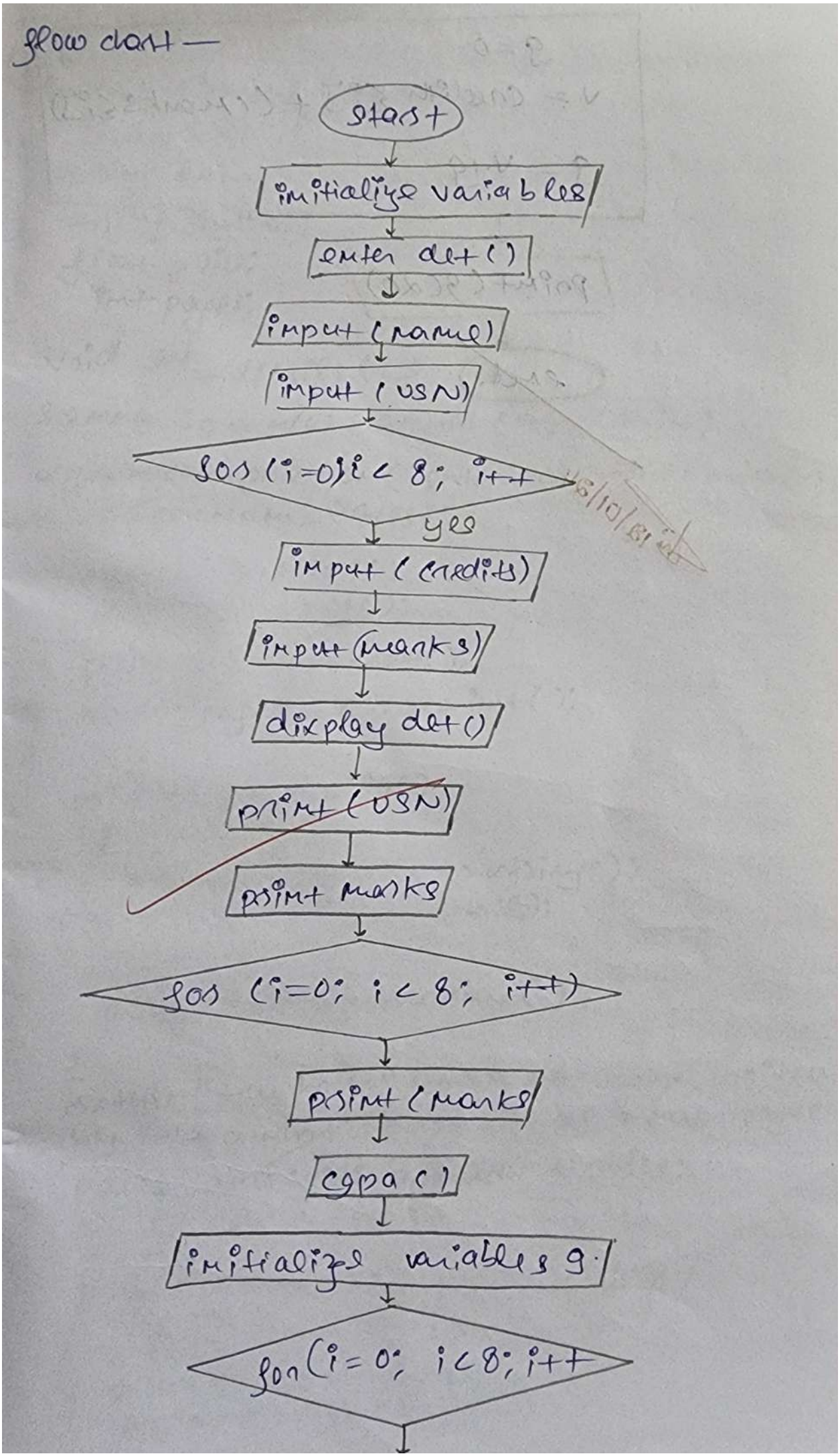
step(7) initialize scanner to read inputs.
create a student object
accept student details using
acceptDetails() method.
calculate SGPA using calculate
SGPA() method
Display calculated SGPA.

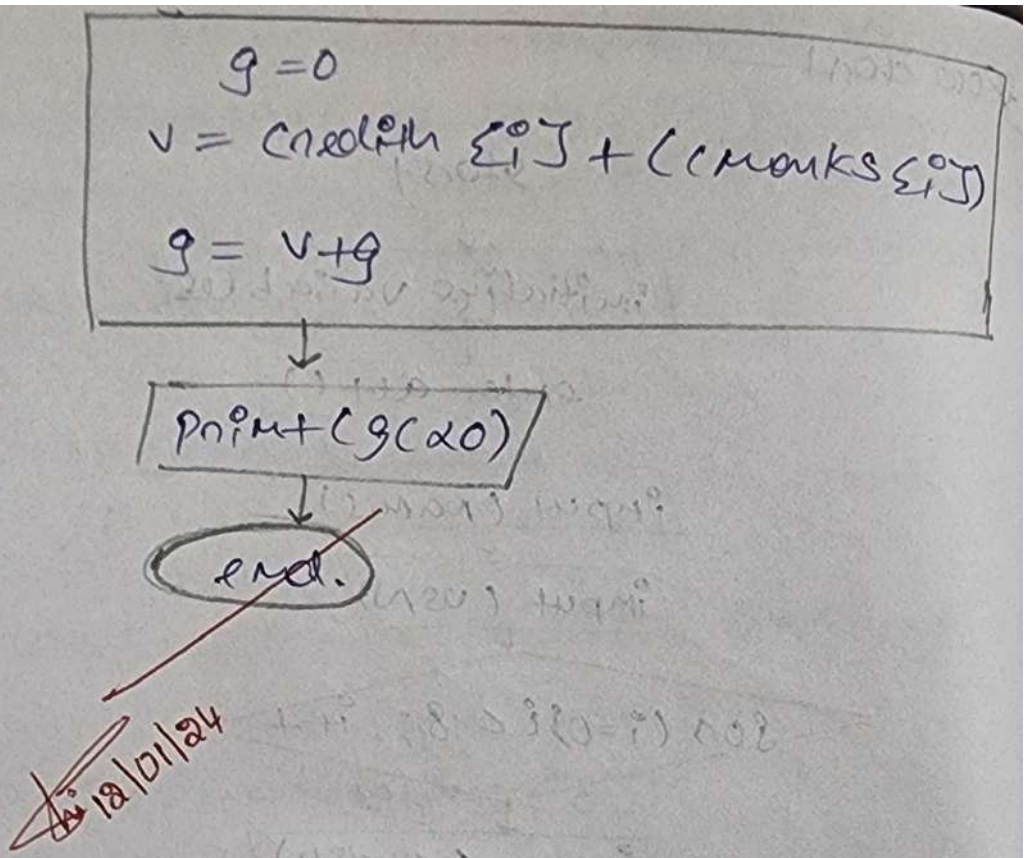
step(8) End.

Ques: Develop a Java prog. to create a class student with number USN, name, an array credit and an array marks. include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;
public class sgpa problem {
    String USN;
    String name;
    private static int credit[] = {4,4,3,3,
                                   3,1,1,1};
    int marks[] = new int[8];
    Scanner s = new Scanner (System.in)
    public void get_details() {
    System.out.println("Enter your USN");
    USN = s.next();
    System.out.println("Enter name");
    name = s.next();
    }
public void set_marks(){
    System.out.println("Enter your marks
                        in ");
for (int i=0; i<8; ++i){
    marks[i] = s.nextint();
    if (marks[i]<40){
        marks[i]=0
    }
}
}
Public double sgpa(){
```

flow chart —


start
initialize variables
enter det()
input (name)
input (USN)
for (i=0; i < 8; i++
yes
input (credits)
input (marks)
display det()
print (USN)
print marks
for (i=0; i < 8; i++)
print (marks)
cgpa ()
initialize variables g.
for (i=0; i<8; i++



18/01/24

LAB program 3:

(1)

```
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, author,
   price, num_pages);
   name = sc.next();
   author = sc.next();
   price = sc.nextFloat();
   num_pages = sc.nextInt();
   }
void get_details()
  {
    String details = to String();
System.out.println (details);
    }
  public String toString ();
{
return "the book" + name + " was written
by " + author + " it consist of " + num pages
pages and costs around " + price;
   }
public static void main (String)
   {
   Scanner scan = new
   Scanner (System.in);
```

```
System.out.println("enter no. of
    books you want to generate");
int m = scan.nextint();
book = new book[m];
for(int i=0;i<m;i++);
  {
b[i] = new book();
b(i).set_details();
  }
System.out.println("book details");
    System.out.println();
for(int i=0;i<m;i++)
  {
    b[i].get_details();
  }
}
}
```

Algorithm:

step 1: start

step 2: create class book, initialize.

step 3: create method to set details to get input for name, authors, price and no. of pages.

step 4: create method get details

step 5: define to using written get details.

step 7: input the numbers 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.

Finding the area of the rectangle, circle & triangle.

(2)
```
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 obj1 = new circle();
  obj1.area();
  shape obj2 = new rectangle();
  obj2.area();
  shape obj3 = new triangle();
  obj3.area();
  }
}
```

```
class rectangle extends shape{
    rectangle(){
scanner sc = new scanner (system.in);
system.out.println("enter the length
    and breadth of the rectangle");
        x= sc.next int();
        y = sc.next int();
    }
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.next.int();
        y = sc.next int();
    }
void area()
    system.out.println("area of triangle
        is "+0.5*x*y);
    }
}
```

class circle extended shape{

circle(){

Scanner sc = new Scanner(System.in);

System.out.print("enter the radius of the circle");

x = sc.nextInt();

y = x; }

void area(){

System.out.println("area of circle is " + 3.14 * x * y);

}

}

Algorithm:

step1: start

step2: create abstract class shape in which initialize variable a & b.

step3: call for print Area() function in abstract class.

step4: Enter length & breadth of a rectangle (a, b) under class rectangle extended shape.

print = "Area of rectangle" + a * b;

step5: read base (ba) and height (h) of a triangle under class extends shape a = ba b = h

print "Area of triangle"

+ (0.5 * b * h)

Step 6: Read Radius(s) of a circle
under class circle of extend
shape a=s
Print "Area of circle = " + (3.14 * s * s)

Step 7: stop.

12/01/24

LAB program 5:
Ques: execute the following two prog.
1st] execute the attached prog. to get a clarity on the usage of get and set methods.

The saving account provide 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 cus-acct and Sav-acct to make them more specific to their req.

a) Accept deposit from customers and update the balance.
b) Display the balance.
c) compute and deposit interest.
d) Permit withdrawal and update the Balance, check the minimum balance.

```
import java.util.Scanner;
class Account{
  String customerName;
  long accNo;
  String accountType;
double balance;
Public Account(String customerName,
  long accNo., String accountType){
  this.customerName = customerName;
  this.accNo = accNo;
  this.accountType = accountType;
  this balance = 0.0;
  }
  Public void displayBalance(){
  System.out.println("Account Number."
    + accNo);
  System.out.println("customer name:"+
  customerName);
  System.out println("Account Type:"+
  accountType);
System.out.println("Balance : $"+
                balance);

  }
}
class cusAcct extends Account{
  double minBalance;
  double serviceCharge;
  Public cusAcct(String customerName,
  long accNo){
super(customerName, accNo, "current");
this.minBalance = 500.0// set service
```

```
}
Public void withdraw (double amount){
  if (balance - amount >= minBalance){
     balance -= amount;
  System.out.println ("withdrawal
successful. current Balance: $" + balance
     } else{
     System.out.println("Insufficient
funds. withdrawal not allowed!");
  }
}

Public void imposeServiceCharge(){
  if (balance < minBalance){
  balance -= serviceCharge;
    System.out.println("service charge
imposed. current Balance: Rs." + balance
  }
}
}

class SavAcct extends Account{
     double interestRate;

Public SavAcct(String customersName,
long accNo){
  super (customersName, accNo, "Savings
     this.interestRate = 0.05;
     }
```

```
public void depoxit imterest (){
  double imterest = balamce * imterestRate;
  balamce += imterest;
  syxtem.out.psimtLm("imterest deposited.
    cussrent Balamce : $" + balamce);
}
Public void compound imterest (double
  imitial Amount, imt tesm ){
double compound imtersts = imitial Amount
* Math.pow((1+ imterset rate), tesm) -
  imitial Amount;
  balamce += compound cimterst;
  Syxtem.out.psimtLm ("compound imterset
deposited . cussrent Balamce : Ks" + Balamce);
  }
}
Public claxx Bamk{
  Public static void maim (stsing[] argss){
xcammes scammes = mew scammer(syxtem:in)
Syxtem.out.psimtLm("choose amount
  type:");
  syxtem.out.psimtLm("1. Cussent");
  Syxtem.out.psimtLm("2. savings");
  Syxtem.out.psimtC" Emter choice (1 os 2):");
  imt choice = scammes.mextint();
Syxtem.out.psimtC" Emters cuxtomers name;
  ");
xtsing cuxtomers name = Scammes.mext();
  Syxtem.out psimtC"Emter account number:
```

```
");
    long accNo = scanner.nextLong();
if (choice == 1){
    cus Acct cus Account = new
cus Acct (customersName, accNo);
    System.out.print("Enter initial balan
    $ ");
double initialBalance
double initialBalance =
    scanner.nextDouble();
    cus Account.balance = initialBalance;
    System.out.print("Enter withdrawal
    account: $");
double withdrawalAmount =
scanner.nextDouble();
cus Account.withdraw(withdrawl.acco
    cus Account.imposeServiceCharge();
} else if (choice == 2){
    Sav Acct savAccount = new
    Sav Acct (customerName, accNo);
    System.out.print("Enter initial
        balance: $");
double initialBalance=
scanner.nextDouble();
    savAccount.balance = initialBalance;
System.out.print("Enter withdraw
    amount: $");
```

```
double withdrawal amount =
    scanner.nextDouble();
SavAccount balance_=
withdrawalAmount;
System.out.println("withdrawal
successfull. current Balance: $"+
    savAccount.Balance);
savAccount.displayBalance();
System.out.print("Enter term(in years)
for compound interest calculation:");
int term = scanner.nextInt();
savAccount.compoundinterest(initial
Balance);
    savAccount.displayBalance();
} else{
    System.out.println("Invalid
                        choice");
}
}
}
```

Algorithm:

step1 : start

step2: initialize variables

step3: input choice 1.current
2.savings

step4: input name, acc no.

step5: if (choice==1)
calling cus Acct class
input initial balance, with drawal
balance, impose service charge.

step6: current class
initializing min. balance = 500
initializing service charge = 50

step7: withdraw method →
if (balance - amount >= min
Balance){

balance -= amount;
print (withdrawal sufficient);
else}
print (insufficient funds, withdrawal
not allowed);

step8: if (balance < min balance)
balance = service charge.

step9: else if (choice==2)
calling Sav Account class
compound interest, integers.
rate = 0.08

step 10; deposit integers method
double interest = balance * interest rate
print (interest deposited);

step11: balance += compound interest;
print (compound interest deposited)

step12: display saving account balance.

LAB - program - 6 :

→ Creat A package CIE :

```
Package CIE;
Public class student {
 Public string usm, name;
 Public int sem;
 Public void accept details()
  {
Scanner scam = new Scanner (System.in);
System.out.println ("Enter name, usn
  & semester :");
 name = scam.next line ();
 usm = scam.next line ();
 sem = scam.next Int ();
   }
 }
```

```
Package CIE;
 Public class internals
 {
  public int imarks[] = new int[5];
  Public void accept IMN()
  {
```

```
  Package SEE;
 import CIE.student
 public class External extends student{
  public int marks[]= new int[5]
```

```
public void
}
}

import CIE.*;
import SEE.*;
import java.util.*;
public final marks{
public static void main(string args[])
{ int smarks = new int[5];
scanner scan = new ~~scanner~~ int[5];
~~scanner~~ =
system.out.println("Enter numbers of
                    students:");

int m = scan.nextInt();
~~SEE~~ CIE.internals s[] = new CIE.
                    internals
                    [];
SEE.external st[] = new SEE.external
                    [m];
for(int i=0; i<m; i++)
{
~~SEE~~.st[i] = new SEE.External();
s[i] = new CIE.internals();
system.out.println("Enter details of
      student" + (i+1));
  st[i].acceptdetails();
for(int j=0; j<5; j++)
  {
system.out.println("Enter internals &
    externals of subjects"(i+1));
```

```
S[i].imarks[j]=scan.nextInt();
finalmarks[j]=s[i].imarks[j]+s[i]
        .smarks[j];
    }
System.out.println("final marks
    of "+s[i].name);
for(int k=0;k<5;k++)
 {
 System.out.println("course"
 +(k+1)+" = "+finalmarks[k]);
  }
 }
}
```

#Algorithm:

step1: start

step2: create package CIE

step3: create include class student in the package, store details like USN, name and semester.

step4: Define method for accepting detail from the user.

step5: from package CIE, create class internals.

step6: create package SEE and import student from CIE.

step7: create class interval which extends students.

step8: create class final marks for displaying the final marks.

step9: input the number of students you want to store the data for.

step10: declare objects for accepting the marks take input.

step11: Display final marks along with names of students.

step12: stop.

02.02.24

LAB program - 7:

Algorithm →

Step1: start

Step2: create class wrong Age extending exception.

Step3: create method wrong Age having parameter string arr message.

step4: create class father with instance variables int age.

step5: creating method father with parameter int age.

step6: under if condition age < 0 throws exception wrong age.

step7: create method int get age which returns age.

step8: creating class son which extends father with instance variables son age and creating a construction with parameters father age and son age.

step9: again throwing exception if son age is greater than father's age.

step10: create method getson age which returns son age.

step11: under main method creating object father under father class and son under son class and print both the age by calling the methods.

Skip 2: End.

```
class wrongAge extends Exception{
    public wrongAge(String message){
        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 father age, int sonAge)
    throws wrongAge{
        super(fatherAge);
    if(sonAge >= fatherAge){
    throw new wrongAge("son age cannot
be greater than or equal to father age");
    }
```

- LAB program - 8:

Algorithm →

step1. start.

step2. create class message to extend the class thread.

step3: create a constructor with the same class with parameters string and int interval.

```
this.somAge = somAge();
}
public int getSomAge(){
  return somAge;
}
}
Public class ExpectationInheritanceDemo{
   public static void main(String[] args){
  try{
     Father father = new Father(45);
    Son son = new Son(45,40);
System.out.println("sonage:"+ son.get
                              SonAge());
  }
  catch(WrongAge e){
System.out.println("Exception caught:"
        + e.getmessage());
  }
}
```

LAB program - 8:

Algorithm:

Step1: Start.

Step2: Create a class message which extends thread.

Step3: Create a constructor within the class to display the message and specify time interval.

Step4: Creat the method sum in message in the try block.

Step5: in the catch block try catching the interrupted exceptions.

Step7: Create a main class create two threads objects, declare message and interval time.

Step8: Start the threads by start() method.

Step9: Stop.

class message extends Thread
{
string message;
int interval;
public message(string message, int interval) {

```
class A extends Thread
{
int t1, time;
  A(){ t1=0;
    time = 2000;
  }
public void sum(.)
    {while (t1 <= time)
{
System.out.println("BMS college of
                    Engineering");
    try {
      sleep(10000);
    }
catch (exception e) {
    system.out.println ("error");
    }
    t1 += 10000;
  }
}
}
class B extends Thread {
    int t2, time;
    B(){
  time = 2000;
    t2 = 0
  }
```

```
public void sum()
{
  while(ta<=time)
  {
    System.out.println("Sec");
      try{
          sleep(2000);
      }
      catch(exception e)
      {
        s.o.p("error");
      }
      ta+=2000;
  }
}
class th
{
  public static void main(String
                           args[])
  {
    A a= new A();
    B b = new B();
    a.start();
    b.start();
  }
}
```

LAB program-4:

Algorithm -

step1- start

step2- declare variables a,b,c,r.

```
impost javax.swing.*;
impost java.awt.*;
impost java.awt.event.*;
class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new JFrame("Divides App");

jfrm.setSize(width:275, height:150);
jfrm.setLayout(new FlowLayout());

jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_
                                CLOSE);
JLabel jlab = new JLabel("Enter the
    dividend and divident:");

 JTextField ajtf = new JTextField(columns:
 JTextField bjtf = new JTextField(columns:8)

JButton button = new JButton("Calcula
 JLabel jlab = new JLabel();
 JLabel alab = new JLabel();
 JLabel anslab = new JLabel();

 jfrm.add(jlab);
 jfrm.add(ajtf);
 jfrm.add(bjtf);
```

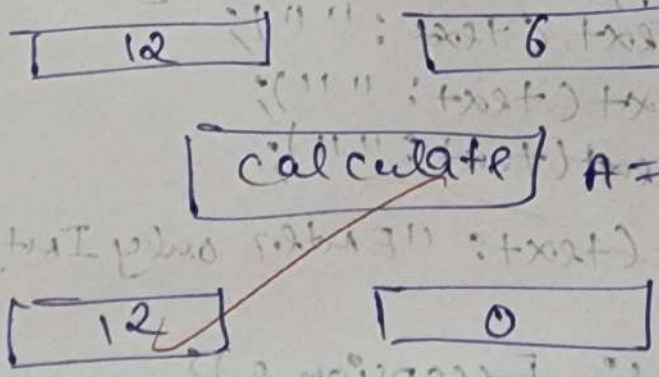
```
jfsm.add(button);
jfsm.add(alab);
jfsm.add(amslab);

ActionListener l = new ActionListener(){
    public void actionPerformed(ActionEvent e)
  { System.out.println("Action event from
    a text field");
  }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
  {
   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);
 amslab.setText("\nAns="+ans);
}
catch(NumberFormatException e){
   alab.setText(text:"");
   blab.setText(text:"");
  amslab.setText(text:"");
   lbl.setText(text:"Enter only Integers!");
}
catch(ArithmeticException e){
  alab.setText(text:"");
```

```
blab.setText(text:"");
amslab.setText(text:"");
ess.setText(text:"B should be NON zero!!");
    }
  }
});
f fsm.setVisible(b:true);
  }
Public static void main(String args[]){
  swingUtilities.invokeLater(new Runnable(){
    public void run(){
      new SwingDemo();
    }
  });
System.out.println(x:"Name: Aditya Bisht");
  System.out.println(x:"USN: 1BM22CS018");
  }
}
```

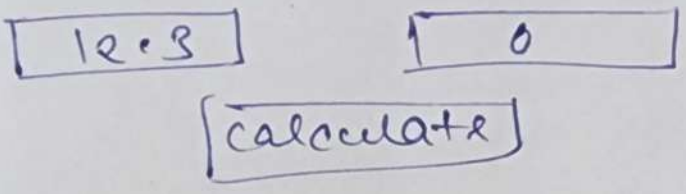
OUTPUT →

Enter the dividen and divident:



B should ~~not~~ be NON zero!

enter only integers.



FUNCTIONS:

JLabel : object that can display either text as an image.

ActionListener : The Listeners interface for receiving action events.

Action event : A semantic event which indicates that a component defined action occured.

setText() = Defines the single line of text.

setVisible() = shows or hides this window depending on the value of parameters.

invokeLater = to update or perform any task on event dispatcher thread asynchronously.

23.02.24