VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

Samriddhi Singh (1BM23CS295)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep 2024-Jan 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by **Samriddhi Singh(1BM23CS295),** who is a bona fide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

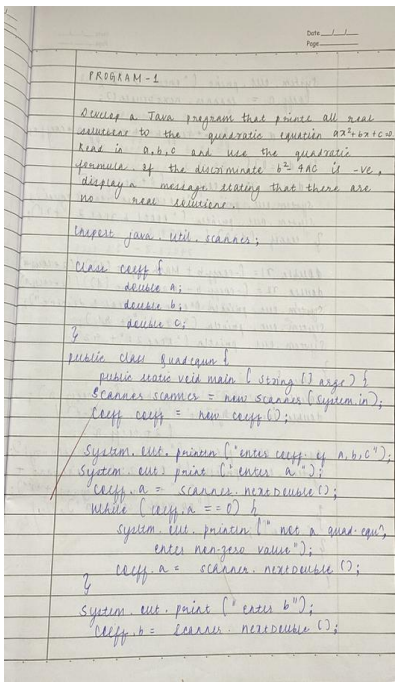
Index

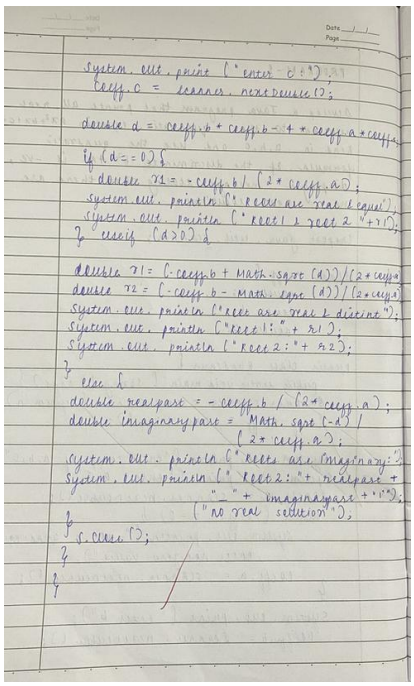
Github Link: samriddhisingh05/1BM23CS295_java

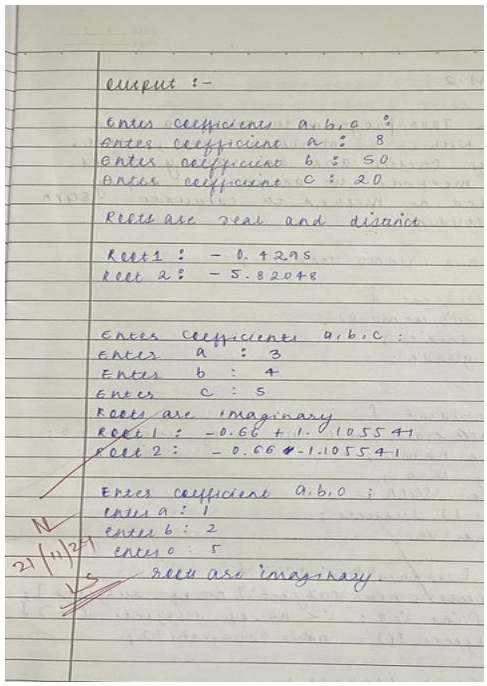
Program 1

Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real
solutions.

Algorithm:







Code:

```
import java.util.Scanner;

class Coeff {
   double a;
   double b;
   double c;
}
public class QuadraticEquation {
   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);
      Coeff coeff = new Coeff();

      System.out.println("Enter the coefficients of a, b, c:");
```

```
        System.out.print("Enter coefficient a: ");
        coeff.a = scanner.nextDouble();
        while (coeff.a == 0) {
            System.out.println("Not a quadratic equation. Please enter a non-zero value for a:");
            coeff.a = scanner.nextDouble();
        }

        System.out.print("Enter coefficient b: ");
        coeff.b = scanner.nextDouble();
        System.out.print("Enter coefficient c: ");
        coeff.c = scanner.nextDouble();

        double d = coeff.b * coeff.b - 4 * coeff.a * coeff.c;

        if (d == 0) {
            double r1 = -coeff.b / (2 * coeff.a);
            System.out.println("Roots are real and equal.");
            System.out.println("Root 1 and Root 2: " + r1);
        } else if (d > 0) {
            double r1 = (-coeff.b + Math.sqrt(d)) / (2 * coeff.a);
            double r2 = (-coeff.b - Math.sqrt(d)) / (2 * coeff.a);
            System.out.println("Roots are real and unique.");
            System.out.println("Root 1: " + r1);
            System.out.println("Root 2: " + r2);
        } else {
            double realPart = -coeff.b / (2 * coeff.a);
            double imaginaryPart = Math.sqrt(-d) / (2 * coeff.a);
            System.out.println("Roots are imaginary.");
            System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
            System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
        }
        scanner.close();
    }
}
```

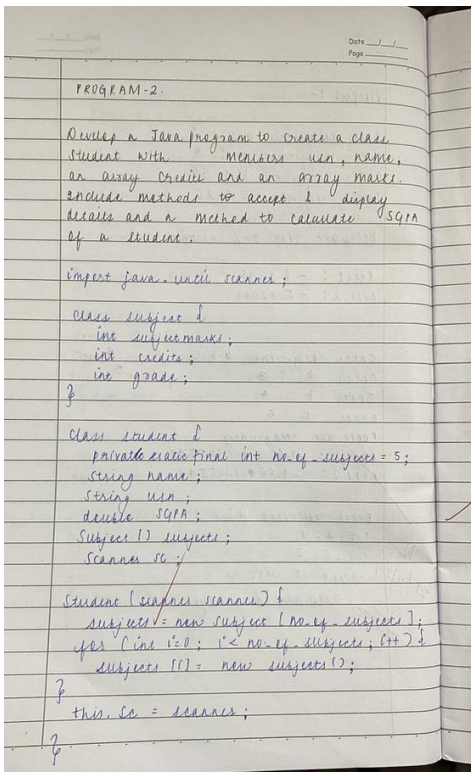
Output:

```
Command Prompt

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>cd 295

C:\295>set path="C:\Program Files\Java\jdk-23\bin"

C:\295>javac QuadraticEquation.java

C:\295>java QuadraticEquation
Enter the coefficients of a, b, c:
Enter coefficient a: 8 50 20
Enter coefficient b: Enter coefficient c: Roots are real and unique.
Root 1: -0.42951766839402206
Root 2: -5.820482331605978

C:\295>javac QuadraticEquation.java

C:\295>java QuadraticEquation
Enter the coefficients of a, b, c:
Enter coefficient a: 3 4 5
Enter coefficient b: Enter coefficient c: Roots are imaginary.
Root 1: -0.6666666666666666 + 1.10554159678513 32i
Root 2: -0.6666666666666666 - 1.10554159678513 32i

C:\295>javac QuadraticEquation.java

C:\295>java QuadraticEquation
Enter the coefficients of a, b, c:
Enter coefficient a: 1 2 5
Enter coefficient b: Enter coefficient c: Roots are imaginary.
Root 1: -1.0 + 2.0i
Root 2: -1.0 - 2.0i

C:\295>
```

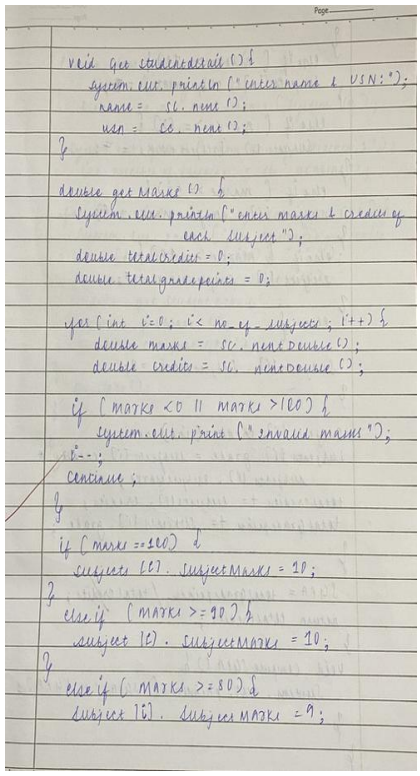
Program 2

Develop a Java program to create a class Student with members usn, name, an array credits and an array marks.Include methods to accept and display details and amethod to calculate SGPA of a student.

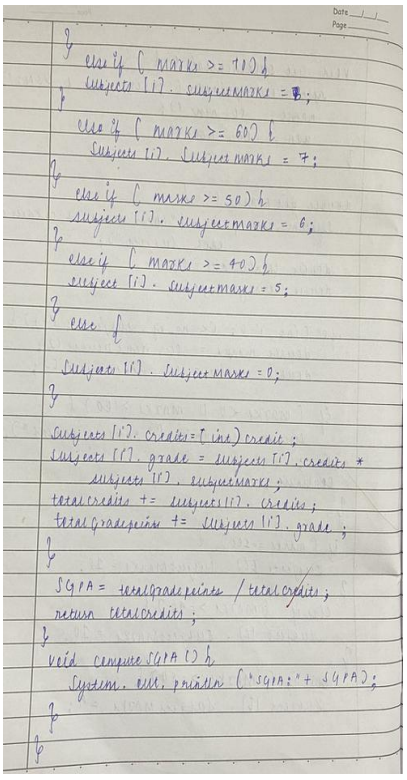
Algorithm:



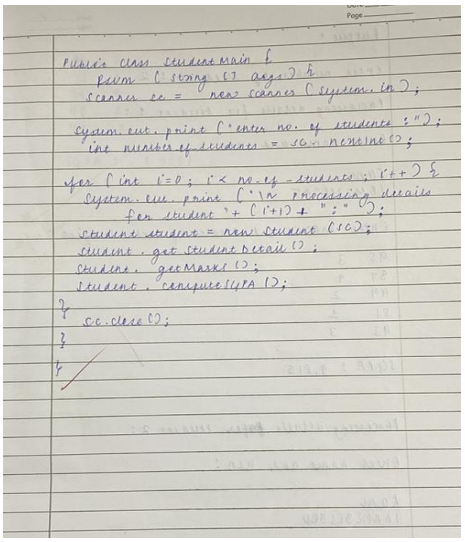
```
PROGRAM-2.

Develop a Java program to create a class
Student with members usn, name,
an array credits and an array marks.
Include methods to accept & display
details and a method to calculate SGPA
of a student.

import java.util.Scanner;

class Subject {
    int subjectmarks;
    int credits;
    int grade;
}

class Student {
    private static final int no_of_subjects = 5;
    String name;
    String usn;
    double SGPA;
    Subject[] subjects;
    Scanner sc;

Student(Scanner scanner) {
    subjects = new Subject[no_of_subjects];
    for (int i=0; i< no_of_subjects; i++) {
        subjects[i] = new subjects();
    }
    this.sc = scanner;
}
```



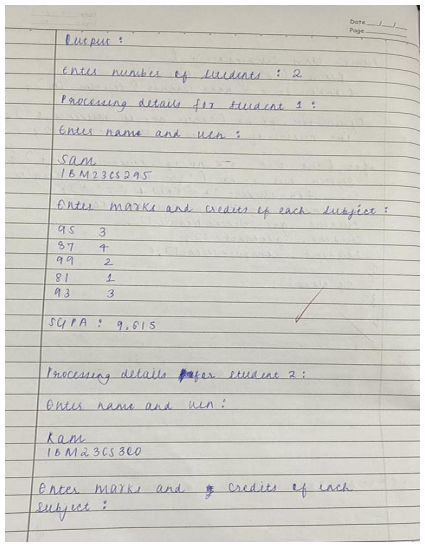
```
void getstudentdetails() {
    System.out.println("enter name & USN:");
    name = sc.next();
    usn = sc.next();
}

double getMarks() {
    System.out.println("enter marks & credits of
                each subject");
    double totalcredits = 0;
    double totalgradepoints = 0;

    for (int i=0; i< no_of_subjects; i++) {
        double marks = sc.nextDouble();
        double credits = sc.nextDouble();

        if (marks<0 || marks>100) {
            System.out.print("enravlid marks");
            i--;
            continue;
        }

        if (marks==100) {
            subjects[i].subjectMarks = 10;
        }
        else if (marks>=90) {
            subject[i].subjectMarks = 10;
        }
        else if (marks>=80) {
            subject[i].subjectMarks = 9;
```



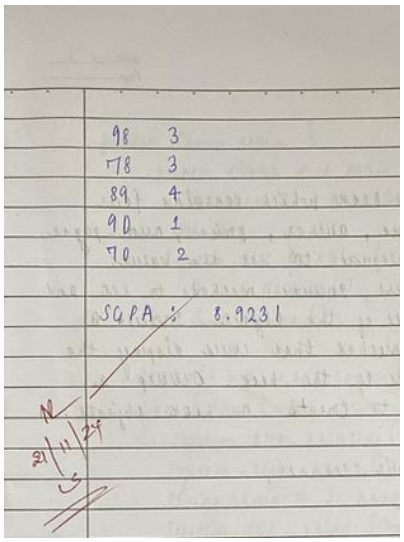
```
        }
        else if (marks>=70) {
            subjects[i].subjectMarks = 8;
        }
        else if (marks>=60) {
            subjects[i].subjectMarks = 7;
        }
        else if (marks>=50) {
            subjects[i].subjectmarks = 6;
        }
        else if (marks>=40) {
            subject[i].subjectmarks = 5;
        }
        else {

            subjects[i].subjectMarks = 0;
        }

        subjects[i].credits=(int) credits;
        subjects[i].grade = subjects[i].credits *
            subjects[i].subjectMarks;
        totalcredits += subjects[i].credits;
        totalgradepoints += subjects[i].grade;
    }

    SGPA = totalgradepoints / totalcredits;
    return totalcredits;
}
void computeSGPA() {
    System.out.println("SGPA:"+ SGPA);
}
}
```



```
Public class StudentMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("enter no. of students:");
        int numbers of students = sc.nextInt();

        for (int i=0; i< no.of-students; i++) {
            System.out.print("\n processing details
                for student "+(i+1)+ ":");
            student student = new student(sc);
            student.getstudentdetail();
            student.getMarks();
            student.computeSGPA();
        }
        sc.close();
    }
}
```



```
Output:

enter numbers of students : 2

Processing details for student 1 :

Enter name and usn :

Sam
1BM23CS295

Enter marks and credits of each subject :

95  3
87  4
99  2
81  1
93  3

SGPA : 9.615

Processing details for student 2 :

Enter name and usn :

Ram
1BM23CS300

Enter marks and credits of each
subject :
```



```
98  3
78  3
89  4
90  1
70  2

SGPA : 8.9231
```

N
21/11/24

Code:

```
import java.util.Scanner;
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}

class Student {
    private static final int NUMBER_OF_SUBJECTS = 8;
    String name;
    String usn;
    double SGPA;
    Subject[] subjects;
    Scanner sc;

    Student(Scanner scanner) {
        subjects = new Subject[NUMBER_OF_SUBJECTS];
        for (int i = 0; i < NUMBER_OF_SUBJECTS; i++) {
            subjects[i] = new Subject();
        }
        this.sc = scanner;
    }

    void getStudentDetail() {
        System.out.println("Enter name and USN:");
        name = sc.next();
        usn = sc.next();
    }

    double getMarks() {
        System.out.println("Enter marks and credits of each subject:");
        double totalCredits = 0;
        double totalGradePoints = 0;

        for (int i = 0; i < NUMBER_OF_SUBJECTS; i++) {
            double marks = sc.nextDouble();
            double credits = sc.nextDouble();

            if (marks < 0 || marks > 100) {
                System.out.println("Invalid marks. Please enter marks between 0 and 100.");
                i--;
                continue;
```

```
            }

            if (marks == 100) {
                subjects[i].subjectMarks = 10;
            } else if (marks >= 90) {
                subjects[i].subjectMarks = 10;
            } else if (marks >= 80) {
                subjects[i].subjectMarks = 9;
            } else if (marks >= 70) {
                subjects[i].subjectMarks = 8;
            } else if (marks >= 60) {
                subjects[i].subjectMarks = 7;
            } else if (marks >= 50) {
                subjects[i].subjectMarks = 6;
            } else if (marks >= 40) {
                subjects[i].subjectMarks = 5;
            } else {
                subjects[i].subjectMarks = 0;
            }

            subjects[i].credits = (int) credits;
            subjects[i].grade = subjects[i].credits * subjects[i].subjectMarks;
            totalCredits += subjects[i].credits;
            totalGradePoints += subjects[i].grade;
        }

        SGPA = totalGradePoints / totalCredits;
        return totalCredits;
    }

    void computeSGPA() {
        System.out.println("SGPA: " + SGPA);
    }
}

public class StudentMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of students: ");
        int numberOfStudents = sc.nextInt();

        for (int i = 0; i < numberOfStudents; i++) {
            System.out.println("\nProcessing details for student " + (i + 1) + ":");
            Student student = new Student(sc);
            student.getStudentDetail();
```

```
            student.getMarks();
            student.computeSGPA();
        }

        sc.close();
    }
}
```

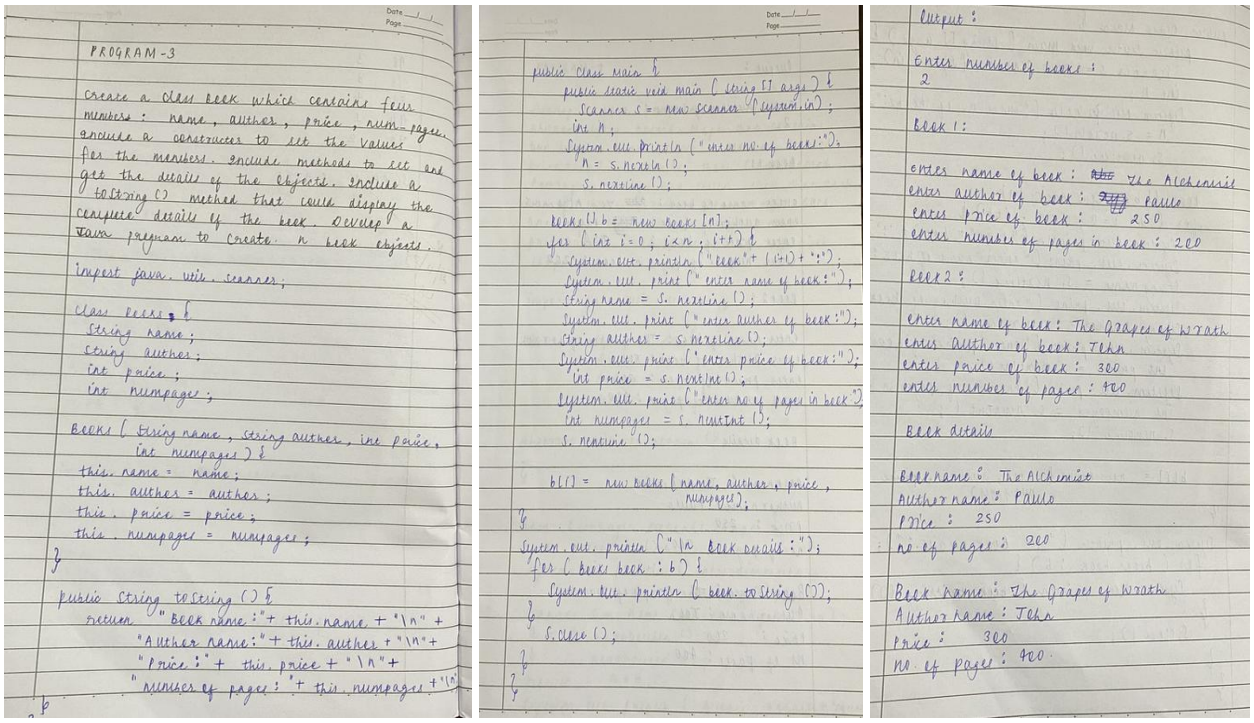
Output:

```
Command Prompt

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>cd 295

C:\295>set path="C:\Program Files\Java\jdk-23\bin"

C:\295>javac StudentMain.java

C:\295>java StudentMain
Enter the number of students: 2

Processing details for student 1:
Enter name and USN:
sam 1bm23cs295
Enter marks and credits of each subject:
95 3 87 4 99 2 81 1 93 3
SGPA: 9.615384615384615

Processing details for student 2:
Enter name and USN:
ram 1bm23cs300
Enter marks and credits of each subject:
98 3 78 3 89 4 90 1 70 2
SGPA: 8.923076923076923

C:\295>
```

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:



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.

```
import java.util.scanner;

class books {
  String name;
  String author;
  int price;
  int numpages;

  Books( String name, String author, int price,
         int numpages ) {
  this. name = name;
  this. author = author;
  this. price = price;
  this. numpages = numpages;
}

public String toString () {
    return "Book name:"+ this.name + "\n" +
           "Author name:"+ this. author + "\n"+
           "Price :"+ this. price + "\n"+
           " number of pages :"+ this. numpages +"\n"
}
```

```
public class main {
    public static void main ( String [] args ) {
      Scanner s = new scanner (System.in);
      int n;
      System.out.println ("enter no of books:");
      n = s.nextIn ();
      s. nextline ();

  Books[] b = new books [n];
  for ( int i = 0 ; i<n ; i++) {
      System. out. println ("book"+ (i+1) + ":");
      System. out. print ("enter name of book:");
      String name = s. nextline ();
      System. out. print ("enter author of book:");
      String author = s. nextline ();
      System. out. print ("enter price of book:");
      int price = s. nextIn ();
      System. out. print ("enter no of pages in book")
      int numpages = s. nextint ();
      s. nextline ();

      b[i] = new books ( name, author, price,
                         numpages);
  }
  System. out. println ("\n book details:");
  for ( books book : b) {
      System. out. println ( book. toString ());
  }
      s. close ();
}
}
```

```
Output :

Enter number of books :
2

Book 1:

enter name of book : The Alchemist
enter author of book : Paulo
enter price of book : 250
enter number of pages in book : 200

Book2 :

enter name of book: The Grapes of Wrath
enter author of book: John
enter price of book : 300
enter number of pages : 400

Book details

Book name : The Alchemist
Author name : Paulo
Price : 250
no of pages : 200

Book name : The Grapes of Wrath
Author name : John
Price : 300
no of pages : 400
```

Code:

```
import java.util.Scanner;

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

    Books(String name, String author, int price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
```

```
        this.numPages = numPages;
    }

    public String toString() {
        return "Book name: " + this.name + "\n" +
               "Author name: " + this.author + "\n" +
               "Price: " + this.price + "\n" +
               "Number of pages: " + this.numPages + "\n";
    }
}

public class Main1 {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n;

        System.out.println("Enter the number of books: ");
        n = s.nextInt();
        Books[] b = new Books[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Book " + (i + 1) + ":");
            System.out.print("Enter name of book: ");
            String name = s.next();
            System.out.print("Enter author of book: ");
            String author = s.next();
            System.out.print("Enter price of book: ");
            int price = s.nextInt();
            System.out.print("Enter number of pages in the book: ");
            int numPages = s.nextInt();

            b[i] = new Books(name, author, price, numPages);
        }

        System.out.println("\nBook Details:");
        for (Books book : b) {
            System.out.println(book.toString());
        }

        s.close();
    }
}
```

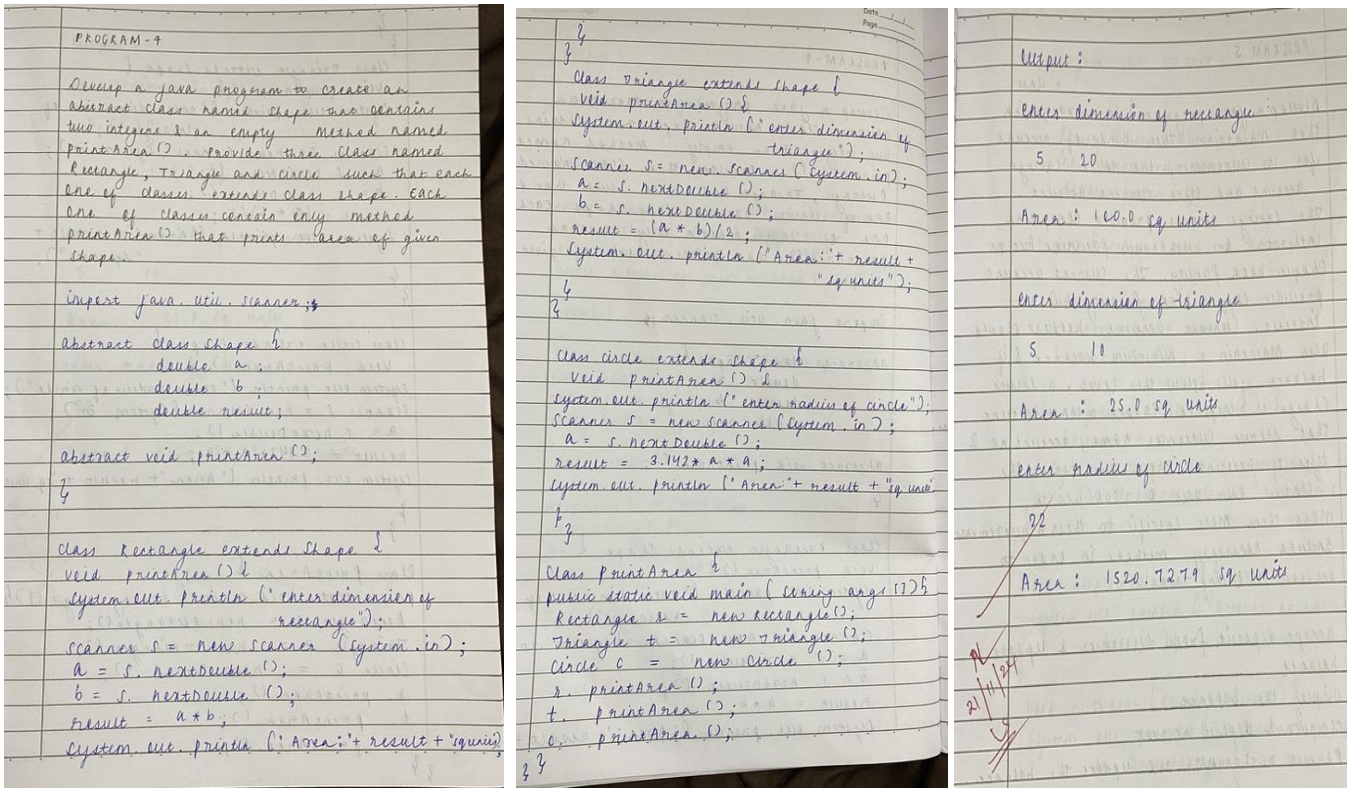
Output:

```
Command Prompt

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>cd 295

C:\295>set path="C:\Program Files\Java\jdk-23\bin"

C:\295>javac StudentMain.java

C:\295>java StudentMain
Enter the number of students: 2

Processing details for student 1:
Enter name and USN:
sam 1bm23cs295
Enter marks and credits of each subject:
95 3 87 4 99 2 81 1 93 3
SGPA: 9.615384615384615

Processing details for student 2:
Enter name and USN:
ram 1bm23cs300
Enter marks and credits of each subject:
98 3 78 3 89 4 90 1 70 2
SGPA: 8.923076923076923

C:\295>
```

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 name 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:



PROGRAM-4

Develop a java program to create an abstract class named shape that contains two integers & an empty method named printArea(). Provide three class named Rectangle, Triangle and circle such that each one of classes extends class shape. Each one of classes contain only method printArea() that prints area of given shape.

```
import java.util.scanner;

abstract class shape {
        double a;
        double b;
        double result;

abstract void printArea();
}

class Rectangle extends Shape {
void printArea() {
System.out.println("enter dimension of rectangle");
scanner s = new scanner (System.in);
a = s.nextDouble();
b = s.nextDouble();
result = a * b;
System.out.println("Area:"+ result + "squnits");
}
}

class Triangle extends shape {
void printArea() {
System.out.println("enter dimension of triangle");
Scanner s = new Scanner (System.in);
a = s.nextDouble();
b = s.nextDouble();
result = (a * b)/2;
System.out.println("Area:"+ result + "sq units");
}
}

class circle extends shape {
void printArea() {
System.out.println("enter radius of circle");
Scanner s = new Scanner (System.in);
a = s.nextDouble();
result = 3.142* a * a;
System.out.println("Area:"+ result + "sq units"
}
}

Class printArea {
public static void main (String args[]){
Rectangle r = new rectangle();
Triangle t = new triangle();
circle c = new circle();
r.printArea();
t.printArea();
c.printArea();
}}
```

Output:

```
enter dimension of rectangle
5 20
Area: 100.0 sq units

enter dimension of triangle
5 10
Area: 25.0 sq units

enter radius of circle
22
Area: 1520.7279 sq units
```

Code:

```
import java.util.Scanner;

abstract class Shape {
    double a;
    double b;
    double result;

    abstract void printArea();
}

class Rectangle extends Shape {
```

```
    void printArea() {
        System.out.println("Enter dimensions of rectangle:");
        Scanner s = new Scanner(System.in);
        a = s.nextDouble();
        b = s.nextDouble();
        result = a * b;
        System.out.println("Area: " + result + " square units");
    }
}

class Triangle extends Shape {
    void printArea() {
        System.out.println("Enter dimensions of triangle:");
        Scanner s = new Scanner(System.in);
        a = s.nextDouble();
        b = s.nextDouble();
        result = (a * b) / 2;
        System.out.println("Area: " + result + " square units");
    }
}

class Circle extends Shape {
    void printArea() {
        System.out.println("Enter radius of circle:");
        Scanner s = new Scanner(System.in);
        a = s.nextDouble();
        result = 3.142 * a * a;
        System.out.println("Area: " + result + " square units");
    }
}

class Printarea {
    public static void main(String args[]) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.printArea();
        t.printArea();
        c.printArea();
}
}
```

Output:

```
Command Prompt

Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>cd 295

C:\295>set path="C:\Program Files\Java\jdk-23\bin"

C:\295>javac Printarea.java

C:\295>java Printarea
Enter dimensions of rectangle:
5 20
Area: 100.0 square units
Enter dimensions of triangle:
5 10
Area: 25.0 square units
Enter radius of circle:
22
Area: 1520.7279999999998 square units

C:\295>
```

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 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 Cur-acct and Sav-acct to 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.

Algorithm:

```
import java.util.Scanner;

Class Account {
    String CustomerName;
    int accountnumber;
    double balance;

    Account (String name, int accno) {

        CustomerName = name;
        accountnumber = accno;
        balance = 0.0;
    }
    void deposit (double amount) {
        balance += amount;
        System.out.println ("Deposited:"+ amount+
            "\n Updated balance:"+ balance);
    }

    void displaybalance () {
        System.out.println ("Account balance:"+
            balance);
    }

    void withdraw(double Amount) {
        System.out.println ("Withdrawal is specific
            to Account type");

    void ComputeInterest () {
        System.out.println ("Interest can't be
            calculated for this account type");
    }
}

Class SavingsAccount extends Account {
    double interestrate = 0.04;

    SavingsAccount ( String name, int accno) {
        super (name, accno);
    }
    void computeinterest () {
        double interest = balance * interestrate;
        balance += interest;
        System.out.println ("Interest added:"+
            interest + "\n Updated balance:"+balance);
    }
    void withdraw ( double amount) {

        if ( balance >= amount) {
            balance -= amount;
            System.out.println ("Withdrawn:"+
            amount + "\n Updated balance:"+ balance);
        }
        else
        {
        System.out.println ("Insufficient balance:");
        }
    }
}

Class CurrentAccount extends Account {
    double minimumbalance = 500.0;
    double serviceCharge = 50.0;

    CurrentAccount ( String name, int accno)
    {
        super ( name, accno); }

    void checkminimumbalance () {

        if ( balance < minimumbalance) {
            System.out.println ("balance is
            below minimum. service charge imposed");
            balance -= serviceCharge;
            System.out.println (" service charge:"+
            serviceCharge + "\n updated balance:"+
            balance);
        }
    }

    void withdraw ( double amount) {
        if ( balance >= amount) {
            balance -= amount;
            System.out.println (" withdrawn:"+
            amount + "\n updated balance:"+
            balance);
            checkminimumbalance ();
        } else {
            System.out.println ("insufficient
            balance");
        }
    }
}

public Class Bank {
    public static void main (String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.print ("enter customer name");
        String name = sc.nextline ();
        System.out.print ("enter acc no.");
```

```
int accno = sc.nextInt();
sc.nextLine();
System.out.print("enter accountype
(savings/current):");
String acctype = sc.nextLine();

Account account;
if (acctype.equalsIgnoreCase("savings"))
{ account = new SavingsAccount(name,
                accno);
} else if
(acctype.equalsIgnoreCase("current"))
        account = new CurrentAccount(name,
                accno);
} else {
        System.out.println("Invalid");
        sc.close();
        return;
}
while (true) {
    System.out.println("\n Menu :");
    System.out.println("1. Deposit");
    System.out.println("2. Withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. Compute Interest");
    System.out.print("5. Exit");
    System.out.print("enter your choice");
    int choice = sc.nextInt();
```

```
switch (choice) {

    case 1 :
        System.out.print("enter deposit amount");
        double depositAmount = sc.nextDouble();
        account.deposit(depositAmount);
        break;

    case 2:
        System.out.print("enter withdrawal
                amount :");
        double withdrawAmount = sc.nextDouble();
        account.withdraw(withdrawAmount);
        break;

    case 3:
        account.displayBalance();
        break;

    case 4:
        account.computeInterest();
        break;

    case 5:
        System.out.println("exiting");
        sc.close();
        return;
    default:
        System.out.print("invalid choice");
}
}
}
```

```
Output :-

Enter customer name : sam
Enter account number : 123
Enter account type (savings/current) : savings

Menu :
1. Deposit
2. Withdraw
3. Display balance
4. Compute Interest
5. Exit

Enter your choice : 1
Enter deposit amount : 500
deposited : 500.0
Updated Balance : 500.0

Enter your choice : 4
Interest added : 20.0
Updated balance : 520.0

Enter your choice : 2
Enter withdrawal amount : 10
Withdrawn : 10.0
Updated Balance : 510.0
```

```
Enter customer name : xyz
Enter account number : 134
Enter account type (savings/current) : current

Enter your choice : 1
Enter deposit amount : 210
Deposited : 210.0
Updated balance : 210.0

Enter your choice : 2
Enter withdrawal amount : 10.
Withdrawn : 10.0
Updated Balance : 200.0
Balance is below minimum. service
charge imposed.
Service Charge : 50.0
Updated balance : 150.0
```

Code:

```
import java.util.Scanner;

class Account {
   String customerName;
   int accountNumber;
   double balance;

   Account(String name, int accNumber) {
      customerName = name;
      accountNumber = accNumber;
      balance = 0.0;
   }

   void deposit(double amount) {
      balance += amount;
      System.out.println("Deposited: " + amount + "\nUpdated Balance: " + balance);
   }

   void displayBalance() {
      System.out.println("Account Balance: " + balance);
   }

   void withdraw(double amount) {
      System.out.println("Withdrawal is specific to account type.");
   }

   void computeInterest() {
      System.out.println("Interest computation not applicable for this account type.");
   }
}

class SavingsAccount extends Account {
   double interestRate = 0.04;

   SavingsAccount(String name, int accNumber) {
      super(name, accNumber);
   }

   void computeInterest() {
      double interest = balance * interestRate;
      balance += interest;
      System.out.println("Interest added: " + interest + "\nUpdated Balance: " + balance);
   }
```

```
    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + "\nUpdated Balance: " + balance);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

class CurrentAccount extends Account {
    double minimumBalance = 500.0;
    double serviceCharge = 50.0;

    CurrentAccount(String name, int accNumber) {
        super(name, accNumber);
    }

    void checkMinimumBalance() {
        if (balance < minimumBalance) {
            System.out.println("Balance is below minimum. Service charge imposed.");
            balance -= serviceCharge;
            System.out.println("Service Charge: " + serviceCharge + "\nUpdated Balance: " + balance);
        }
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + "\nUpdated Balance: " + balance);
            checkMinimumBalance();
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

public class Bank{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter customer name: ");
        String name = sc.nextLine();
        System.out.print("Enter account number: ");
        int accNumber = sc.nextInt();
        sc.nextLine();
```

```
System.out.print("Enter account type (Savings/Current): ");
String accType = sc.nextLine();

Account account;
if (accType.equalsIgnoreCase("Savings")) {
    account = new SavingsAccount(name, accNumber);
} else if (accType.equalsIgnoreCase("Current")) {
    account = new CurrentAccount(name, accNumber);
} else {
    System.out.println("Invalid account type.");
    sc.close();
    return;
}

while (true) {
    System.out.println("\nMenu:");
    System.out.println("1. Deposit");
    System.out.println("2. Withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. Compute Interest");
    System.out.println("5. Exit");
    System.out.print("Enter your choice: ");
    int choice = sc.nextInt();

    switch (choice) {
        case 1:
            System.out.print("Enter the deposit amount: ");
            double depositAmount = sc.nextDouble();
            account.deposit(depositAmount);
            break;
        case 2:
            System.out.print("Enter the withdrawal amount: ");
            double withdrawAmount = sc.nextDouble();
            account.withdraw(withdrawAmount);
            break;
        case 3:
            account.displayBalance();
            break;
        case 4:
            account.computeInterest();
            break;
        case 5:
            System.out.println("Exiting.");
            sc.close();
            return;
        default:
```

```
                System.out.println("Invalid choice.");
            }
        }
    }
}
```

Output:

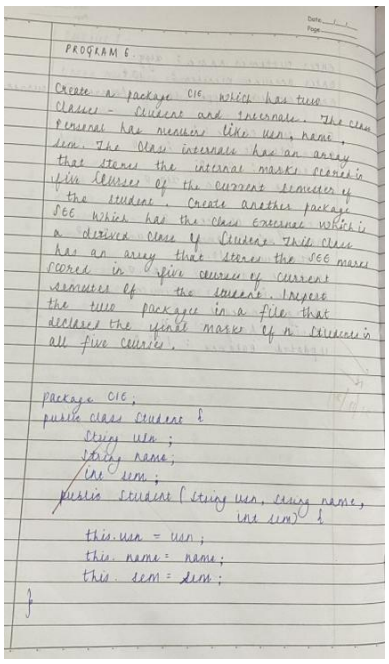
```
Command Prompt
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd\

C:\>cd 295

C:\295>set path="C:\Program Files\Java\jdk-23\bin"

C:\295>javac Bank.java

C:\295>java Bank
Enter customer name: sam
Enter account number: 123
Enter account type (Savings/Current): savings

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 1
Enter the deposit amount: 500
Deposited: 500.0
Updated Balance: 500.0

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 2
Enter the withdrawal amount: 10
Withdrawn: 10.0
Updated Balance: 490.0

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 5
Exiting.
```

```
C:\295>javac Bank.java

C:\295>java Bank
Enter customer name: xyz
Enter account number: 234
Enter account type (Savings/Current): current

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 1
Enter the deposit amount: 210
Deposited: 210.0
Updated Balance: 210.0

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 2
Enter the withdrawal amount: 10
Withdrawn: 10.0
Updated Balance: 200.0
Balance is below minimum. Service charge imposed.
Service Charge: 50.0
Updated Balance: 150.0

Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 5
Exiting.

C:\295>
```

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 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.

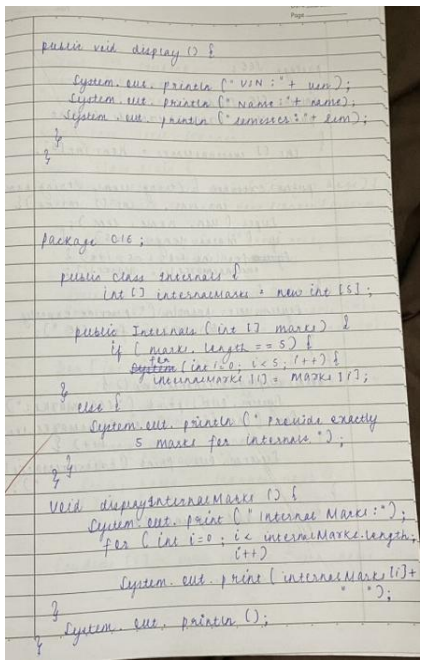
Algorithm:



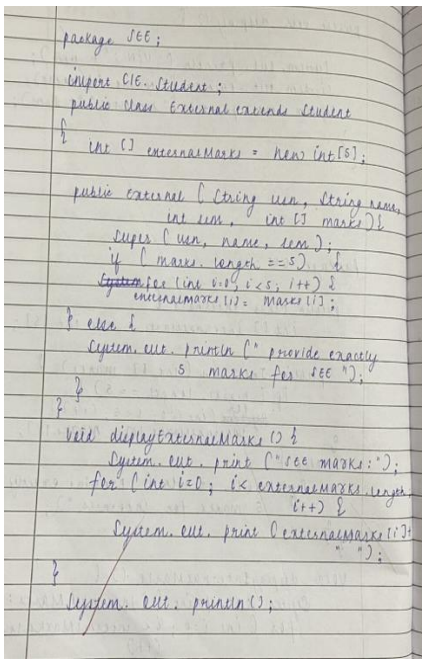
PROGRAM 6

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 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 current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

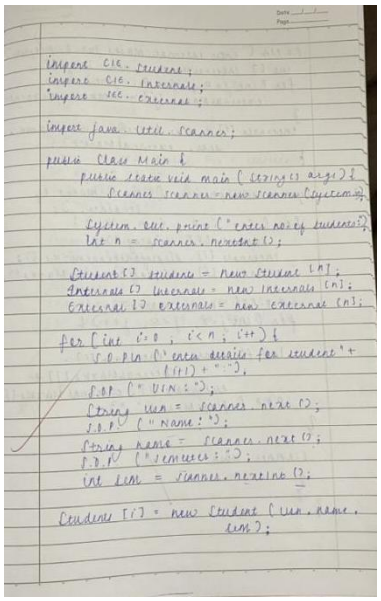
```
package CIE;
public class Student {
    String usn;
    String name;
    int sem;
    public Student (String usn, String name,
                    int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }
```



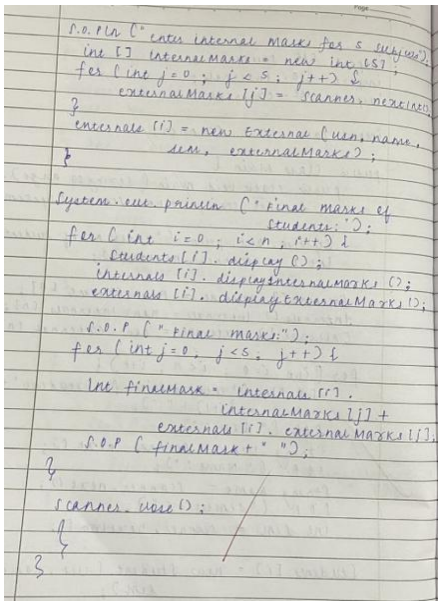
```
public void display () {
    System.out.println ("USN :" + usn);
    System.out.println ("NAME :" + name);
    System.out.println ("semester :" + sem);
}
}

package CIE;

public class Internals {
    int [] internalMarks = new int [5];

    public Internals (int [] marks) {
        if (marks.length == 5) {
            for (int i=0; i<5; i++)
                internalMarks [i] = marks [i];
        } else {
            System.out.println ("provide exactly
                5 marks for internals");
        }
    }

    void displayInternalMarks () {
        System.out.print ("Internal Marks :");
        for (int i=0; i< internalMarks.length;
                i++)
            System.out.print (internalMarks [i] +
                " ");
    }
    System.out.println ();
}
```



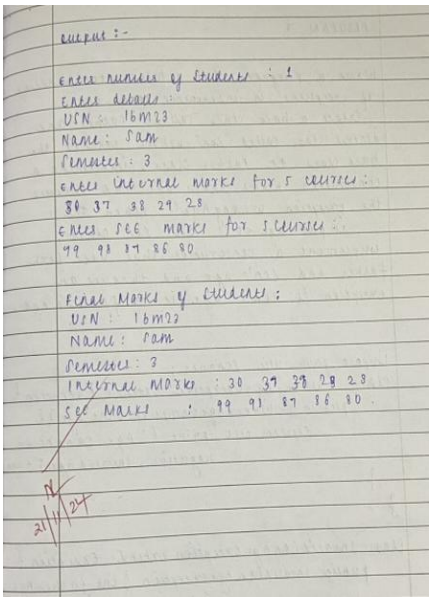
```
package SEE;

import CIE.Student;
public class External extends Student
{
    int [] externalMarks = new int [5];

    public External (String usn, String name,
                     int sem, int [] marks) {
        super (usn, name, sem);
        if (marks.length == 5) {
            for (int i=0; i<5; i++) {
                externalMarks [i] = marks [i];
        } else {
            System.out.println ("provide exactly
                5 marks for SEE");
        }
    }

    void displayExternalMarks () {
        System.out.print ("SEE marks :");
        for (int i=0; i< externalMarks.length;
                i++) {
            System.out.print (externalMarks [i] +
                " ");
        }
    System.out.println ();
```



```
import CIE.Student;
import CIE.Internals;
import SEE.External;

import java.util.Scanner;

public class Main {
    public static void main (String[] args) {
        Scanner scanner = new Scanner (System.in);

        System.out.print ("enter no of students");
        int n = scanner.nextInt ();

        Student [] students = new Student [n];
        Internals [] internals = new Internals [n];
        External [] externals = new External [n];

        for (int i=0 ; i<n ; i++) {
            S.O.P.ln ("enter details for student" +
                (i+1) + ":");
            S.O.P ("USN :");
            String usn = scanner.next ();
            S.O.P ("Name :");
            String name = scanner.next ();
            S.O.P ("Semester :");
            int sem = scanner.nextInt ();

            Student [i] = new Student (usn, name,
                sem);
```



```
S.O.P.ln ("enter internal marks for 5 subjects");
int [] internalMarks = new int [5];
for (int j=0 ; j<5 ; j++) {
    externalMarks [j] = scanner.nextInt();
}
internals [i] = new External (usn, name,
                sem, externalMarks);
}

System.out.println ("Final marks of
                students:");
for (int i=0 ; i<n ; i++) {
    students [i].display ();
    internals [i].displayInternalMarks ();
    externals [i].displayExternalMarks ();

    S.O.P ("Final marks:");
    for (int j=0 ; j<5 ; j++) {

        int finalMark = internals [i].
                internalMarks [j] +
                externals [i].externalMarks [j];
        S.O.P (finalMark + " ");
    }

    scanner.close ();
}
}
```



```
output :-

enter number of students : 1
enter details :
USN : 1BM23
Name : Sam
Semester : 3
enter internal marks for 5 courses :
30 37 38 29 28
enter SEE marks for 5 courses :
99 93 87 86 90

Final Marks of students :
USN : 1BM23
Name : Sam
Semester : 3
Internal Marks : 30 37 38 29 28
SEE Marks : 99 93 87 86 90
```

Code: CIE

Internals.java

```
package CIE;

public class Student {
    String usn;
    String name;
    int sem;

    public Student(String usn, String name, int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }

    public void display() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}

package CIE;

public class Internals {
    int[] internalMarks = new int[5];

    public Internals(int[] marks) {
        if (marks.length == 5) {
            System.arraycopy(marks, 0, internalMarks, 0, 5);
        } else {
            System.out.println("Please provide exactly 5 marks for internals.");
        }
    }

    void displayInternalMarks() {
        System.out.print("Internal Marks: ");
        for (int i = 0; i < internalMarks.length; i++) {
            System.out.print(internalMarks[i] + " ");
        }
        System.out.println();
    }
}
```

SEE:

Student.java

```
package SEE;

import CIE.Student;

public class External extends Student {
   int[] externalMarks = new int[5];

   public External(String usn, String name, int sem, int[] marks) {
      super(usn, name, sem);
      if (marks.length == 5) {
         System.arraycopy(marks, 0, externalMarks, 0, 5);
      } else {
         System.out.println("Please provide exactly 5 marks for SEE.");
      }
   }

   void displayExternalMarks() {
      System.out.print("SEE Marks: ");
      for (int i = 0; i < externalMarks.length; i++) {
         System.out.print(externalMarks[i] + " ");
      }
      System.out.println();
   }
}
```

Main.java

```
import CIE.Student;
import CIE.Internals;
import SEE.External;

import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of students: ");
      int n = scanner.nextInt();
```

```
Student[] students = new Student[n];
Internals[] internals = new Internals[n];
External[] externals = new External[n];

for (int i = 0; i < n; i++) {
    System.out.println("Enter details for student " + (i + 1) + ": ");
    System.out.print("USN: ");
    String usn = scanner.next();
    System.out.print("Name: ");
    String name = scanner.next();
    System.out.print("Semester: ");
    int sem = scanner.nextInt();

    students[i] = new Student(usn, name, sem);

    System.out.println("Enter internal marks for 5 courses:");
    int[] internalMarks = new int[5];
    for (int j = 0; j < 5; j++) {
        internalMarks[j] = scanner.nextInt();
    }
    internals[i] = new Internals(internalMarks);

    System.out.println("Enter SEE marks for 5 courses:");
    int[] externalMarks = new int[5];
    for (int j = 0; j < 5; j++) {
        externalMarks[j] = scanner.nextInt();
    }
    externals[i] = new External(usn, name, sem, externalMarks);
}

System.out.println("\nFinal Marks of Students:");
for (int i = 0; i < n; i++) {
    students[i].display();
    internals[i].displayInternalMarks();  // Using the standard for loop to display internal marks
    externals[i].displayExternalMarks();  // Using the standard for loop to display external marks

    System.out.print("Final Marks: ");
    for (int j = 0; j < 5; j++) {
        int finalMark = internals[i].internalMarks[j] + externals[i].externalMarks[j];
        System.out.print(finalMark + " ");
    }
    System.out.println("\n");
}

scanner.close();
```

```
    }
}
```

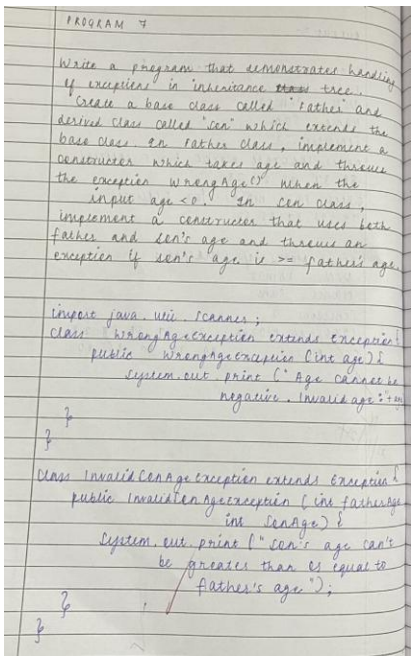
Output:

```
C:\Windows\System32\cmd.e

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\295>javac marks.java

C:\295>java marks
Enter the number of students: 2
Enter details for student 1:
USN: 1bm23
Name: sam
Semester: 3
Enter internal marks for 5 courses:
30 39 29 28 30
Enter SEE marks for 5 courses:
86 87 89 90 99
Enter details for student 2:
USN: 1bm24
Name: xyz
Semester: 2
Enter internal marks for 5 courses:
39 34 35 36 34
Enter SEE marks for 5 courses:
99 98 87 86 80

Final Marks of Students:
USN: 1bm23
Name: sam
Semester: 3
Internal Marks: 30 39 29 28 30
SEE Marks: 86 87 89 90 99
Final Marks: 116 126 118 118 129

USN: 1bm24
Name: xyz
Semester: 2
Internal Marks: 39 34 35 36 34
SEE Marks: 99 98 87 86 80
Final Marks: 138 132 122 122 114


C:\295>
```

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 the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that uses both father and son's age and throws an exception if son's age is >=father's age.

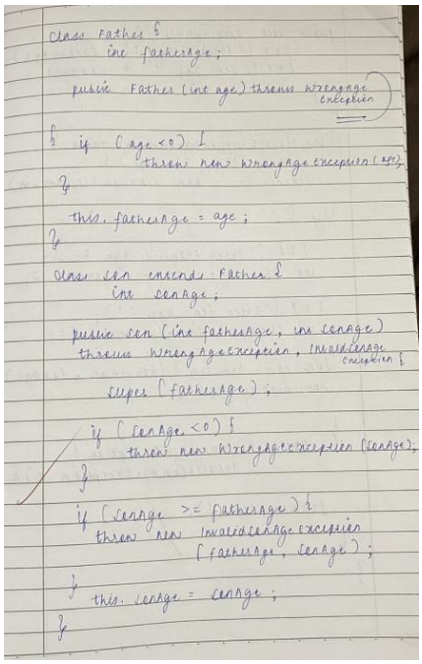
Algorithm:



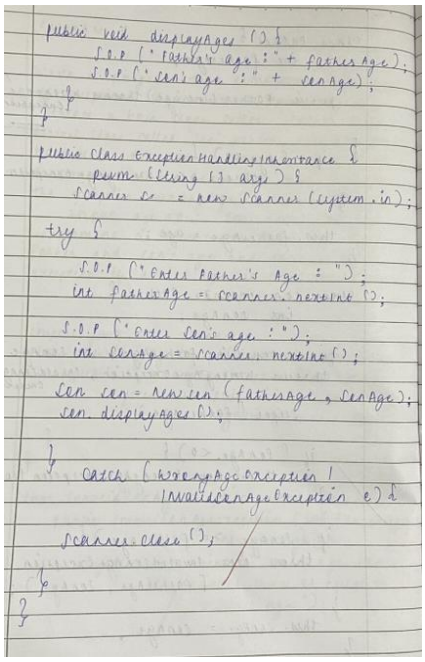
PROGRAM 7

Write a program that demonstrates handling of exceptions in inheritance ~~tree~~ 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 age and throws the exception WrongAge() when the input age < 0. In Son class, implement a constructor that uses both father and son's age and throws an exception if son's age is >= father's age.

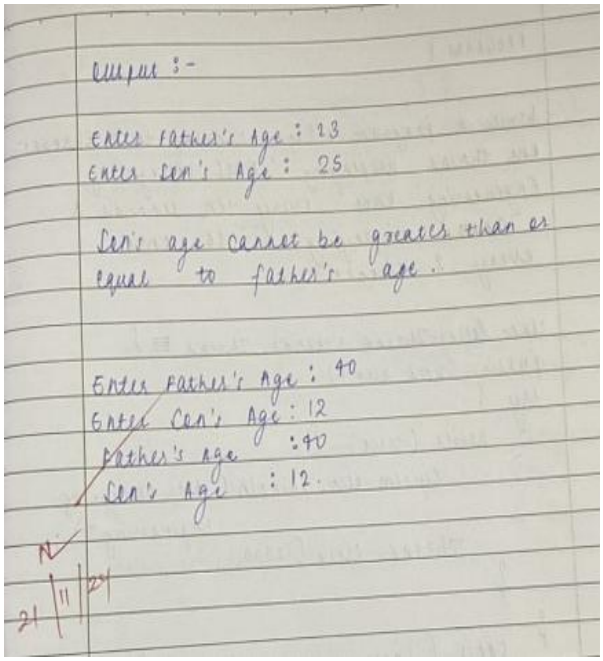
```
import java.util.Scanner;
class WrongAgeException extends Exception {
    public WrongAgeException (int age) {
        System.out.print ("Age cannot be
                negative. Invalid age :+ age
    }
}

class InvalidSonAgeException extends Exception {
    public InvalidSonAgeException (int fatherAge,
                int sonAge) {
        System.out.print ("Son's age can't
                be greater than or equal to
                father's age.");
    }
}
```



```
class Father {
    int fatherAge;

    public Father (int age) throws WrongAge
                Exception

    {  if (age < 0) {
            throw new WrongAgeException (age);
    }

    this.fatherAge = age;
}

class Son extends Father {
    int sonAge;

    public Son (int fatherAge, int sonAge)
    throws WrongAgeException, InvalidSonAge
                Exception {

        super (fatherAge);

        if (sonAge < 0) {
            throw new WrongAgeException (sonAge);
        }

        if (sonAge >= fatherAge) {
            throw new InvalidSonAgeException
                (fatherAge, sonAge);
        }
        this.sonAge = sonAge;
    }
```



```
public void displayAge () {
    S.O.P ("Father's age :" + fatherAge);
    S.O.P ("Son's age :" + sonAge);
}
}

public class ExceptionHandlingInheritance {
    public (String [] args) {
        Scanner sc = new Scanner (System.in);

    try {
        S.O.P ("Enter Father's Age : ");
        int fatherAge = scanner.nextInt ();

        S.O.P ("Enter Son's age :");
        int sonAge = scanner.nextInt ();

        Son son = new Son (fatherAge, sonAge);
        son.displayAge ();

    }  catch (WrongAgeException |
                InvalidSonAgeException e) {

        Scanner.close ();
    }
}
```



Output :-

Enter Father's Age : 23
Enter Son's Age : 25

Son's age cannot be greater than or equal to father's age.

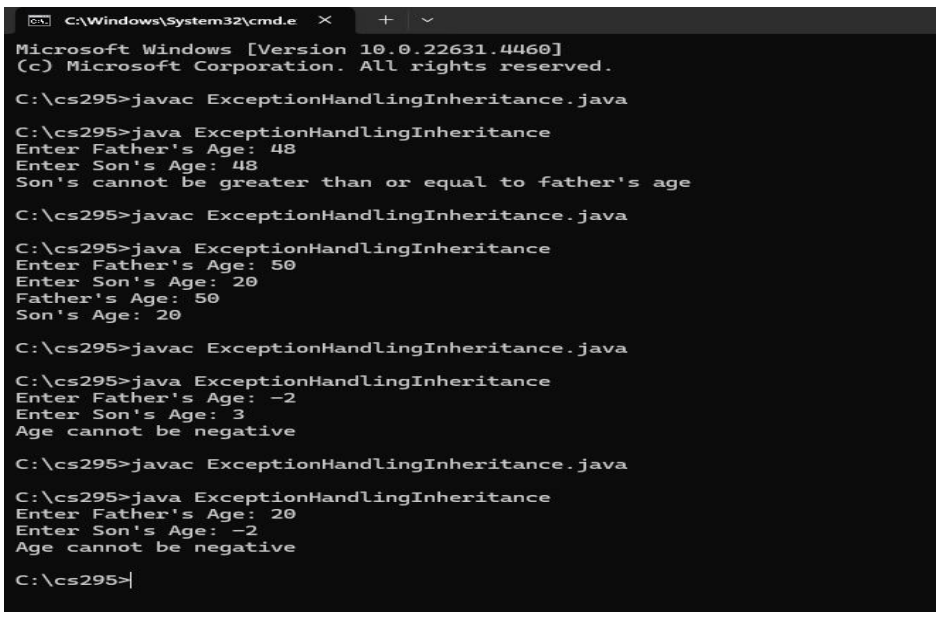
Enter Father's Age : 40
Enter Son's Age : 12
Father's Age : 40
Son's Age : 12.

Code:

```
import java.util.Scanner;
class WrongAgeException extends Exception {
public WrongAgeException(int age) {
System.out.println("Age cannot be negative. Invalid age: " + age);
    }
}
class InvalidSonAgeException extends Exception {
public InvalidSonAgeException(int fatherAge, int sonAge) {
System.out.println("Son's cannot be greater than or equal to father's age");
    }
}
class Father {
    int fatherAge;

public Father(int age) throws WrongAgeException {
   if (age < 0) {
   throw new WrongAgeException(age);
   }
    this.fatherAge = age;
    }
}
class Son extends Father {
    int sonAge;

 public Son(int fatherAge, int sonAge) throws WrongAgeException, InvalidSonAgeException {
   super(fatherAge);
   if (sonAge < 0) {
   throw new WrongAgeException(sonAge);
   }
   if (sonAge >= fatherAge) {
   throw new InvalidSonAgeException(fatherAge, sonAge);
   }
    this.sonAge = sonAge;
    }

public void displayAges() {
System.out.println("Father's Age: " + fatherAge);
System.out.println("Son's Age: " + sonAge);
    }
}
public class ExceptionHandlingInheritance {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
```

```
        try {
            System.out.print("Enter Father's Age: ");
            int fatherAge = scanner.nextInt();

            System.out.print("Enter Son's Age: ");
            int sonAge = scanner.nextInt();

            Son son = new Son(fatherAge, sonAge);
            son.displayAges();

        } catch (WrongAgeException | InvalidSonAgeException e) {

        } finally {
            scanner.close();
        }
    }
}
```

Output:



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:

```
Class DisplayThread extends Thread {
  String message;
  int interval;

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

  public void run () {
    try {
      while (true) {
```

```
      S.O.println (message);
      Thread.sleep ( interval * 1000); } }
      catch ( InterruptedException e)
      { S.O.println ("Thread interrupted :" + message); }}

      public static void main (String [] args) {

      DisplayThread thread1 = new DisplayThread
                                ("BMSCE", 10);
      DisplayThread Thread2 = new DisplayThread
                                ("CSE", 2);

      }
    }
```

28/11/24

```
output :-
BMS college of engineering
CSE
CSE
CSE
CSE
CSE
BMS college of engineering
CSE
CSE
CSE
CSE
CSE
BMS college of engineering
:
:
```

Code:

```
class DisplayThread extends Thread {
       private String message;
       private int interval;
Public DisplayThread(String message, int interval)
{
     this.message = message;
     this.interval = interval; }
 public void run()
 {
try {
while (true) {
     System.out.println(message); Thread.sleep(interval * 1000);  }
}
catch (InterruptedException e)
{
           System.out.println("Thread interrupted: " + message); } }
public static void main(String[] args) {

  DisplayThread thread1 = new DisplayThread("BMS College of Engineering", 10);
  DisplayThread thread2 = new DisplayThread("CSE", 2);
 }
}
```

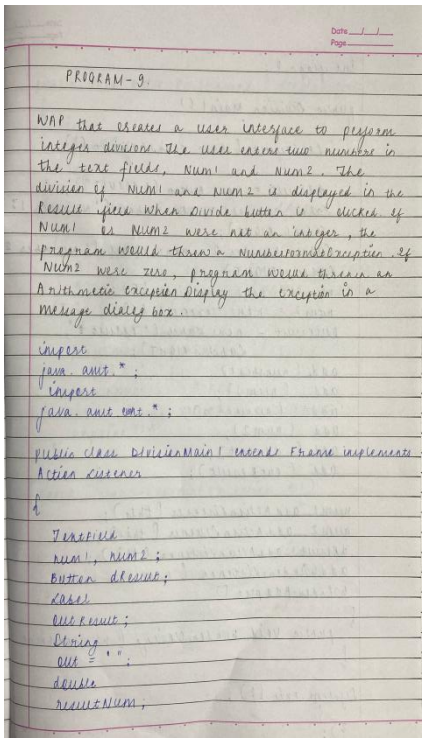
Output:

```
E:\1BM23CS295>java DisplayThread
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
```

Program 9

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 Arithmetic Exception Display the exception in a message dialog box.

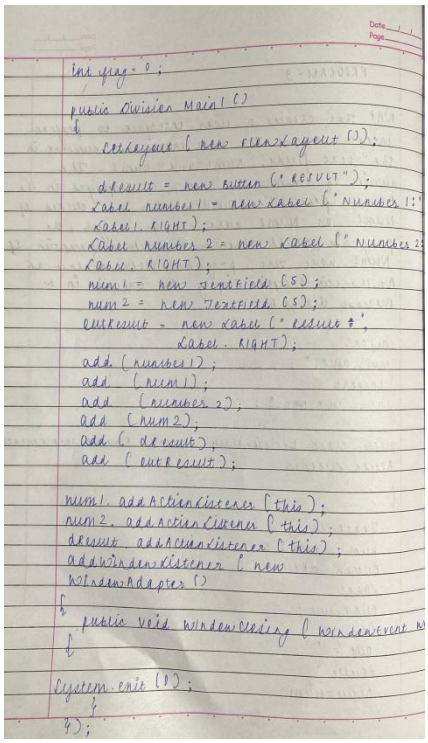
Algorithm:



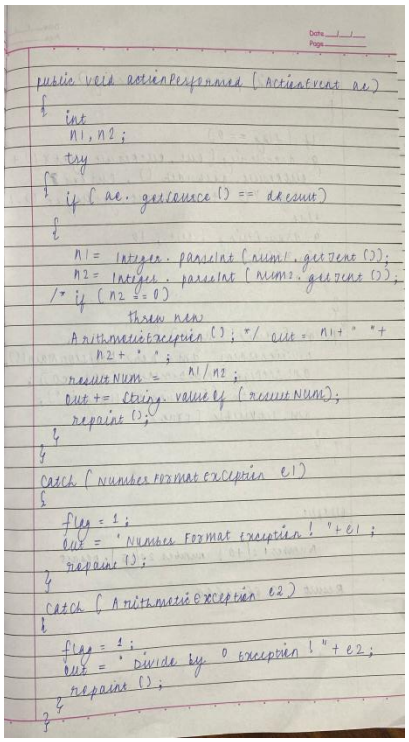
PROGRAM-9

WAP 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 Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were zero, program would throw an Arithmetic Exception Display the exception in a message dialog box.

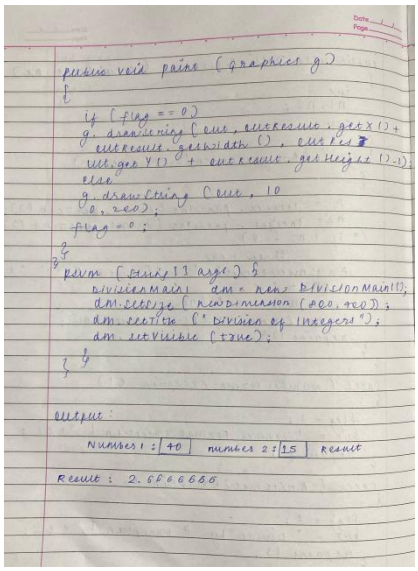
```
import
java.awt.*;
import
java.awt.event.*;

public class DivisionMain1 extends Frame implements
ActionListener
{
    TextField
    num1, num2;
    Button dresult;
    Label
    outresult;
    String
    out = "";
    double
    resultNum;
```



```
int flag = 0;

public DivisionMain1()
{
    setLayout(new FlowLayout());

    dresult = new Button("RESULT");
    Label number1 = new Label("Number 1:
    Label.RIGHT);
    Label number2 = new Label("Number 2:
    Label.RIGHT);
    num1 = new TextField(5);
    num2 = new TextField(5);
    outresult = new Label("Result :",
                Label.RIGHT);
    add(number1);
    add(num1);
    add(number2);
    add(num2);
    add(dresult);
    add(outresult);

    num1.addActionListener(this);
    num2.addActionListener(this);
    dresult.addActionListener(this);
    addWindowListener(new
    WindowAdapter()
    {
        public void windowClosing(WindowEvent we
    }

    System.exit(0);
    }
});
```



```
public void actionPerformed(ActionEvent ae)
{
    int
    n1, n2;
    try
    {
    if (ae.getSource() == dresult)
    {
        n1 = Integer.parseInt(num1.getText());
        n2 = Integer.parseInt(num2.getText());
/* if (n2 == 0)
        throw new
        ArithmeticException(); */ out = n1+" "+
        n2+" ";
        resultNum = n1/n2;
        out += String.valueOf(resultNum);
        repaint();
    }
    }
    catch (NumberFormatException e1)
    {
        flag = 1;
        out = "Number Format exception! "+e1;
        repaint();
    }
    catch (ArithmeticException e2)
    {
        flag = 1;
        out = "Divide by 0 exception! "+e2;
        repaint();
    }
```



```
public void paint(Graphics g)
{
    if (flag == 0)
    g.drawString(out, outresult.getX()+
    outresult.getWidth(), outresult
    .getY() + outresult.getHeight()-1);
    else
    g.drawString(out, 10
    0, 200);
    flag = 0;
}
}
public static void main(String[] args) {
    DivisionMain1 am = new DivisionMain1();
    am.setSize(new Dimension(400, 400));
    am.setTitle("Division of Integers");
    am.setVisible(true);
}
}

output:

Number1 : 40  number 2 : 15  Result

Result : 2.6666666
```

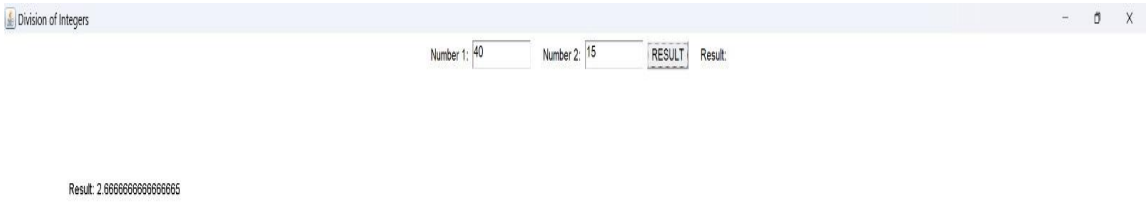
Code:

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

public class DivisionMain1 extends Frame implements ActionListener {
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = "";
    double resultNum;
    int flag = 0;

    public DivisionMain1() {
        setLayout(new FlowLayout());

        dResult = new Button("RESULT");
        Label number1 = new Label("Number 1:", Label.RIGHT);
        Label number2 = new Label("Number 2:", Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
        outResult = new Label("Result:", Label.RIGHT);

        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);

        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
```

```
    setSize(300, 200);
    setVisible(true);
}

public void actionPerformed(ActionEvent ae) {
    int n1, n2;
    try {
        if (ae.getSource() == dResult) {
            n1 = Integer.parseInt(num1.getText());
            n2 = Integer.parseInt(num2.getText());

            if (n2 == 0) {
                throw new ArithmeticException("Cannot divide by zero");
            }

            resultNum = (double) n1 / n2;
            out = "Result: " + resultNum;
            flag = 0;
        }
    } catch (NumberFormatException e1) {
        flag = 1;
        out = "Number Format Exception: " + e1.getMessage();
    } catch (ArithmeticException e2) {
        flag = 1;
        out = "Divide by Zero Exception: " + e2.getMessage();
    }
    repaint();
}

public void paint(Graphics g) {
    if (flag == 0) {
        g.drawString(out, 100, 150);
    } else {
        g.drawString(out, 100, 200);
        flag = 0;
    }
}

public static void main(String[] args) {
    DivisionMain1 dm = new DivisionMain1();
```

```
        dm.setSize(new Dimension(800, 400));
        dm.setTitle("Division of Integers");
        dm.setVisible(true);
    }
}
```

Output:

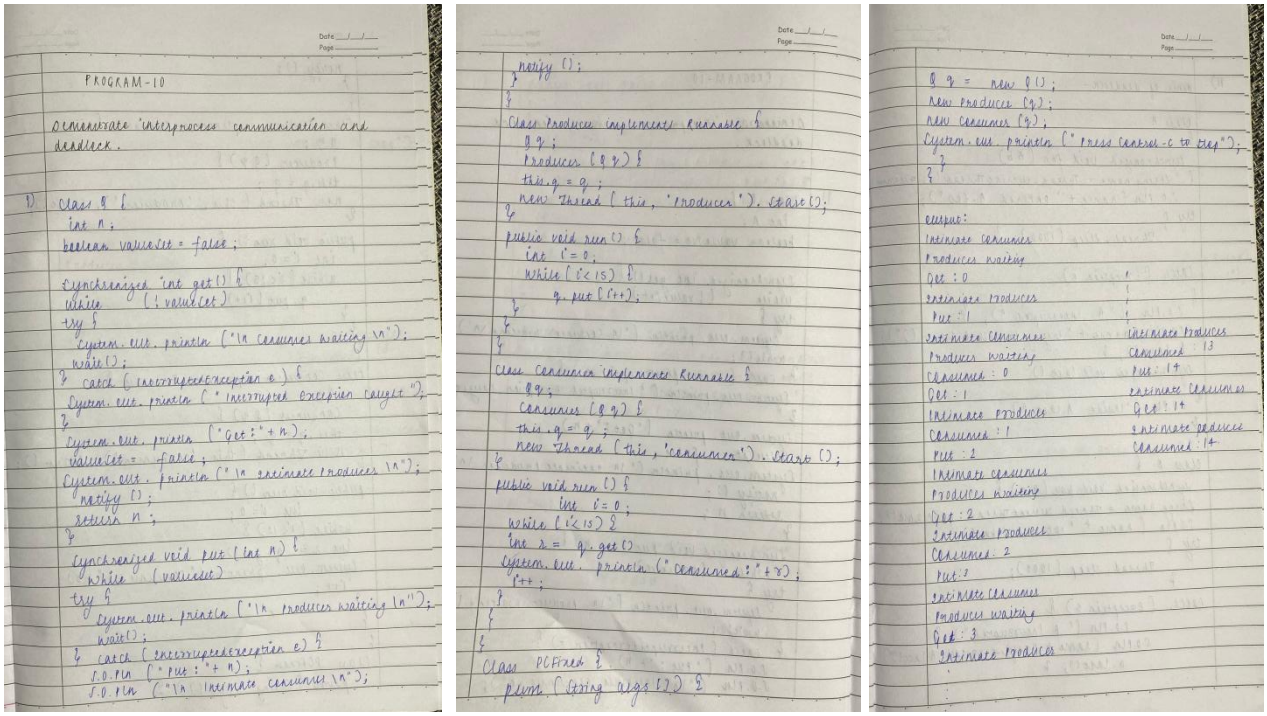


Program 10

Demonstrate Inter process Communication and deadlock

Algorithm:

i)



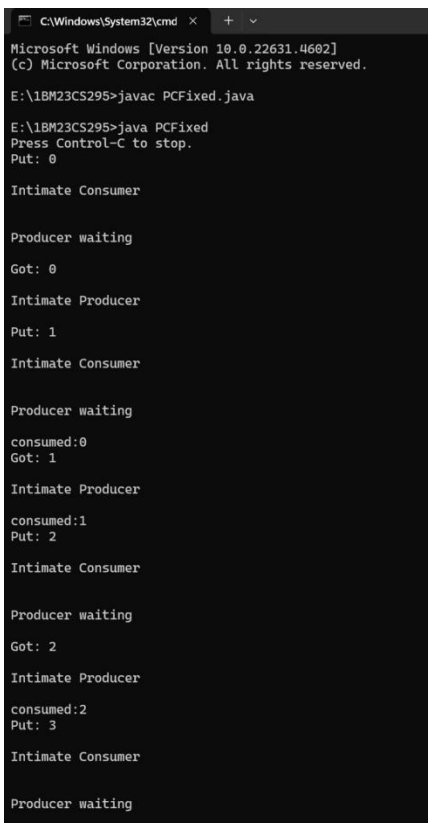
Code:

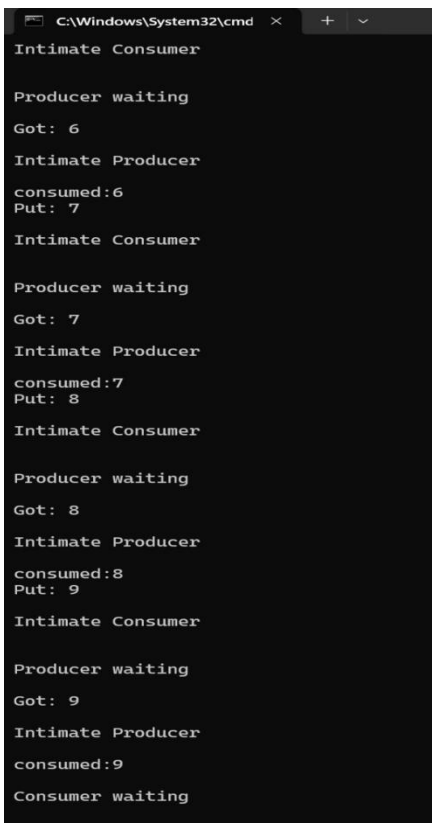
```
class Q {
    int n;
    boolean valueSet = false;

    synchronized int get() {
        while (!valueSet)
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\nIntimate Producer\n");
        notify();
        return n;
```

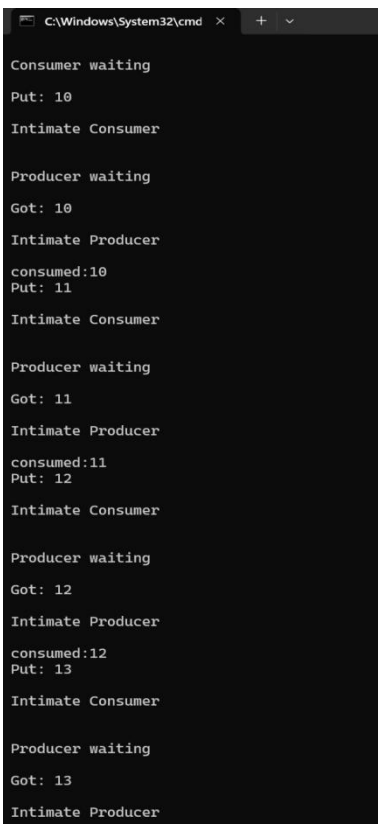
```
    }

    synchronized void put(int n) {
        while (valueSet)
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;

    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();
```

```
            System.out.println("consumed: " + r);
            i++;
        }
    }
}

class PCFixed {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

Output:

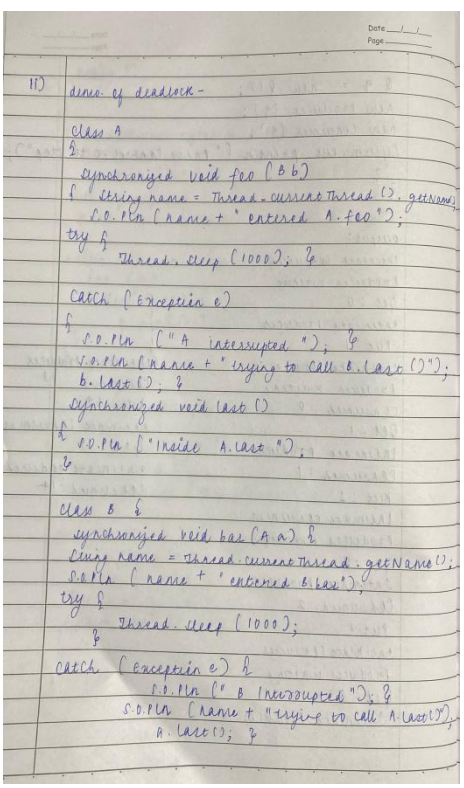




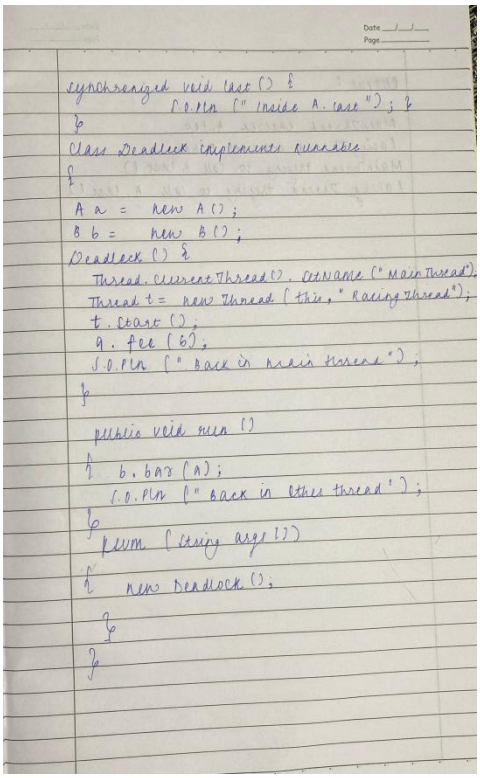


```
C:\Windows\System32\cmd

Producer waiting

Got: 12

Intimate Producer

consumed:12
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

consumed:13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

consumed:14

E:\1BM23CS295>
```

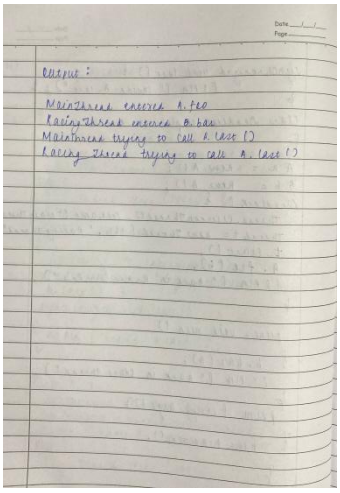
ii) Algorithm:



```
ii) demo. of deadlock -

class A
{
  synchronized void foo (B b)
  { String name = Thread.currentThread().getName();
    S.O.Println (name + " entered A.foo");
    try {
        Thread.sleep (1000); }

    catch (Exception e)
    {
      S.O.Println ("A interrupted "); }
      S.O.Println (name + " trying to call B.last()");
      b.last(); }
    synchronized void last()
    { S.O.Println ("Inside A.last");
    }

class B {
  synchronized void bar (A a) {
  String name = Thread.currentThread.getName();
  S.O.Println (name + " entered B.bar");
  try {
      Thread.sleep (1000);
    }
  catch (Exception e) {
        S.O.Println (" B Interrupted "); }
        S.O.Println (name + " trying to call A.last()");
          A.last(); }
```



```
synchronized void last () {
        S.O.Println (" Inside A.last "); }
}
class Deadlock implements runnable
{
A a = new A();
B b = new B();
Deadlock () {
  Thread.currentThread().setName ("MainThread");
  Thread t = new Thread (this, "RacingThread");
  t.start ();
  a.foo (b);
  S.O.Println (" back in main thread ");
}

public void run ()
{ b.bar (a);
  S.O.Println (" back in other thread ");
}
psvm (String args[])
{ new Deadlock();
  }
}
```



```
output :

MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call A.last()
Racing thread trying to call A.last()
```

Code:

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

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

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

    synchronized void last() {
        System.out.println("Inside B.last");
    }
}
```

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

   Deadlock() {
      Thread.currentThread().setName("MainThread");
      Thread t = new Thread(this, "RacingThread");
      t.start();

      a.foo(b); // get lock on a in this thread.
      System.out.println("Back in main thread");
   }

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

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

Output: