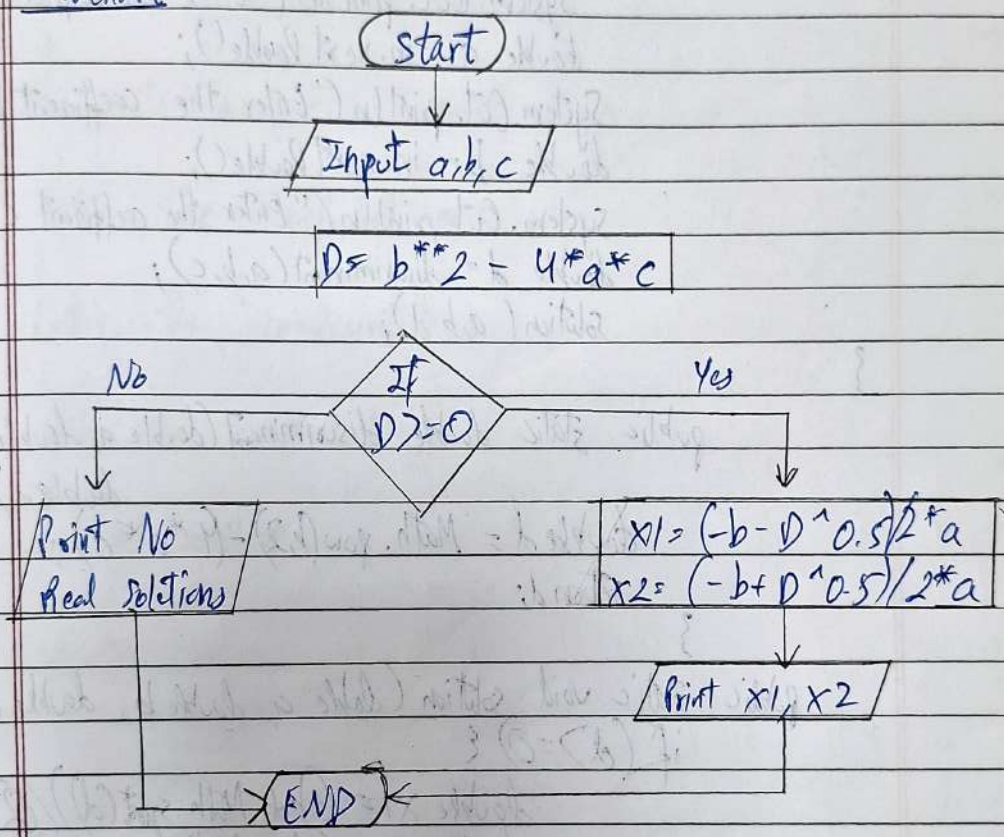
Program-1: Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a,b,c and use the quadratic formula. If the discriminant b^2-4ac is negative, display a message stating that there are no real solutions.

Flowchart:



Algorithm:

Step-1: Read a,b,c coefficients of quadratic equation.
Step-2: Calculate discriminant $D = b^2 - 4*a*c$
Step-3: Check if discriminant >=0
Step-4: If true, calculate roots x1 and $x2 = (-b \pm b^{0.5})/2*a$
Print roots
If false, print no real solutions

Code:

```
import java.util.*;
public clas Quadratic {
        public static Void main(String[] args) {
            Scanner in= new Scanner(System.in);
            System.Out.println("Enter the coefficient a:");
            double a= in.nextDouble();
            System.Out.println("Enter the coefficient b:");
            double b= in.nextDouble();
            System.Out.println("Enter the coefficient c:");
            double d= discriminant(a,b,c);
            Solution(a,b,d);
}
        public static double discriminant(double a, double b,
                                          double c) {
            double d= Math.pow(b,2)-(4*a*c);
            return d;
        }
    public static void solution(double a, double b, double d) {
        if(d>=0) {
                double x1= (-b+Math.sqrt(d))/(2*a);
                double x2= (-b-Math.sqrt(d))/(2*a);
            System.Out.println("The roots are"+x1+
                               "and"+x2);
        }
    else { System.Out.println("No real solutions");}
}
```

Output:

Enter the coefficient a:
1
Enter the coefficient b:
2
Enter the coefficient c:
1
The roots are -1.0 and -1.0

Enter the coefficient a:
1
Enter the coefficient b:
1
Enter the coefficient c:
1
No real solutions

```
PS C:\Users\sarth\Downloads\Java lab programs\program1> javac prog1.java
PS C:\Users\sarth\Downloads\Java lab programs\program1> java prog1
enter the coefficients
1 2 1
r1=-1.0
r2=-1.0
```

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

Algorithm:

Step-1: Start
step-2: import java.util(Scanner and java.lang.Math)
step-3: Create classes Student and Sgpacalc
Step 4: Create public static void main()
Step-5: Create a student object called stud.
step-6: Initialize attributes name, usn, num, sgpa and arrays marks, credits, grades
Step-7: Call method stud.accept_details()
Step-8: Prompt the user for name and usn
Step-9: for i=0; i<num_subs; i++
if (mark[i] >=40 and marks[i] <=100)
Step-10: grades[i]+= Math+floor(marks[i]/100)+1
~~Step-11~~: print name and usn
~~Step-10~~: Call method stud.display_details()
Step-11: Call method calculate_sgpa()
Step-12: for i=0; i<Num_subs; i++
num+= credit[i]*grades[i]
den+= credits[i]
Step-13: Sgpa = num/den
step-14: return sgpa
Step-15: display sgpa
Step-16: Stop

Program:

```
import java.util.Scanner;
import java.lang.Math;
```

```
class student {
    int num_subs = 8; double sgpa, num, den;
    String name, usn;
    double credits[] = new double[num_subs];
    double marks[] = new double[num_subs];
    double grades[] = new double[num_subs];

    void accept_details() {
        Scanner reader = new Scanner(System.in);
        System.out.println("Enter name: ");
        name = reader.nextLine();
        System.out.println("Enter usn:");
        usn = reader.nextLine();
        for(int i=0; i < num_subs; i++) {
            system.out.println("Enter credits:");
            credits[i] = reader.nextDouble();
            system.out.println("Enter marks:");
            marks[i] = reader.nextDouble();
        }
    double calculate_sgpa() {
        for(int i=0; i < num_subs; i++) {
            if(marks[i] >= 40 && marks[i] <= 100) {
                grades[i] = Math.floor(marks[i]/10) + 1;
            }
            else {
                grades[i] = 0;
            }
        }
        for(int i=0; i < num_subs; i++) {
            num += grades[i] * credits[i];
            den += credits[i];
        }
        sgpa = num/den;
        return sgpa;
    }
```

```
void display_details() {
    System.out.println("Name: " + name);
    System.out.println("USN: " + usn);
    System.out.println("Credits");
    for (i=0; i< num_subs; i++) {
        System.out.println(Credits[i] + " ");
    }
    System.out.println(" ");
    System.out.print("Marks: ");
    for (int i=0; i< num_subs; i++) {
        System.out.println(marks[i] + " ");
    }
    System.out.println(" ");
    System.out.println("SGPA: " + Calculate_Sgpa());
}
}

public class Sgpacalc {
    public static void main(String [] args) {
        Student stud = new Student();
        stud.accept_details();
        stud.display_details();
    }
}
```

Output:

```
Enter name:
Rohit
Enter USN:
1BM22CS235
Enter credits:
4
Enter marks:
90
```

Enter credits:
4
Enter marks:
92
Enter credits:
3
Enter marks:
87
Enter credits:
3
Enter marks:
95
Enter credits:
3
Enter marks:
92
Enter credits:
1
Enter marks:
97
Enter credits:
1
Enter marks:
96
Enter credits:
1
Enter marks:
95
Name: Rohit
USN: 1BM22CS235
Credits: 4.0, 4.0, 3.0, 3.0, 3.0, 1.0, 1.0, 1.0
Marks: 90.0, 92.0, 87.0, 95.0, 92.0, 97.0, 96.0, 95.0
SGPA: 9.85

```
Enter your name:
Rohit
Enter your usn:
1BM22CS235
Enter the number of subject:
4
Enter the marks of subject 1 :
90
Enter credits of subject 1 :
4
Enter the marks of subject 2 :
92
Enter credits of subject 2 :
3
Enter the marks of subject 3 :
87
Enter credits of subject 3 :
3
Enter the marks of subject 4 :
95
Enter credits of subject 4 :
3
Name: Rohit USN: 1BM22CS235
The marks of a subject 1 : 90
The credits of the subject : 4
The marks of a subject 2 : 92
The credits of the subject : 3
The marks of a subject 3 : 87
The credits of the subject : 3
The marks of a subject 4 : 95
The credits of the subject : 3
The SGPA of USN: 1BM22CS235 Name: Rohit is : 9.769230769230770
```

Program-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.

Algorithm:

Step-1: Create class Book

Step-2: Declare members name, author, price, and number of pages as num_pages.

Step-3: Create a constructor public Book(String name, String author, Double price, int num_of_pages).

Step-4: Set values of members in constructor.

Step-5: Create the following methods to set the name, author, price and number of pages.
void setName(String Name), void Author(String Author)
void setPrice(Double Price), void setPages(int pages)

Step-6: Create the following methods to return, name, author, price and num_pages. String getName(), String getAuthor(), double getPrice(), double getPages().

Step-7: Create method toString() to return the details in string format.

Step-8: Return string containing details in toString().

Step-9: Create class Program declared Public.

Step-10: Read the number of books for which details have to be entered.

Step-11: Declare array books of Book type with n objects.

Step-12: Initialize each object in array with new Book().

Step-13: Read name, author, price & num_pages.

Step-14: Use set methods of class Book to set the values of members.

Step-15: Invoke toString() method in class Book to print details of all the books.

Code:

```
import java.util.*;
class Book {
    private String name;
    private String author;
    private double price;
    private int num_pages;

    public Book() { }
    public Book(String name, String author, double price, int
                                                num_pages)
    {
        this.author = author;
        this.name = name;
        this.price = price;
        this.num_pages = num_pages; }
    public void setName(String name) {
        this.name = name; }
    public String getName()
        {return name;}
    public void setAuthor(String author) {
        this.author = author; }
    public void getAuthor() {return author;}
    public void setPrice(double price) {
        this.price = price; }
    public void getprice() {
        return price; }
    public void set_pages(int num_pages) {
        this.num_pages = num_pages; }
    public void getPages() {
        return num_pages; }
    public String toString() {
        return "Book Name: " + name + " Author: " + author +
            " Price: " + price + "Pages: " + num_pages; }
}
```

```
public class Program {
    public static void main(String [] args)
    { Scanner in = new Scanner(System.in);
      System.out.println("Enter number of books: ");
      int n = in.nextInt();
      Book[] books = new Book[n];
      for(int i=0; i<n; i++)
      {   books[i] = new Book();
          System.out.println("Enter book name: ");
          String name = in.next(); in.nextLine;
          System.out.println("Enter book author: ");
          String author = in.next();
          in.nextLine();
          System.out.println("Enter book price: ");
          double price = in.nextDouble();
          System.out.println("Enter number of pages: ");
          int num_pages = in.nextInt();
          books[i].setName(name);
          books[i].setAuthor(author);
          books[i].setPrice(price);
          book[i].setPages(num_pages);
          System.out.println("Book: "+books[i].getName()+
      "Author: "+book[i].getAuthor+"details entered");
    }

      for(int i=0;i<n;i++)
      { String s = books.toString();
        System.out.println(s);
      }
    }
}
```

Output:

```
Enter number of books:
2
Enter book name: A
Enter book author: B
Enter the book price: 254
Enter the number of pages: 568
Book: A author: B details entered
Enter the book name: C
Enter book author: D
Enter the price of book: 456
Enter number of pages: 590
Book: C Author: D details Entered

Book Name: A
Author Name: B
Price: 234.0
Pages: 568

Book Name: C
Author Name: D
Price: 456.0
Pages: 590
```

```
Enter number of books:
2
Enter the name of book:
A
Enter the author of book:
B
Enter the price of the book:
254
Enter the number of pages:
568
Book: A Author: A details entered.
Enter the name of book:
C
Enter the author of book:
D
Enter the price of the book:
456
Enter the number of pages:
590
Book: C Author: C details entered.
Book Name: A Author Name: B Price: 254.0 Pages: 568
Book Name: C Author Name: D Price: 456.0 Pages: 590
```

Program-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.

Algorithm:

Step-1: Create an abstract class Shape
Step-2: Declare members int a, int b;
Step-3: Create an abstract method printArea() with no body.
Step-4: Create a class: Rectangle that extends class Shape.
Step-5: Declare members int a, int b.
Step-6: Create method printArea that calculates area using a & b and prints it, where area = a*b.
Step-7: Create a class: Triangle that extends class Shape.
Step-8: Declare members int a, int b.
Step-9: Create a constructor Triangle that initializes or sets the values of a & b.
Step-10: Create method void PrintArea to calculate & print area given by 0.5*a*b
Step-11: Create a class: circle that extends shape.
Step-12: Declare member int a.
Step-13: Create a constructor Circle to initialize a.
Step-14: Create void printArea() to print area given by area = 3.14*a*a.
Step-15: Create public class program
Step-16: Create objects circle, triangle, rectangle or class Shape.
Step-17: Invoke printArea method of each object.

Code:

```
abstract class Shape {
    private int a,b;
    Shape() { }
    abstract void printArea();
}
class Rectangle extends Shape {
    private int a,b;
    Rectangle(int a, int b)
    { this.a=a;
      this.b=b;
    }
    void printArea()
    { System.out.println("Area of rectangle is: " + (a*b)); }
}
class Triangle extend Shape {
    private int a,b;
    Triangle(int a, int b)
    { this.a=a; this.b=b; }
    void printArea()
    { System.out.println("Area of triangle is: " + (0.5*a*b)); }
}
class Circle extends Shape {
    private int a;
    Circle(int a) { this.a=a; }
    void printArea()
    { System.out.println("Area of Circle: " + (3.14*a*a)); }
}
public class Program {
    public static void main(String[] args)
    { Shape r = new Rectangle(3,6);
      Shape t = new Triangle(4,6);
```

```
        Shape c = new Circle(5);
        r.printArea();
        t.printArea();
        c.printArea();
    }
}
}
```

Output:

```
Area of rectangle is: 18.0
Area of triangle is: 12.0
Area of circle is: 78.5
```

```
The area of rectangle is: 18
The area of triangle is: 12.0
The area of circle is: 78.5
```

Program-5: Develop a Java Program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores account details and achieve the following: accept deposit & update, display balance, compute and deposit interest, withdraw and update balance.

Algorithm:

Step-1: Create a class Account with data members String name, String type, Int Account_name, double deposit.
Step-2: Create a method Info to get the above data.
Step-3: Create a method details which prints the above data.
Step-4: Create a class savings that extends (inherits) Account.
Step-5: Create a method inside savings deposit(double amount) which deposits amount.
Step-6: Create a method inside savings withdraw (double amount) which deducts from balance.
Step-7: Create method Interest (int time, int rate)
Step-8: Amount after interest deposit is given by $A = P*(1+r/100)^t$.
Step-9: All methods display balance.
Step-10: Create class current which inherits Account
Step-11: Create method deposit (double amount) which reads balance entered and updates it.
Step-12: Create Method withdraw (double amount) which deducts from overall balance and implements check().

Step-13: Create method check() which checks if balance < 20000.
If ~~the~~ ~~bal~~ balance = balance − 500.

Step-14: Create class Bank

Step-15: Read Name, account type, account-num

Step-16: Input user ~~advice~~ choice for account type.

Step-17: If account type is savings:
(i) Savings object is created
(ii) details are input through info function inherited by Account.
(iii) User choice for methods in savings class.

Step-18: If ~~account~~ account_type is current:
(i) current object is created
(ii) details input are sent through current object which inherits info.
(iii) user choice for method is current class.

Program:

```
import java.util.*;
class Account {
    String name;
    String type;
    int acc_num;
    double dep;
    public void info(String name, String type, int acc_num, double dep) {
        this.name = name;
        this.type = type;
        this.acc_num = acc_num;
        this.dep = dep;
    }
public details() {
    System.out.println("Name: " + name);
    System.out.println("Account_type" + type);
```

```
System.out.println("Account Number: " + acc_num);
System.out.println("Current Balance: " + dep); }
}

class Savings extends Account {
    public void deposit(double amount) {
            dep = dep + amount;
            System.out.println("Balance: " + dep);
    }
    public void withdraw(double amount) {
    {   if (dep < amount)
            { System.out.println("Insufficient funds"); }
        else
            dep = dep - amount;
        System.out.println("Balance: " + dep);
    }
    public void interest(double t, double r) {
        double dep1 = dep * Math.pow((1+r/100), t);
        System.out.println("Interest: " + dep1 - dep);
        dep = dep1;
    System.out.println("Interest deposited amount: " + dep);
}

class Current extends ~~Amount~~ Account {
    public void deposit(~~double~~ amount) {
            dep = dep + amount;
            System.out.println("Balance: " + dep);
    }
    public void ~~withdraw~~ withdraw(double amount) {
        if (dep < amount) { System.out.println("Insufficient
                                                ~~fund~~ funds") }
        else { dep = dep - amount; }
        check(dep);
    }
```

```
public void check (double amt)
 { if (amt<2000)
      { if (amt<500) { dep=0; }
        else { dep = dep-500; }
     System.out.println ("Insufficient Funds! Amount less than
                                    Rs.2000 "); }
     System.out.println ("Balance: " +dep);
   }
}
public class Bank {
     public static void main (String[] args)
     { Scanner in = new Scanner(System.in);
      int c1=1;
     while (c1=1)
      { System.out.println("Enter name: ");
        String name = in.next(); in.nextLine();
        System.out.println ("Enter Account Number: ");
        int acc_no = in.nextInt(); int choice1;
        System.out.println ("1. Savings \n2. Current ");
        System.out.println (" Enter Account type: ");
        choice1 = in.nextInt();
     switch (choice1) {
        case 1:
          Savings s = new Savings();
          System.out.println ("Enter deposit:");
          double balance = in.nextDouble();
          s.into (name, type: "savings", acc_no, balance);
          s.details();
          System.out.println ("1. Deposit 2. Withdraw 3. Interest 4. Exit");
          int choice2;
     do { System.out.println ("Enter your choice");
          choice2 = in.nextInt();
          switch (choice2) {
             case 1:
```

```
        System.out.println("Enter amount: ");
        double amount = in.nextDouble();
        s.deposit(amount);
        break;
    case 2:
        System.out.println("Enter amount: ");
        double amount2 = in.nextDouble();
        s.withdraw(amount2);
        break;
    case 3:
        System.out.println("Enter the period: ");
        double time = in.nextDouble();
        System.out.println("Enter rate: ");
        double rate = in.nextDouble();
        s.interest(time, rate);
        break;
    case 4:
        break;
    default:
        System.out.println("Invalid Choice");
    }
} while (choice2 >=1 && choice2 <=3);
break;

case 2:
    current c = new current();
    do{
        System.out.println("Enter deposit (>2000)");
        balance = in.nextDouble();
    } while (balance <2000);
    c.into(name, "current", acc_no, balance);
    c.details();
    System.out.println("1. Deposit 2. Withdraw. 3. Exit");
    int choice3;
    System.out.println("Enter choice");
```

```
choice3 = in.nextInt();
switch(choice3)
do { case 1:
    System.out.println("Enter amount:");
    double amount1 = in.nextDouble();
    c.deposit(amount1); break;
  case 2:
    System.out.println("Enter amount:");
    double amount2 = in.nextDouble();
    c.withdraw(amount2);
    break;
  case 3:
    break;
  default:
    System.out.println("Invalid Choice");
} while(choice3>=1 && choice3<=2);
default:
    System.out.println("Invalid choice");
} System.out.println("Enter 1 to continue or 0 to exit");
  int c2 = in.nextInt(); } }
}
```

Output:

```
Enter name: Mohan
Enter account number = 6796352
Enter account type: 1
Enter deposit: 3000
Account type: Savings
Account Number: 6796352
Cresent Balance: 3000.0
1. Repsit 2. withdraw 3. Interest 4. Exit.
Enter your choice:
1
Enter your amount:
1000
```

Balance= 3000
Enter choice : 3
Enter rate: 7
Interest: 1800.344
Interest Deposited Amount = 9800.344
Enter choice: 4
Enter 1 to continue or 0 to exit: 1
Enter account type: 2
Enter deposit (>2000): 4500
Name: Rahul
Account type: Cossent
Account Number: 6457786
Cossent Balance: 4500
1. Deposit 2. Withdraw 3. Exit
Enter choice: 2
Enter amount: 2400
Insufficient funds!! Amount less than 2000
Balance: 1100.0
Enter choice: 3
Enter 1 to continue or 0 to exit: 0

2/12/24

```
Enter Name:
Mohan
Enter Account Number:
6796352
1.Savings 2.Current
Enter Account Type:
1
Enter deposit
3000
Name: Mohan
Account Type: Savings
Account Number: 6796352
Current Balance: 3000.0
1.Deposit 2.Withdraw 3.Interest 4.Exit
Enter your choice:
1
Enter amount:
1000
Balance: 4000.0
Enter your choice:
3
Enter time period:
3
Enter rate:
7
Interest: 900.1720000000005
Interest Deposited Amount : 4900.1720000000005
Enter your choice:
4
Enter 1 to continue or 0 to exit
1
Enter Name:
Rahul
Enter Account Number:
6457786
1.Savings 2.Current
Enter Account Type:
2
Enter deposit(>2000)
4500
Name: Rahul
Account Type: Current
Account Number: 6457786
Current Balance: 4500.0
1.Deposit 2.Withdraw 3.Exit
Enter your choice:
2
Enter amount:
2400
Balance: 2100.0
Enter your choice:
2
Enter amount:
2400
Enter insufficient funds.
Balance: 2100.0
Enter your choice:
3
Invalid Choice
Enter 1 to continue or 0 to exit
0
```

Program-6: Create a package CIE which has two classes - Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

Algorithm:

1. Create the CIE package:
   (i) Define the 'Student' class in the CIE package with members like 'usn', 'name' and 'sem'.
   (ii) Define the 'Internals' class in the CIE package with an array to store the internal marks scored in five courses of the current semester for the student.

2. Create the SEE package:
   (i) Define the 'Externals' class in the SEE package which is a derived class of the 'Student' class.
   (ii) The 'Externals' class should have an array to store the SEE marks scored in five courses of the current semester for the student.

3. Import the package:
   • Import the CIE and SEE package in a file that declares the final marks of students in all 5 courses.

Program:

```
package CIE;
import java.util.*;
public class Student {
        public String name;
        public string usn;
        public int sem;
        public void display() {
            Scanner sc = new Scanner (System.in);
            System.out.println("Name: ");
            name = sc.next();
            System.out.println("USN: ");
            usn = sc.next();
            System.out.println("Sem: ");
            sem = sc.nextInt();
        }
}
```

```
package ~~SEE~~ CIE;
import java.util.*;
public class Internals extends Student {
        public ~~class~~ double ciem[];
        public void display() {
            ciem = new double[5];
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter CIE marks out of 50:");
            for(int i=0; i<5; i++) {
                ciem[i] = c.nextDouble();
            }
        }
}
```

```
package SEE;
import CIE.*;
import java.util.*;
public class Externals extends CIE.Student {
        public double seem[];
        public void display() {
            seem = new double[5];
            Scanner s = new Scanner(System.in);
            System.out.println("SEE marks for 5 subjects
                    out of 100: ");
        for (int i=0; i<5; i++) {
            seem[i] = s.nextDouble();
        }
    }
}
```

```
import CIE.*;
import SEE.*;
import java.util.*;
public class Main {
    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter no. of Students:");
        n = sc.nextInt();
        CIE.Student st[] = new CIE.Student[n];
        CIE.Internals in[] = new CIE.Internals[n];
        SEE.External ex[] = new SEE.externals[n];
    for(int i=0; i<n; i++) {
        st[i] = new CIE.Student();
        in[i] = new CIE.Internals();
        ex[i] = new SEE.Externals();
        st[i].display();
        in[i].display();
```

```
            ext[i].display();
            System.out.println("Total marks of "+st[i].name+"\n");
            for(int j=0;j<5;j++){
                System.out.println(in[i].ciem[j] +ext[i].seem[j]/2);
            }
        }
    }
}
```

Output:

```
Enter no. of students:
3
Name:
A
USN:
1BM17CS005
Sem:
2
Enter cie marks out of 50:
48
47
46
48
44
Enter SEE marks out of 100:
84
88
86
82
80
Total marks of A:
90.0
```

91.0
89.0
89.0
84.0
Name:
B
USN:
1BM19CS192
Sem:
5
Enter CIE marks out of 50:
43
41
39
37
45
SEE marks out of 100:
78
80
84
86
82
Total Marks of B:
82.0
81.0
81.0
80.0
86.0

Name:
C
USN:
1BM21CS083
Sem:
6
Enter CIE marks out of 50:
47
45
43
49
48
Enter SEE marks out of 100:
84
88
94
90
96
Total Marks of C:
89.0
89.0
90.0
94.0
96.0

```
Enter no. of students:
2
Name:
A
USN:
1BM17CS005
Sem:
2
Eneter cie marks out of 50:
48
47
46
48
44
SEE marks for 5 subjects out of 100:
84
88
86
82
80
Total Marks of A

90.0
91.0
89.0
89.0
84.0
Name:
B
USN:
1BM19CS192
Sem:
5
Eneter cie marks out of 50:
43
41
39
37
45
SEE marks for 5 subjects out of 100:
78
80
84
86
82
Total Marks of B

82.0
81.0
81.0
80.0
86.0
```

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

Algorithm:

1. Define the Father Class:
   - Create a class called 'Father'.
   - Implement the constructor that takes the age parameter.
   - Inside the constructor, check if the age is <0.
   - If the age is less than 0, throw a WrongAge exception.

2. Define the Son Class:
   - Create a class called 'Son' that inherits from 'Father'
   - Implement a constructor that takes both father's age and son's age as parameters.
   - Inside the constructor, call the base class constructor with the father's age.
   - Check if the son's age is greater than or equal to the father's age.
   - If the son's age is ~~greater than~~ or equal to the father's age, throw an exception.

3. Define the WrongAge Exception class:
   - Create a class called WrongAge to represent the exception.
   - This class can be a simple subclass of Exception or a custom exception class.

4. Write the main program:
- Instantiate objects of the Father and Son classes, passing appropriate ages as parameters to their constructors.
- Handle any exceptions that may be thrown by using try-except blocks.

Program:

```
import java.util.*;
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message);
    }
}
class Father {
    int fatherAge;
    public Father(int fatherAge) throws WrongAge {
        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
        this.fatherAge = fatherAge;
    }
}
class Son extends Father {
    int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge);
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age must be less than
                                Father's age");
        }
        this.sonAge = sonAge;
    }
}
```

```
public class FatherSon {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter father's age and son's age:");
        int fa = sc.nextInt();
        int sa = sc.nextInt();
        try {
            Son s = new Son(fa, sa);
            System.out.println("Father's age: " + s.fatherAge);
            System.out.println("Son's age: " + s.sonAge);
        }
        catch(WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

Enter father's age and son's age:
45 25
father's age: 45
son's age: 25

Enter father's age and son's age:
32 43
Error:
son's age must be less than father's age.

Enter father's age and son's age:
20
56
Error: Son's age must be less than father's ~~age~~ age

```
Enter father's age and son's age:
45 25
Father's age: 45
Son's age: 25
PS C:\Users\sarth\Downloads\Java lab programs\prog7> java fatherson
Enter father's age and son's age:
32 43
Error: Son's age must be less than Father's age
```

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

Algorithm:

1. Define class A which extend Thread class:
   (i) Define a class named 'A' that extends Thread.
   (ii) Initialize instance variables t1 and time to represent time intervals.
   (iii) Implement the constructor to set initial values for t1 and time.
   (iv) Override the run method to execute the thread logic.
   (v) Inside the run method, use a while loop to run until t1 exceeds time.
   (vi) within the loop, print "BMS College of Engineering" and catch any exceptions that occur during sleep.

SSB 16/2/24

2. Define class B extends Thread:
   (i) Define a class 'B' that extends Thread.
   (ii) Initialize instance variables t2 and time similar to class A.
   (iii) Implement the constructor to set initial values for t2 and time.
   (iv) Override the run method to execute the Thread logic.
   (v) Inside the run method, use a while loop to run until t2 exceeds time.
   (vi) Within the loop, print "CSE" and catch any exceptions that occur during sleep.

3. Define the main class:
   (i) Create a class named Th to contain the main method.
   (ii) Inside the main method, instantiate objects of classes A and B.
   (iii) Start both threads using the start method.

Program :

```
class NewThread implements Runnable {
    String Name;
    Thread t;
    int n;
    NewThread(String threadname, int n) {
        Name = threadname;
        this.n=n;
        t= new Thread (this, Name) {
        System.out.println("New Thread:" +t);
        t.start();
    }

    public void run() {
        try {
            for (int i=0; i<10; i++) {
                System.out.println(Name + ": " + i);
                Thread.sleep(n);
            }
        }
```

```
        catch (InterruptedException e) {
            System.out.println(Name + "Interrupted");
        }
        System.out.println("Exiting");
    }
}

public class Prog8 {
    public static void main(String[] args) {
        NewThread ob1 = new NewThread("CSE", 2000);
        NewThread ob2 = new NewThread("BMS College of
                        Engineering", 10000);
    }
}
```

Output:

New Thread Thread[#21, CSE, 5, main]
New Thread Thread[#22, BMS College of Engineering, 5, main]
BMS College of Engineering: 0
CSE: 0
CSE: 1
CSE: 2
CSE: 3
CSE: 4
BMS College of Engineering: 1
CSE: 5
CSE: 6
CSE: 7
CSE: 8
CSE: 9
BMS College of Engineering: 2
Exiting
BMS College of Engineering: 3
BMS College of Engineering: 4

BMS College of Engineering : 5
BMS College of Engineering : 6
BMS College of Engineering : 7
BMS College of Engineering : 8
BMS College of Engineering : 9
Exiting

```
New Thread Thread[#21,CSE,5,main]
New Thread Thread[#22,BMS College Of Engineering,5,main]
CSE : 0
BMS College Of Engineering : 0
CSE : 1
CSE : 2
CSE : 3
CSE : 4
BMS College Of Engineering : 1
CSE : 5
CSE : 6
CSE : 7
CSE : 8
CSE : 9
BMS College Of Engineering : 2
Exiting
BMS College Of Engineering : 3
BMS College Of Engineering : 4
BMS College Of Engineering : 5
BMS College Of Engineering : 6
BMS College Of Engineering : 7
BMS College Of Engineering : 8
BMS College Of Engineering : 9
Exiting
```

Program-9: WAP that creates a user interface to perform integer divisions. The user enters two numbers in the text field, Num1 and num2. The division of num1 and num2 is displayed in the result field when the Divide button is clicked. If num1 or num2 were not an integer, the program would throw a NumberFormatException. If num2 were zero, the program could throw an ArithmeticException display the exception in a message dialogue box.

Algorithm:

1. Start
2. Create a class named SwingDemo.
3. Define a constructor for the SwingDemo class.
4. Inside the constructor:
   (a) Create a JFrame named jfrom with the title "Divider App" and set its size to 275 x 150 pixels.
   (b) Set the layout of the jframe to FlowLayout.
   (c) Set the default close operation to exit on close.
   (d) Create a JLabel named jlab with the text "Enter the divider and dividend:"
   (e) Create two JTextFields named ajtf and bjtf for user input.
   (f) Create a JButton named button with the label "Calculate".
   (g) Create three JLabels named err, alab, and blab for displaying error messages and input values.
   (h) Add the components to the JFrame in the following order: err, jlab, ajtf, bjtf, button, alab, blab, anslab.
   (i) Add "Action Listener" to ajtf and bjtf, printing "Action event from a text field" when triggered.
   (j) Add ActionListener to the button:
   i) Parse the integer values from ajtf and bjtf.
   2) Calculate the division of a by b and display the result in anslab.

3) Handle NumberFormatException by clearing alab, blab, anslab and displaying "Enter only Integers!" in err.
4) Handle ArithmeticException by clearing alab, blab, anslab and displaying "B should be NON zero!" in err.
5. Define the main method:
a) Invoke the SwingUtilities.invokeLater method to create the frame on the event dispatching thread.
b) Instantiate a new SwingDemo object.
6. End

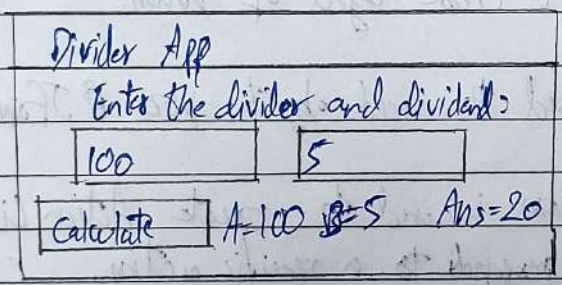
Program:

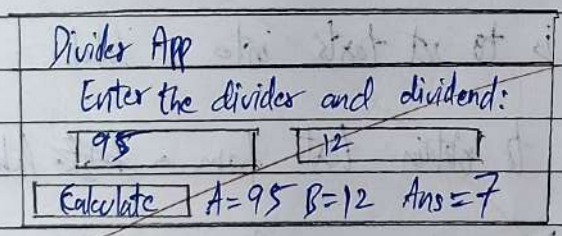
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(295, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel jlab = new JLabel("Enter the 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 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("\n A = "+a);
            blab.setText("\n B = "+b);
            anslab.setText("\n Ans = "+ans);
        }
        catch(NumberFormatException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            exc.setText("Enter only Integers!");
        }
        catch(ArithmeticException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            exc.setText("B should be NON zero!");
```

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

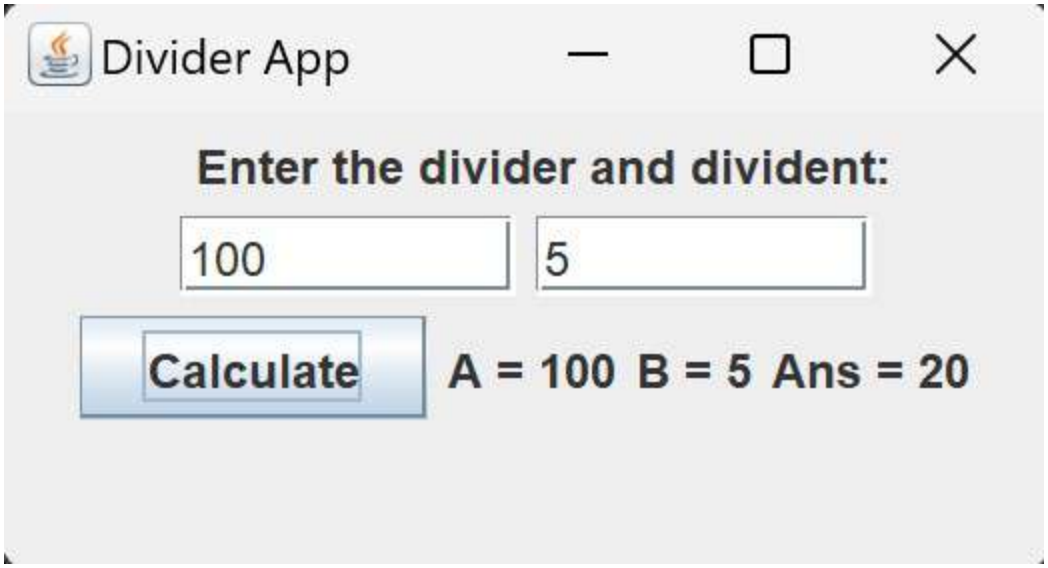
Output:





Definitions:

1) JFrame class: In swing functionalities, the container used for an application is called as JFrame. It acts like the main window where components like labels, buttons, textfields are added to create a GUI.


Divider App
Enter the divider and divident:
100
5
Calculate
A = 100 B = 5 Ans = 20

2) jfrm.setSize: Resizes this component so that it has width w and height h.

3) jfrm.setLayout: Used to set layout of the container

4) jfrm.FlowLayout: Used to arrange the components in a line. Default layout of applet or ~~layout~~ panel.

5) jfrm.setDefaultCloseOperation: Used to specify one of the several options for the close button.

6) JLabel is the class used to create label objects

7) JTextField is used to create textfield

8) JButton is used to create object of button.

9) jfr.add() is used to add objects into JFrame.

10) ActionListener class is used to create Action Listener objects which responds to a specific action.

11) ob.setText(" ") is to set texts into a label.

12) ob.getText() is to obtain texts from a label.

13) jfr.setVisible(true) is to set JFrame object to be visible.

22/2/24