**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

***Submitted by***

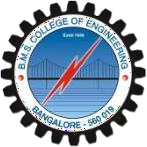
SHREYAS.R.ACHAR (1BM23CS320)

***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 **SHREYAS.R.ACHAR(1BM23CS320),** who is bonafide 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 Dr. Kavitha Sooda**

Associate Professor, Professor and Head,

Department of CSE, Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 24-10-24 | Quadratic equation and display all solutions | 1 - 5 |
| 2 | 1 -11-24 | Display student details and SGPA calculation | 6 - 11 |
| 3 | 7-11-24 | Implementation of array of objects and tostring() methods | 12-15 |
| 4 | 14-11-24 | Creating abstract class shape and abstract method named printarea() | 16-19 |
| 5 | 21-11-24 | 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. | 17-26 |
| 6 | 28-11-24 | Demonstration of use of pakages using concepts of student and internals | 26-33 |
| 7 | 5-12-24 | a program that demonstrates handling of exceptions in inheritance tree. | 34-37 |
| 8 | 19-12-24 | 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 | 38-41 |
| 9 | 26-12-24 | Write a program that creates a user interface to perform integer divisions | 42-50 |
| 10 | 26-12-24 | Demonstrate Interprocess communication and deadlock | 50-53 |

WEEK 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.

Source Code:

import java.util.Scanner;

class Quadratic {

    float d;

    Scanner sc = new Scanner(System.in);

    void solver()

    {

        System.out.println("enter the values of a,b, and c");

        int a = sc.nextInt();

        int b = sc.nextInt();

        int c = sc.nextInt();

        if (a == 0) {

            System.out.println("invalid equation");

        }

        else{

            d= b\*b - 4\*a\*c;

            System.out.println(d);

            System.out.println("the solutions are");

            if(d>0){

                System.out.println("roots are unique ");

                double r1 = (-b+Math.sqrt(d))/(2\*a);

                double r2 = (-b-Math.sqrt(d))/(2\*a);

                System.out.println(r1 +" " + r2);

            }

            if(d==0){

                System.out.println("roots are equal ");

                double r = -b/(2\*a);

                System.out.println(r);

            }

            if(d<0){

                System.out.println("There are no real roots" );

            }

        }

    }

}

public class Main {

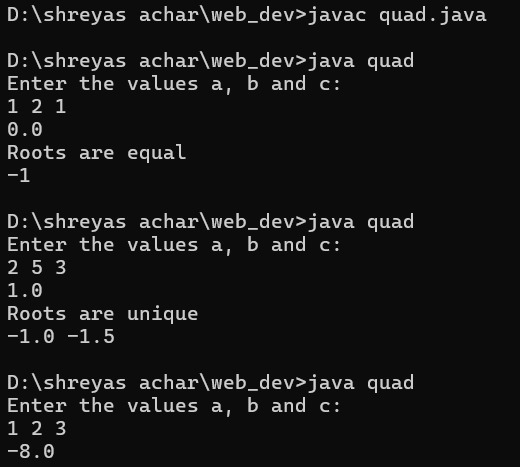
    public static void main(String[] args) {

        Quadratic q1 = new Quadratic();

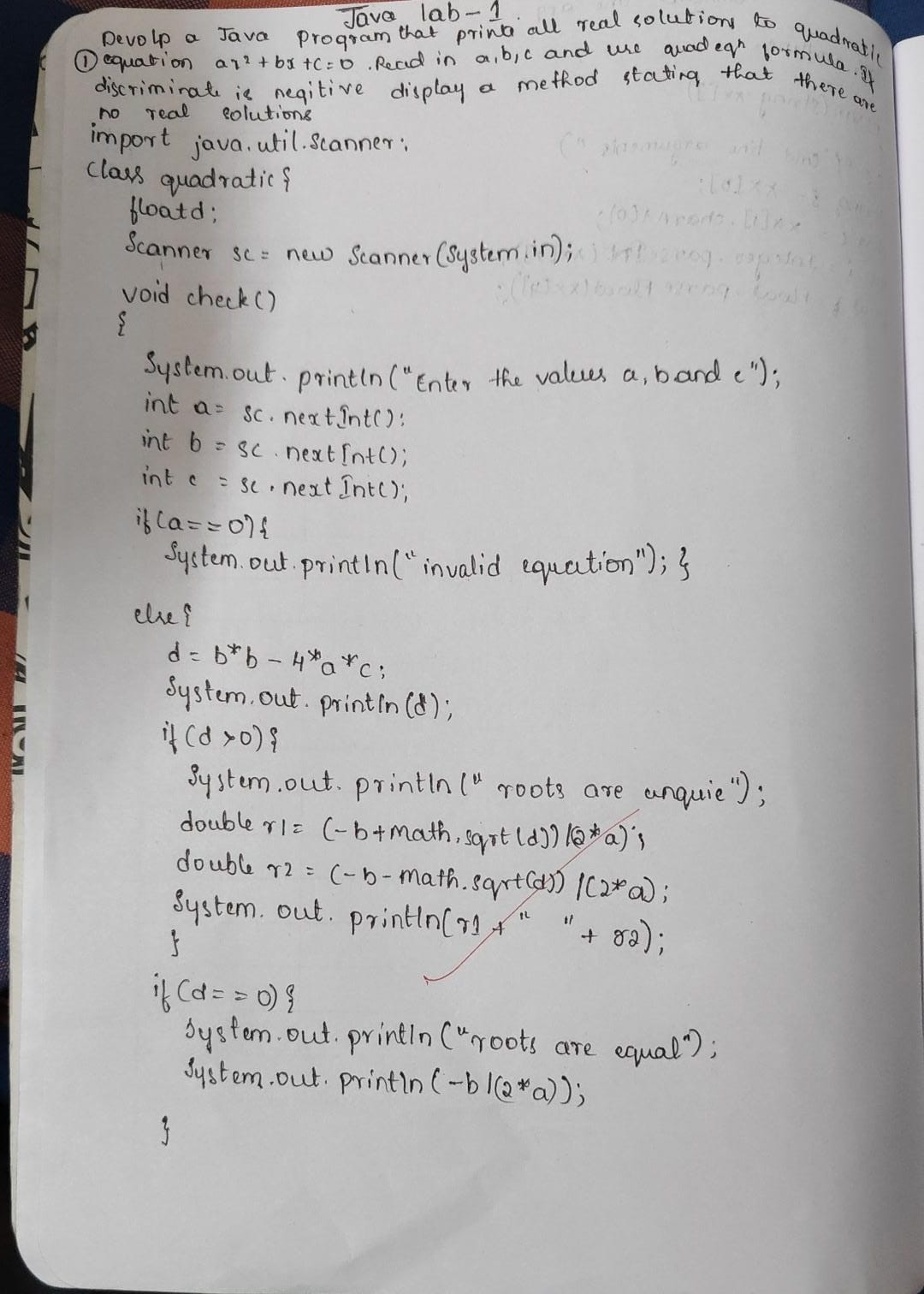
         q1.solver();

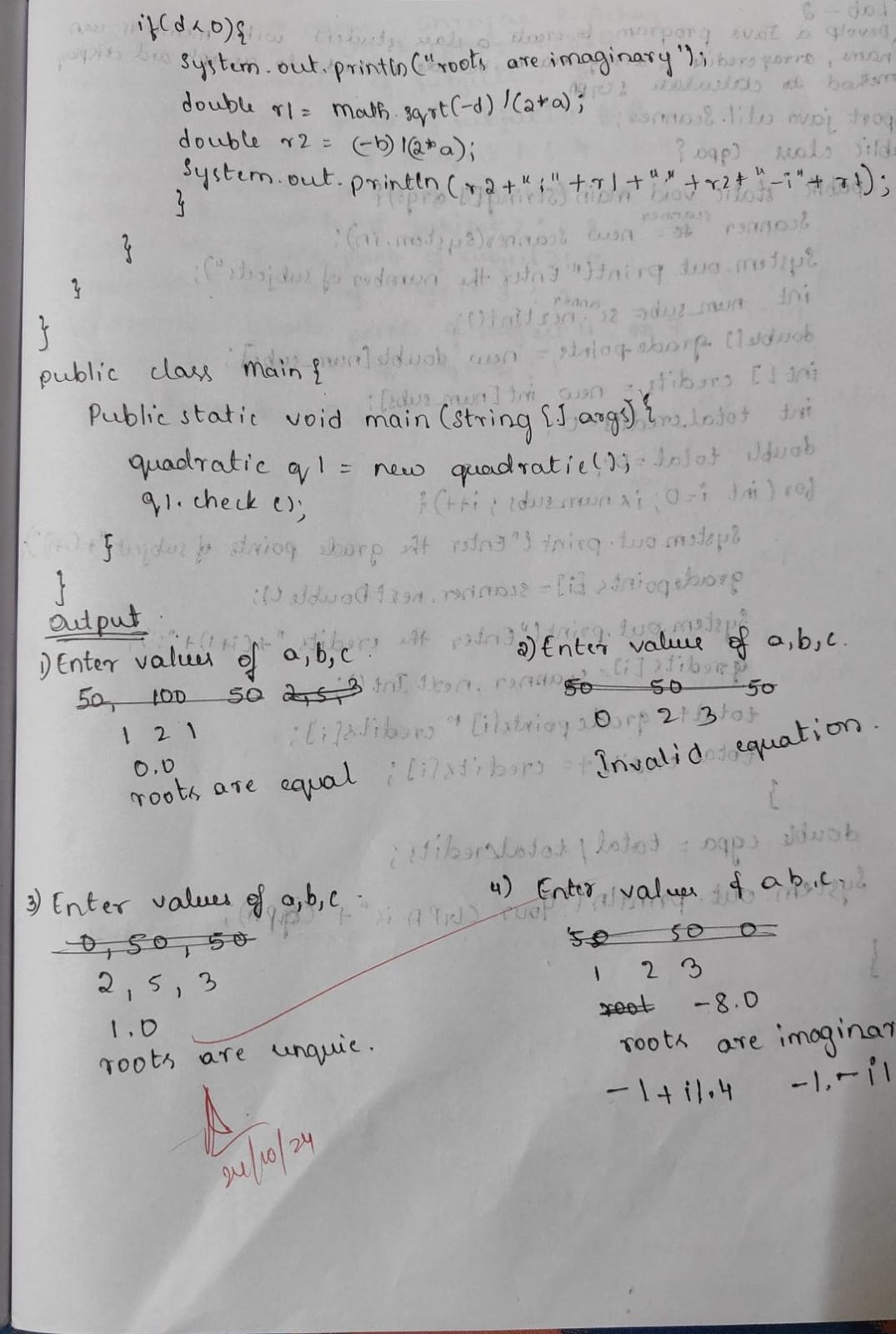
  }

}



Written Code & Output:





WEEK 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 a method to calculate SGPA of a student.

Source Code:

import java.util.Scanner;

class Student {

    String usn;

    String name;

    int numSubjects;

    int[] credits;

    int[] marks;

    double sgpa;

    public void acceptDetails() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter USN: ");

        usn = sc.nextLine();

        System.out.print("Enter Name: ");

        name = sc.nextLine();

        System.out.print("Enter the number of subjects: ");

        numSubjects = sc.nextInt();

        credits = new int[numSubjects];

        marks = new int[numSubjects];

        for (int i = 0; i < numSubjects; i++) {

            System.out.print("Enter credits for subject " + (i + 1) + ": ");

            credits[i] = sc.nextInt();

            System.out.print("Enter marks for subject " + (i + 1) + ": ");

            marks[i] = sc.nextInt();

        }

    }

    public void displayDetails() {

        System.out.println("\nStudent Details:");

        System.out.println("USN: " + usn);

        System.out.println("Name: " + name);

        System.out.println("Subjects and Marks:");

        for (int i = 0; i < numSubjects; i++) {

            System.out.println("Subject " + (i + 1) + ": Marks = " + marks[i] + ", Credits = " + credits[i]);

        }

    }

    public void calculateSGPA() {

        int totalCredits = 0;

        int totalGradePoints = 0;

        for (int i = 0; i < numSubjects; i++) {

            int grade = calculateGrade(marks[i]);

            totalGradePoints += grade \* credits[i];

            totalCredits += credits[i];

        }

        sgpa = (double) totalGradePoints / totalCredits;

    }

    private int calculateGrade(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 if (marks >= 40) {

            return 5;

        } else {

            return 0;

        }

    }

    public void displaySGPA() {

        System.out.printf("SGPA:" + sgpa);

    }

    public static void main(String[] args) {

        Student student = new Student();

        student.acceptDetails();

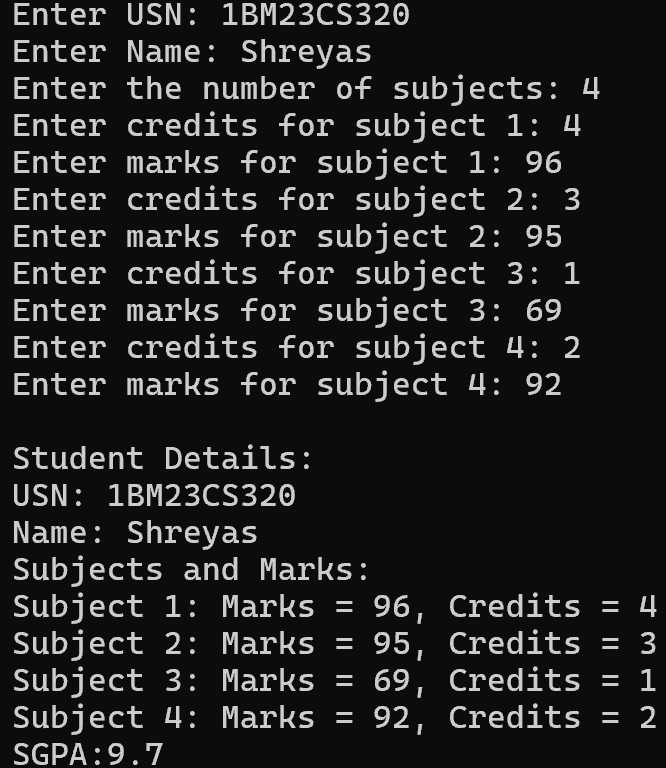
        student.displayDetails();

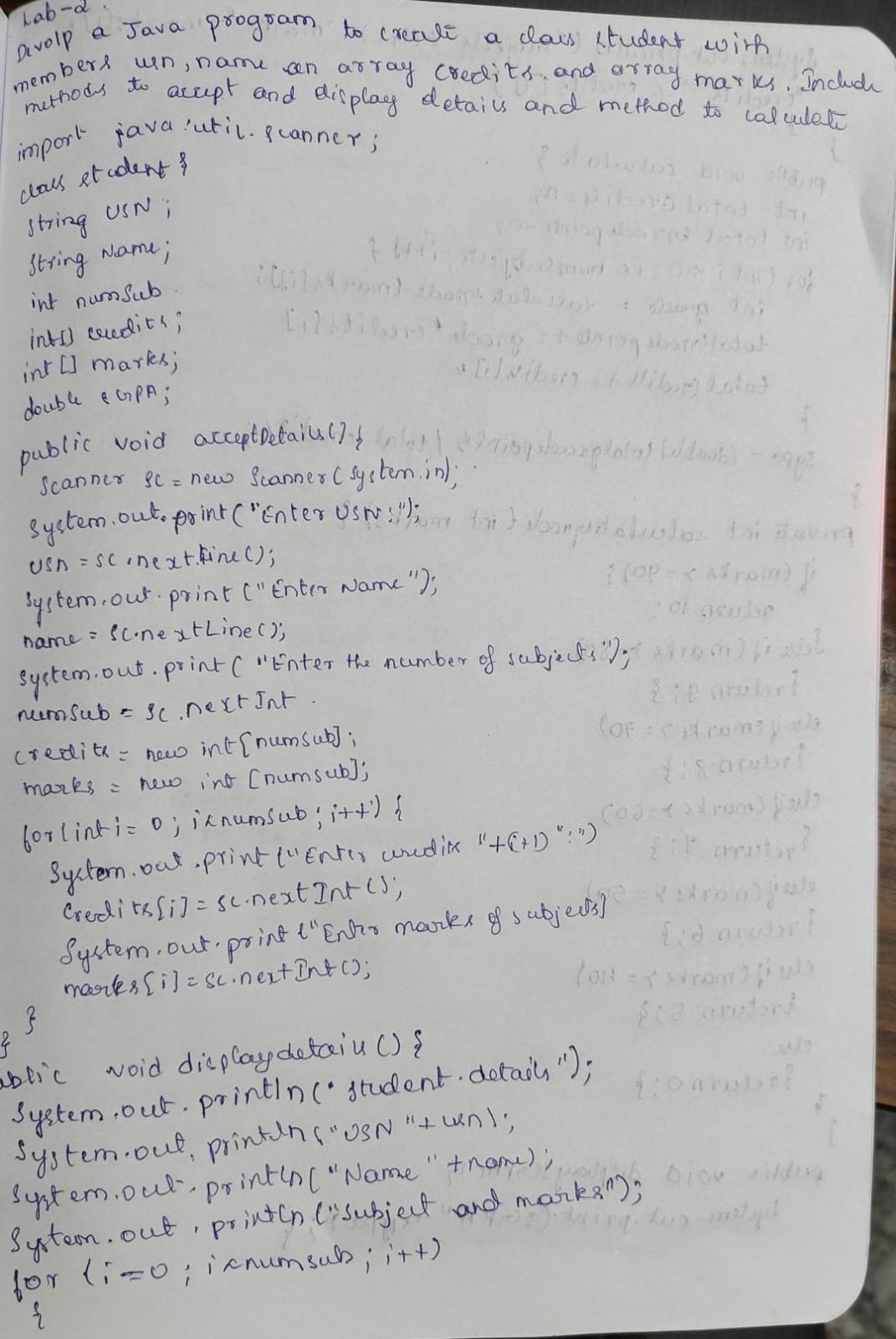
        student.calculateSGPA();

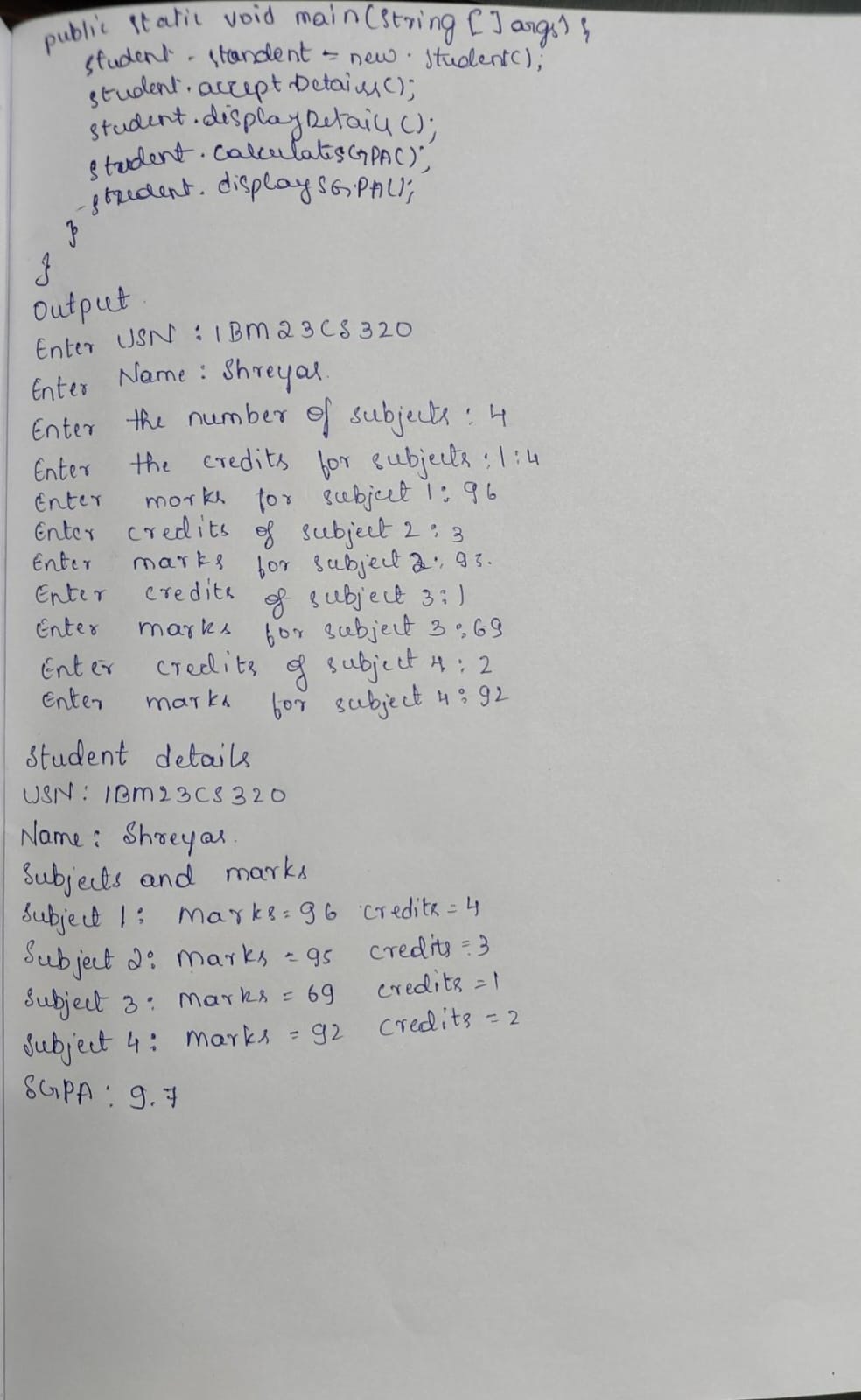
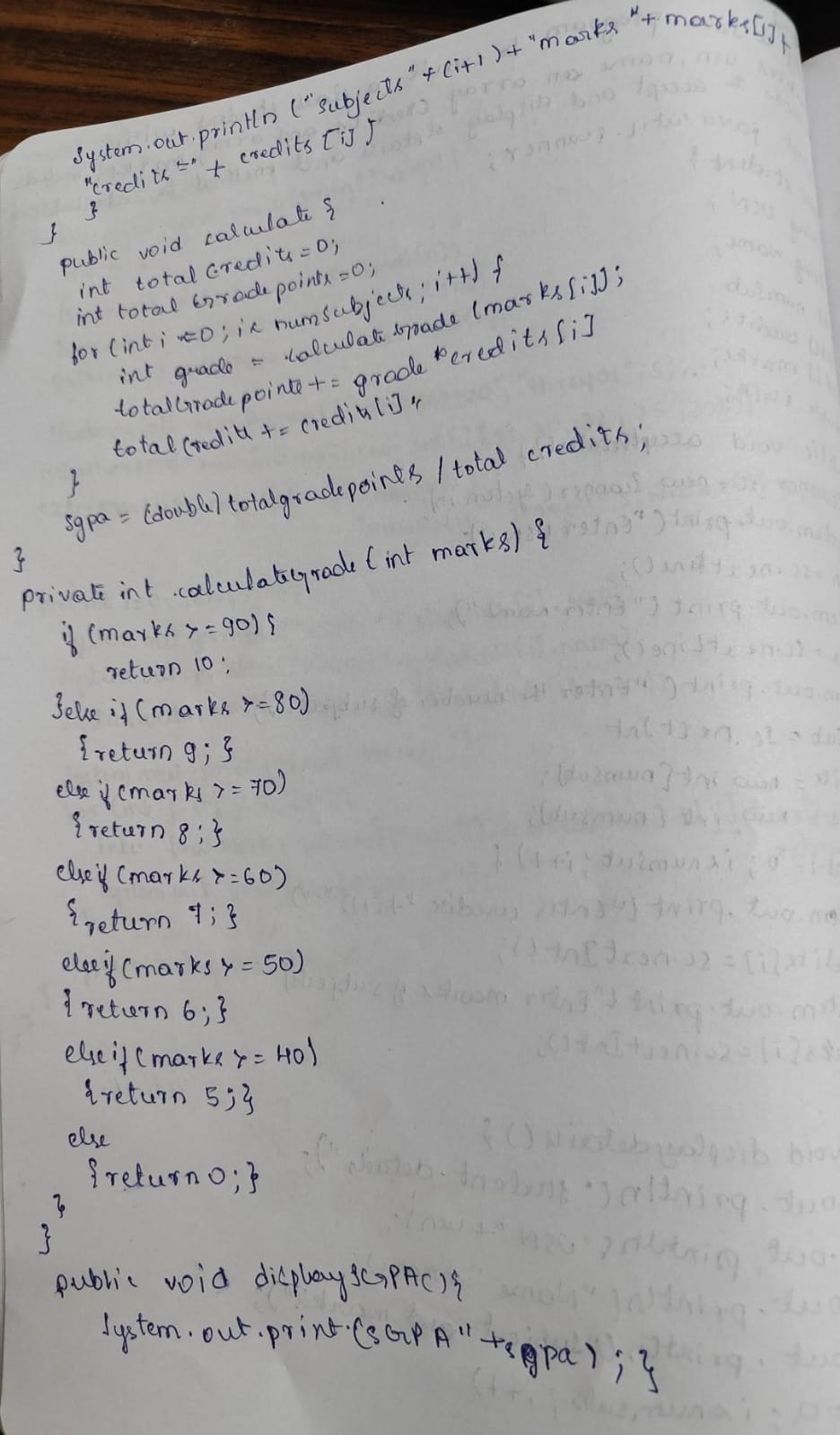
        student.displaySGPA();

    }

}







WEEK 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.

Source Code:

import java.util.Scanner;

class Book {

    int price;

    String author;

    String name;

    int pages;

    public Book(int price, String author, String name, int pages) {

        this.price = price;

        this.author = author;

        this.name = name;

        this.pages = pages;

    }

    public void setter() {

        System.out.println("enter the price,author,name and pages of the book");

        Scanner sc = new Scanner(System.in);

         this.price=sc.nextInt();

         this.author= sc.next();

         this.name=sc.next();

         this.pages=sc.nextInt();

    }

    public void getter() {

        System.out.println("Book Details:");

        System.out.println("Price:"+price);

        System.out.println("Author:"+author);

        System.out.println("Name:"+name);

        System.out.println("Pages:"+pages);

    }

    public String toString() {

                return "these are book details";

    }

}

public class Pro {

    public static void main(String[] args) {

        Scanner s1 = new Scanner(System.in);

        System.out.println("enter the number of books");

        int n = s1.nextInt();

       Book []b1 = new Book[n];

        for(int i=0;i<n;i++){

            b1[i] = new Book(200,"sachin","The Pride",111);

            b1[i].getter();

            b1[i].setter();

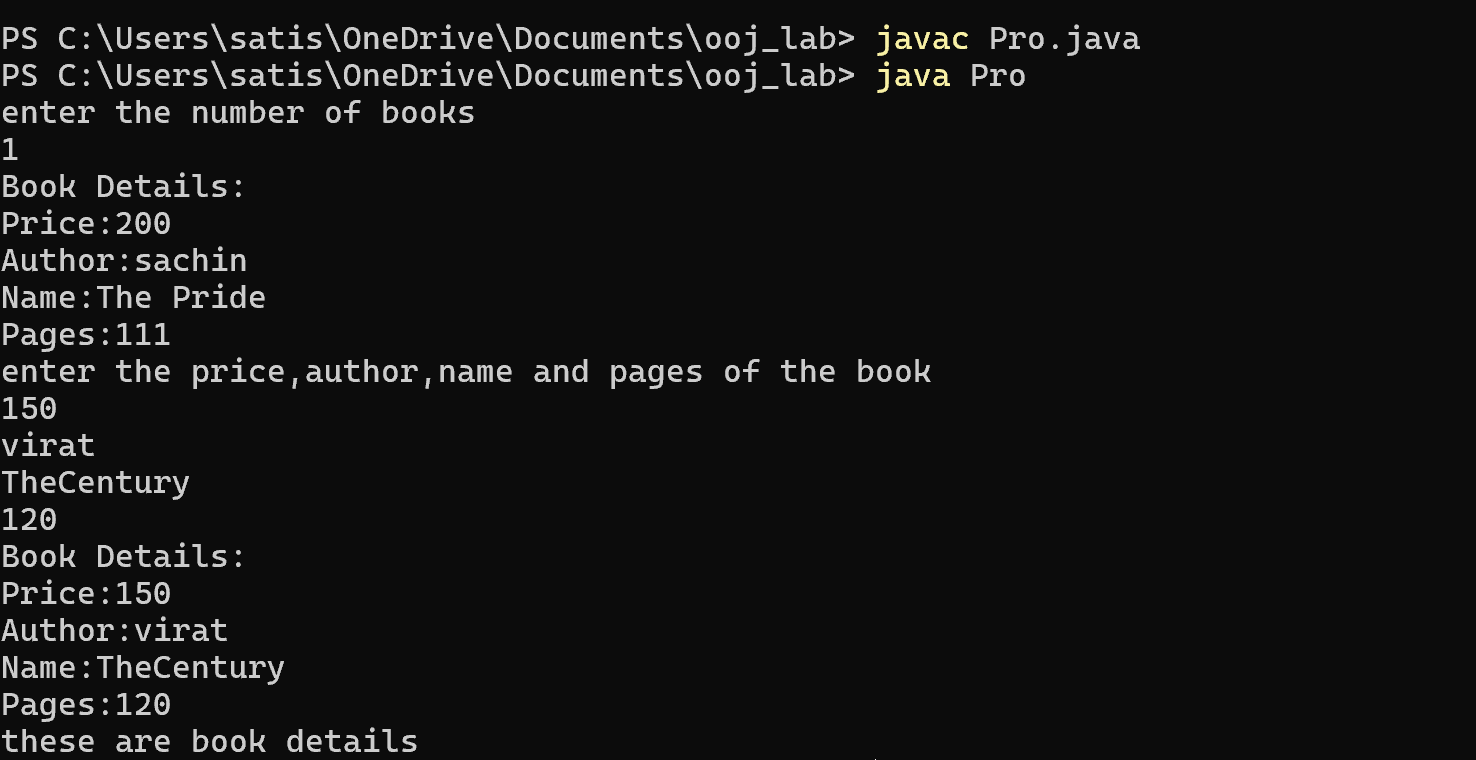
            b1[i].getter();

         System.out.println(b1[i]);

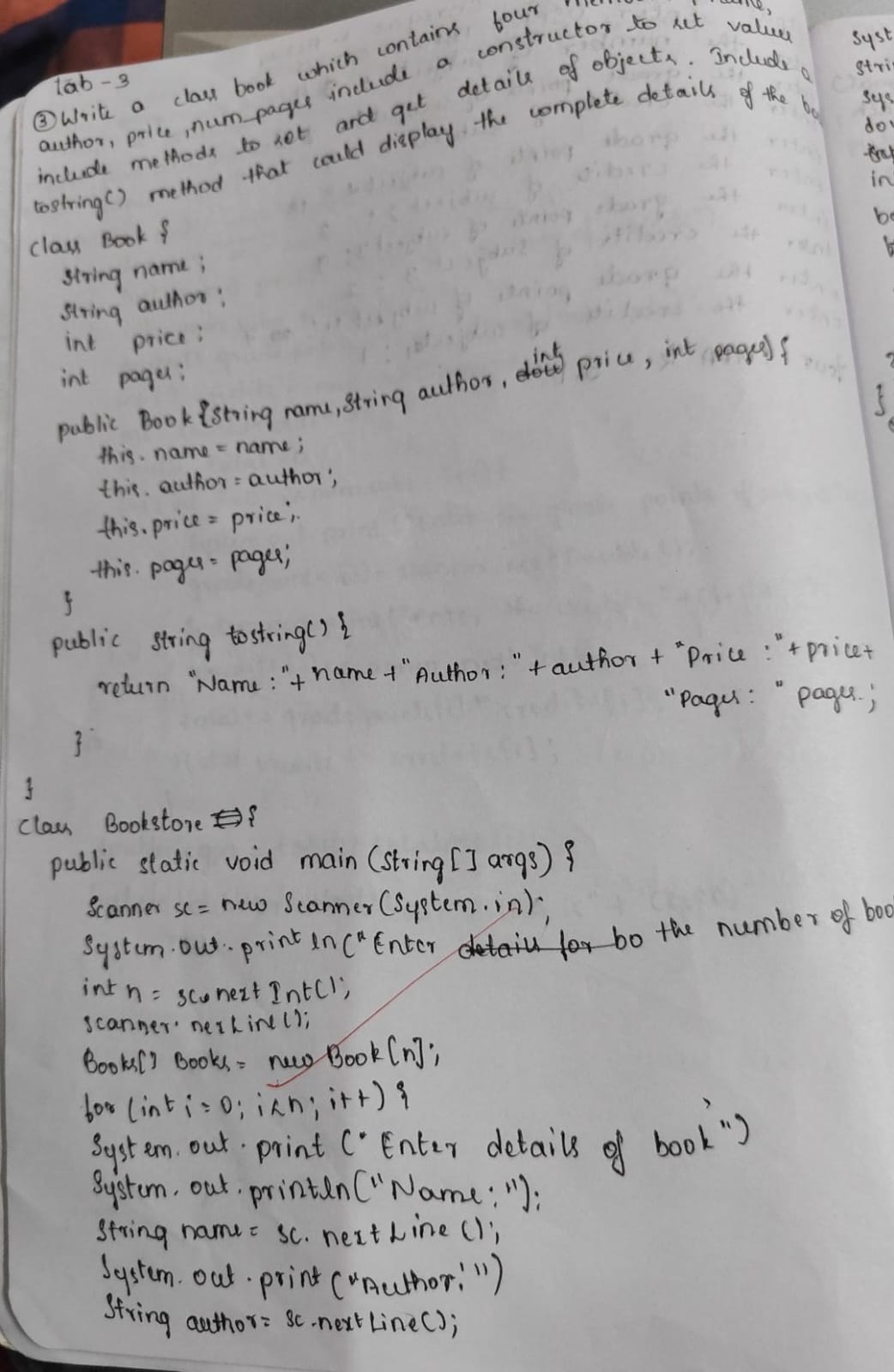
        }

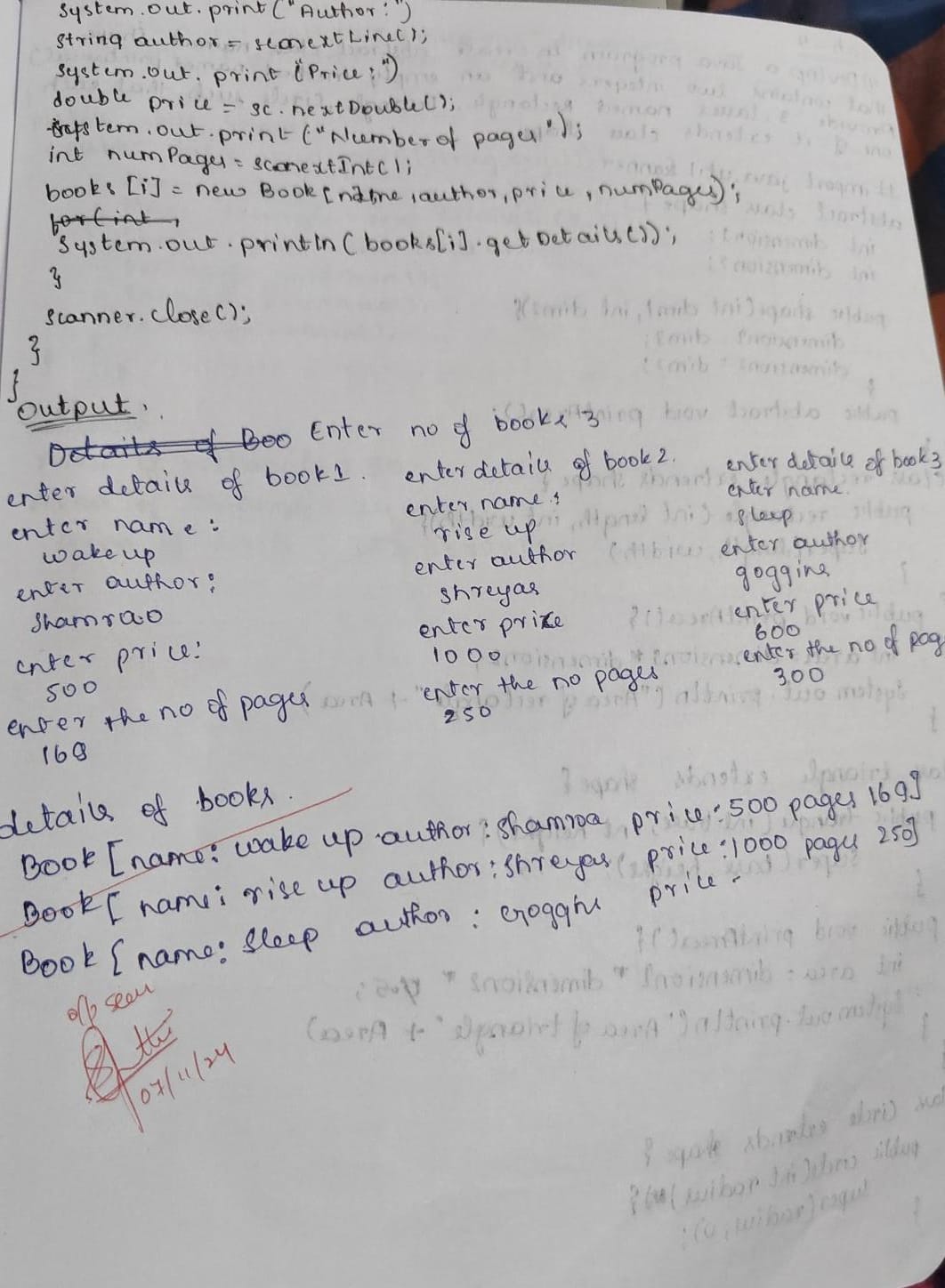
    }

}

Output:

Written code and output:





WEEK 4 :

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

Source Code:

abstract class Shape {

    int dim1;

    int dim2;

    abstract void printArea();

}

class Rectangle extends Shape {

    public Rectangle(int length, int width) {

        this.dim1 = length;

        this.dim2 = width;

    }

    void printArea() {

        int area = dim1 \* dim2;

        System.out.println("Area of Rectangle: " + area);

    }

}

class Triangle extends Shape {

    public Triangle(int base, int height) {

        this.dim1 = base;

        this.dim2 = height;

    }

    void printArea() {

        double area = 0.5 \* dim1 \* dim2;

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

    }

}

class Circle extends Shape {

    public Circle(int radius) {

        this.dim1 = radius;

        this.dim2 = 0;

    }

    void printArea() {

        double area = Math.PI \* dim1 \* dim1;

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

    }

}

public class Main {

    public static void main(String[] args) {

        Shape rectangle = new Rectangle(8,9);

        Shape triangle = new Triangle(8, 6);

        Shape circle = new Circle(14);

        rectangle.printArea();

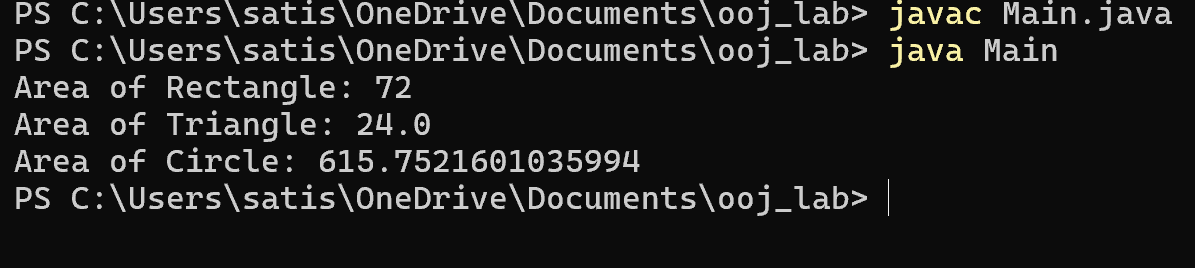
        triangle.printArea();

        circle.printArea();

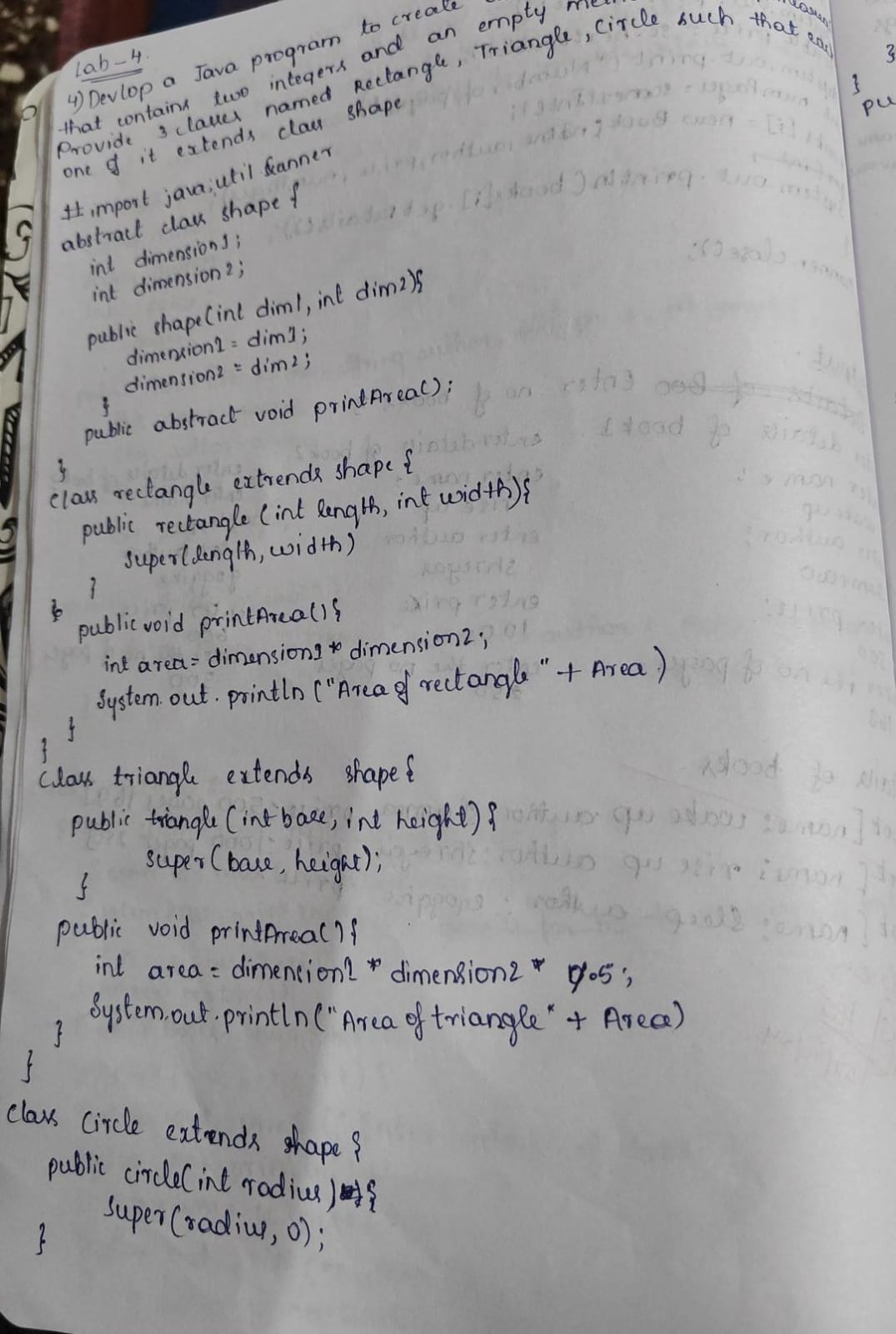
    }

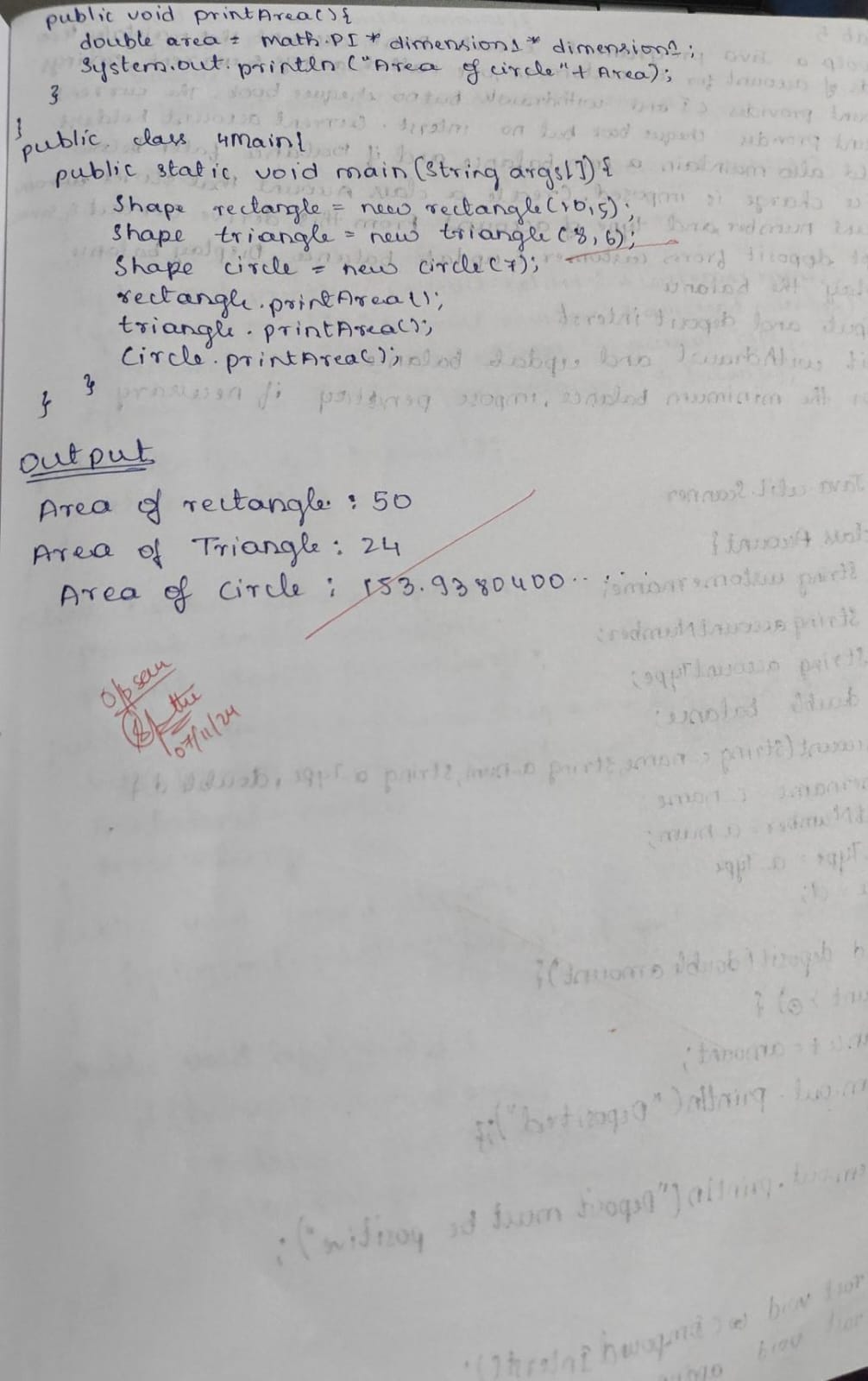
}

Output:



Written code and output:





WEEK 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.

Source Code :

import java.util.Scanner;

class Account {

     String customerName;

     int accountNumber;

     String accountType;

     double balance;

    public Account(String customerName, int accountNumber, String accountType) {

        this.customerName = customerName;

        this.accountNumber = accountNumber;

        this.accountType = accountType;

        this.balance = 0.0;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Amount deposited: " + amount);

            System.out.println("Updated balance: " + balance);

        } else {

            System.out.println("Invalid deposit amount!");

        }

    }

    public void displayBalance() {

        System.out.println("Balance: " + balance);

    }

}

class SavAcct extends Account {

    private double interestRate;

    public SavAcct(String customerName, int accountNumber, double interestRate) {

        super(customerName, accountNumber, "Savings");

        this.interestRate = interestRate;

    }

    public void computeAndDepositInterest() {

        double interest = balance \* (interestRate / 100);

        balance += interest;

        System.out.println("Interest added: " + interest);

        System.out.println("Updated balance: " + balance);

    }

    public void withdraw(double amount) {

        if (amount <= balance) {

            balance -= amount;

            System.out.println("Amount withdrawn: " + amount);

            System.out.println("Updated balance: " + balance);

        } else {

            System.out.println("Insufficient balance!");

        }

    }

}

class CurAcct extends Account {

   double minimumBalance;

    double serviceCharge;

    public CurAcct(String customerName, int accountNumber, double minimumBalance, double serviceCharge) {

        super(customerName, accountNumber, "Current");

        this.minimumBalance = minimumBalance;

        this.serviceCharge = serviceCharge;

    }

    public void withdraw(double amount) {

        if (amount <= balance) {

            balance -= amount;

            System.out.println("Amount withdrawn: " + amount);

            if (balance < minimumBalance) {

                imposePenalty();

            }

            System.out.println("Updated balance: " + balance);

        } else {

            System.out.println("Insufficient balance!");

        }

    }

    private void imposePenalty() {

        balance -= serviceCharge;

        System.out.println("Balance fell below minimum. Service charge imposed: " + serviceCharge);

    }

}

public class Bank {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Choose account type:\n1. Savings Account\n2. Current Account");

        int choice = scanner.nextInt();

        scanner.nextLine();

        System.out.println("Enter customer name: ");

        String name = scanner.nextLine();

        System.out.println("Enter account number: ");

        int accNum = scanner.nextInt();

        if (choice == 1) {

            System.out.println("Enter interest rate for savings account: ");

            double interestRate = scanner.nextDouble();

            SavAcct savAccount = new SavAcct(name, accNum, interestRate);

            System.out.println("Enter amount to deposit: ");

            double deposit = scanner.nextDouble();

            savAccount.deposit(deposit);

            savAccount.computeAndDepositInterest();

            System.out.println("Enter amount to withdraw: ");

            double withdrawAmount = scanner.nextDouble();

            savAccount.withdraw(withdrawAmount);

        } else if (choice == 2) {

            System.out.println("Enter minimum balance for current account: ");

            double minBalance = scanner.nextDouble();

            System.out.println("Enter service charge for falling below minimum balance: ");

            double serviceCharge = scanner.nextDouble();

            CurAcct curAccount = new CurAcct(name, accNum, minBalance, serviceCharge);

            System.out.println("Enter amount to deposit: ");

            double deposit = scanner.nextDouble();

            curAccount.deposit(deposit);

            System.out.println("Enter amount to withdraw: ");

            double withdrawAmount = scanner.nextDouble();

            curAccount.withdraw(withdrawAmount);

        } else {

            System.out.println("Invalid account type selected.");

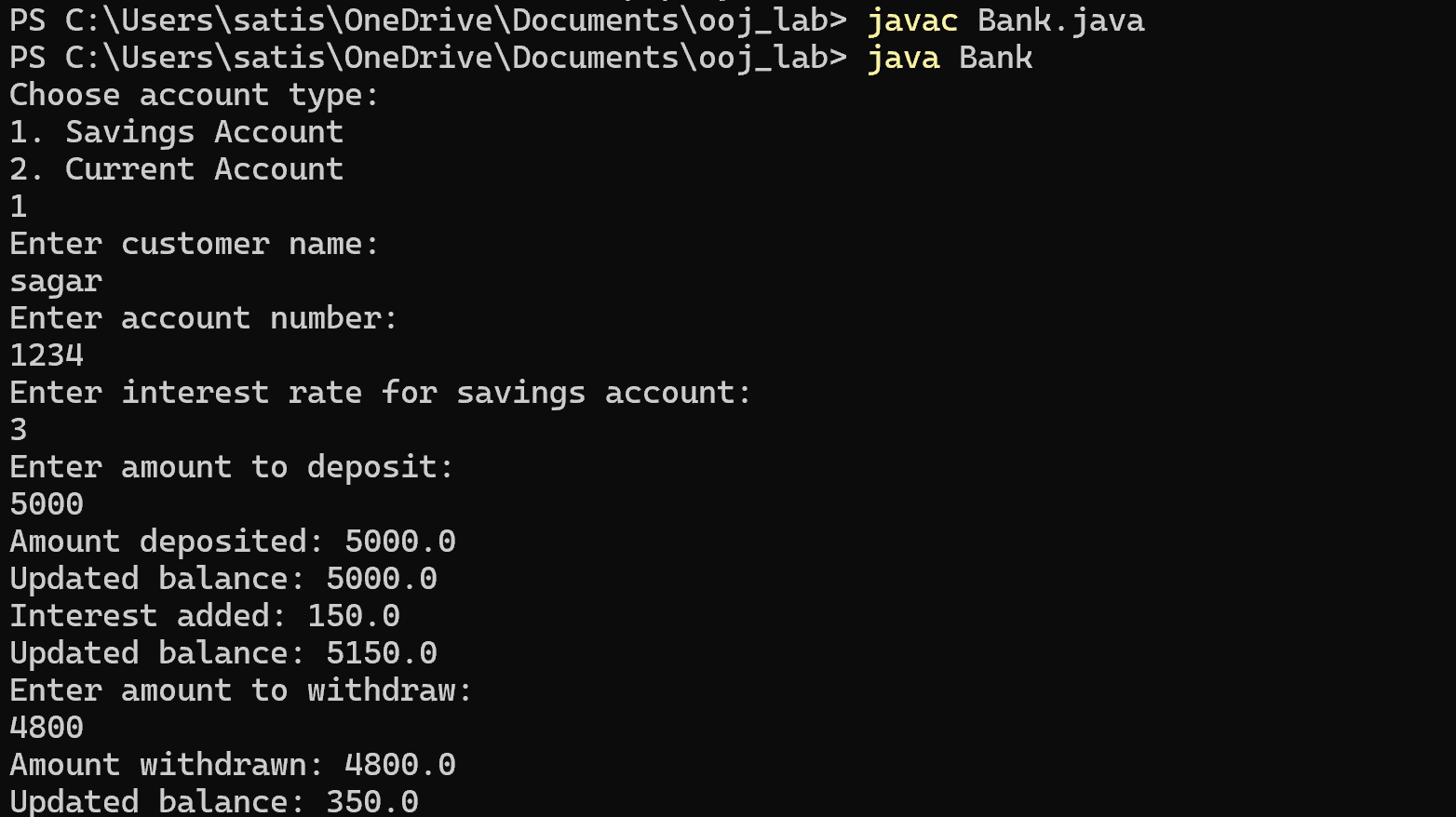
        }

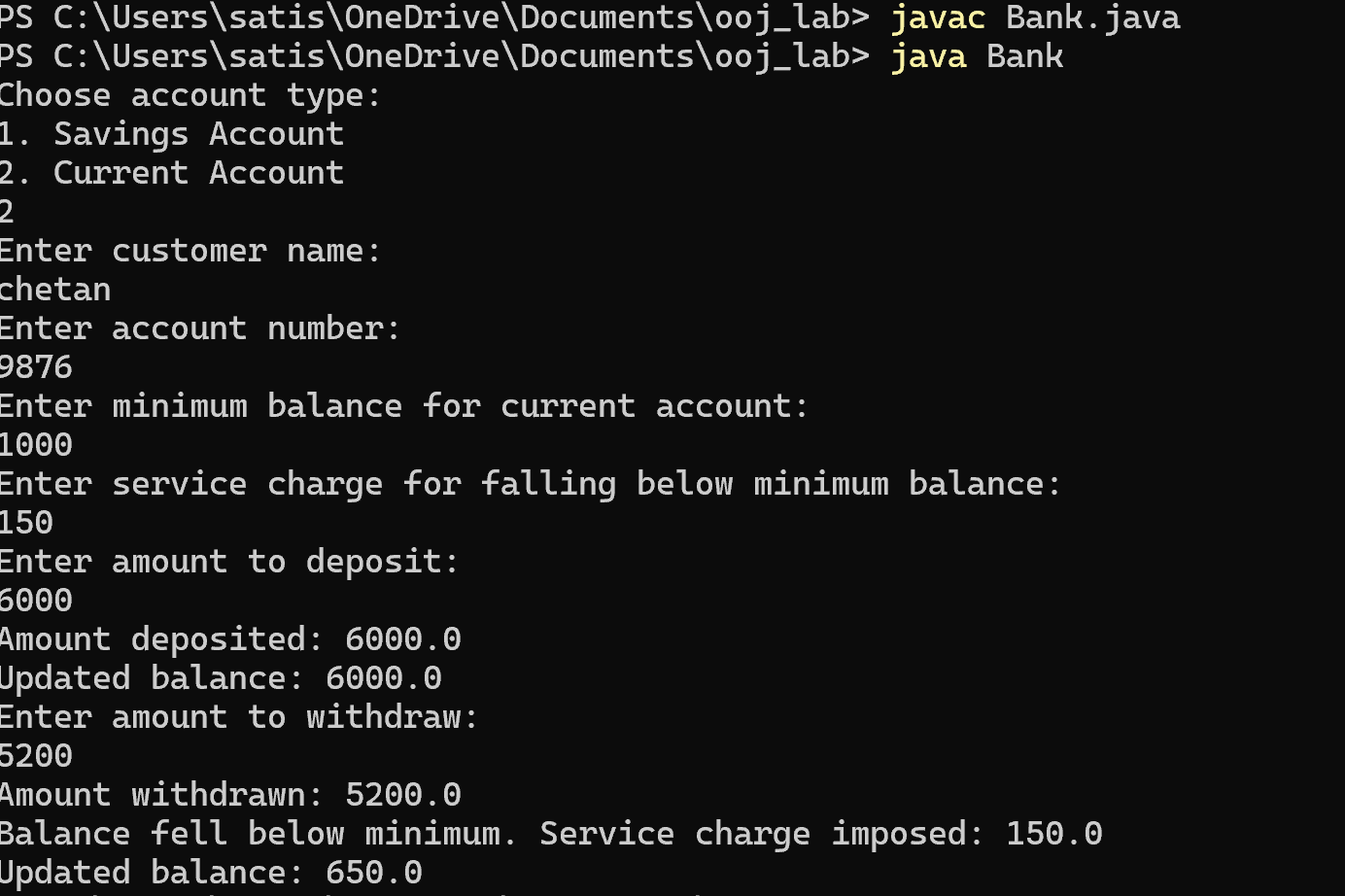
        scanner.close();

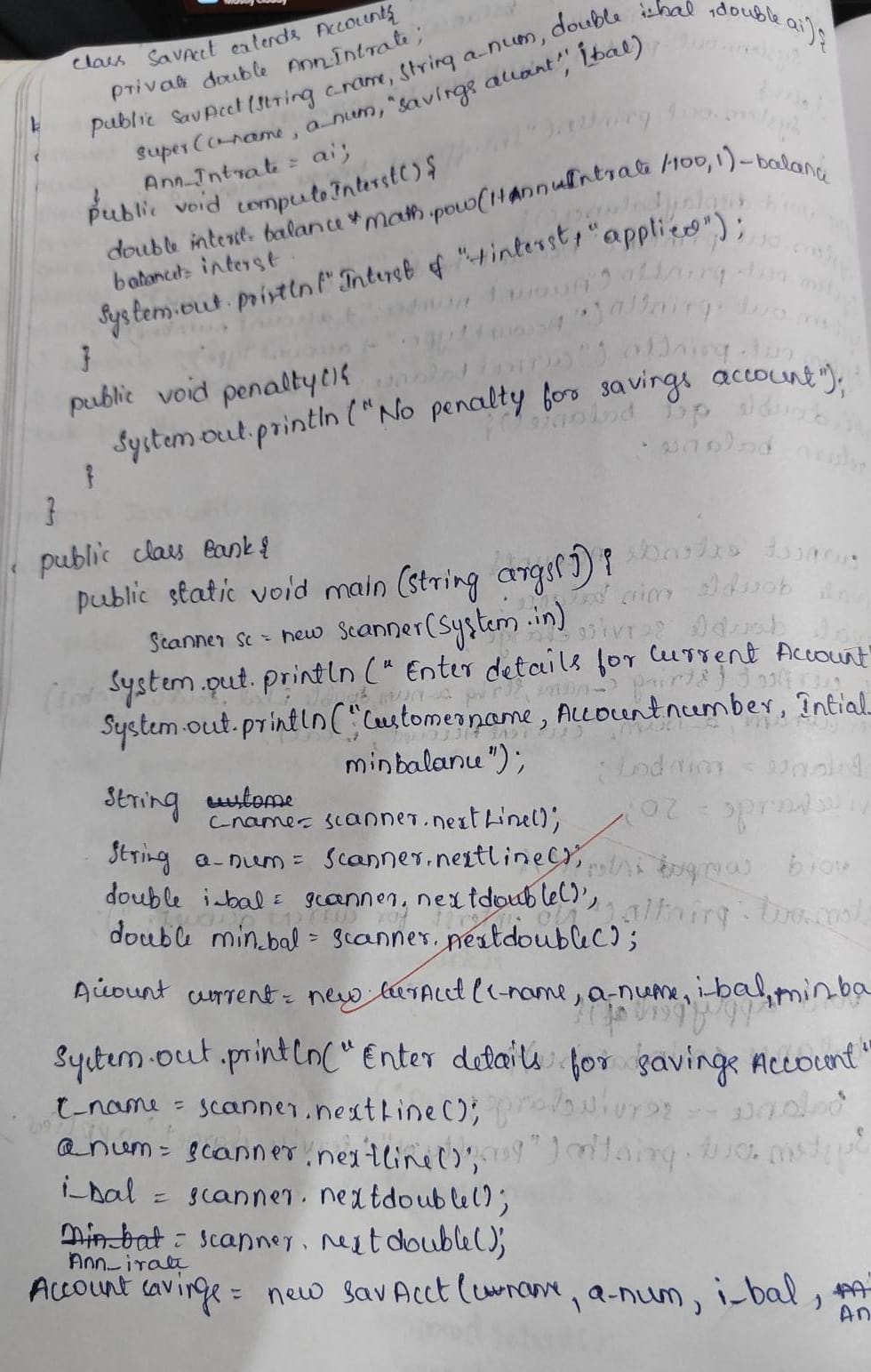
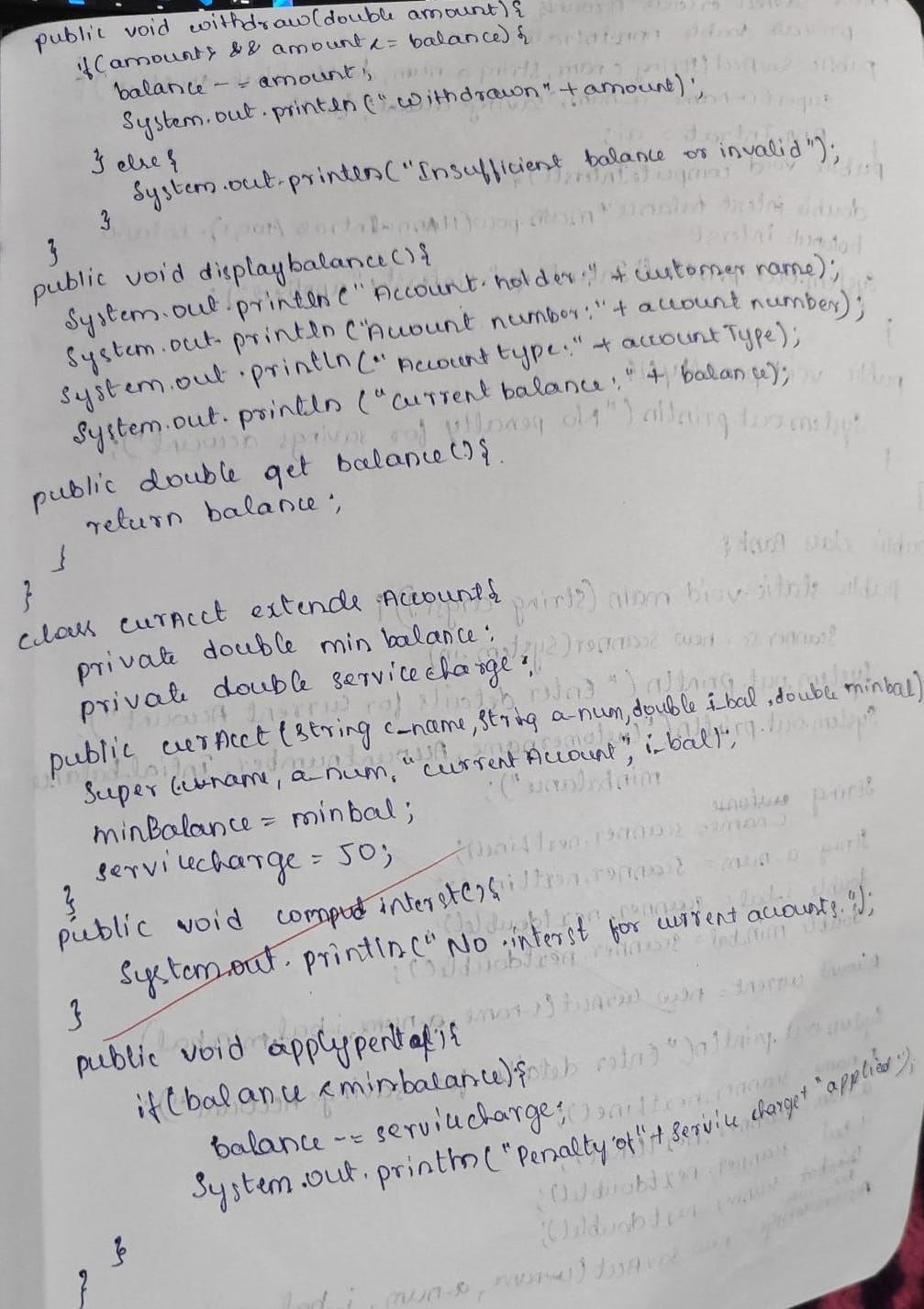
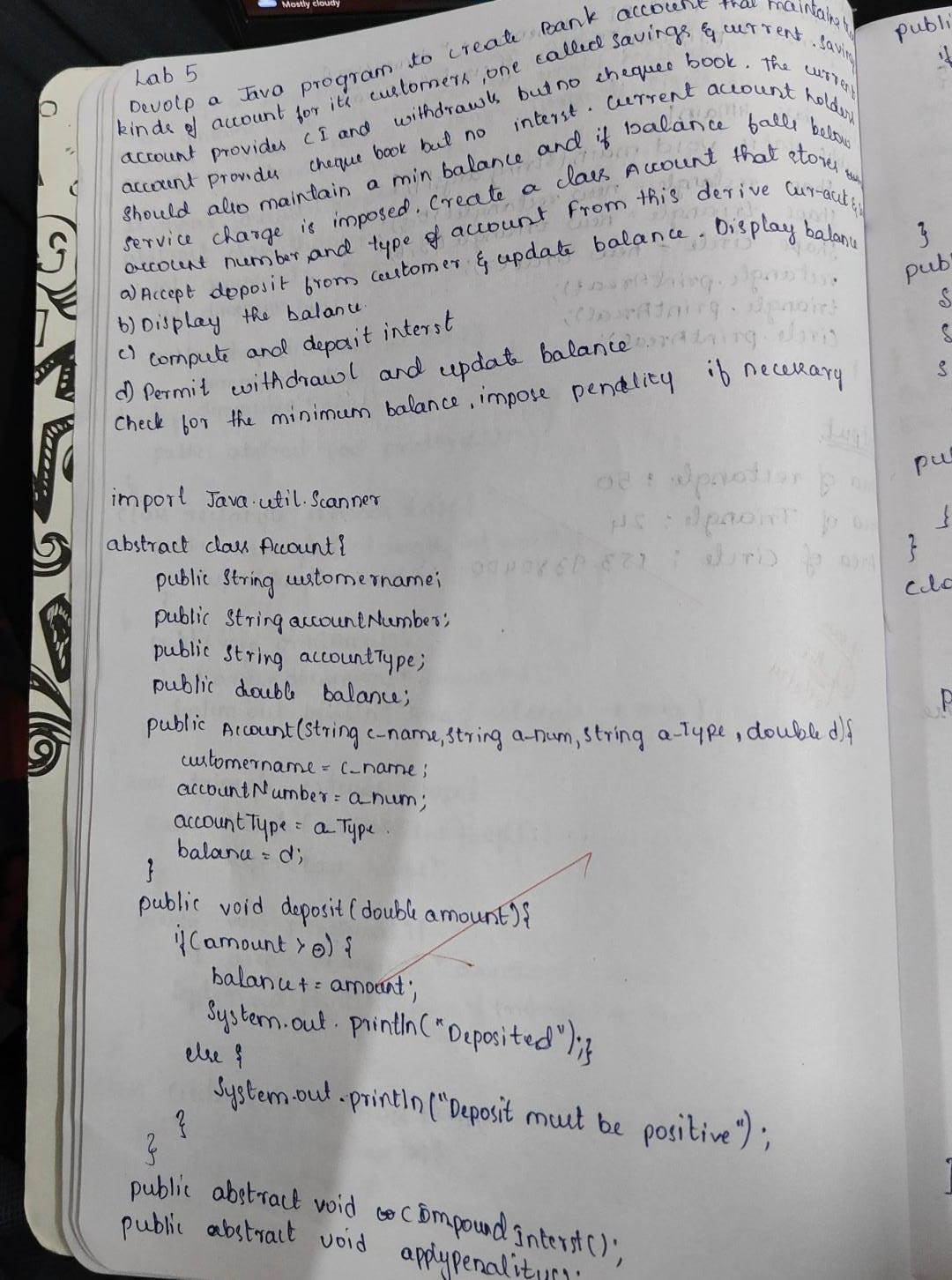
    }

}

Output :







WEEK 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.

Source Code :

import CIE.Internals;

import SEE.External;

import java.util.Scanner;

public class Studentmarks {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of students: ");

        int n = scanner.nextInt();

        scanner.nextLine();

        Internals[] cieStudents = new Internals[n];

        External[] seeStudents = new External[n];

        for (int i = 0; i < n; i++) {

            System.out.println("Enter details for CIE Student " + (i + 1) + ": ");

            System.out.print("USN: ");

            String usn = scanner.nextLine();

            System.out.print("Name: ");

            String name = scanner.nextLine();

            System.out.print("Semester: ");

            int sem = scanner.nextInt();

            int[] internalMarks = new int[5];

            System.out.println("Enter internal marks for 5 courses: ");

            for (int j = 0; j < 5; j++) {

                internalMarks[j] = scanner.nextInt();

            }

            cieStudents[i] = new Internals(usn, name, sem, internalMarks);

            scanner.nextLine();

            System.out.println("Enter details for SEE Student " + (i + 1) + ": ");

            System.out.print("USN: ");

            usn = scanner.nextLine();

            System.out.print("Name: ");

            name = scanner.nextLine();

            System.out.print("Semester: ");

            sem = scanner.nextInt();

            int[] externalMarks = new int[5];

            System.out.println("Enter external marks for 5 courses: ");

            for (int j = 0; j < 5; j++) {

                externalMarks[j] = scanner.nextInt();

            }

            seeStudents[i] = new External(usn, name, sem, externalMarks);

            scanner.nextLine();

        }

        System.out.println("\nFinal Marks for all students:");

        for (int i = 0; i < n; i++) {

            cieStudents[i].displayStudentDetails();

            cieStudents[i].displayInternalMarks();

            seeStudents[i].displayStudentDetails();

            seeStudents[i].displayExternalMarks();

            int[] internalMarks = cieStudents[i].getInternalMarks();

            int[] externalMarks = seeStudents[i].getExternalMarks();

            int[] finalMarks = new int[5];

            for (int j = 0; j < 5; j++) {

                finalMarks[j] = internalMarks[j] + externalMarks[j];

            }

            System.out.print("Final Marks: ");

            for (int mark : finalMarks) {

                System.out.print(mark + " ");

            }

            System.out.println("\n");

        }

        scanner.close();

    }

}

package CIE;

public class Internals extends Student {

    private int[] internalMarks = new int[5];

    public Internals(String usn, String name, int sem, int[] internalMarks) {

        super(usn, name, sem); // Call parent constructor

        this.internalMarks = internalMarks;

    }

    public void displayInternalMarks() {

        System.out.print("Internal Marks: ");

        for (int mark : internalMarks) {

            System.out.print(mark + " ");

        }

        System.out.println();

    }

    public int[] getInternalMarks() {

        return internalMarks;

    }

}

package CIE;

public class Student {

    protected String usn;

    protected String name;

    protected int sem;

    public Student(String usn, String name, int sem) {

        this.usn = usn;

        this.name = name;

        this.sem = sem;

    }

    public void displayStudentDetails() {

        System.out.println("USN: " + usn + ", Name: " + name + ", Semester: " + sem);

    }

}

package SEE;

import CIE.Student;

public class External extends Student {

    private int[] externalMarks = new int[5];

    public External(String usn, String name, int sem, int[] externalMarks) {

        super(usn, name, sem);

        this.externalMarks = externalMarks;

    }

    public void displayExternalMarks() {

        System.out.print("External Marks: ");

        for (int mark : externalMarks) {

            System.out.print(mark + " ");

        }

        System.out.println();

    }

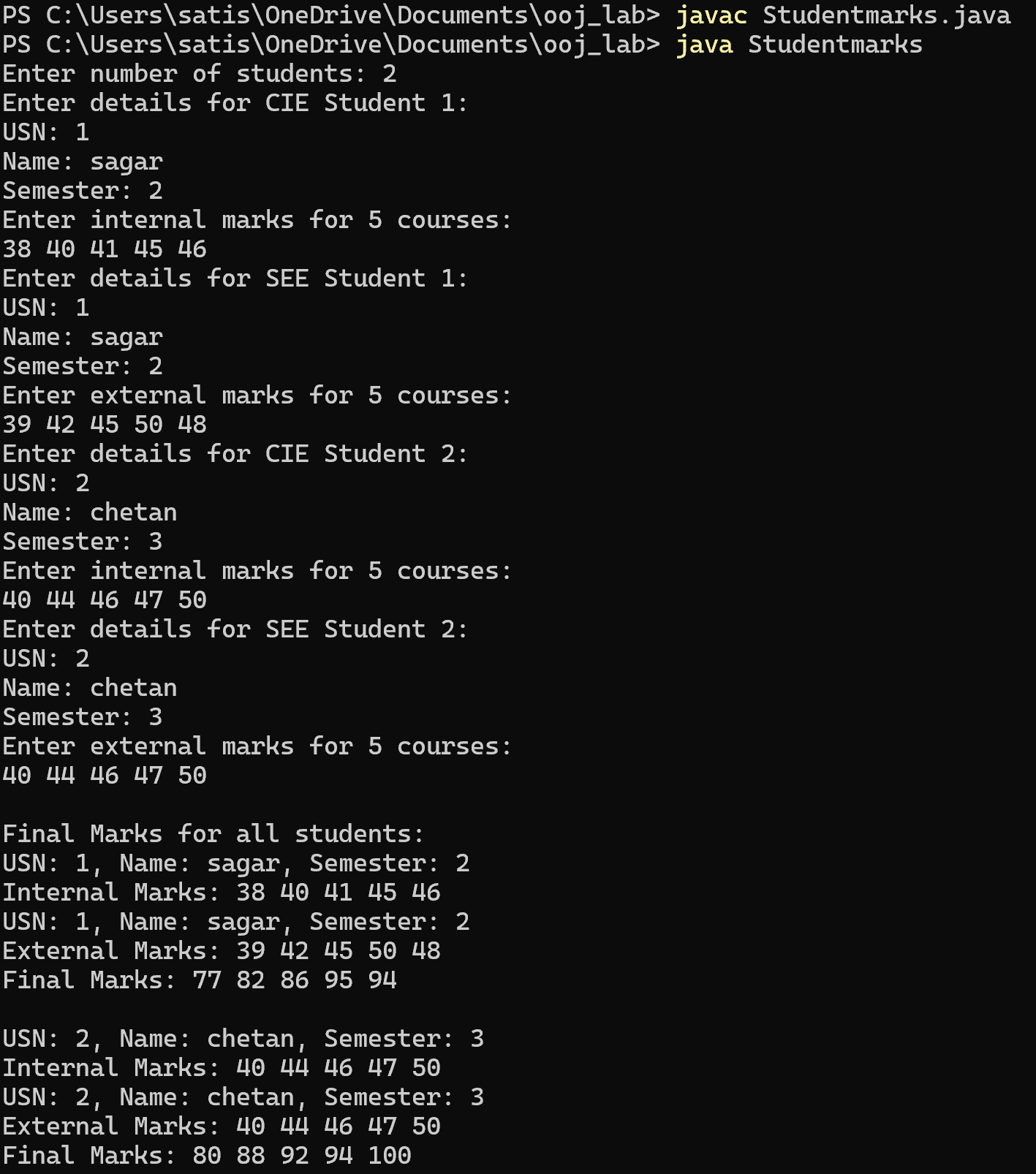
    public int[] getExternalMarks() {

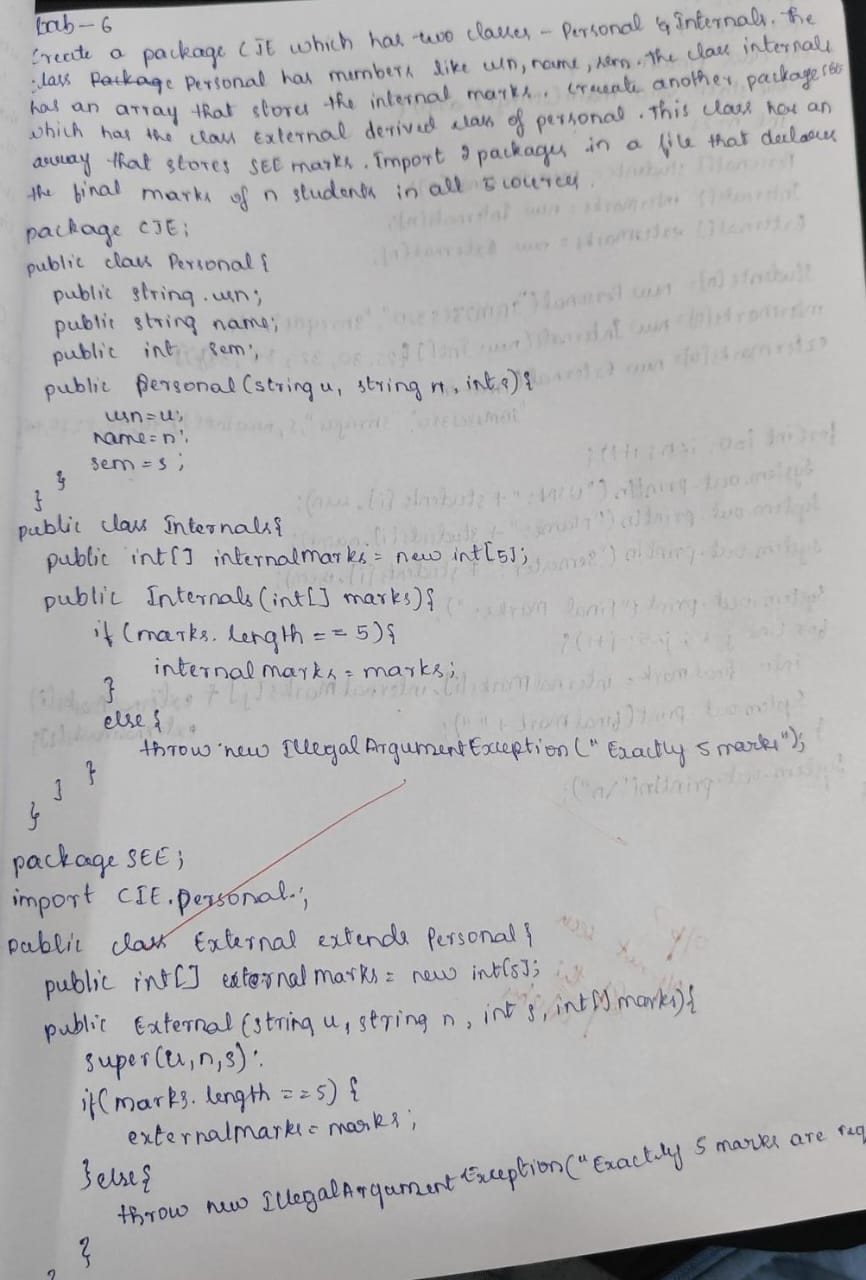
        return externalMarks;

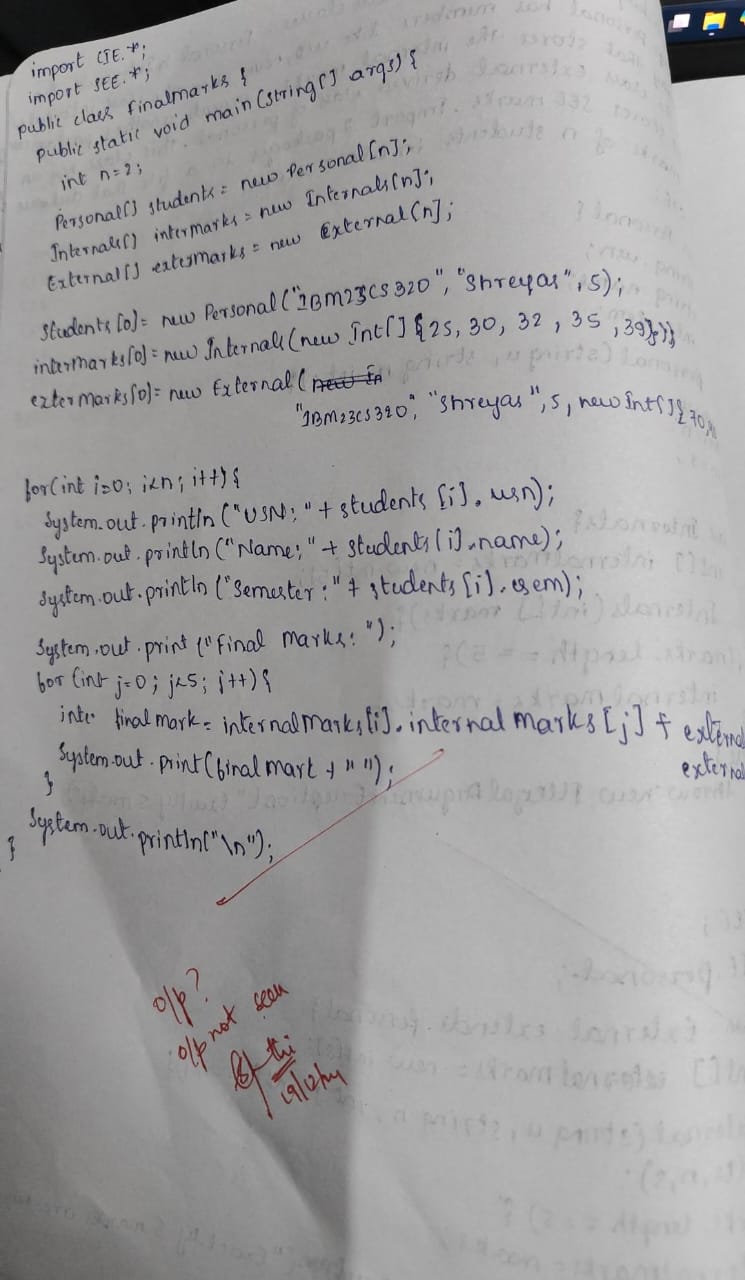
    }

}

Output :



Written code and output: 



WEEK 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.

Source Code :

import java.util.Scanner;

class WrongAgeException extends Exception {

    public WrongAgeException(String message) {

        super(message);

    }

}

class SonAgeException extends Exception {

    public SonAgeException(String message) {

        super(message);

    }

}

class Father {

     int age;

    public Father(int age) throws WrongAgeException {

        if (age <= 0) {

            throw new WrongAgeException("Wrong age");

        }

        this.age = age;

    }

    public int getAge() {

        return age;

    }

}

class Son extends Father {

     int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAgeException, SonAgeException {

        super(fatherAge);

        if (sonAge >= fatherAge) {

            throw new SonAgeException("Son's age cannot be greater than or equal to father's age");

        }

        if(sonAge <= 0){

        throw new WrongAgeException("Wrong age");

        }

        this.sonAge = sonAge;

    }

    public int getSonAge() {

        return sonAge;

    }

}

public class FatherSon{

    public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);

            System.out.print("Enter Father's Age: ");

            int fatherAge = sc.nextInt();

            System.out.print("Enter Son's Age: ");

            int sonAge = sc.nextInt();

            try {

                Son son = new Son(fatherAge, sonAge);

                System.out.println("Accepted Succesfully");

            }

            catch (WrongAgeException e) {

                System.out.println(e.getMessage());

            }

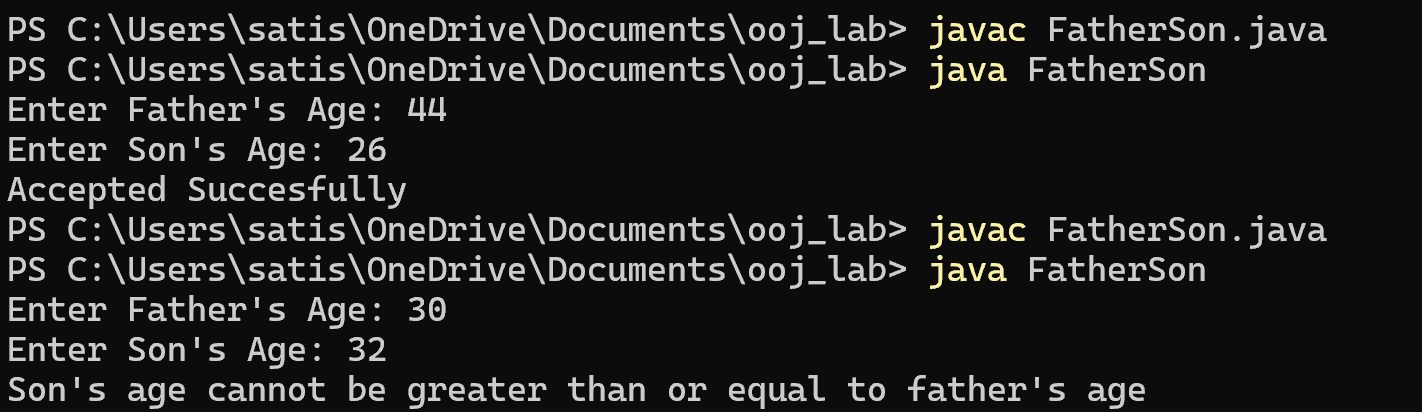
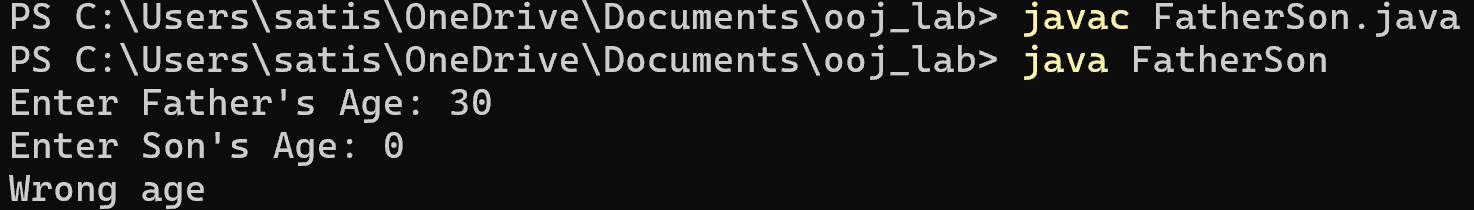
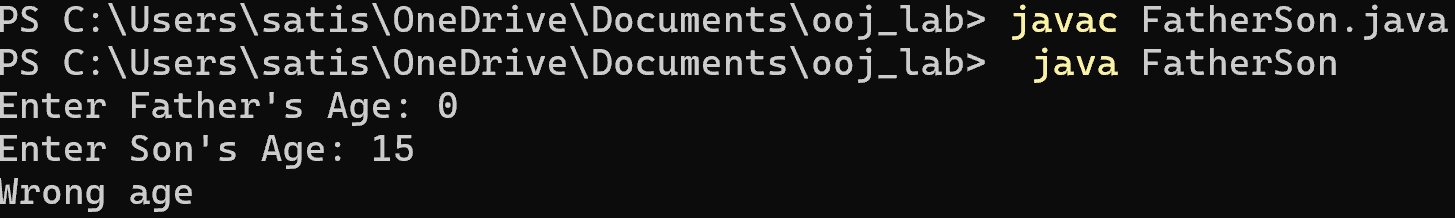
            catch (SonAgeException e) {

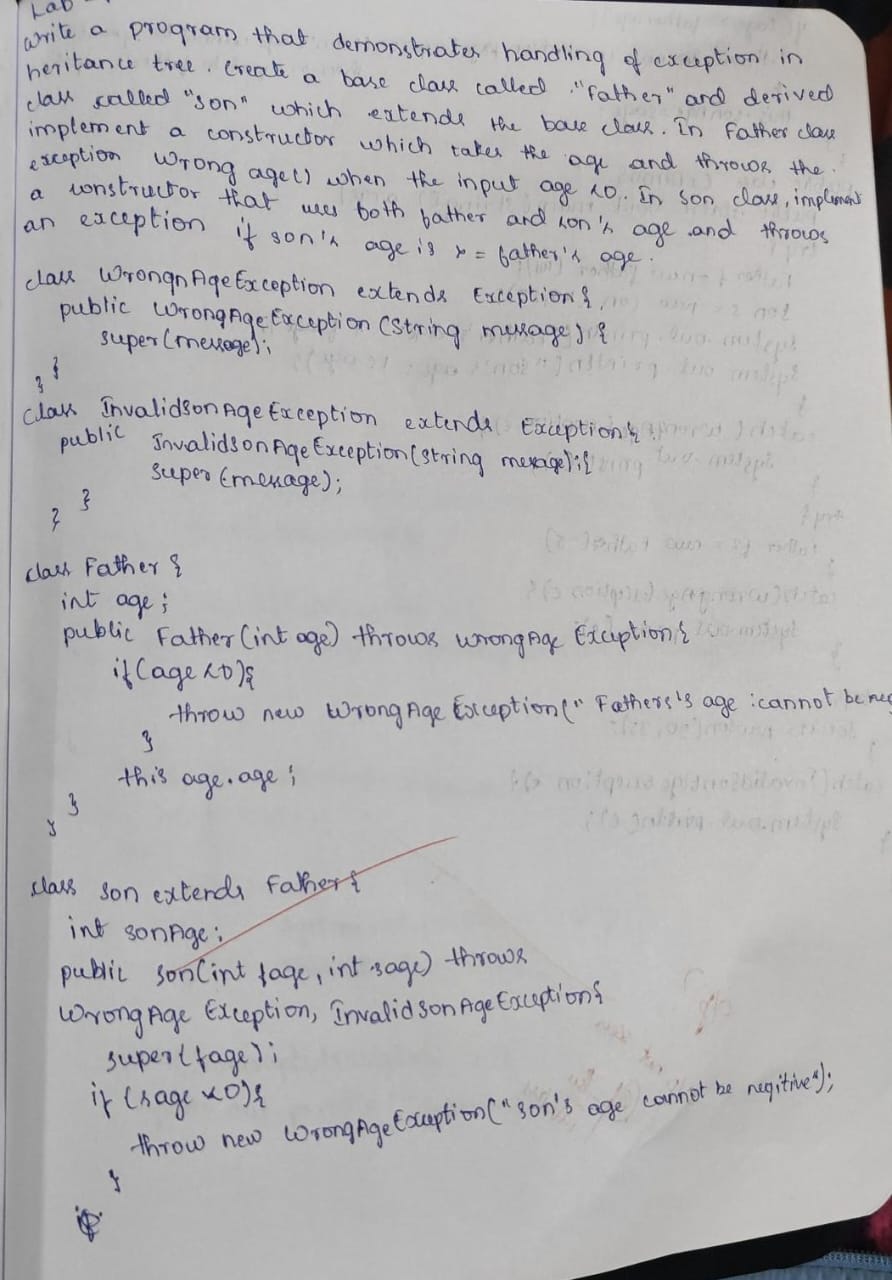
                System.out.println(e.getMessage());

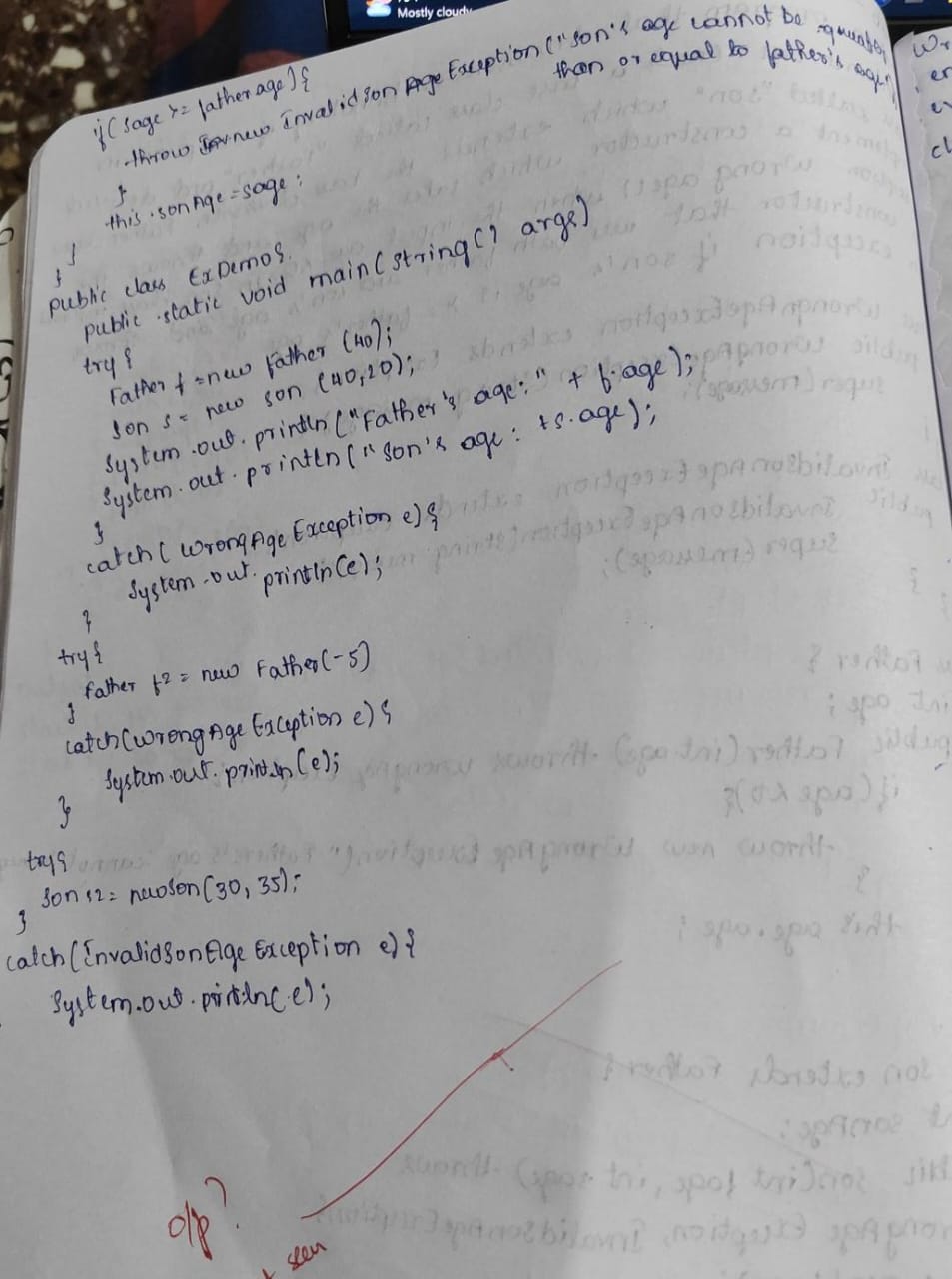
            }

       }

  }

Output :

Written Code & Output :



WEEK 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.

Source Code :

class ThreadDemo extends Thread{

    public void run(){

    while(true){

    System.out.println("BMS College Of Engineering");

    try{

    Thread.sleep(10000);

    }

    catch(InterruptedException e){

    e.printStackTrace();

    }

     }

  }

}

class CSEThread extends Thread{

    public void run(){

    while(true){

    System.out.println("CSE");

    try{

    Thread.sleep(2000);

    }

    catch(InterruptedException e){

    e.printStackTrace();

    }

     }

  }

}

public class Demo{

    public static void main(String[] args){

    ThreadDemo t1 = new ThreadDemo();

    CSEThread t2 = new CSEThread();

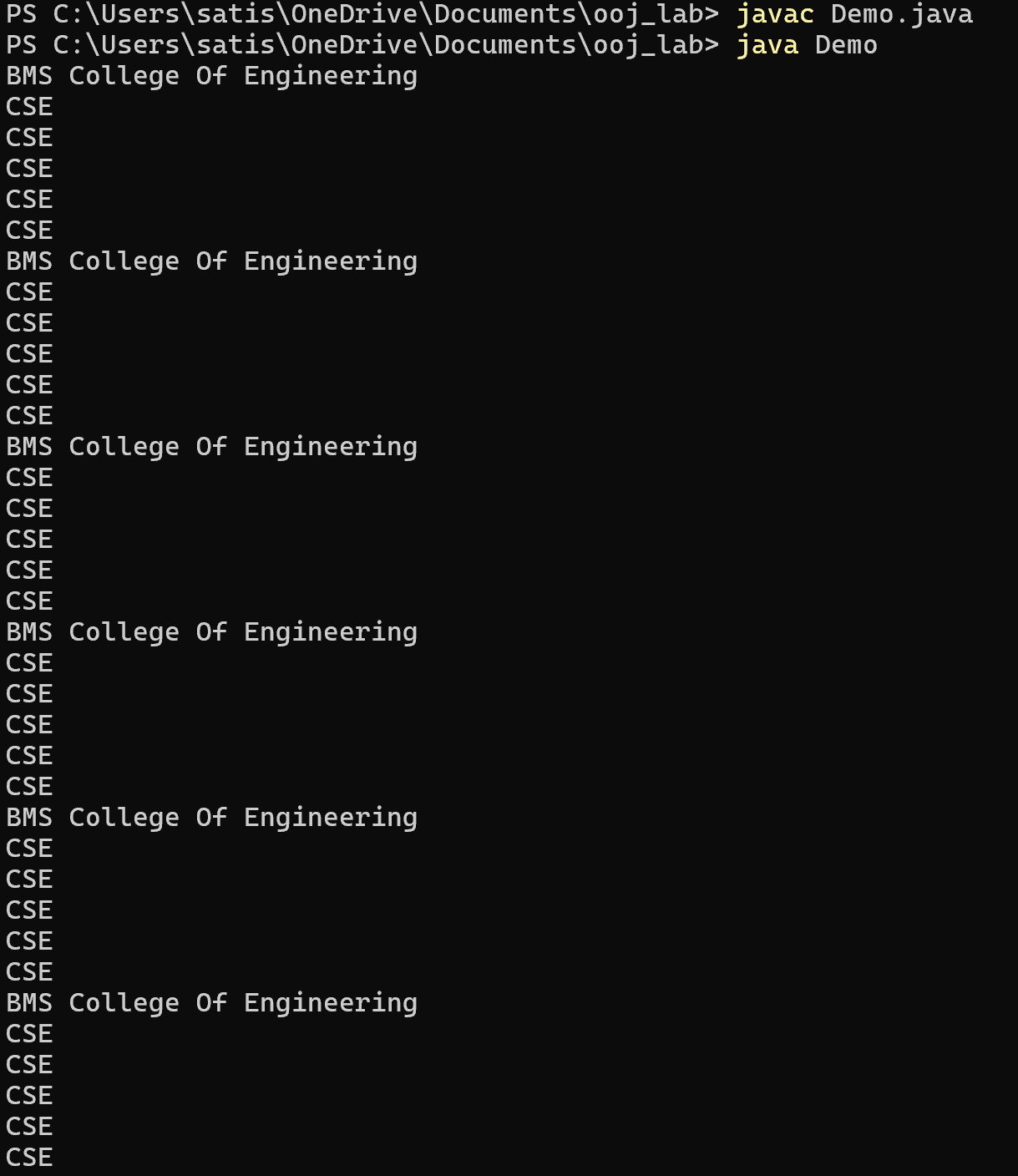
    t1.start();

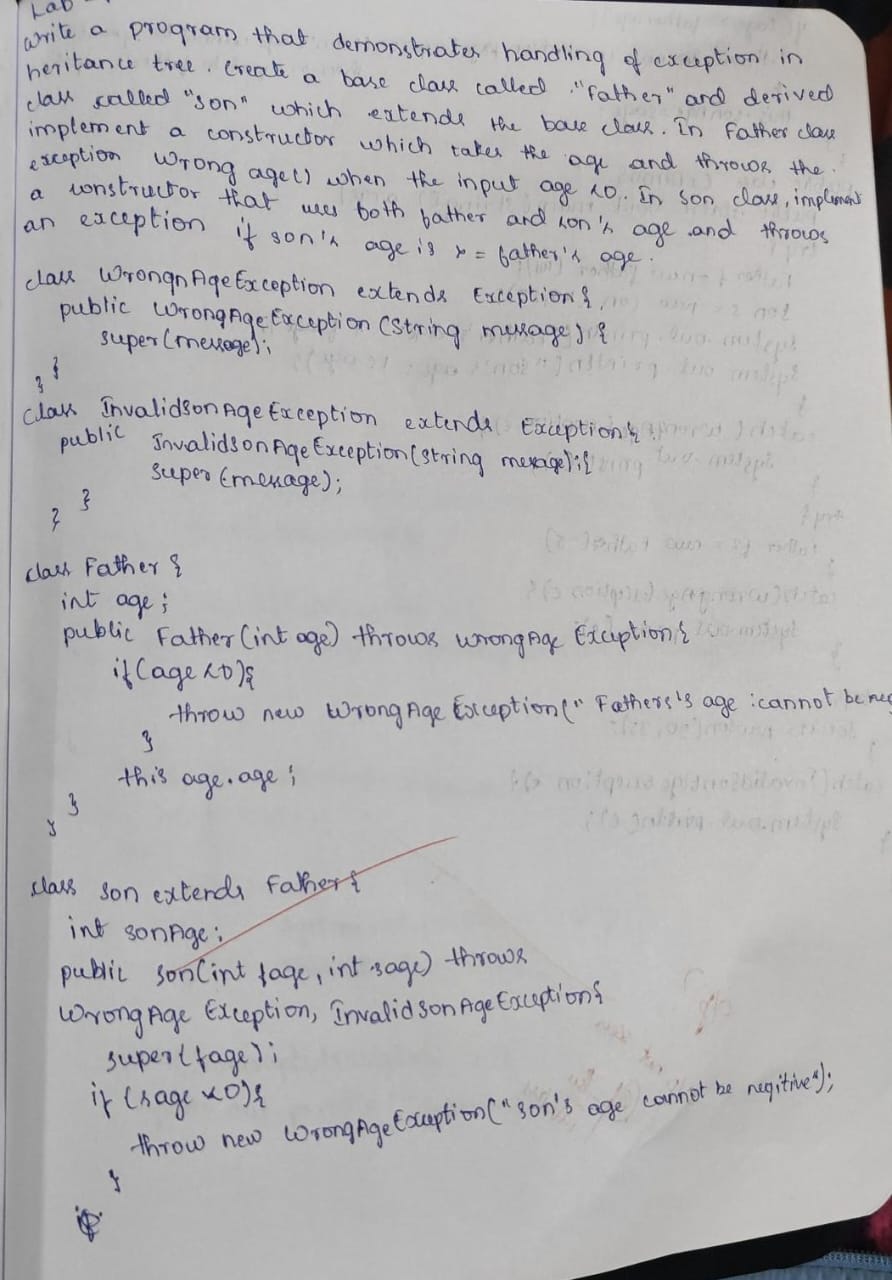
    t2.start();

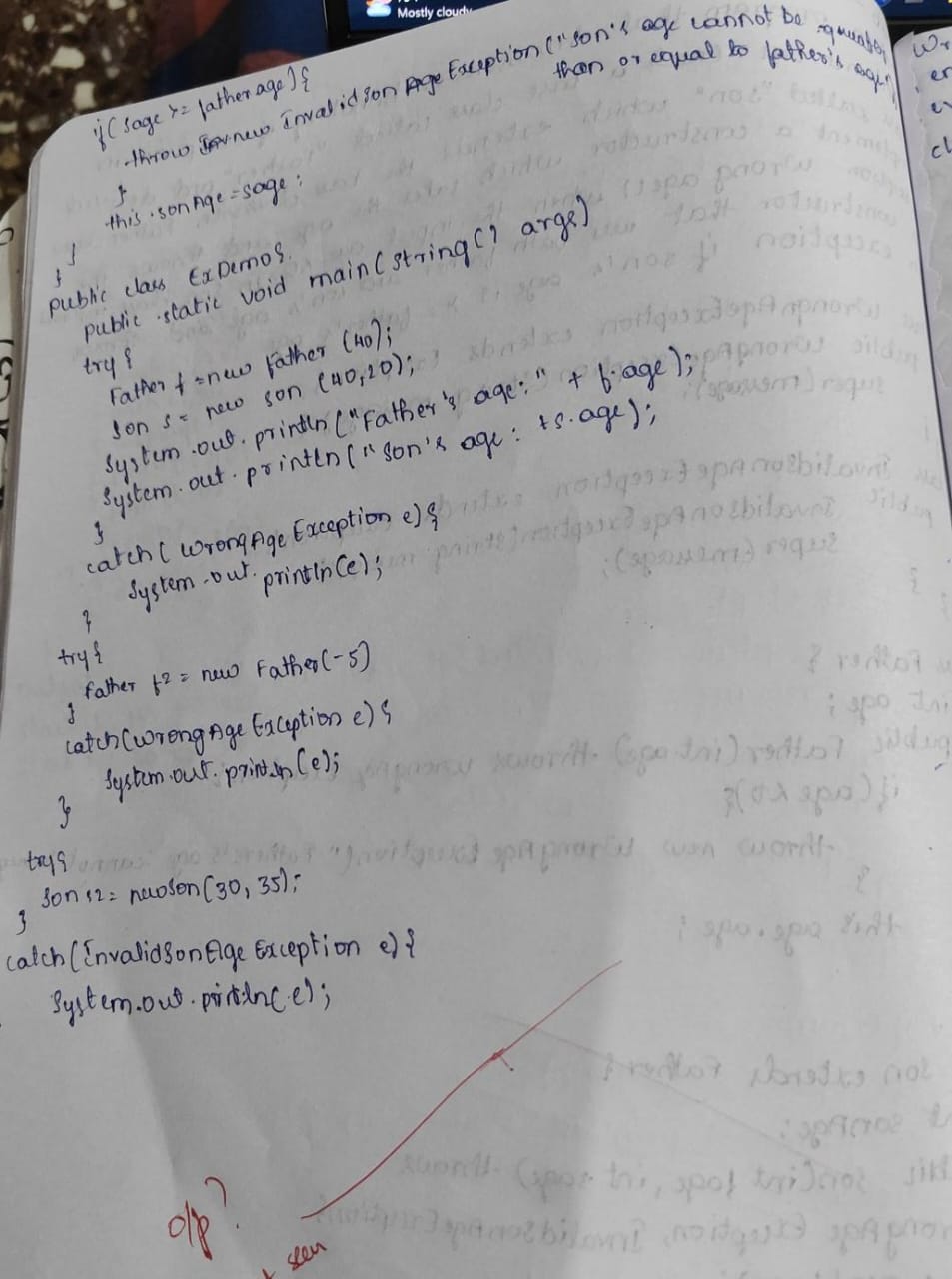
     }

 }

Output :



Written code and output: 



**LABORATORY 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.

**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()-8);

else g.drawString(out,100,200); flag=0;

}

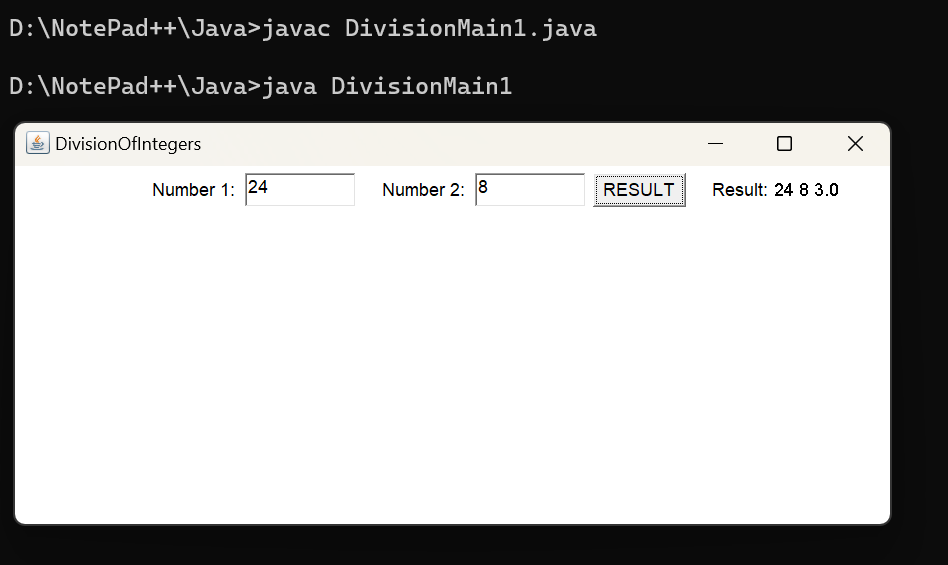
public static void main(String[] args)

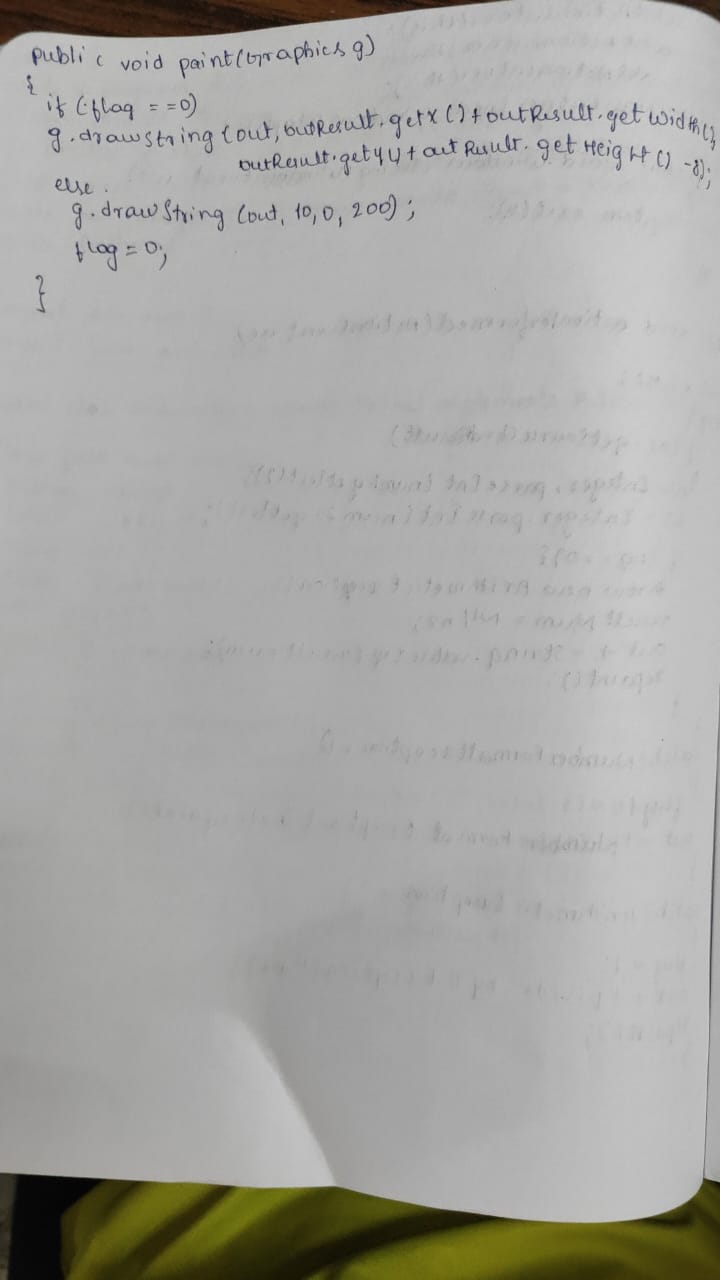
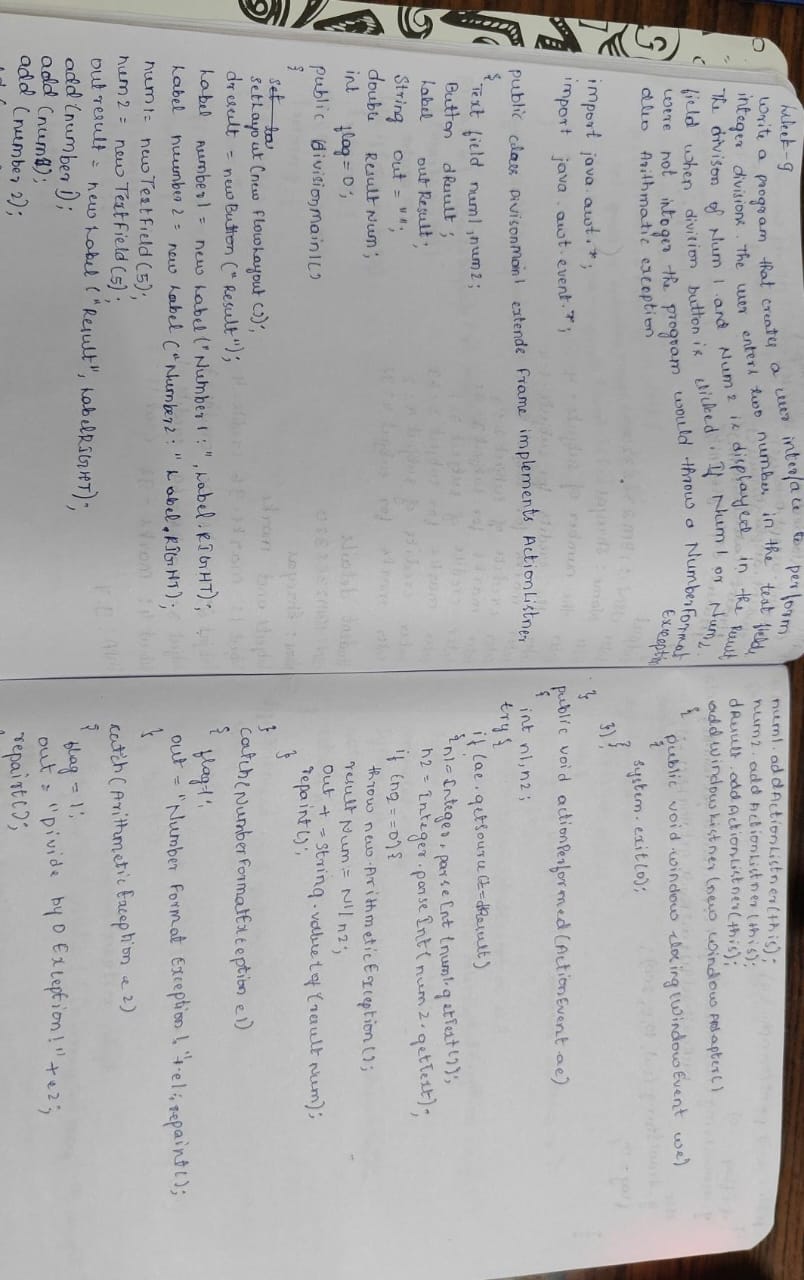
{

DivisionMain1 dm=new DivisionMain1(); dm.setSize(new Dimension(800,400)); dm.setTitle("DivisionOfIntegers"); dm.setVisible(true);

}

**OUTPUT**

****



Demonstrate Interprocess communication and deadlock

**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.");**

**}**

**}**

**ii. Demonstration of deadlock**

**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 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); // get lock on a in this thread.**

**System.out.println("Back in main thread");**

**}**

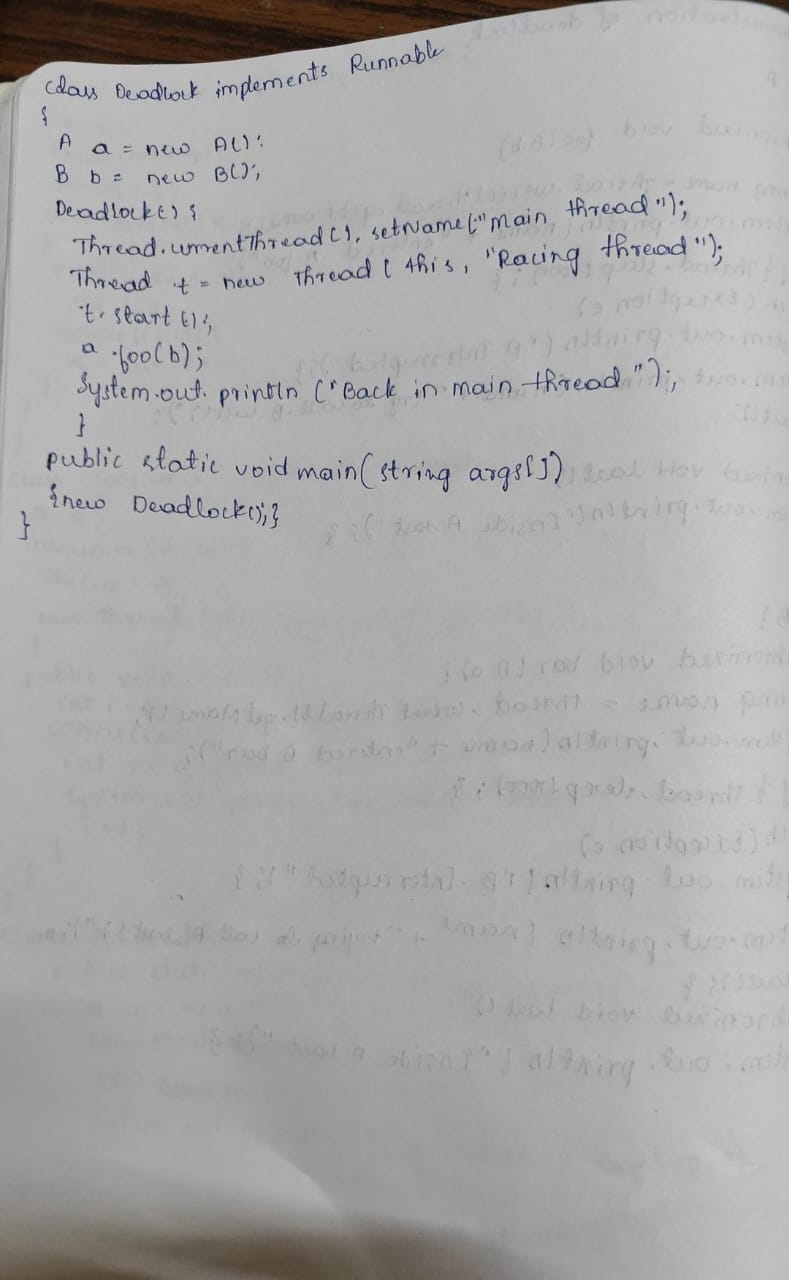
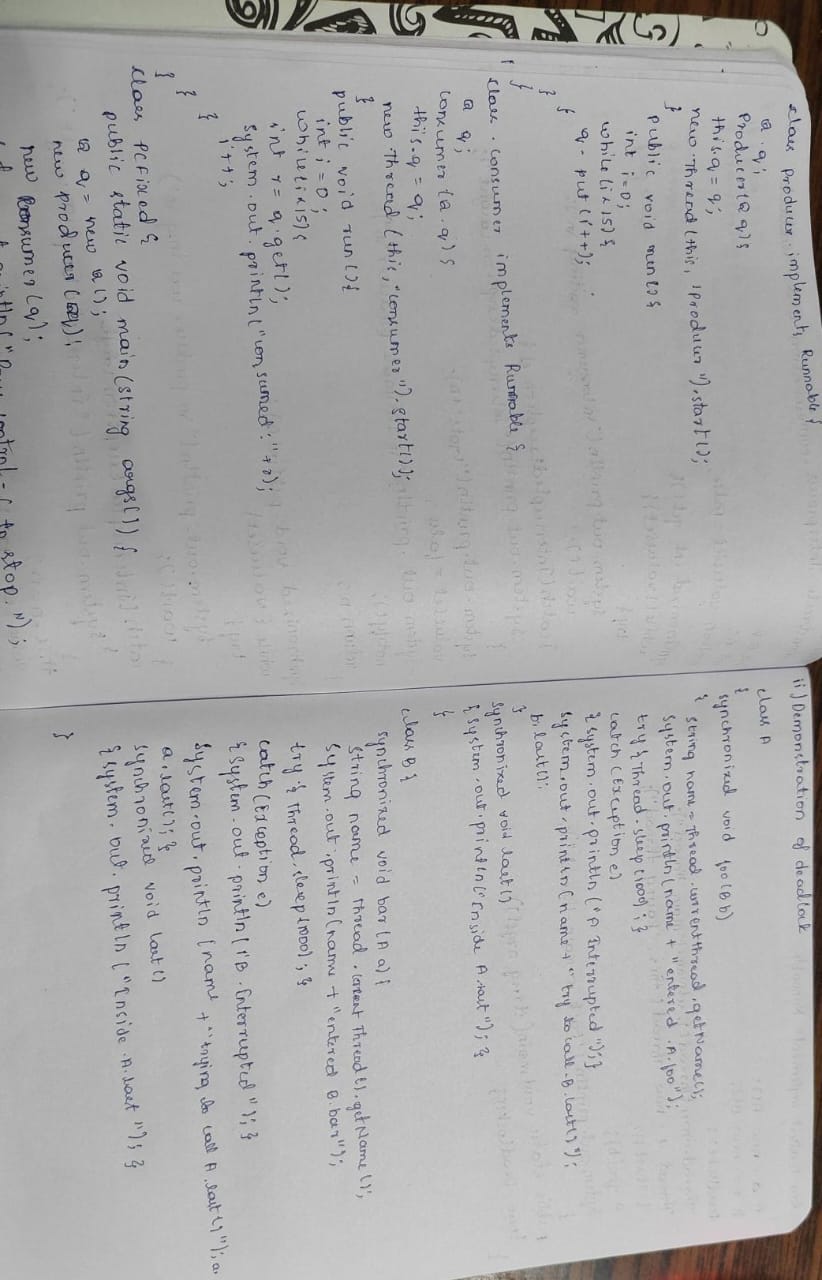
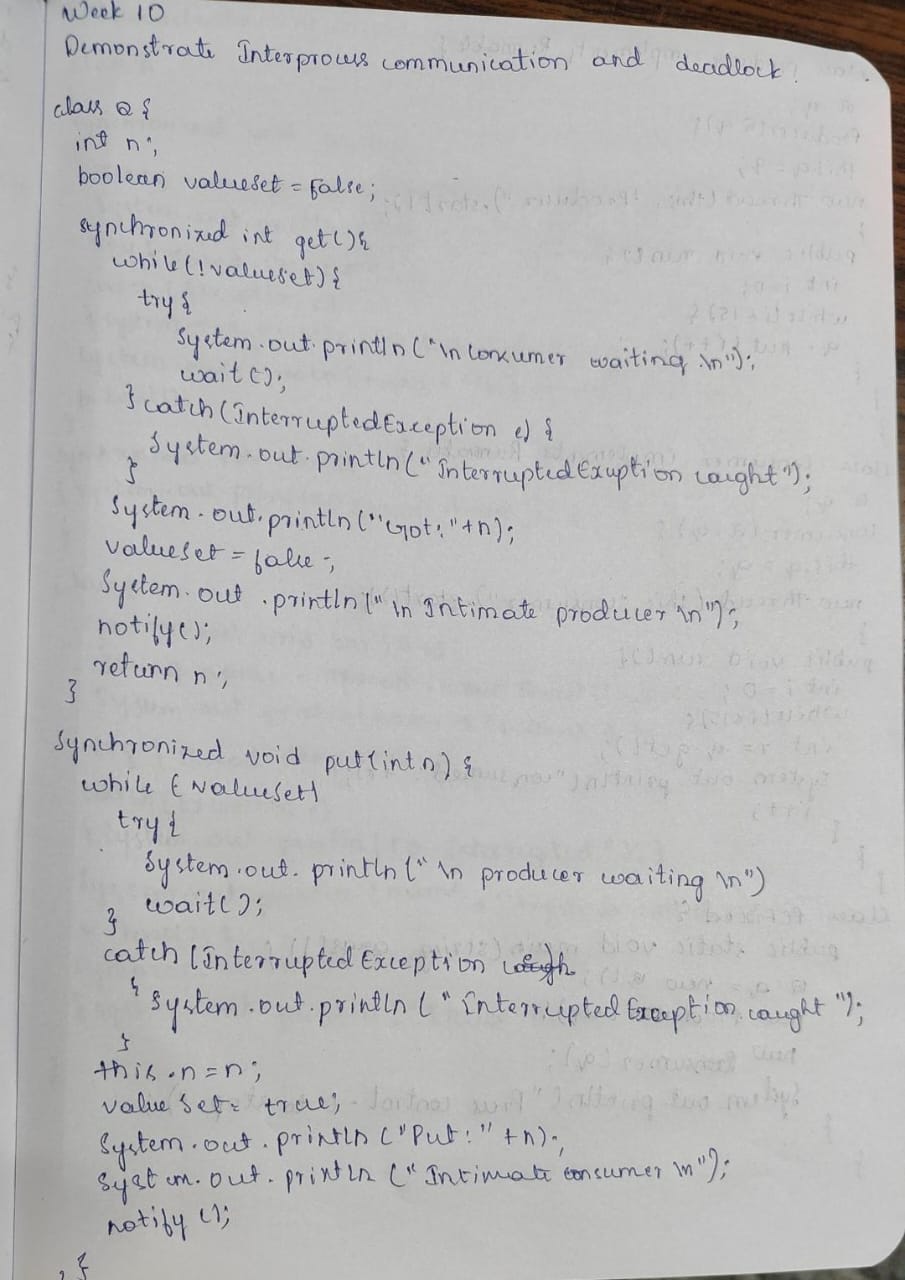
**public void run() { b.bar(a); // get lock on b in other thread.**

**System.out.println("Back in other thread");**

**}**

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

**}**

****