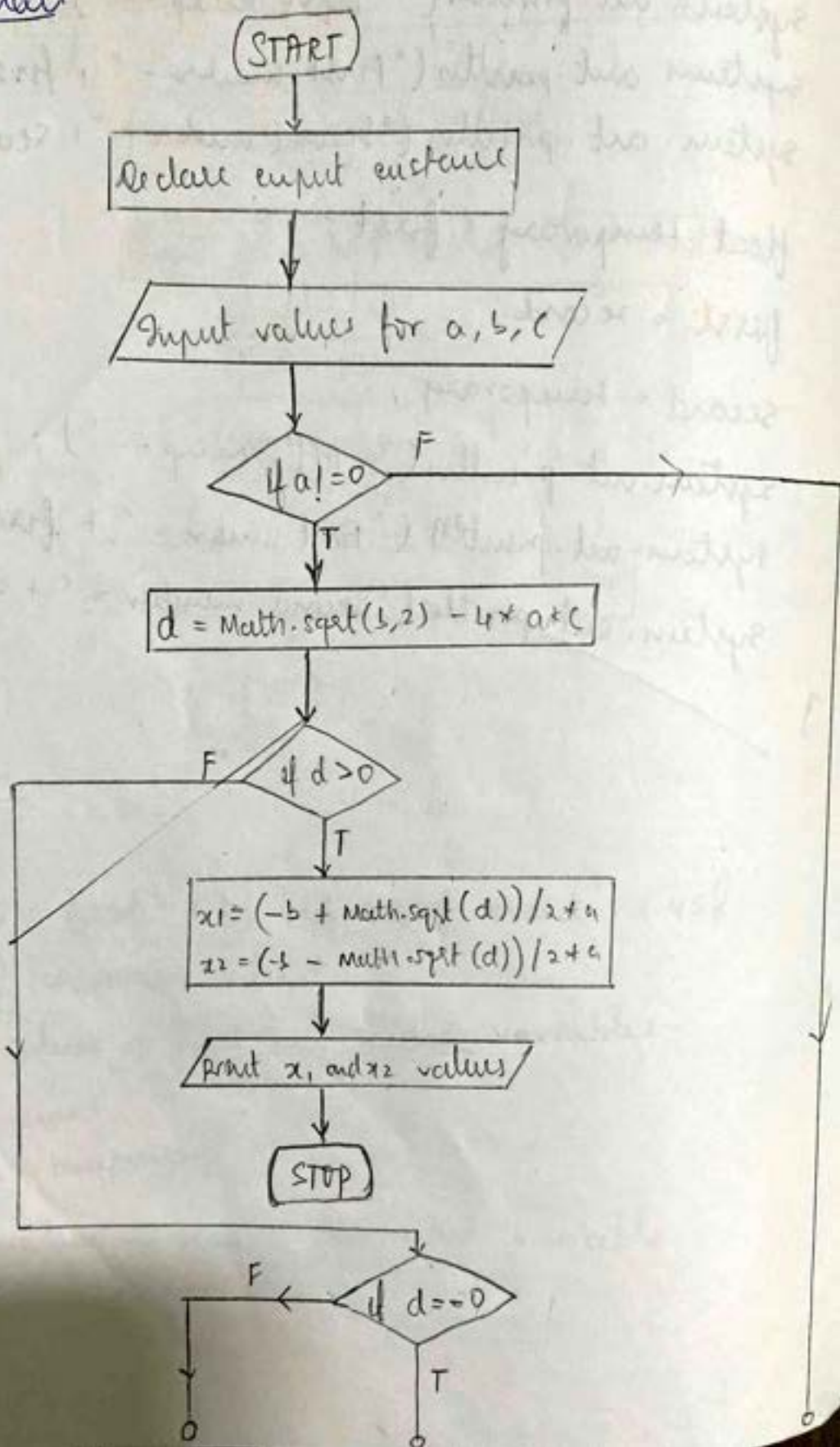
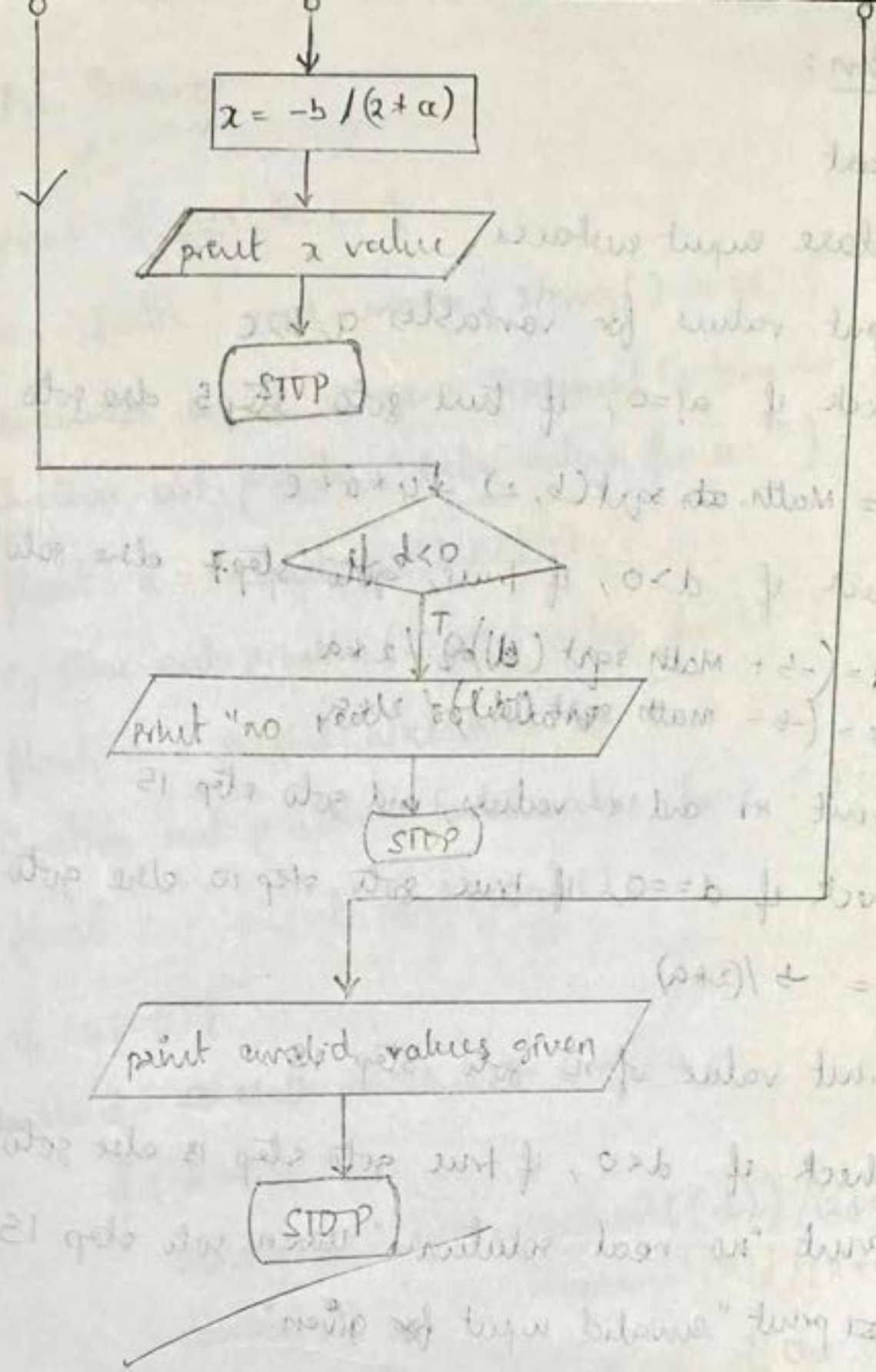
Lab Program 1

~~Q:-~~

Q. Develop a Java program that prints all real to the quadratic equation $ax^2 + bx + c$.

A: Flowchart




x = -b / (2 * a)
print x value
STOP
if d<0
T
F
print "no real solution"
STOP
print invalid values given
STOP

Algorithm:

1. Start
2. Declare input instance
3. Input values for variables a, b, c
4. check if a!=0, if true goto step5 else goto step 15
5. d= Math. ~~ab~~ sqrt(b, 2) - 4*a*c
6. check if d>0, if true, goto step7 else goto step 9
7. x1 = (-b + Math.sqrt (d)) / 2*a
   x2 = (-b - math.sqrt (d)) / 2*a
8. Print x1 and x2 values, and goto step 15
9. check if d==0, if true goto step 10 else goto step 12
10. x = -b /(2*a)
11. print value of x goto step 15.
12. check if d<0, if true goto step 13 else goto step
13 Print "no real solutions" then goto step 15
14. ~~pr~~ print "invalid input ~~for~~ given"
15. Stop

Code:

```
import. util. Scanner

public class Quadratic() {
    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);
        System. out. println(" Enter value for a: ");
        float a = input. nextFloat();
        system. out. println (" Enter value for b: ");
        float b = input. nextFloat();
        System. out. println (" Enter value for c: ");
        float c = input. nextFloat();

        if (a!=0) {
        double d= Math. pow(b, 2) - 4 * a * c;
            if ( d>0){
                double x1 = (-b + Math.sqrt(d)) /(2*a);
                double x2 = (-b - Math.sqrt(d)) /(2*a);
                System. out. println(" Roots of the Quadratic
                  Equation are" + x1 + "and" + x2 );
            }
        else if(d==0) {
                double x = -b /(2*a);
                System. out. println(" Root of the Quadratic
                  Equation is" + x);
            }
```

```
        else {
            System.out.println("The Quadratic Equation has
            no real roots");
        }

    }
    else {
        System.out.println("Invalid Input");
    }
  }
}
```

Output: 1:

Enter the coefficient of x^2

1

Enter the coefficient of x

-5

Enter the constant value

6

The roots of the equation are 3.0 and 2.0;

Output 2:

Enter coefficient of x^2

0

Enter coefficient of x

4

Enter constant value

6

Invalid Input

Output 3:

Enter coefficient of x^2

1

Enter coefficient of x

4

Enter constant value

5

The Quadratic Equation has no real ~~roots~~ roots

Output 4:

Enter the coefficient of x^2

1

Enter the coefficient of x

-2

Enter the constant value

1

The root of the equation is : 1.0

22/12/23

```
Enter the coeffecient of x^2: 1
Enter the coeffecient of x: -3
Enter the constant value: 2
The Quadratic Equation has two real values for x.
Solution 1:  2.0
Solution 2:  1.0
Do you want to continue? (y/n): y

Enter the coeffecient of x^2: 1
Enter the coeffecient of x: -2
Enter the constant value: 1
The Quadratic Equation has one real value for x.
Solution :  1.0
Do you want to continue? (y/n): y

Enter the coeffecient of x^2: 2
Enter the coeffecient of x: 1
Enter the constant value: 3
The Quadratic Equation has no real values for x.
Do you want to continue? (y/n): n


-Abhinav Raghu
 1B22CS005
```

Lab Program - 2

29/12/23

Q. 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.

Ans.

```
import java.util.Scanner;

class Student {
        String usn;
        string name;
        int [] credits;
        int [] marks;

        Student() {
            ~~String usn;~~
            credits = new int[5];
            ~~String name;~~
            marks = new int [5];
        }
        public void ~~E~~details() {
            Scanner scanner = new Scanner (System.in);
            System.out.println ("Enter USN");
            usn = scanner.next();
```

```
System.out.println("Enter name:")
name = scanner.next();
System.out.println("Enter credits and marks
                    for 5 subjects:");

for (int i=0; i<5; i++) {
    system.out.println("Enter credits:");
    credits[i] = scanner.nextInt();
    System.out.println("Enter marks:");
    marks[i] = scanner.nextInt();
}

}
public void display() {
    System.out.println("USN:" + usn);
    system.out.println("Name:" + name);
    system.out.println("Marks" ~~marks~~ and
                        credits");

    for(int i=0; i<5; i++) {
        ~~system.out.~~ println("credits:" + credits[i]);
        System.out.println("Marks:" + marks[i]);
    }
}
```

```
public double SGPA {
    double total_credits = 0;
    double total_sum = 0;
    for (int i = 0; i < 5; i++) {
        total_credits += credits[i];
        total_sum += gradepoint(marks[i]) * credits
    }
    return total_sum / total_credits;
}
public int gradepoint (int marks) {
    if (marks >= 90) {
        return 10;
    } else if (marks >= 80) {
        return 9;
    } else if (marks >= 70) {
        return 8;
    } else if (marks >= 60) {
        return 7;
    } else if (marks >= 50 {
        return 6;
    } else {
        return 0;
    }
}
```

```
}
public class Main {
    public static void main(String[] args) {
        student student = new Student();
        student.details();
        System.out.println("\n student
                                details");

        student.display();
        System.out.println("\nSGPA:" +
        student.SGPA());

    }
}
```

Algorithm:

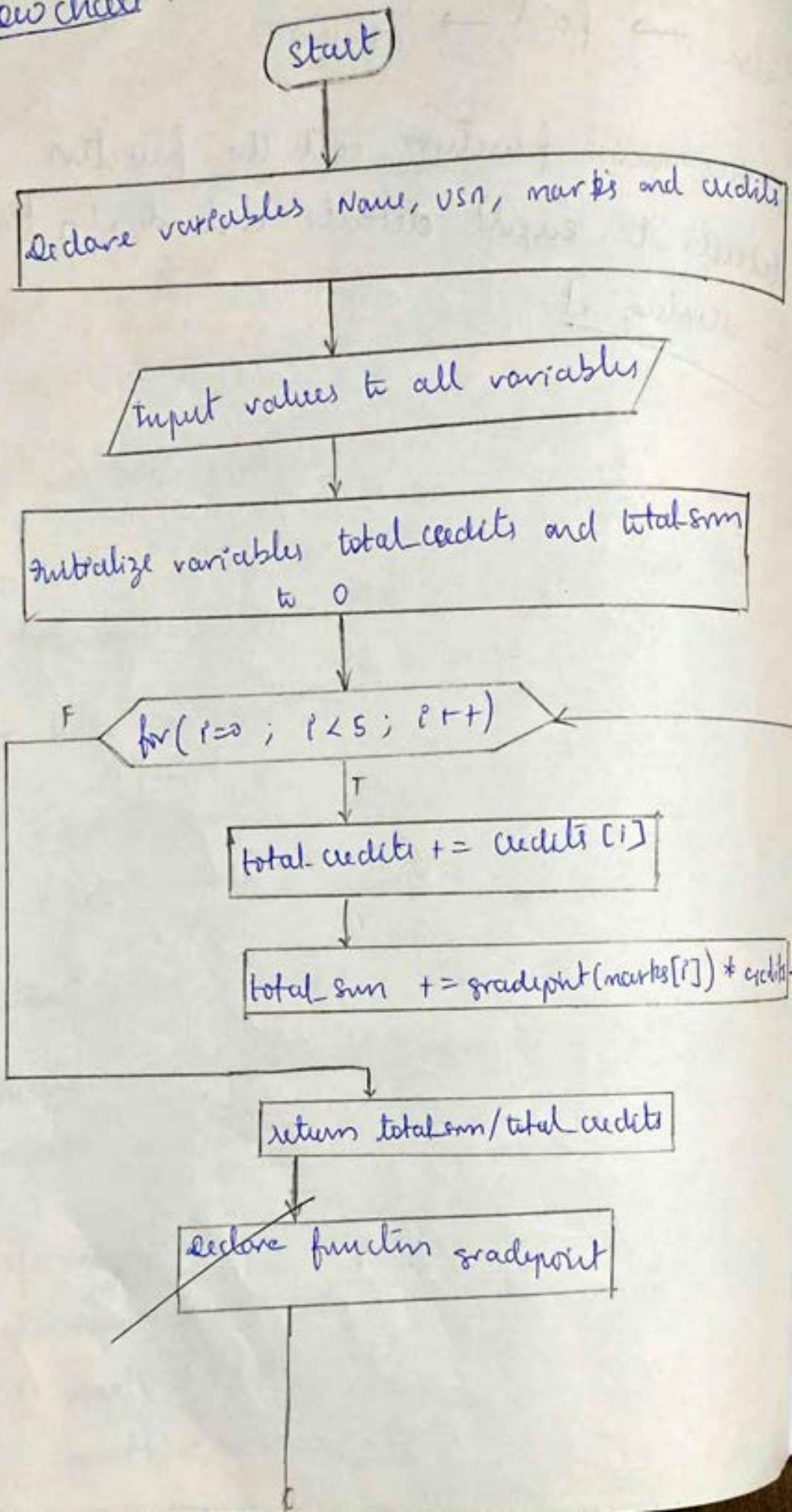
1. Start
2. Declare variables USN, name, credits, marks;
3. Input values for USN, name, credits, marks; ~~in a method~~ using a method
4. Create another method to calculate SGPA
5. In SGPA method, initialize variables ~~credits and total~~ total_credits and total_sum to 0;
6. for loop condition (i=0; i<5; i++)
7. total_credits += credits [i]
   total_sum += gradepoint (marks[i]) * credit
7. return value for SGPA function as total_sum/total_credit
8. create gradepoint method. and check for marks condition
9. ~~if marks >=90 → return 10~~
   ~~if marks >=80 → return 9~~
   if marks >=70 → return 8
   if marks >=60 → return 7

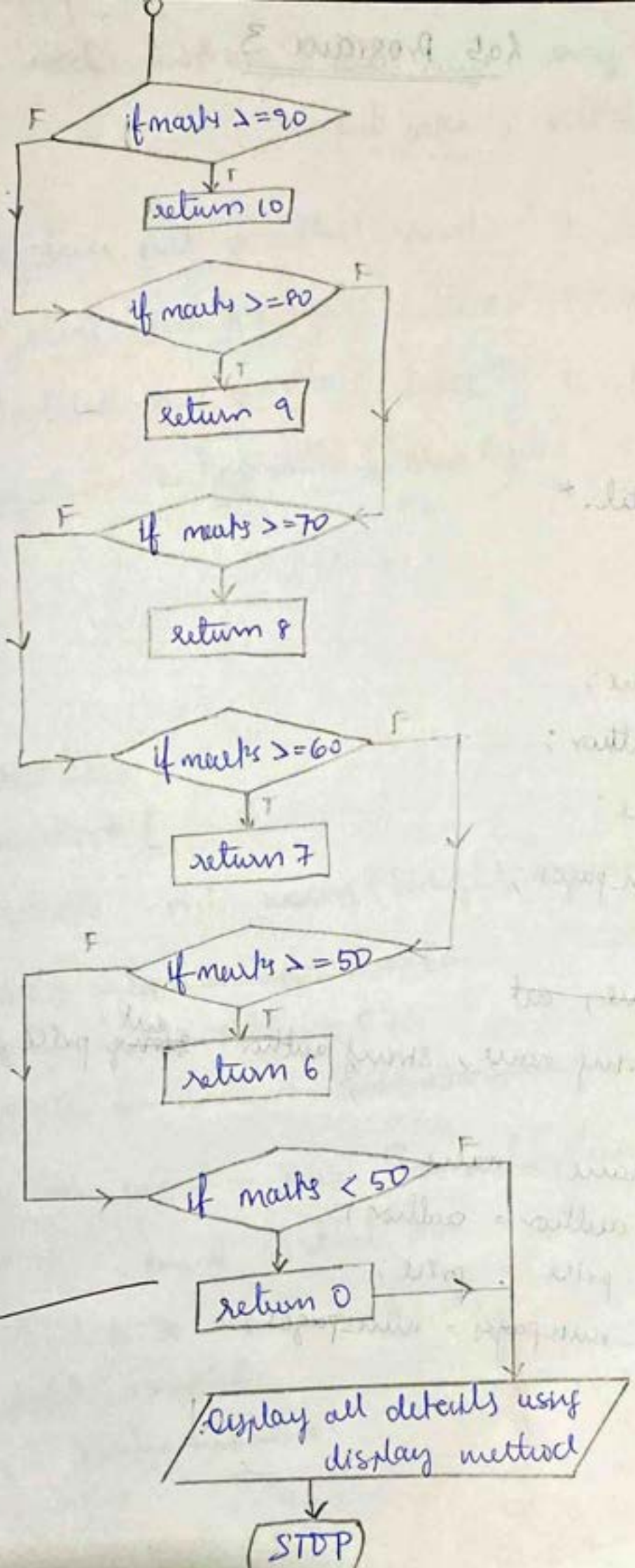
if marks >= 50 → return 6

else → fail → return 0

10. In the main function call the function details to input details and display function to display it.

Flow chart:


Start
Declare variables Name, USN, marks and credits
Input values to all variables
Initialize variables total_credits and total_sum to 0
for(i=0 ; i<5 ; i++)
F
T
total_credits += credits[i]
total_sum += gradepoint(marks[i]) * credits
return total_sum/total_credits
Declare function gradepoint


if marks >=90
F
T
return 10
if marks >=80
F
T
return 9
if marks >=70
F
return 8
if marks >=60
F
T
return 7
if marks >=50
F
T
return 6
if marks < 50
F
return 0
Display all details using display method
STOP

29/12/23

```
Enter USN: 1BM22CS005
Enter Name: Abhinav Raghu

Enter Number of Subjects: 5
Enter number of credits for Subject 1: 4
Enter marks recieved for Subject 1: 92

Enter number of credits for Subject 2: 3
Enter marks recieved for Subject 2: 89

Enter number of credits for Subject 3: 2
Enter marks recieved for Subject 3: 97

Enter number of credits for Subject 4: 1
Enter marks recieved for Subject 4: 98

Enter number of credits for Subject 5: 4
Enter marks recieved for Subject 5: 88

Student details:
USN: 1BM22CS005
Name: Abhinav Raghu
Credits for Subject 1: 4
Marks recieved for Subject 1: 92
Credits for Subject 2: 3
Marks recieved for Subject 2: 89
Credits for Subject 3: 2
Marks recieved for Subject 3: 97
Credits for Subject 4: 1
Marks recieved for Subject 4: 98
Credits for Subject 5: 4
Marks recieved for Subject 5: 88
SGPA: 9.5

-Abhinav Raghu
 1B22CS005
```

Lab Program - 3

12/01/2

Q:

Code:

```
import java.util.*

class Book {
    string name;
    string author;
    int price;
    int num_pages;

    ~~Book (name, int~~
    Book (string name, string author, ~~string~~ int price, int numpages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numpages = num_pages;
    }
```

```
public void toString (String name, String author,
                      int price, int num_pages)
{
    System.out.println("Name: " + name);
    System.out.println("Author: " + author);
    System.out.println("Price:" + price);
    System.out.println("Num_Pages: " + num_pages);
}

}

~~public static void~~
class ProgramBook {
  public static void main(String[], argument) {
    ~~Book[] b = new Book[n~~
    ~~int n = 3;~~ int i = 0;
    ~~Book[] b = new Book[n];~~ System.out.println("Enter no. of books:");
    Scanner input = new Scanner(System.in);
    int n = input.nextInt();
    Book[] b = new Book[n];
    ~~while (n>=0){~~
        ~~b[n] = input.n~~
```

```
while (n>=0){
  System.out.println("Enter name:");

  ~~b[i]= new Book~~

  int n = input.next()

  System.out.println("Enter author:");

  int a = input.next();

  System.out.println("Enter price");

  int p = input.nextInt();

  System.out.println("Enter num pages:");

  int num = input.nextInt();

  Book b[n]= new Book(n, a, p, num);

  n= n-1;  b[n].toString();
}

}
```

Algorithm:

1. start
2. create class Book with variables name, author, price and num-pages; and a parametric constructors
3. create to string method to display the values.
4. In the main create a scanner instance to input values for the variables ~~and~~ mentioned in Book class.
5. After inputting the values pass those values through the constructor.
6. call function string to print it
7. Stop

```
Enter the number of Books: 2

Enter details for Book 1:
Name: The Catcher in the Rye
Author: J.D. Salinger
Price: 14.99
Number of Pages: 224
Details: A coming-of-age novel capturing teenage angst.

Enter details for Book 2:
Name: The Hobbit
Author: J.R.R. Tolkien
Price: 19.99
Number of Pages: 310
Details:

Book [name=The Catcher in the Rye, author=J.D. Salinger, price=14.99, num_pages=224 ,details: A coming-of-age novel capturing teenage angst.]

Book [name=The Hobbit, author=J.R.R. Tolkien, price=19.99, num_pages=310 ,details: ]


-Abhinav Raghu
 1B22CS005
```

Lab - Program - 4

Q.

Q

code:

```
abstract class shape {
        int dimension 1;
        int dimension 2;
        void printArea() {
        }
}
class Rectangle extends shape {
        public Rectangle (int length, int width) {
                this.dimension1 = length;
                this.dimension2 = width;
        }

        void printArea() {
                int area = dimension1 * dimension2;
                System.out.println("Rectangle Area:" +
                                                area);
        }
}
```

```
class Triangle extends Shape {
    public Triangle (int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    void printArea() {
        double area = 0.5 * dimension1 *
                          dimension2;

        System.out.println("Triangle area:"
                          + area);
    }

}

class Circle extends Shape {
    public Circle (int radius) {
        this.dimension1 = radius;
        this.dimension2 = 0;

    void printArea() {
        double area = Math.PI * dimension1
                          * dimension1;

        System.out.println("Circle area:"
                          + area);
```

```
class Program_Shape{
    public static void main(String[], argument){
        Rectangle rec = new Rectangle(5,10);
        Triangle tri = new Triangle(4,7);
        circle cir = new Circle(2);

        rec.printArea();
        tri.printArea();
        cir.printArea();
    }
}
```

Algorithm

1. start
2. create abstract class shape and create variables dimension 1 and dimension 2 and a function print Area
3. create classes rectangle, ~~inporting all~~ triangle, circle that extends to class shape
4. specify different printArea method in each to override the existing printArea method

5. set the area in ~~rectangle~~ triangle class as

area = 0.5 * dimension1 * dimension2;

and area in rectangle class as

area = dimension 1 * dimension2

and area in circle class as

area = Math.PI * dimension1 * dimension2;

6. In main method, create constructors and call the printArea() method to print the values of the areas of ~~call the~~ rectangle, circle and triangle classes.

Practical

```
Area of Rectangle: 180
Area of Triangle: 40.0
Area of circle: 78.5398163397448


-Abhinav Raghu
 1B22CS005
```

Lab - 5

19/01/24

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account ~~for its customer~~ and the other current account. The savings account ~~and the~~ provides compound interest and withdrawal facilities but no check book facility. The current account provides check book facility but no interest. Current account holder 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 device the class Cur-acct and Sav-acct make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

(a) Accept deposit from customer and update the balance

(b) Display the balance

(c) Compute and deposit interest

(d) Permit withdrawal and update the balance.

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

```
import java.util.Scanner

class Account {
    String customerName;
    long accountNumber;
    String accountType;
    double balance;
    public Account (String customerName, long accountNumber,
                    String accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0.0;
    }
    public void deposit(double amount) {
        balance = balance + amount;
        System.out.println("Updated balance:" + balance)
    }
    public void displayBalance() {
        System.out.println("Account Balance: " + balance);
    }
}
```

```
class CurrAcct extends Account {
        double minimumB = 1000 ;
        double serviceCharge = 40 ;

        public CurrAcct (String customerName, long
                        accountNumber) {
            this.customerName = customerName;
            this.accountNumber = accountNumber;
            this.accountType = 'Current';
            this.balance = 0.0;
        }
        public void deposit (double amount) {
                balance = balance + amount;
            System.out.println("Updated Balance:" +
                                balance);

                minBalance();
        }
        public void minBalance() {
            if ( balance < minBalance) {
                balance = balance - serviceCharge;
            System.out.println("
                "Updated Balance:" + balance);
            }
        }
}
```

```
class SavAcct extends Account {
    double intrestRate = 0.05;

    public SavAcct (String customerName, long
                    accountNumber) {

        this.customer = customerName;
        this.accountNumber = accountNumber;
        this.accountType = "savings";
        this.balance = 0;
    }

    public void computeIntrest() {
        double intrest = balance*(1+(.005/12)) - balance;
        balance = balance + intrest
        System.out.println("Updated Balance:"
        + Balance);
    }

    public void Deposit(double amount) {
        balance = balance + amount;
        System.out.println("Updated: "+balance);
        computeIntrest();
    }
```

```
    public void withdraw(double amount){
        if (amount <= balance) {
            balance = balance - amount;
            System.out.println("Updated Balance:" +
                                balance);
        }
        else {
            System.out.println("Bank account balance
                                is not enough");
        }
    }
}

class Bank {
    public static void main(String[], args) {
        CurrAcct currentAccount = new CurrAcct("John Doe",
                                    9234561110);
        SavAcct savingsAccount = new SavAcct("Will Smith",
                                    98765432);

        ~~SavingsAccount~~
        CurrentAccount.deposit(1500);
        CurrentAccount.displayBalance();
```

```
SavingAccount.deposit(2000);
SavingAccount.displaybalance();
SavingAccount.withdraw(500);
}

}
```

Algorithm

1. Start
2. Create class Account and declare variables customername, accountNumber, accountType, balance.
3. In the account class create methods Account, deposit, and display balance, ~~which are used to~~ where account method is used to enter the customer details, ~~and~~ deposit method is used to increase the balance amount with the deposited amount. and the displaybalance method is used to show the account balance.
4. Create class CurrAcct extending to Account class. ~~Data~~ declare the minimum balance and service charge variable and initialize some values.

5. create methods to get customer details, deposit amount and another method to apply service charge if the balace is less than the minimum required balance.

6. Create class Savacct that extends to class account. ~~create~~ ~~dat~~ initialize the variable intrestRate to 0.05

7. create methods in that to get customer details, deposite amount and another method to compute the compound intrest rate using the formula

balance * (1 + (0.05/12)) - balance.

and also create a method to reduce balance amount upon withdrawal and also show if the amount to be withdrawn is more than the available balance.

8. create the main class Bank.

9. In the main method of this class ~~create variable~~ create objects currentAccount and savings account which are used to input values for current account and the savings account.

10. provide values to the accounts and display the balance and the updated balance

11. STOP.

```
Enter the number of customers: 4

Enter details for Customer 1:
Customer Name: Rajesh
Account Type (Savings/Current): Savings
Account Number: 453489
Initial Balance: 80000

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 3
Account Number: 453489
Customer Name: Rajesh
Account Type: Savings
Current Balance: 80000.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 1
Enter deposit amount: 1400
Deposit successful. Updated balance: 81400.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 2
Enter withdrawal amount: 400
Withdrawal successful. Updated balance: 81000.0
```

```
Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 3
Account Number: 453489
Customer Name: Rajesh
Account Type: Savings
Current Balance: 81000.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 4
Enter interest rate for savings account: 10
Interest deposited. Updated balance: 89100.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 5

Enter details for Customer 2:
Customer Name: Suraj
Account Type (Savings/Current): Current
Account Number: 334578
Initial Balance: 90000
Minimum Balance: 20000
Service Charge: 1500
```

```
Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 3
Account Number: 334578
Customer Name: Suraj
Account Type: Current
Current Balance: 90000.0

Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 2
Enter withdrawal amount: 10000
Withdrawal successful. Updated balance: 80000.0

Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 2
Enter withdrawal amount: 70000
Withdrawal successful, but below minimum balance.
Excess amount: 50000.0
Balance after excess withdrawal: 80000.0
Service charge imposed. Updated balance: 8500.0
```

```
Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 3
Account Number: 334578
Customer Name: Suraj
Account Type: Current
Current Balance: 8500.0

Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 1
Enter deposit amount: 10000
Deposit successful. Updated balance: 18500.0

Operations on Current Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Move to next customer
Enter your choice (1-4): 4

Enter details for Customer 3:
Customer Name: Lokesh
Account Type (Savings/Current): Savings
Account Number: 45763
Initial Balance: 50000

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Move to next customer
Enter your choice (1-5): 5
```

```
Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Exit
Enter your choice (1-5): 2
Enter withdrawal amount: 400
Withdrawal successful. Updated balance: 51000.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Exit
Enter your choice (1-5): 3
Account Number: 45763
Customer Name: Lokesh
Account Type: Savings
Current Balance: 51000.0

Operations on Savings Account:
1. Deposit
2. Withdraw
3. Display Balance
4. Deposit Interest
5. Exit
Enter your choice (1-5): 5

Abhinav Raghu
```

Program - 6 02/02/2024

Q. 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 package SEE which has the class External which is a derived class of student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

A: ①

```
package CIE;
public class Student {
    string usn, name;
    int sem;
}
public class Internals extends Student {
    int [] internalMarks = new int [5];
}
```

(2)

```
package SEE;
import CIE.Student;
public class External extends Student {
    int[] seeMarks = new int[5];
}
```

(3)

```
import CIE.Internals;
import SEE.Internals;

public class Main {
    public static void main(String[] args){
        int n=5;
        Internals[] internalStudents = new Internals[n];
        External[] externalStudents = new External[n];
        for (int i=0; i<n ; i++) {
            internalStudents[i] = new Internals();
            internalStudents[i].usn = "USN" + i;
            internalStudents[i].name = "Student" +i;
            internalStudents[i].sem = 2;
            internalStudents[i].internalMarks =
            new int[] { 90, 75, 90, 85, 88};
```

```
external students[i] = new External();
external students[i].usn = "USN" + i;
external students[i].name = "Student" + i;
external students[i].sem = 2;
external students[i]. ~~= new~~ seeMarks =
        new int[] {75, 80, 88, 92, 76};

}
for (int i=0; i<n; i++) {
    System.out.println("Student" + i +"
                        - Final Marks:");

    for (int j=0; j<5; j++) {
        int finalMarks =
        external students[i].externalMarks[j]
        + external students[i].seeMarks[j];

        System.out.println("Course" + (j+1) +":" +
                                finalMarks);
    }
    System.out.println();

}
}
}
```

Algorithm:

~~1. Create package CIE~~

1. start.

2. (i) Create package CIE and create class student. and declare variables name, usn and sem with appropriate data type

(ii) create another class ~~and~~ Internals that extends student class. And declare array internals of size 5 to store the internal marks

3. (i) Create package SEE and import the package CIE which was already created.

(ii) create a class Externals that extend the student class from CIE package. And then declare array seeMarks with size 5.

4. (i) Create another Main file and import the packages CIE and SEE already created.

(ii) In the main function initialize n with 5.

(iii) Create objects called ~~Internals[n]~~ internal students and external students from the classes Internals and Extend which were declared in the imported packages.

(iv) create a for loop which runs for n times to take n inputs for internals and externals

(v) create another for loop which again runs n times to print the final marks for each course

5. stop.

2/2/24

```
Number of Students: 2
Student 1

Enter USN: 1
Enter Name: A
Enter Sem: 3

Student 2

Enter USN: 2
Enter Name: B
Enter Sem: 2

Enter number of courses each student has:  5

Enter Internal marks out of 50.

Student 1
Course 1: 30
Course 2: 45
Course 3: 42
Course 4: 48
Course 5: 50

Student 2
Course 1: 25
Course 2: 30
Course 3: 44
Course 4: 36
Course 5: 38
```

```
External marks:
Course 1: 41
Course 2: 41
Course 3: 46
Course 4: 47
Course 5: 49

Total marks:
Course 1: 71
Course 2: 86
Course 3: 88
Course 4: 95
Course 5: 99
----------------------


----------------------
Student 2
USN: 2
Name: B
Sem: 2

Internal marks:
Course 1: 25
Course 2: 30
Course 3: 44
Course 4: 36
Course 5: 38

External marks:
Course 1: 38
Course 2: 40
Course 3: 43
Course 4: 40
Course 5: 46

Total marks:
Course 1: 63
Course 2: 70
Course 3: 87
Course 4: 76
Course 5: 84
----------------------
```

Abhinav Raghu

Program - 7

16/0

Q: Write a program that demonstrates handling exceptions on inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class implement a constructor which takes the age & throws the exception WrongAge() when the age < 0. In Son class, ~~implid~~ implement a constructor that cases both father and son's age and throws an exception if son's age >= father's age.

A:
```
import java.util.*;
class WrongAgeException extends Exception {
        WrongAgeException(String message) {
                super(message);
        }

}
```

```
class Father {
    private int age;

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

    int getAge() {
        return age;
    }
}

class Son extends Father {
    private int sonAge;

    Son ( int fatherAge, int sonAge) throws
        -WrongAgeException {
            super (fatherAge);
            if (sonAge >= fatherAge) {
                throw new WrongAgeException ("Son's
                    age should be less than father's age ");
            }
```

```
        this.sonAge = sonAge;
    }
    int getSonAge() {
        return sonAge;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        try {
            int fatherAge = input.nextInt();
            int sonAge = input.nextInt();
            Son sonInstance = new Son(fatherAge, sonAge);
            System.out.println("Father's age: " +
                sonInstance.getAge());
            System.out.println("Son's age: " +
                sonInstance.getSonAge());
        }
```

```
catch (NumberFormatException e) {
    System.out.println("Please enter valid
                        integer ages...");
}

catch (WrongAgeException e) {
    System.out.println("Error: " + e);
}
}
}
```

Program-7 Algorithm:

1) Start

2)(i) Create a class WrongAgeException which extends Exceptions

3)(ii) ~~create a~~ define the WrongAgeException constructor to print the error message.

3. (i) Create class Father

(ii) create variable age

(iii) ~~create a constructor~~ define the constructor Father to check if the age is greater than 0 i.e., to check if the entered age value is positive or not.

(iv) If the age value is not positive throw the WrongAgeException

(v) create an method to return the age value.

4. (i) create the class Son that extends Father class

(ii) define the constructor for Son which accepts the the sonage and fatherage and checks if the son's age is greater than ~~than~~ or equal to fatherage if true, then the WrongAgeException is raised.

(iii) here ~~also~~ also a method named son's age is defined to return son's age.

5. In the main, we create ~~extra~~ instances of the classes we have created and provide the age input for the son and Father.

6. At last after checking all the validity of the entered age. print the son's and father's age.

7. write a try-catch block to handle the wrongAgeException.

8. Stop.

```
Error: Invalid Age


Abhinav Raghu
```

```
Error: Son's age should be less than father's age

Abhinav Raghu
```

Program - 8

Q. 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.

A:

```
class DisplayThread extends Thread {
    private final String string message;
    private final int interval;

    DisplayThread (String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

    @Override
    public void run() {
        int i=0; i<=5
        while (true) {
            System.out.println (message);
            try {
                Thread.sleep (interval *1000); }
            catch (InterruptedException e) {
                e.printStackTrace();
            } i++;
        } }
}
```

```
class DisplayMessages {
    public static void main(String[] args){
        Thread t1 = new DisplayThread("BMS
            College of Engineering", 10);

        Thread t2 = new DisplayThread("CSE", 2);

        t1.start();
        t2.start();
    }
}
```

Pranal

Program-8 Algorithm

1) Start

2) (i) Create a class named Display Thread which extends Threads.

(ii) Create two variables message and interval.

(iii) ~~create~~ define the constructor to initialize these variables with the input.

(iv) Override the method run() and ~~make a~~ create a while loop which prints the message at intervals mentioned by the user.

3) (i) In the main method create the thread instances t1 and t2.

(ii) t1 instance should pass "BMS College of Engineering" as the message and 10 ~~and~~ as the interval which ~~means~~ intends to have 10s ~~det~~ interval.

(iii) t2 instance should pass "CSE" as the message and 2 as the interval to print CSE at intervals ~~of~~ 2s.

(iv) write t1.start() and t2.start() to start execution

4. Stop.

```
Abhinav Raghu
CSE
BMS
CSE
CSE
CSE
CSE
BMS
CSE
CSE
CSE
CSE
CSE
BMS
CSE
CSE
CSE
CSE
CSE
BMS
CSE
CSE
CSE
CSE
CSE
CSE
CSE
BMS
```

LAB - 9 23/02/24

Q) Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

A)

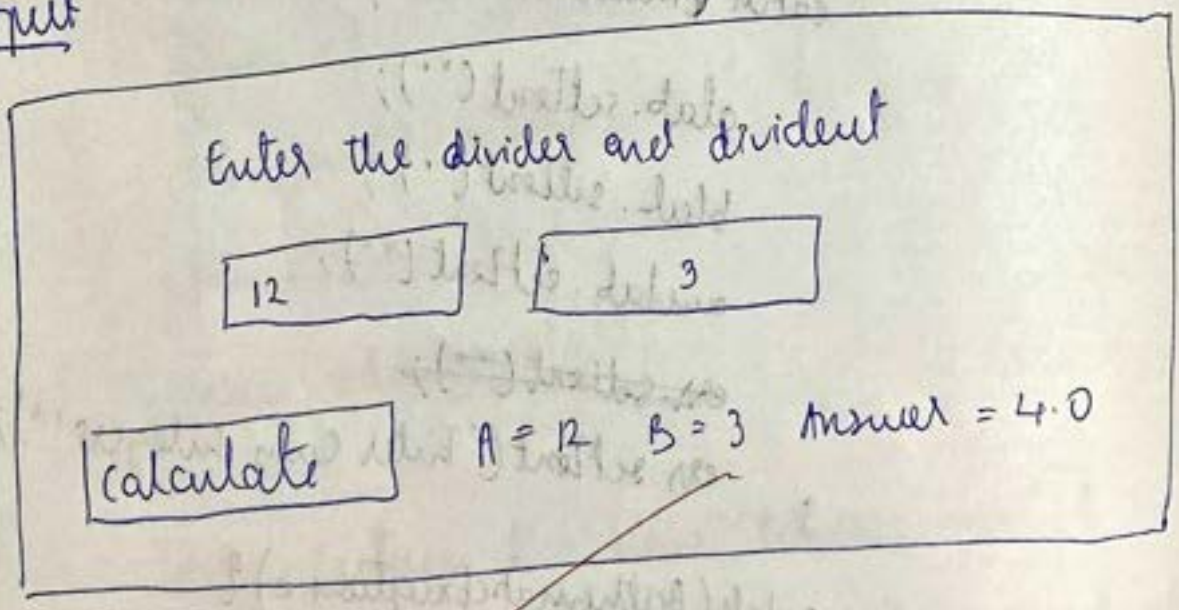
```
import javax.swing.*
import java.awt.*
import java.awt.event.*;

public 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
                                  dividend:");
```

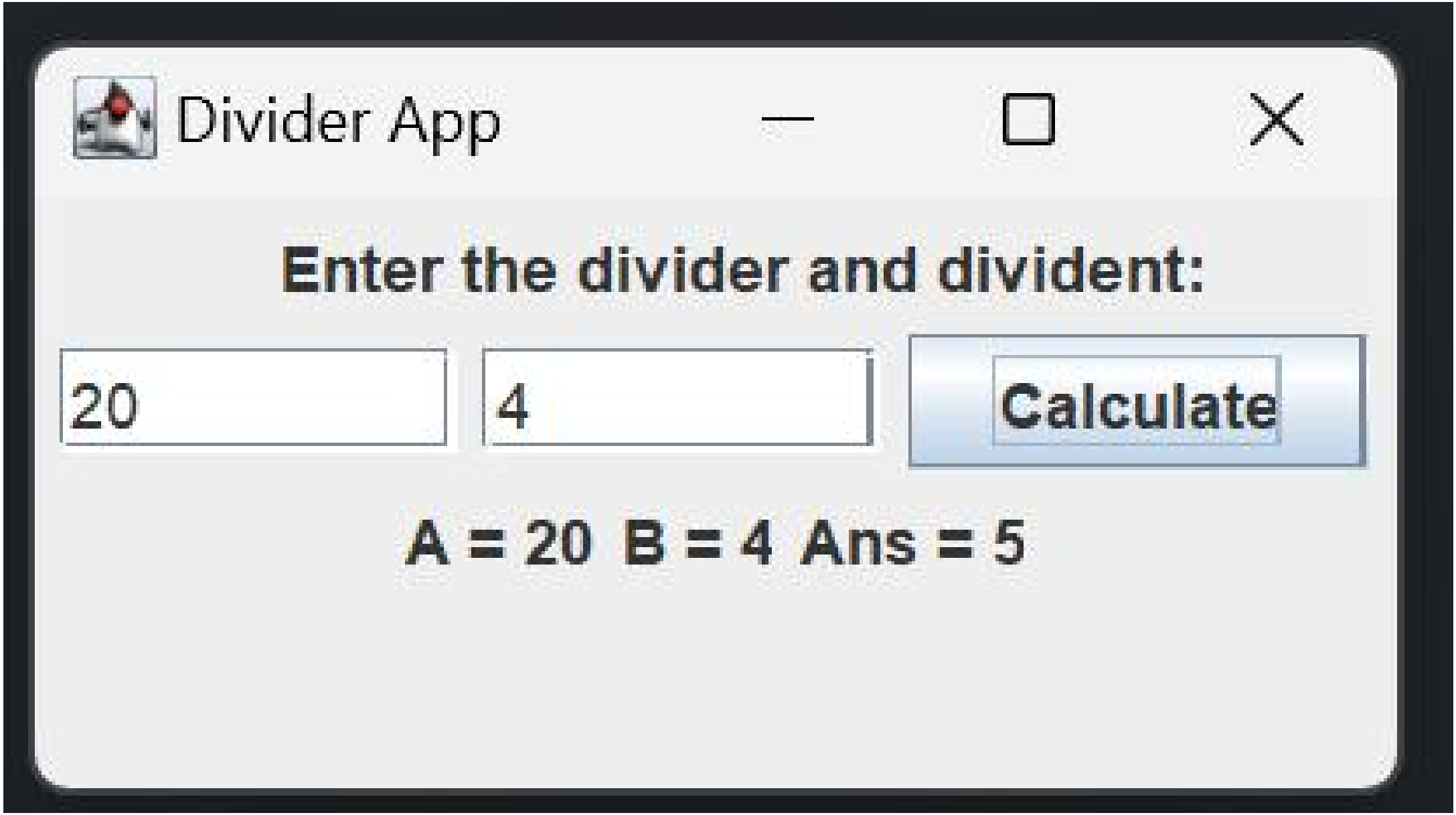
```
JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8)

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(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListner l = new ActionListner(){
public void actionPerformed(ActionEvent evt){
System.out.println("Action event from 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("In A = " +a);
            blab.setText("In B = " +b);
            anslab.setText("InAns = " + ans);
        }
        catch(NumberFormatException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            ~~err.setText("");~~
            err.setText("Enter Only Integers!");
        }
        catch(ArithmaticException e){
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("B should be NON zero");
        }
    }
});
```

```
frm.setVisible(true);

}

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

output




Divider App
Enter the divider and divident:
20
4
Calculate
A = 20 B = 4 Ans = 5

Functions

setSize()

⇒ Resize a component to have the specified width & height.

setLayout()

⇒ It allows us to set the layout of the container to to FlowLayout, BorderLayout, GridLayout, etc.

actionPerformed()

⇒ It helps manage user interactions with GUI components.

setText()

⇒ It is used to set the text inside a label or other screen element

getText()

⇒ It returns the value from the single-line-text field.

23.02.24