**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**Object Oriented Java Programming (23CS3PCOOJ)**

***Submitted by***

**Khalid Rabbani (1BM23CS150)**

***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 to 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 (23CS3PCOOJ)” carried out by **Khalid Rabbani (1BM23CS150),** 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. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

|  |  |
| --- | --- |
| Lab faculty Incharge Name :  Assistant Professor :  Department of CSE, BMSCE | Dr. Jyothi S Nayak  Professor & HOD  Department of CSE, BMSCE |

**Index**

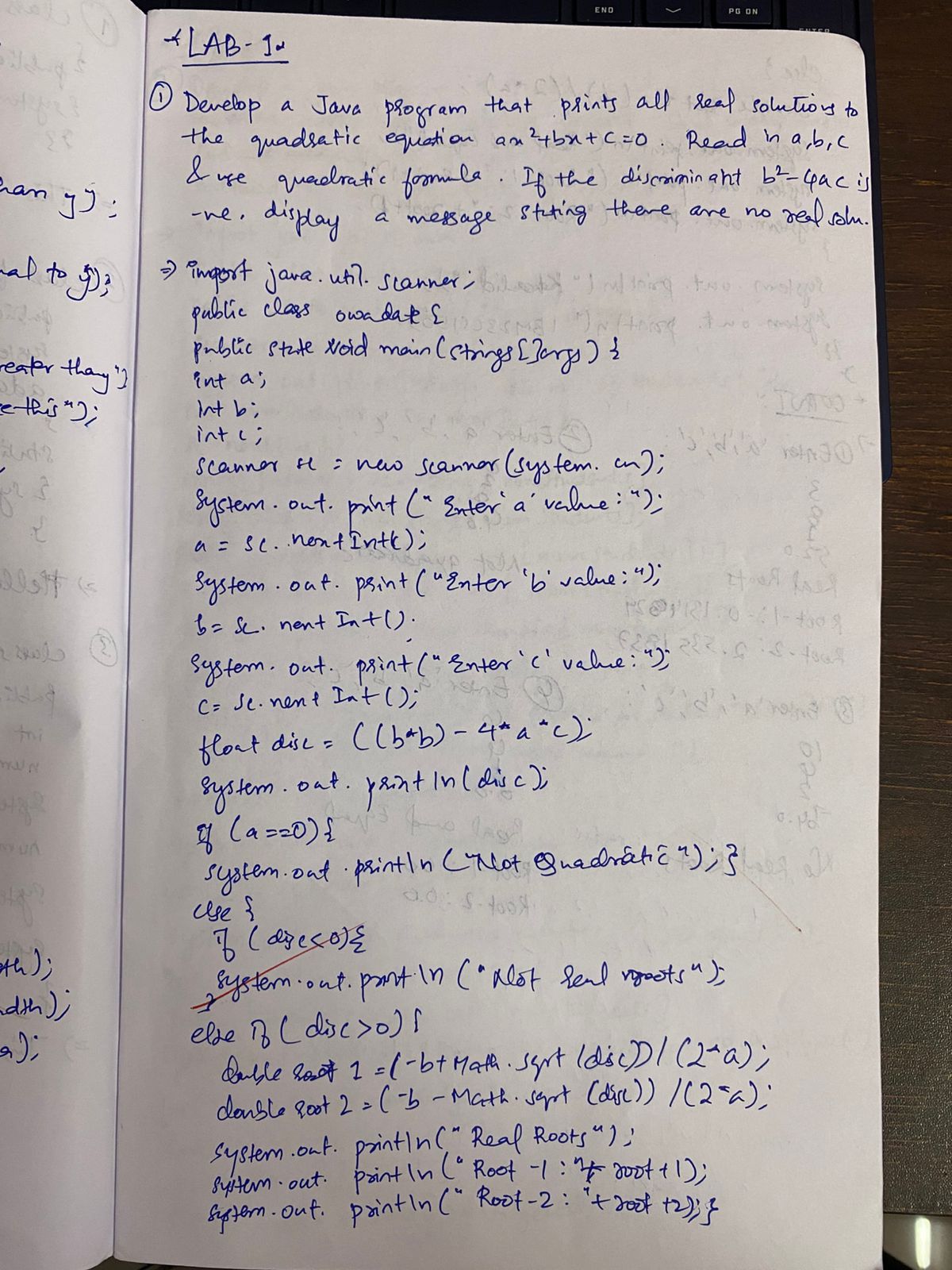
|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 30/09/2024 | Implement Quadratic Equation |  |
| 2 | 07/10/2024 | SGPA Calculation |  |
| 3 | 14/10/2024 | Book Store |  |
| 4 | 21/10/2024 | Area of Shapes |  |
| 5 | 28/10/2024 | Bank - Savings & Current |  |
| 6 | 11/11/2024 | Packages |  |
| 7 | 28/11/2024 | Exceptions |  |
| 8 | 28/11/2024 | Threads |  |
| 9 | 28/11/2024 | Interface - Open End SwingDemo |  |
| 10 | 28/11/2024 | 4 Demonstrate IPC and Deadlock |  |

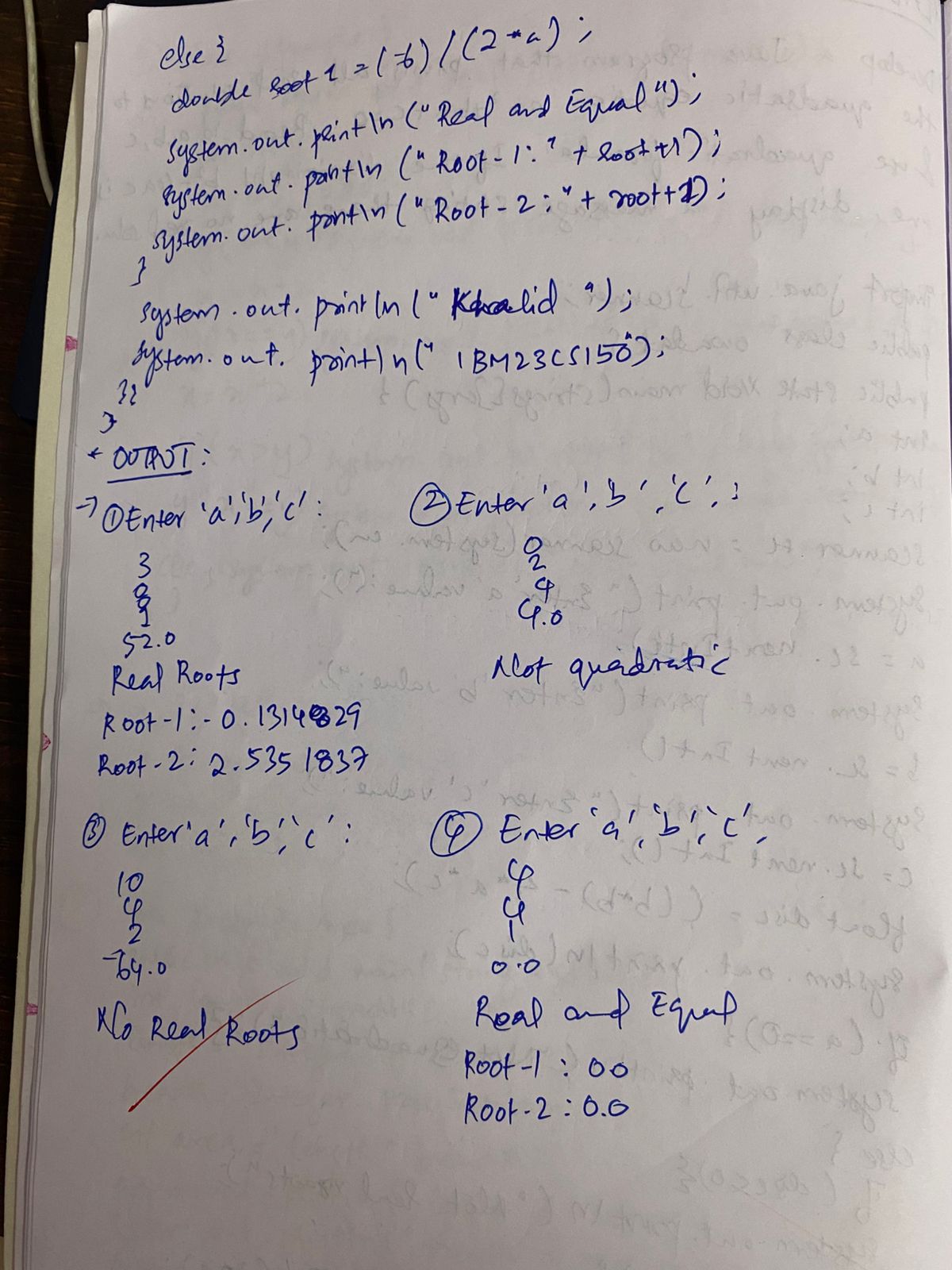
Github Link:

https://github.com/kkhhalliidd/Lab-Programs

**Program 1**

Implement Quadratic Equation





Code:

import java.util.Scanner;

class Main {

public static void main(String[] args) {

class Quadratic {

double a, b, c;

double r1, r2, d;

Quadratic() {

Scanner input = new Scanner(System.in);

System.out.println("This program calculates the roots of a quadratic equation of the form ax^2 + bx + c = 0.");

do {

System.out.print("Enter the value of a (not 0): ");

a = input.nextDouble();

if(a==0) System.out.print("Not a quadratic equation" ) ;

} while (a == 0);

System.out.print("Enter the value of b: ");

b = input.nextDouble();

System.out.print("Enter the value of c: ");

c = input.nextDouble();

d = b\*b-4\*a\*c;

if(d==0) {

r1 = -b/(2\*a);

System.out.println("Roots are real and equal: " + r1);

}else if(d > 0){

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

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

System.out.println("Root1 is " + r1 + ", Root2 is " + r2);

} else {

System.out.println("The roots are imaginary.");

}

}

}

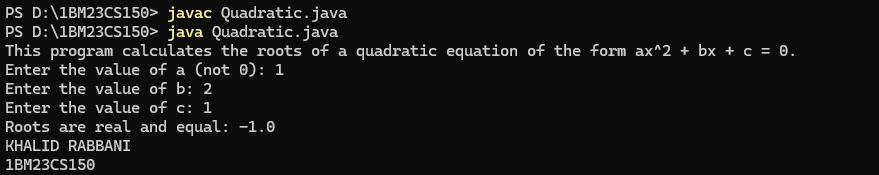
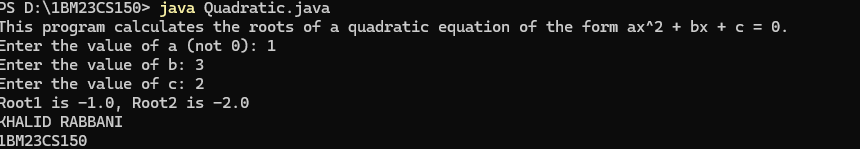
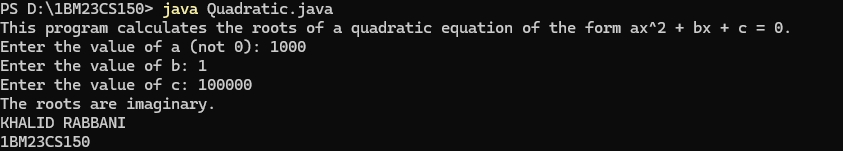
Quadratic q = new Quadratic();

System.out.println("KHALID RABBANI" ) ;

System.out.print("1BM23CS150") ;

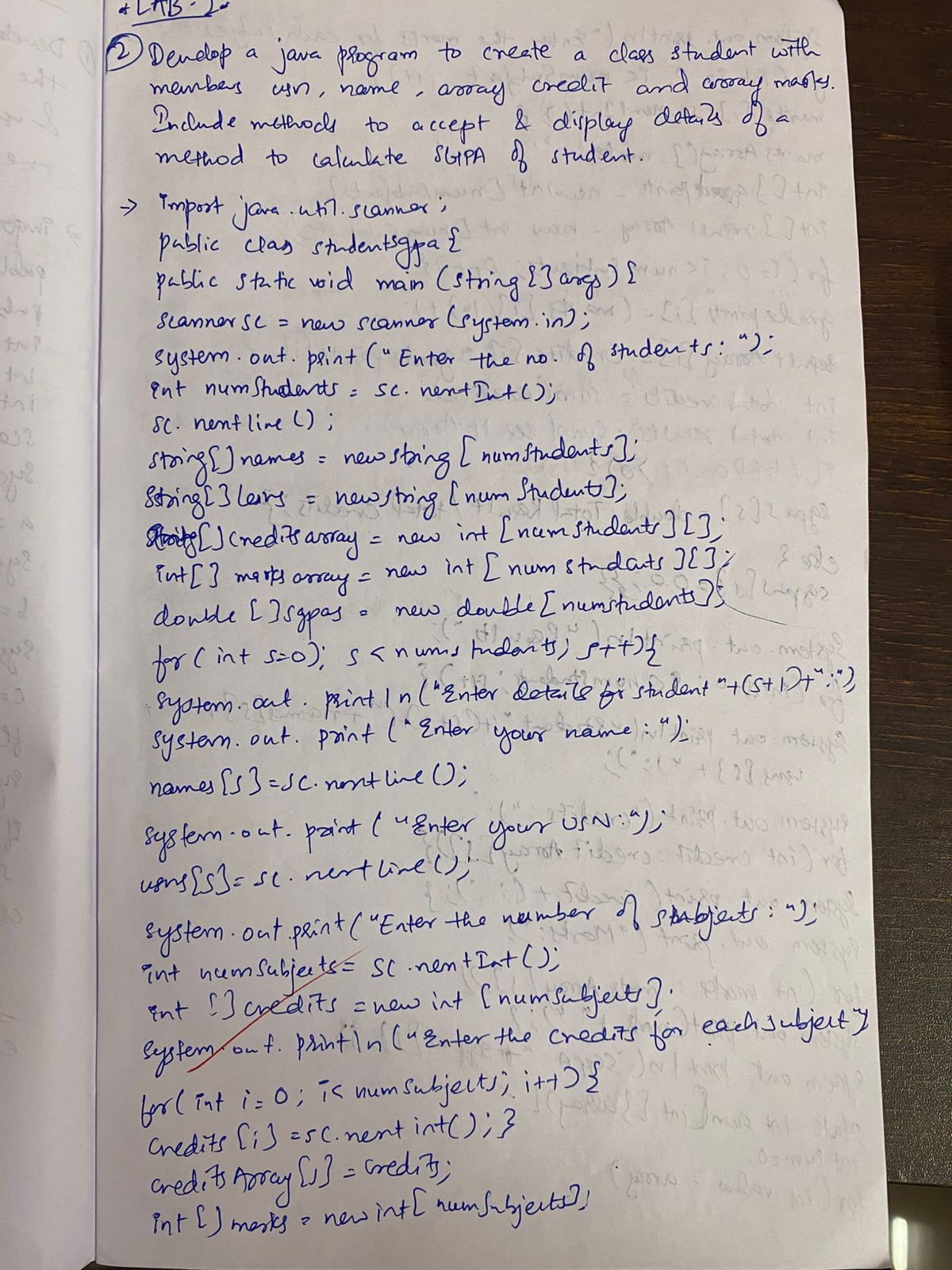
}

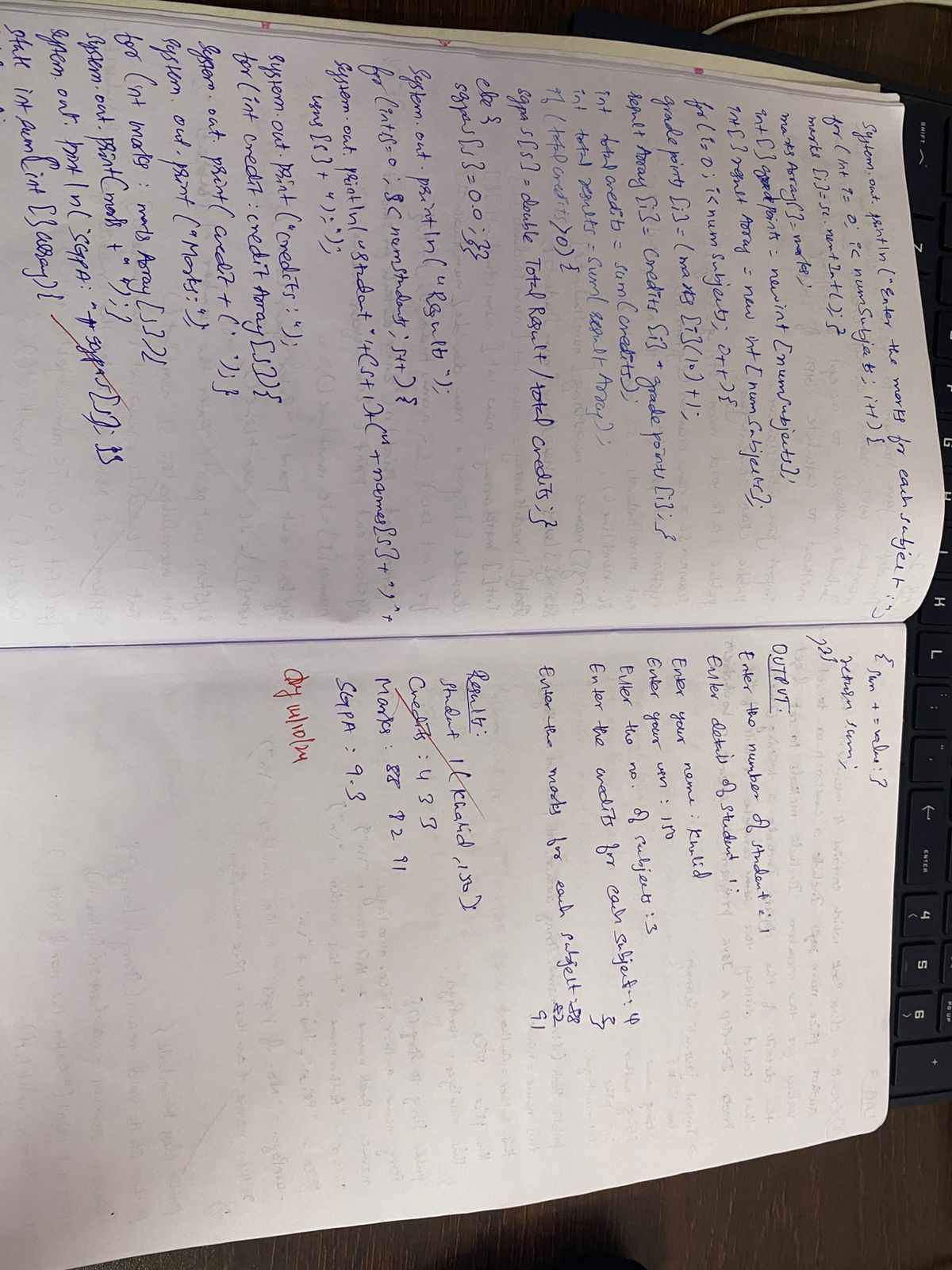
}



**Program 2:**

SGPA Calculator





Code:

import java.util.Scanner;

class Subject {

int subjectMarks;

int credits;

int grade;

public void calculateGrade() {

if (subjectMarks >= 90) {

grade = 10;

} else if (subjectMarks >= 80) {

grade = 9;

} else if (subjectMarks >= 70) {

grade = 8;

} else if (subjectMarks >= 60) {

grade = 7;

} else if (subjectMarks >= 50) {

grade = 6;

} else if (subjectMarks >= 40) {

grade = 5;

} else {

grade = 0;

}

}

}

class Student {

String name;

String usn;

double SGPA;

Subject subject[];

Scanner s;

Student() {

int i;

subject = new Subject[8];

for (i = 0; i < 8; i++) {

subject[i] = new Subject();

}

s = new Scanner(System.in);

}

public void getStudentDetails() {

System.out.print("Enter student name: ");

name = s.nextLine();

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

usn = s.nextLine();

}

public void getMarks() {

for (int i = 0; i < 8; i++) {

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

subject[i].subjectMarks = s.nextInt();

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

subject[i].credits = s.nextInt();

subject[i].calculateGrade();

if (subject[i].subjectMarks > 100) {

System.out.println("Invalid marks, should not exceed 100.");

subject[i].subjectMarks = 0;

} else if (subject[i].subjectMarks < 0) {

System.out.println("Invalid marks, should not be negative.");

subject[i].subjectMarks = 0;

}

}

}

public void computeSGPA() {

double totalGradePoints = 0;

int totalCredits = 0;

for (Subject subj : subject) {

totalGradePoints += (subj.grade \* subj.credits);

totalCredits += subj.credits;

}

if (totalCredits > 0) {

SGPA = totalGradePoints / totalCredits;

} else {

SGPA = 0.0;

}

}

public void displayResults() {

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

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

System.out.printf("SGPA: %.2f%n", SGPA);

}

}

public class Main {

public static void main(String[] args) {

Student s1 = new Student();

s1.getStudentDetails();

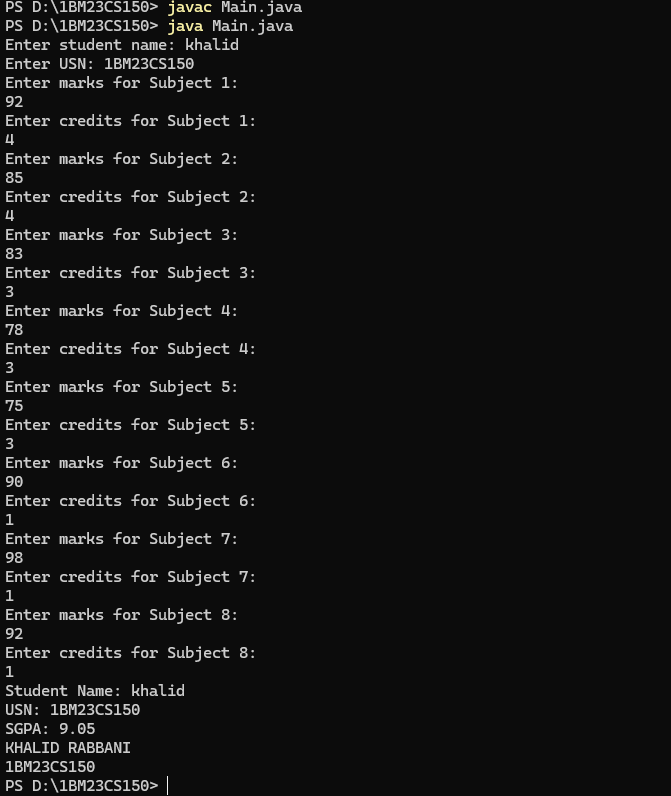
s1.getMarks();

s1.computeSGPA();

s1.displayResults();

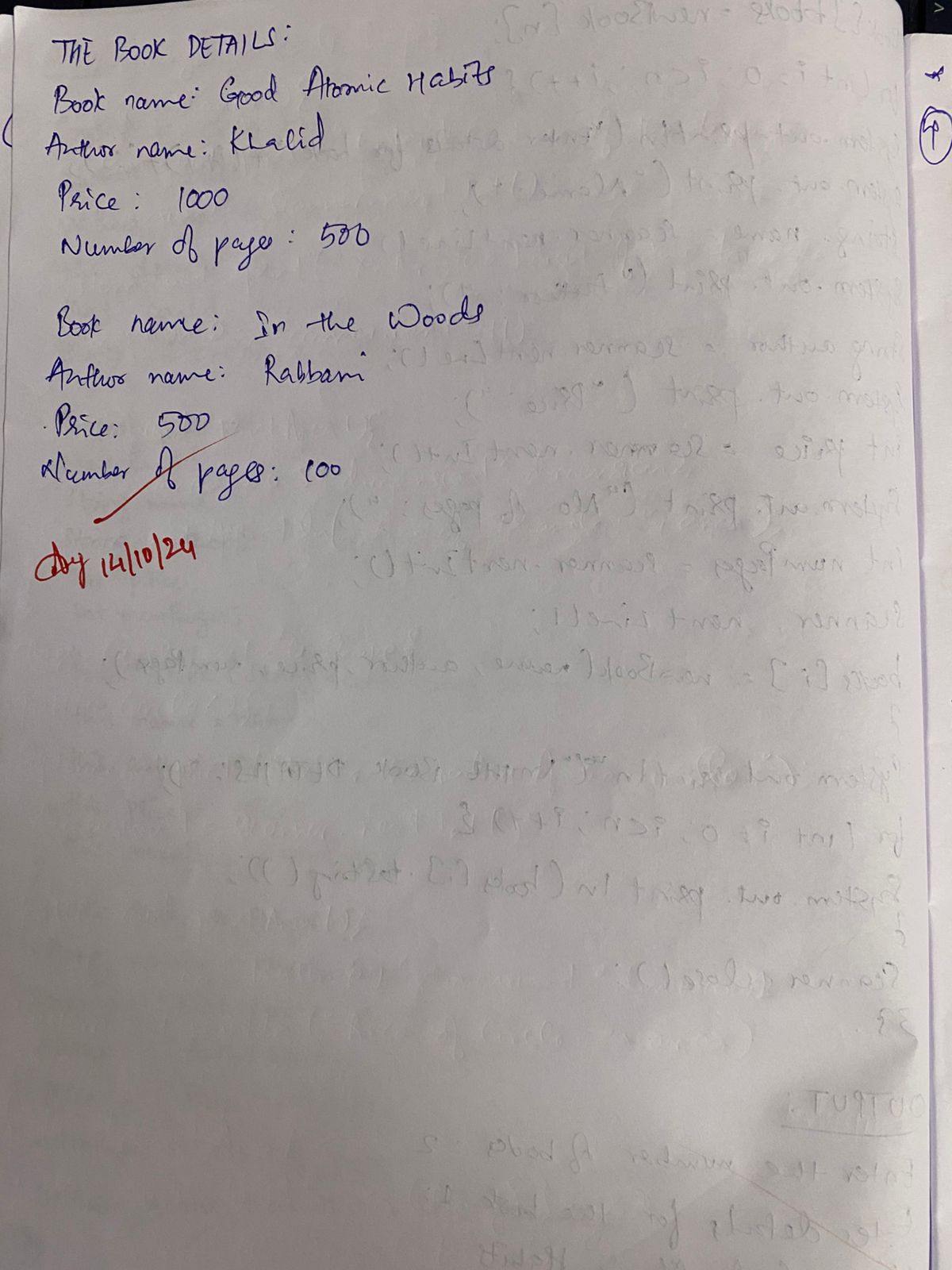
}

}



**Program 3:**

Displaying the details of book using tostring method



Code:

import java.util.Scanner;

class Book {

String name;

String author;

int price;

int numPages;

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

this.name = name;

this.author = author;

this.price = price;

this.numPages = numPages;

}

public String toString() {

String name, author, price, numPages;

name = "Book name: " + this.name + "\n";

author = "Author name: " + this.author + "\n";

price = "Price: " + this.price + "\n";

numPages = "Number of pages: " + this.numPages + "\n";

return name + author + price + numPages;

}}

public class Main\_book {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

scanner.nextLine();

Book[] books = new Book[n];

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

System.out.println("Enter details for the book " + (i + 1) + ":");

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

String name = scanner.nextLine();

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

String author = scanner.nextLine();

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

int price = scanner.nextInt();

System.out.print("Number of pages: ");

int numPages = scanner.nextInt();

scanner.nextLine();

books[i] = new Book(name, author, price, numPages);

}

System.out.println("\nTHE BOOK DETAILS:");

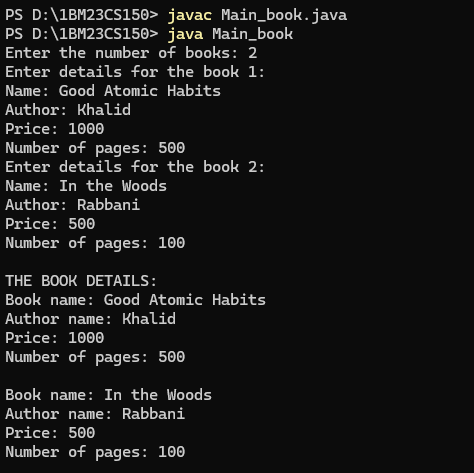
for (int i = 0; i < n; i++) {

System.out.println(books[i].toString());

}

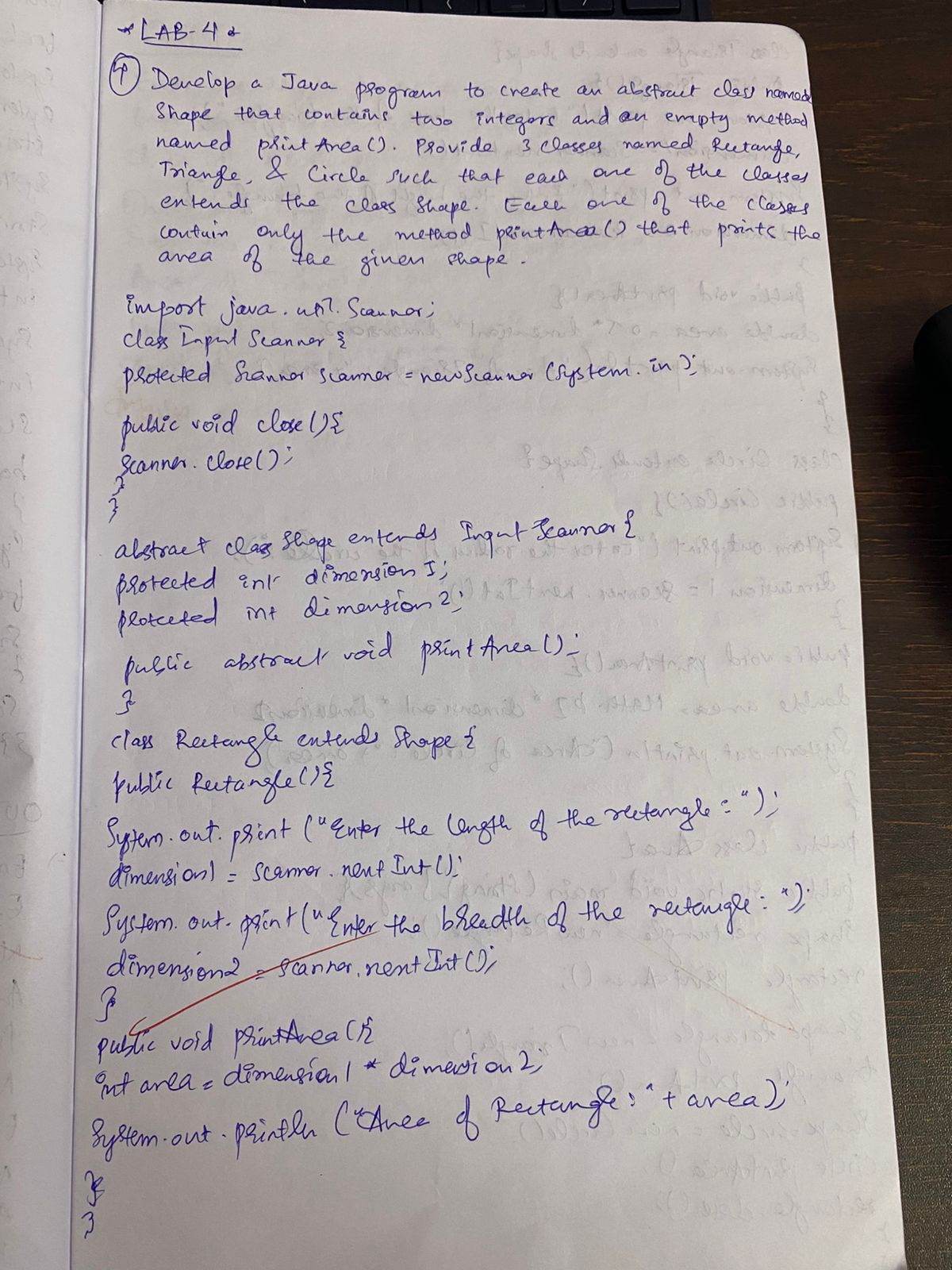
scanner.close();

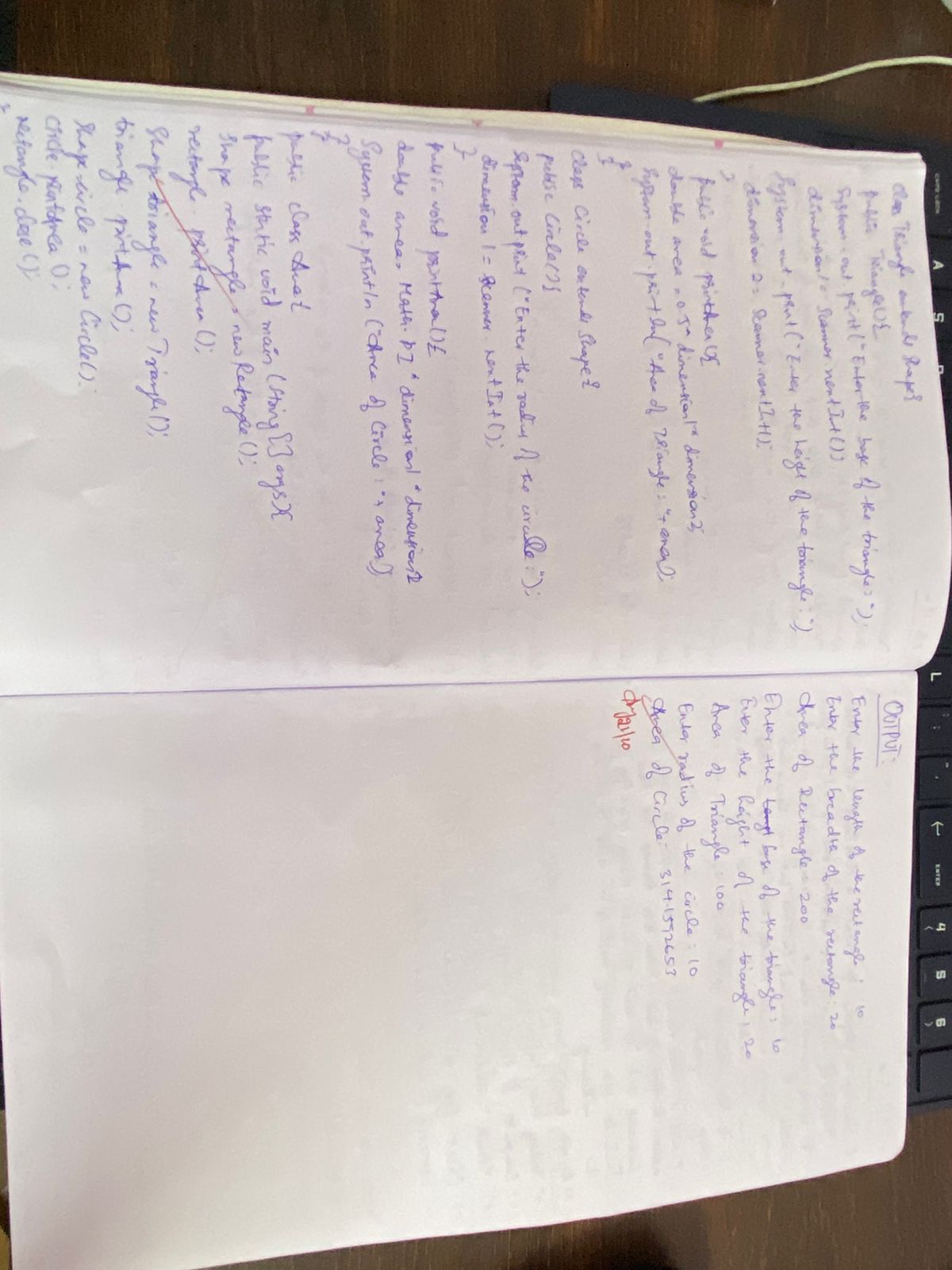
}}



**Program 4:**

Abstract Class





Code:

import java.util.Scanner;

class InputScanner {

protected Scanner scanner = new Scanner(System.in);

public void close() {

scanner.close();

}

}

abstract class Shape extends InputScanner {

protected int dimension1;

protected int dimension2;

public abstract void printArea();

}

class Rectangle extends Shape {

public Rectangle() {

System.out.print("Enter the length of the rectangle: ");

dimension1 = scanner.nextInt();

System.out.print("Enter the breadth of the rectangle: ");

dimension2 = scanner.nextInt();

}

public void printArea() {

int area = dimension1 \* dimension2;

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

}

}

class Triangle extends Shape {

public Triangle() {

System.out.print("Enter the base of the triangle: ");

dimension1 = scanner.nextInt();

System.out.print("Enter the height of the triangle: ");

dimension2 = scanner.nextInt();

}

public void printArea() {

double area = 0.5 \* dimension1 \* dimension2;

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

}

}

class Circle extends Shape {

public Circle() {

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

dimension1 = scanner.nextInt();

}

public void printArea() {

double area = Math.PI \* dimension1 \* dimension1;

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

}

}

public class Area {

public static void main(String[] args) {

Shape rectangle = new Rectangle();

rectangle.printArea();

Shape triangle = new Triangle();

triangle.printArea();

Shape circle = new Circle();

circle.printArea();

rectangle.close();

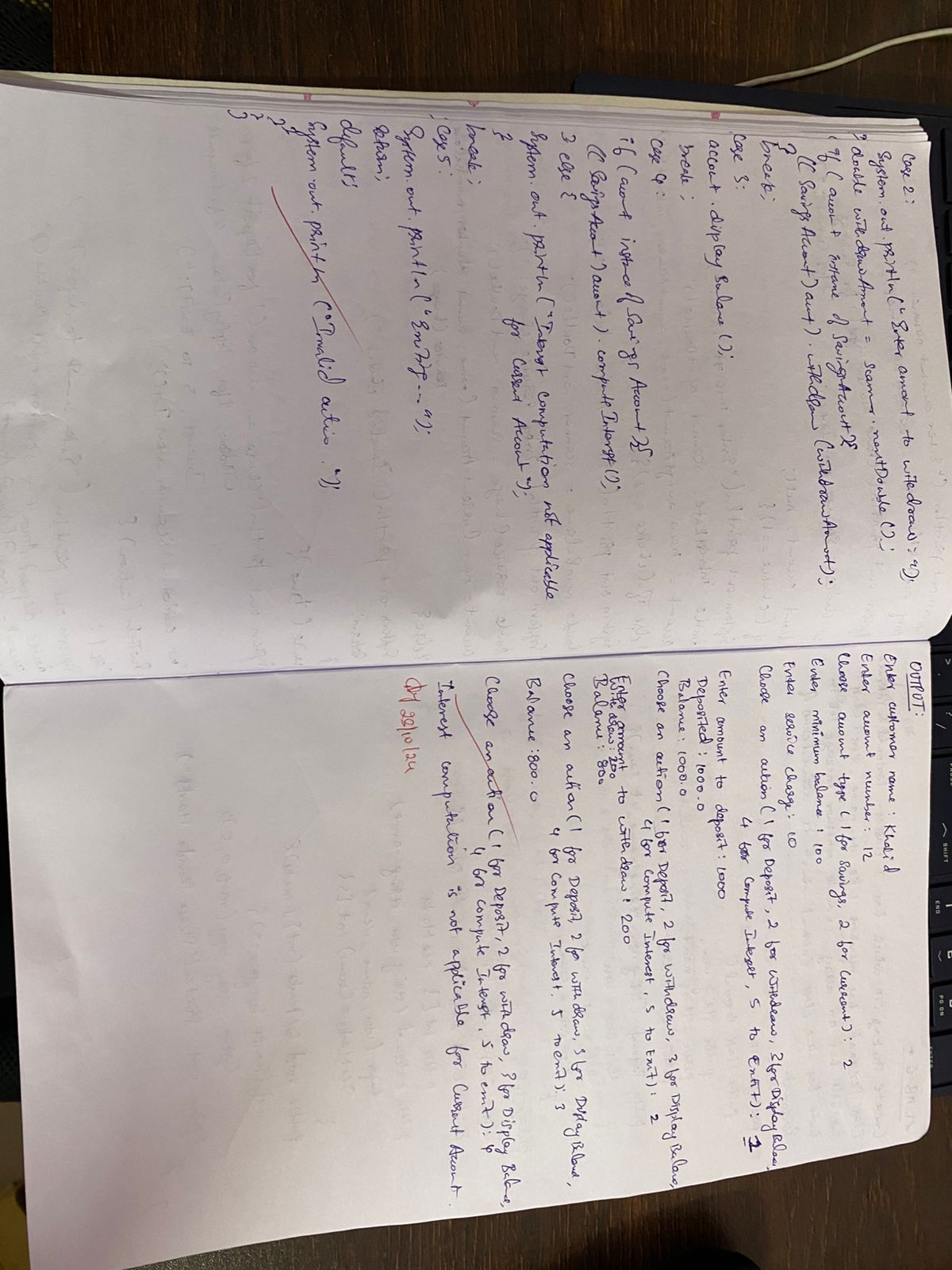
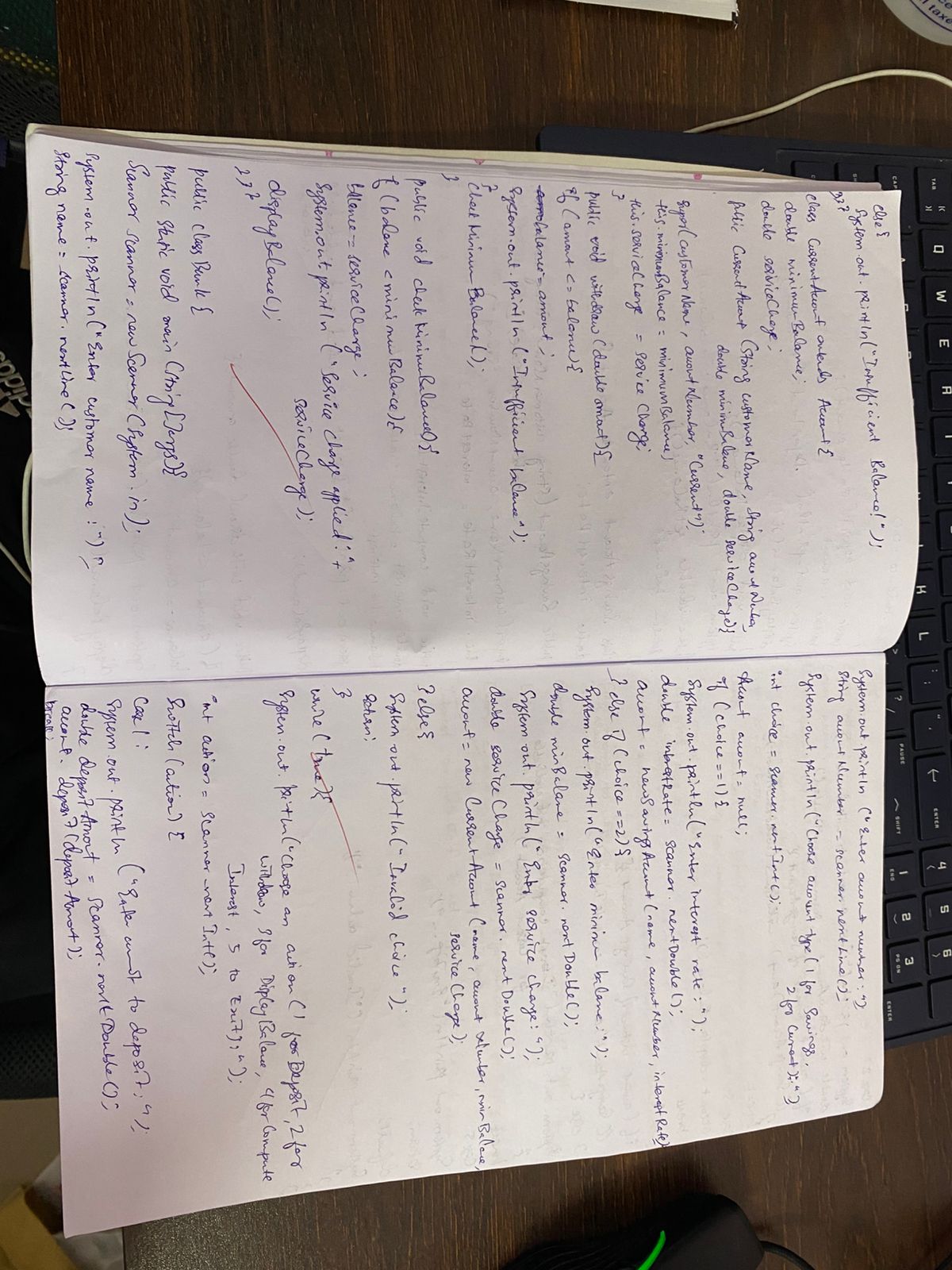
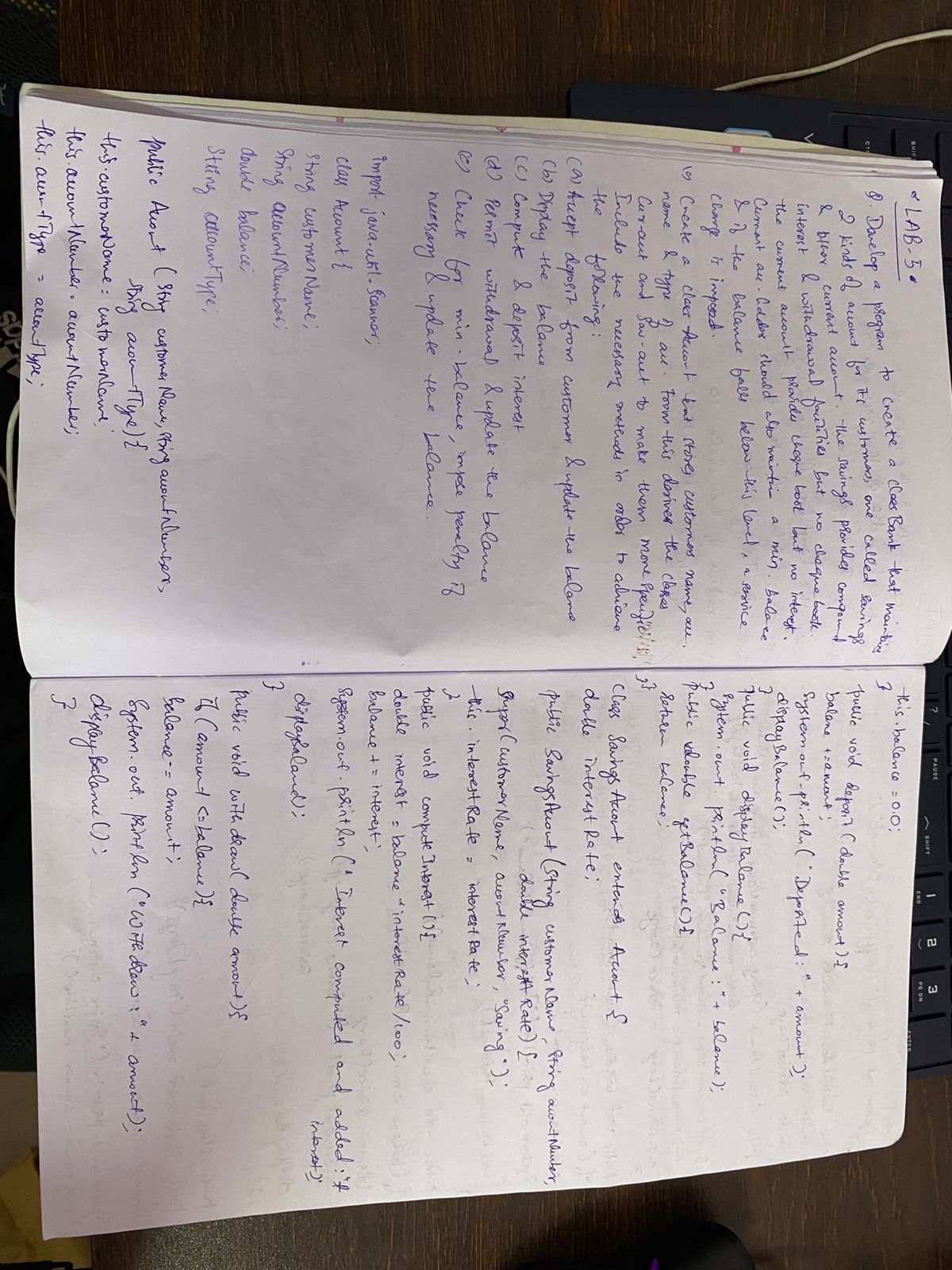
}

}



**Program 5:**

Bank



Code:

import java.util.Scanner;

class Account {

String customerName;

String accountNumber;

double balance;

String accountType;

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

this.customerName = customerName;

this.accountNumber = accountNumber;

this.accountType = accountType;

this.balance = 0.0;

}

public void deposit(double amount) {

balance += amount;

System.out.println("Deposited: " + amount);

displayBalance();

}

public void displayBalance() {

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

}

public double getBalance() {

return balance;

}

}

class SavingsAccount extends Account {

double interestRate;

public SavingsAccount(String customerName, String accountNumber, double interestRate) {

super(customerName, accountNumber, "Savings");

this.interestRate = interestRate;

}

public void computeInterest() {

double interest = balance \* interestRate / 100;

balance += interest;

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

displayBalance();

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrew: " + amount);

displayBalance();

} else {

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

}

}

}

class CurrentAccount extends Account {

double minimumBalance;

double serviceCharge;

public CurrentAccount(String customerName, String 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("Withdrew: " + amount);

displayBalance();

} else {

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

}

checkMinimumBalance();

}

public void checkMinimumBalance() {

if (balance < minimumBalance) {

balance -= serviceCharge;

System.out.println("Service charge applied: " + serviceCharge);

displayBalance();

}

}

}

public class Bank {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String name = scanner.nextLine();

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

String accountNumber = scanner.nextLine();

System.out.println("Choose account type (1 for Savings, 2 for Current):");

int choice = scanner.nextInt();

Account account = null;

if (choice == 1) {

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

double interestRate = scanner.nextDouble();

account = new SavingsAccount(name, accountNumber, interestRate);

} else if (choice == 2) {

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

double minBalance = scanner.nextDouble();

System.out.println("Enter service charge:");

double serviceCharge = scanner.nextDouble();

account = new CurrentAccount(name, accountNumber, minBalance, serviceCharge);

} else {

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

return;

}

while (true) {

System.out.println("Choose an action (1 for Deposit, 2 for Withdraw, 3 for Display Balance, 4 for Compute Interest, 5 to Exit):");

int action = scanner.nextInt();

switch (action) {

case 1:

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

double depositAmount = scanner.nextDouble();

account.deposit(depositAmount);

break;

case 2:

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

double withdrawAmount = scanner.nextDouble();

if (account instanceof SavingsAccount) {

((SavingsAccount) account).withdraw(withdrawAmount);

} else if (account instanceof CurrentAccount) {

((CurrentAccount) account).withdraw(withdrawAmount);

}

break;

case 3:

account.displayBalance();

break;

case 4:

if (account instanceof SavingsAccount) {

((SavingsAccount) account).computeInterest();

} else {

System.out.println("Interest computation is not applicable for Current Account.");

}

break;

case 5:

System.out.println("Exiting...");

return;

default:

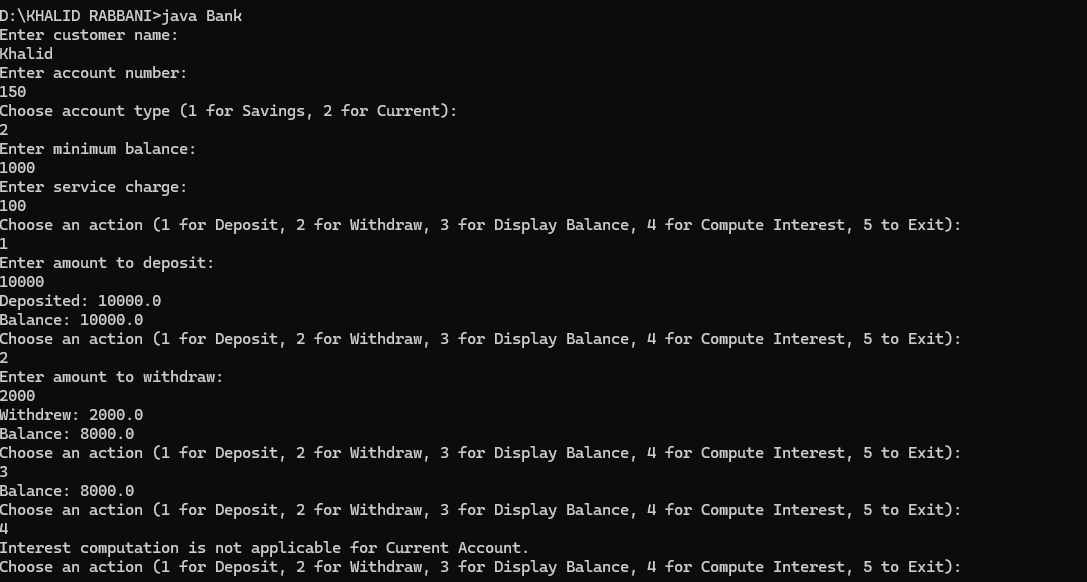
System.out.println("Invalid action.");

}

}

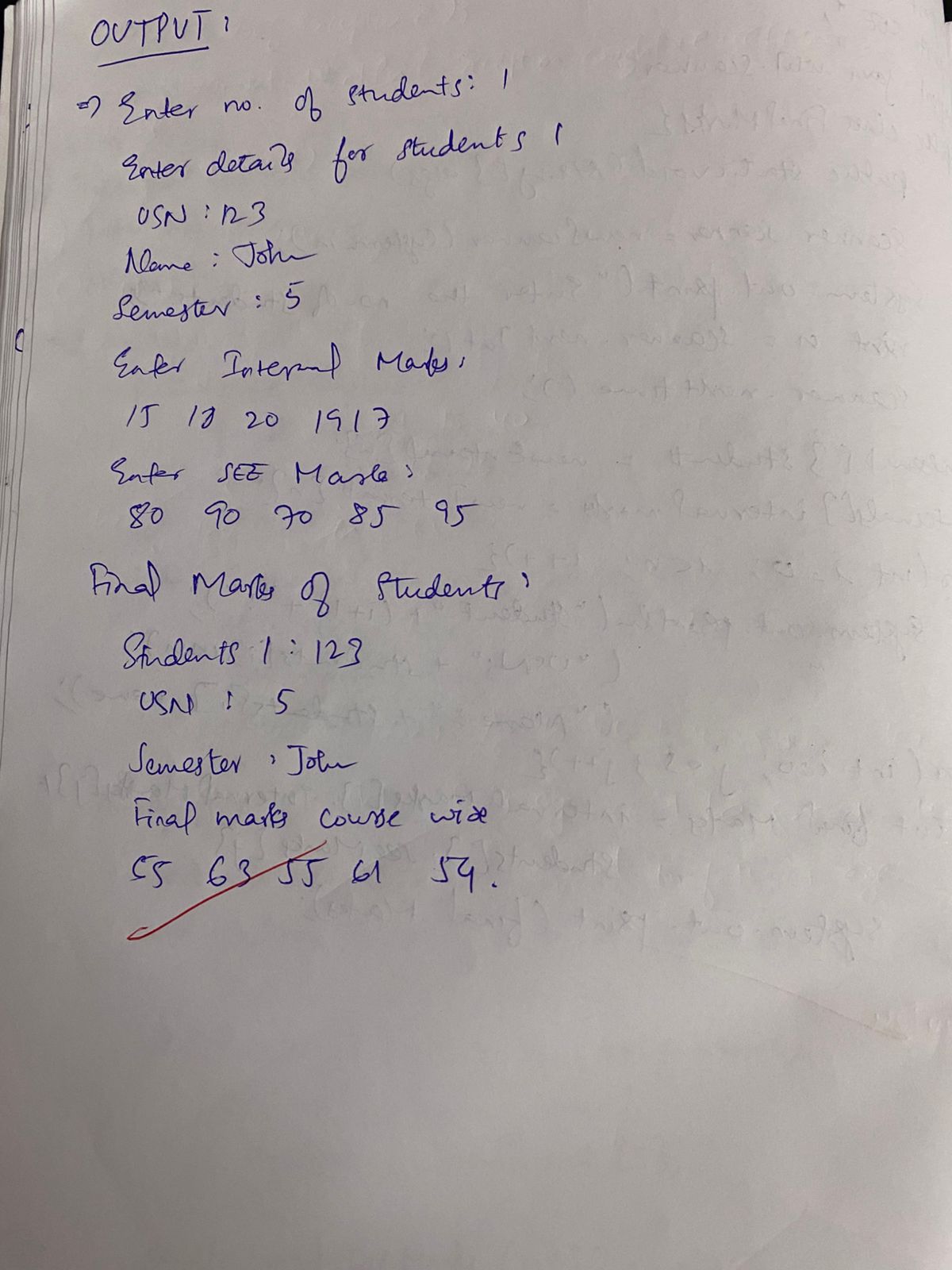
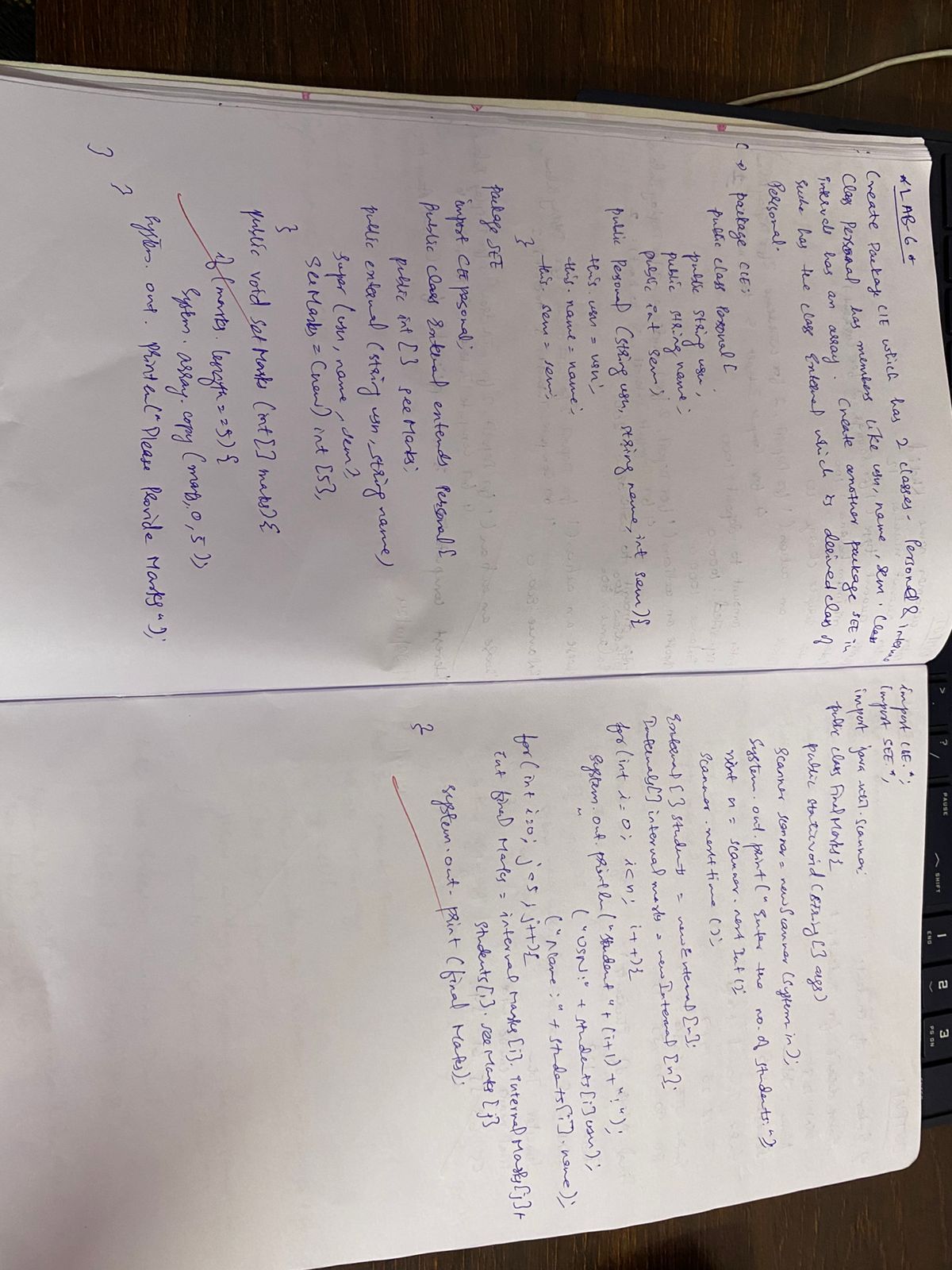
}

}



**Program 6:**

Package CIE and SEE



Code:

CIE (Student):

package CIE;

import SEE.Externals;

import java.util.Scanner;

public class Student {

public String usn;

public String name;

public int sem;

public void inputStudentDetails() {

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 Semester: ");

sem = sc.nextInt();

}

public void displayStudentDetails() {

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

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

System.out.println("Semester: " + sem);

}

public void calcFinalMarks(Internals cie, Externals see) {

displayStudentDetails();

System.out.println("Final Marks for 5 subjects:");

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

int finalMarks = cie.cieMarks[i] + (see.seeMarks[i] / 2);

System.out.println("Subject " + (i + 1) + ": CIE = " + cie.cieMarks[i] +

", SEE = " + see.seeMarks[i] +

", Final = " + finalMarks);

}

}

}

CIE (Internals)

package CIE;

import java.util.Scanner;

public class Internals extends Student {

public int[] cieMarks = new int[5];

public void inputCIEmarks() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter CIE marks for 5 subjects:");

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

System.out.print("Subject " + (i + 1) + ": ");

cieMarks[i] = sc.nextInt();

}

}

}

SEE

package SEE;

import CIE.Student;

import java.util.Scanner;

public class Externals extends Student {

public int[] seeMarks = new int[5];

public void inputSEEmarks() {

Scanner sc = new Scanner(System.in);

System.out.println("Enter SEE marks for 5 subjects:");

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

System.out.print("Subject " + (i + 1) + ": ");

seeMarks[i] = sc.nextInt();

}

}

}

MAIN

import CIE.Internals;

import SEE.Externals;

import CIE.Student;

import java.util.Scanner;

class PrintInfo {

static void print() {

System.out.println("Name: Khalid Rabbani");

System.out.println("USN: 1BM23CS150");

}

}

public class Main {

public static void main(String[] args) {

PrintInfo.print();

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

Internals[] cieStudents = new Internals[n];

Externals[] seeStudents = new Externals[n];

Student[] students = new Student[n];

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

System.out.println("\nEnter details for Student " + (i + 1) + ":");

cieStudents[i] = new Internals();

seeStudents[i] = new Externals();

students[i] = new Student();

students[i].inputStudentDetails();

cieStudents[i].inputCIEmarks();

seeStudents[i].inputSEEmarks();

}

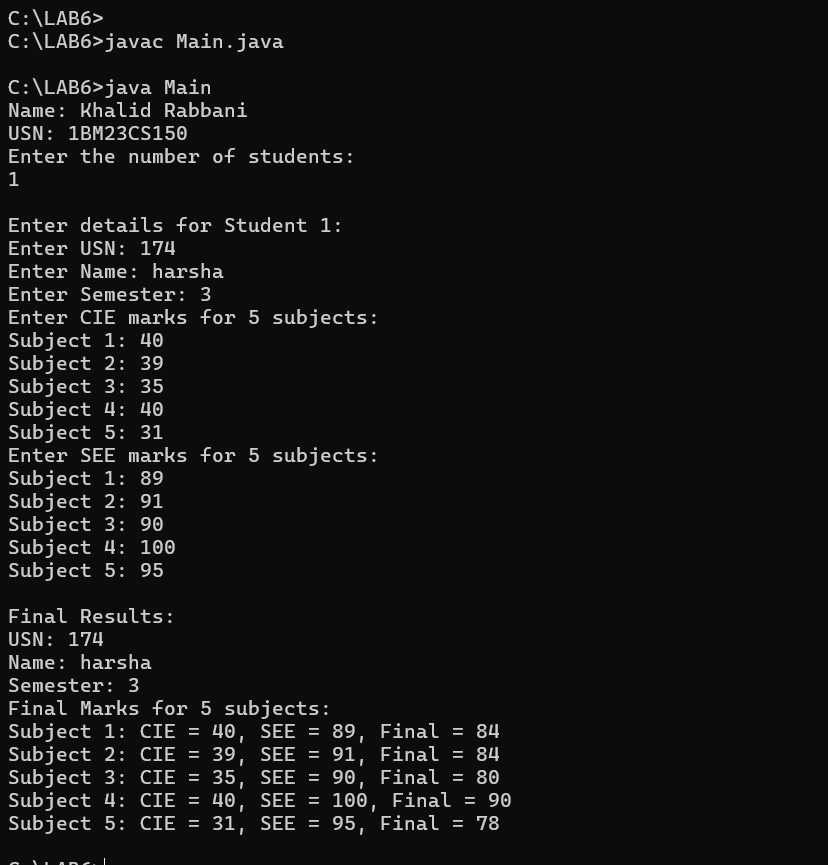
System.out.println("\nFinal Results:");

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

students[i].calcFinalMarks(cieStudents[i], seeStudents[i]);

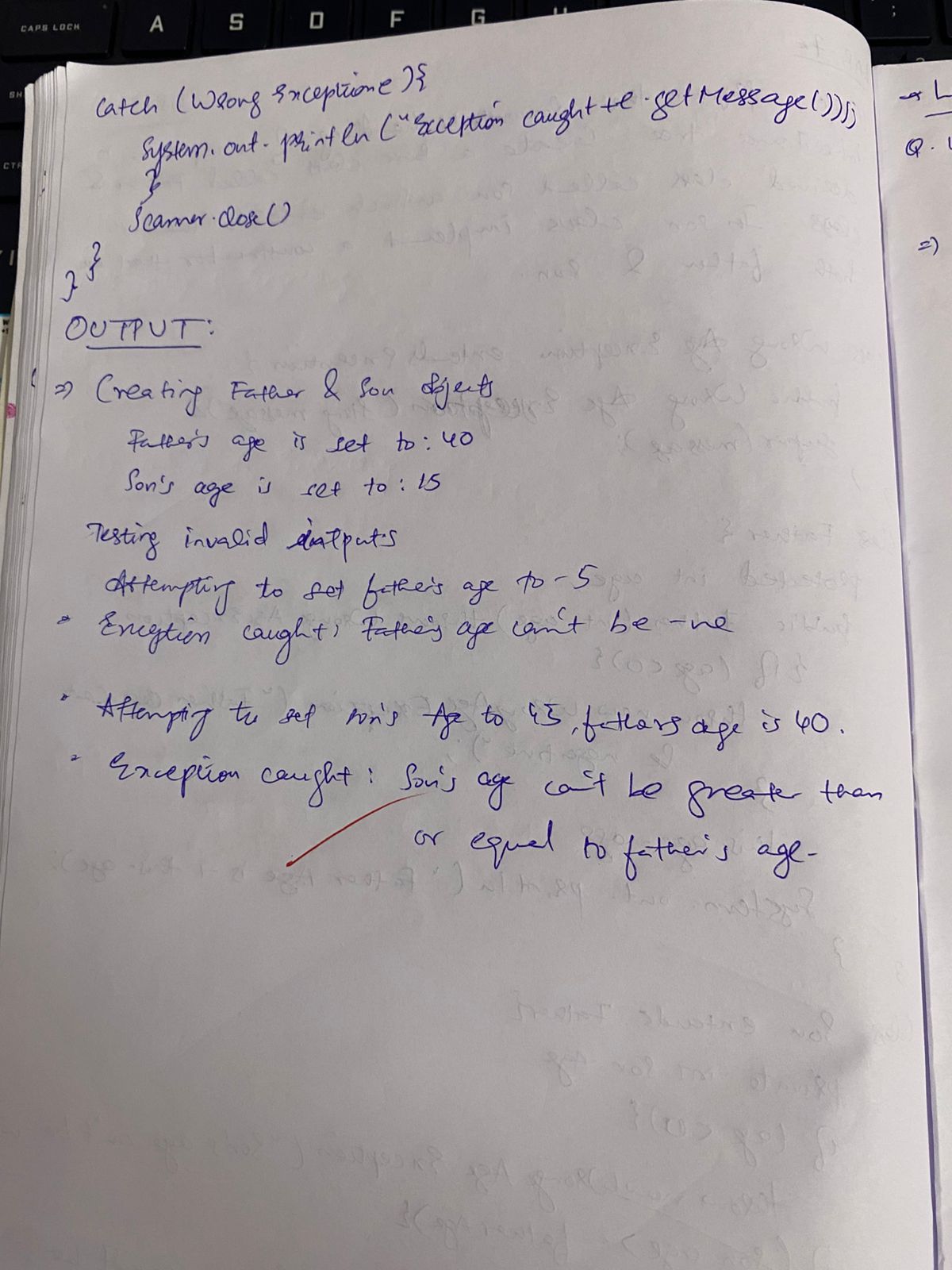
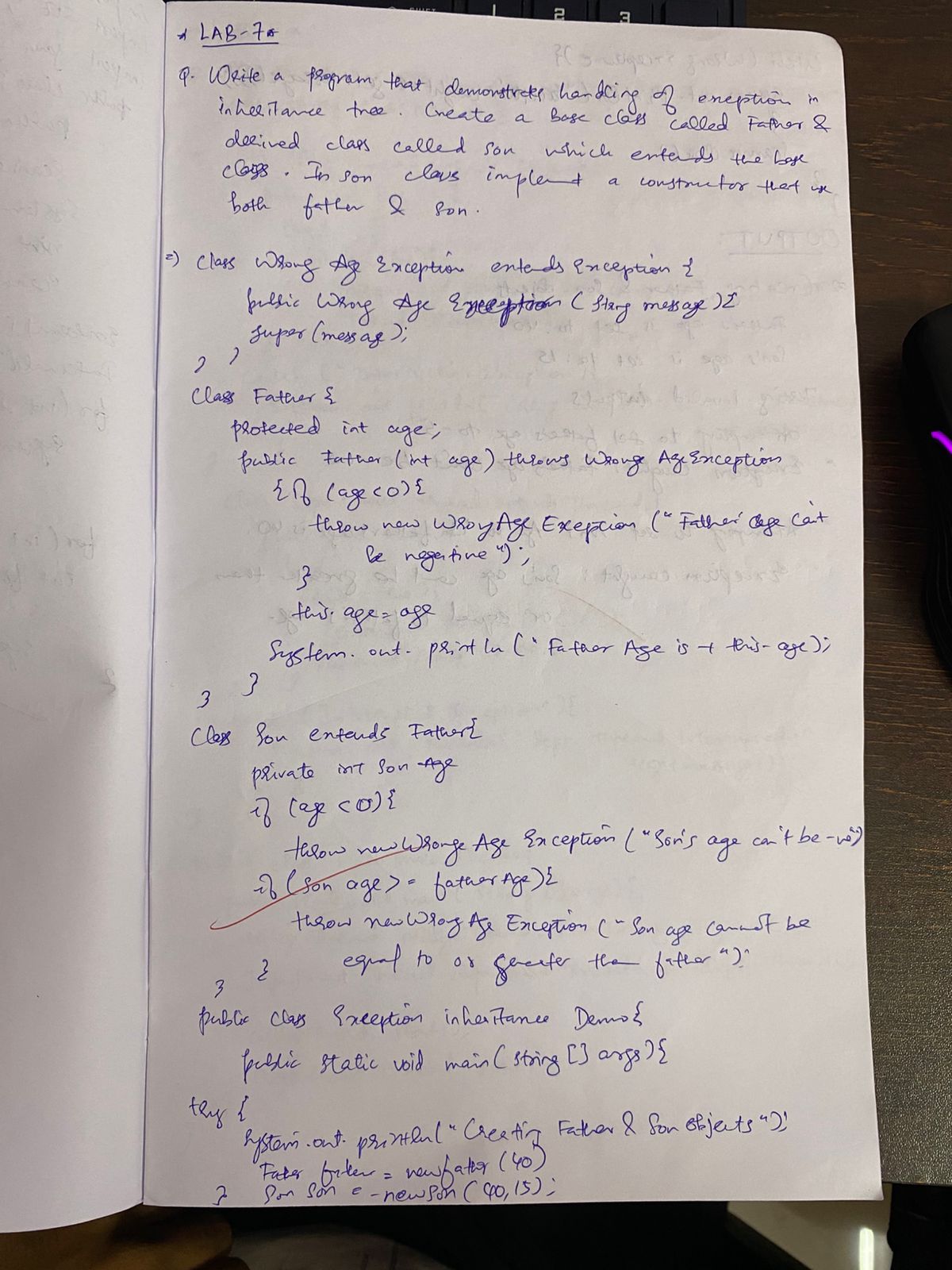
}

}

}

**Program 7:**

Demonstrates handling of exceptions in inheritance tree.



Code:

import java.util.Scanner;

class WrongAge extends Exception {

public WrongAge() {

super("Age Error");

}

public WrongAge(String message) {

super(message);

}

}

class InputScanner {

Scanner s = new Scanner(System.in);

public int getInput() {

return s.nextInt();

}

}

class Father extends InputScanner {

protected int fatherAge;

public Father() throws WrongAge {

System.out.print("Enter father's age: ");

fatherAge = getInput();

if (fatherAge < 0) {

throw new WrongAge("Age cannot be negative");

}

}

public void displayFatherAge() {

System.out.println("Father's age: " + fatherAge);

}

}

class Son extends Father {

private int sonAge;

public Son() throws WrongAge {

super(); // Call Father's constructor

System.out.print("Enter son's age: ");

sonAge = getInput(); // Read son's age

if (sonAge >= fatherAge) {

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

} else if (sonAge < 0) {

throw new WrongAge("Age cannot be negative");

}

}

public void displaySonAge() {

System.out.println("Son's age: " + sonAge);

}

}

public class Main {

public static void main(String[] args) {

try {

Son son = new Son();

son.displayFatherAge();

son.displaySonAge();

} catch (WrongAge e) {

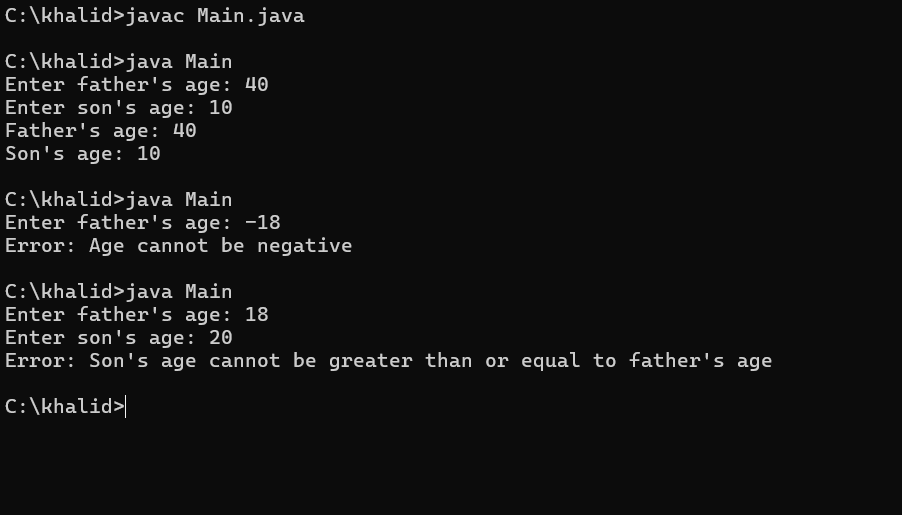
System.out.println("Error: " + e.getMessage());

}

System.out.println("Harsha Vardhan\n1BM23CS136");

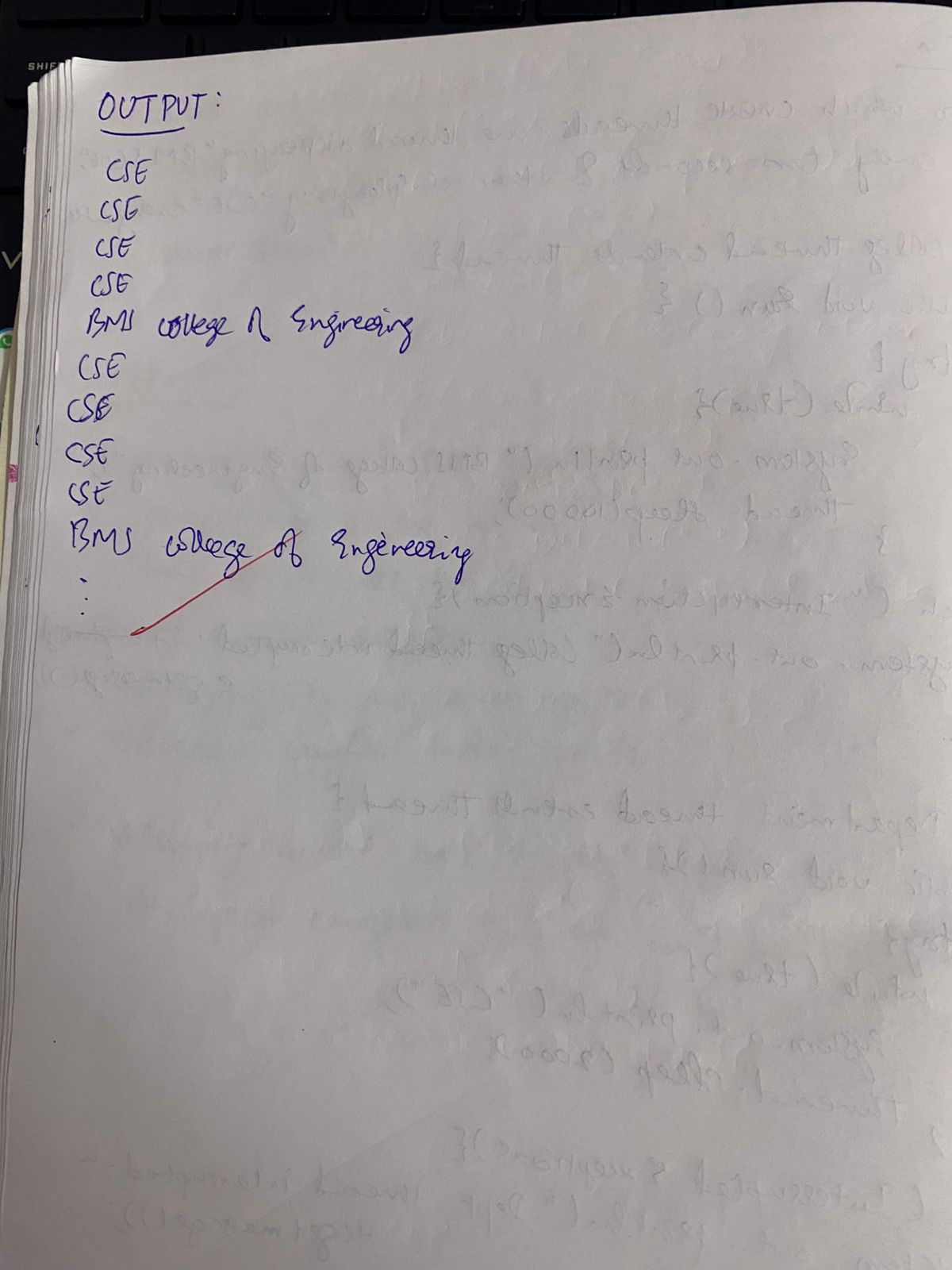
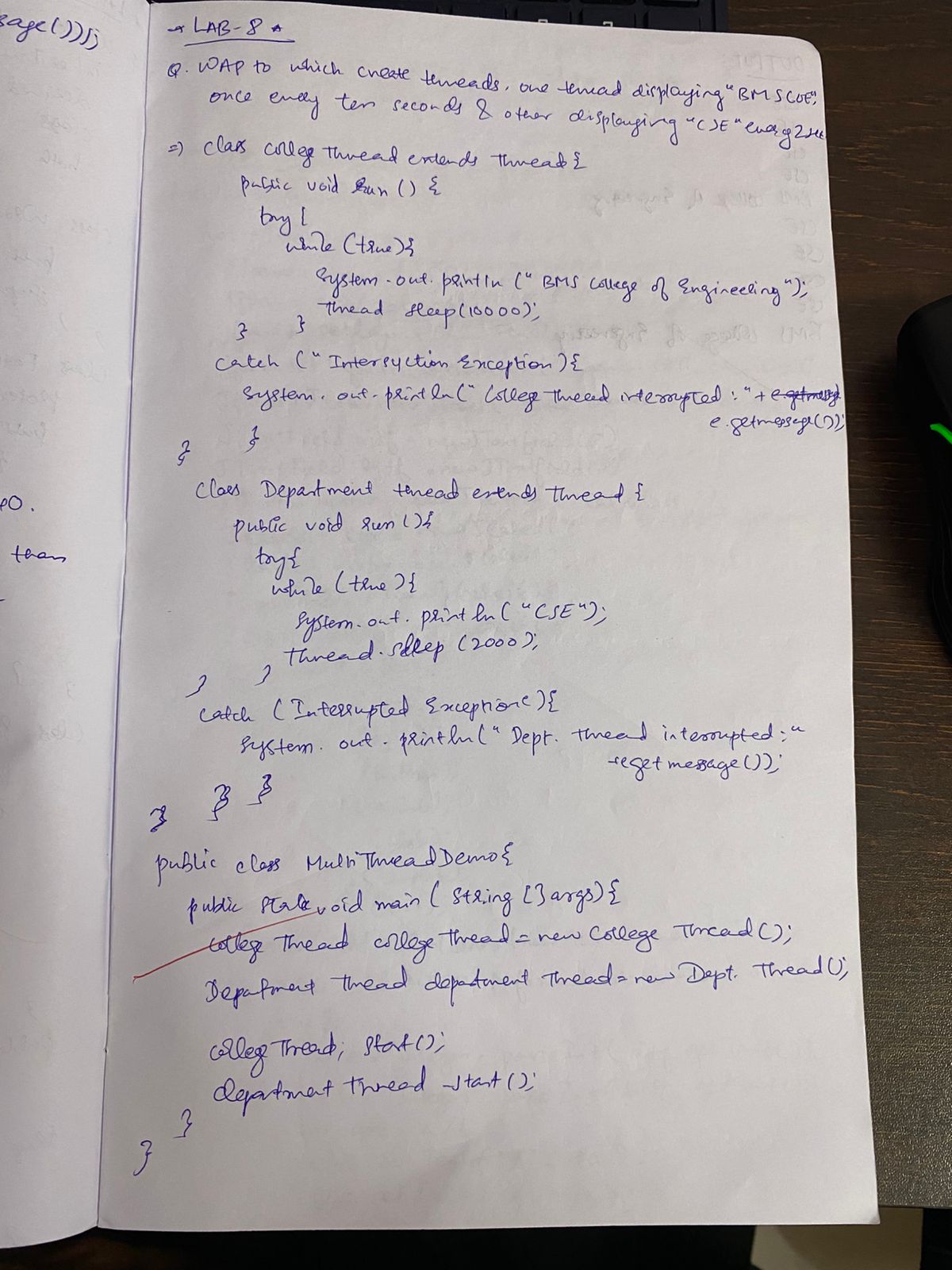
}

}



**Program 8:**

Creates 2 threads



Code:

class CollegeThread extends Thread {

@Override

public void run() {

try {

while (true) {

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

Thread.sleep(10000);

}

} catch (InterruptedException e) {

System.out.println("CollegeThread interrupted: " + e.getMessage());

}

}

}

class DepartmentThread extends Thread {

@Override

public void run() {

try {

while (true) {

System.out.println("CSE");

Thread.sleep(2000);

}

} catch (InterruptedException e) {

System.out.println("DepartmentThread interrupted: " + e.getMessage());

}

}

}

public class Lab\_8 {

public static void main(String[] args) {

CollegeThread collegeThread = new CollegeThread();

DepartmentThread departmentThread = new DepartmentThread();

collegeThread.start();

departmentThread.start();

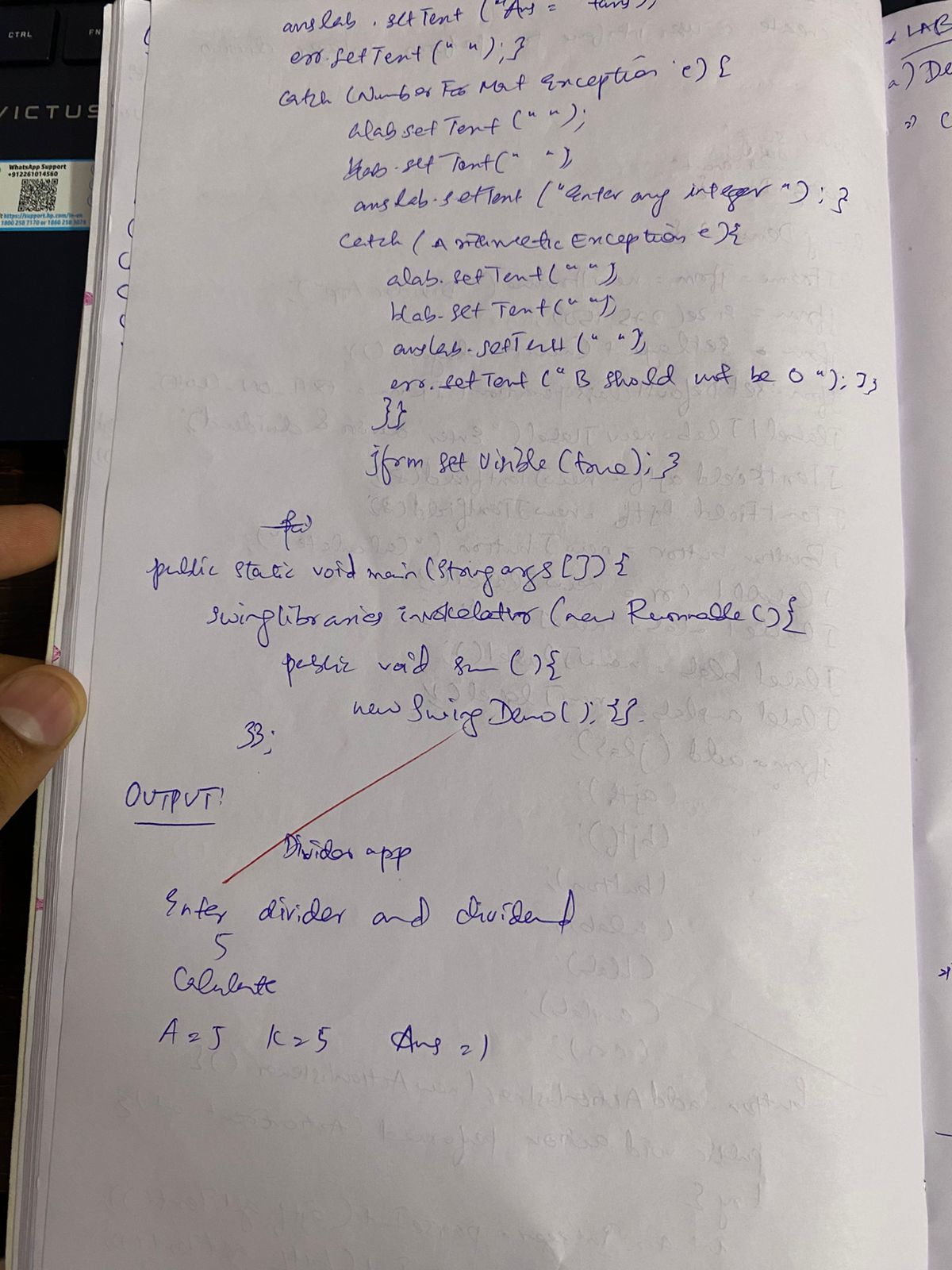
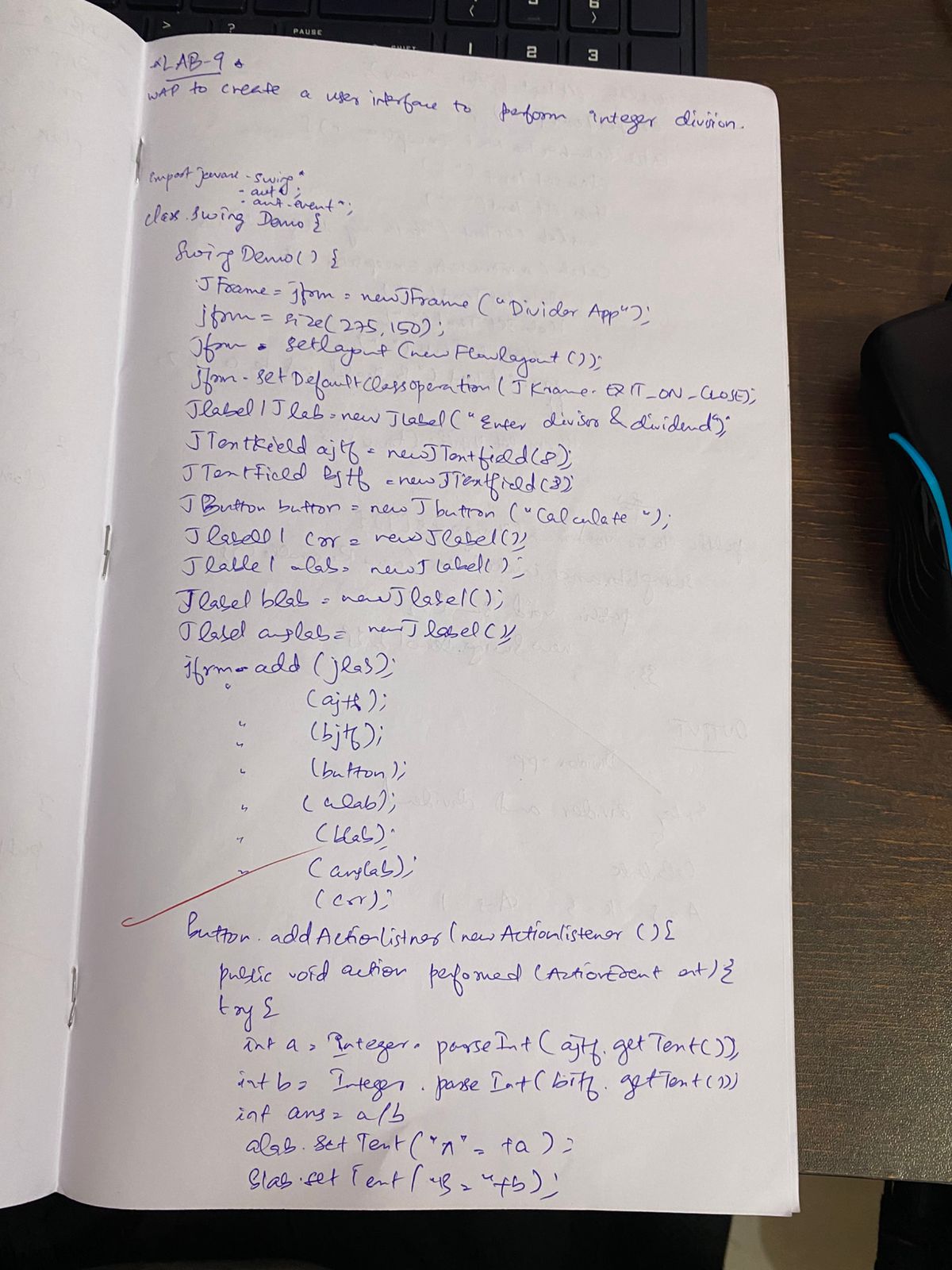
}

}



**PROGRAM-9:**

User interface to perform division



CODE:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class SwingDemo{

SwingDemo(){

JFrame jfrm = new JFrame("Divider App");

jfrm.setSize(275, 150);

jfrm.setLayout(new FlowLayout());

jfrm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel jlab = new JLabel("Enter the divider and divident:");

JTextField ajtf = new JTextField(8);

JTextField bjtf = new JTextField(8);

JButton button = new JButton("Calculate");

JLabel err = new JLabel();

JLabel alab = new JLabel();

JLabel blab = new JLabel();

JLabel anslab = new JLabel();

jfrm.add(err);

jfrm.add(jlab);

jfrm.add(ajtf);

jfrm.add(bjtf);

jfrm.add(button);

jfrm.add(alab);

jfrm.add(blab);

jfrm.add(anslab);

ActionListener l = new ActionListener() {

public void actionPerformed(ActionEvent evt) { System.out.println("Action event from a text field"); } };

ajtf.addActionListener(l);

bjtf.addActionListener(l);

button.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent evt) { try{ int a = Integer.parseInt(ajtf.getText()); int b = Integer.parseInt(bjtf.getText()); int ans = a/b;

alab.setText("\nA = " + a);

blab.setText("\nB = " + b);

anslab.setText("\nAns = "+ ans);

}

catch(NumberFormatException e){

alab.setText("");

blab.setText("");

anslab.setText("");

err.setText("Enter Only Integers!"); }

catch(ArithmeticException e){

alab.setText("");

blab.setText("");

anslab.setText("");

err.setText("B should be NON zero!"); } }

});

jfrm.setVisible(true);

}

public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable(){ public void run(){

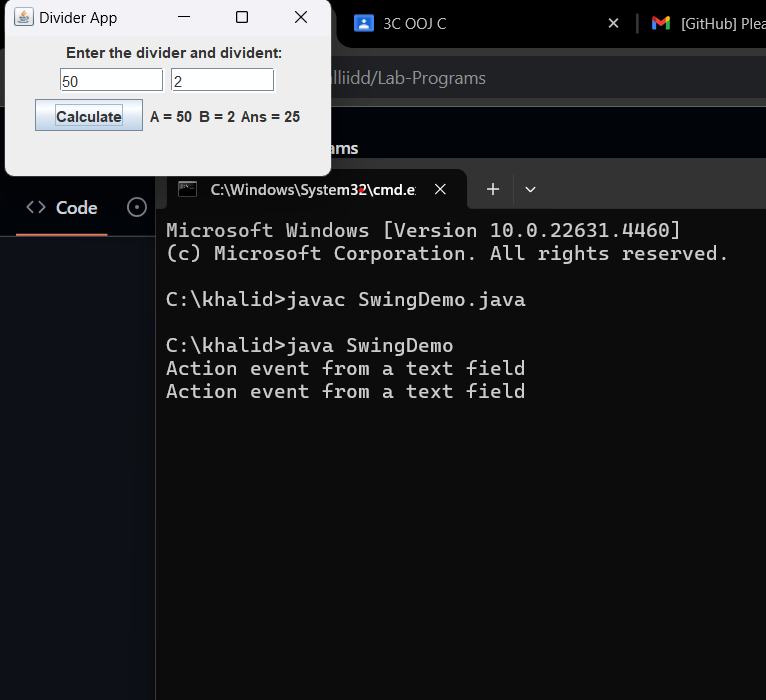
new SwingDemo();

}

});

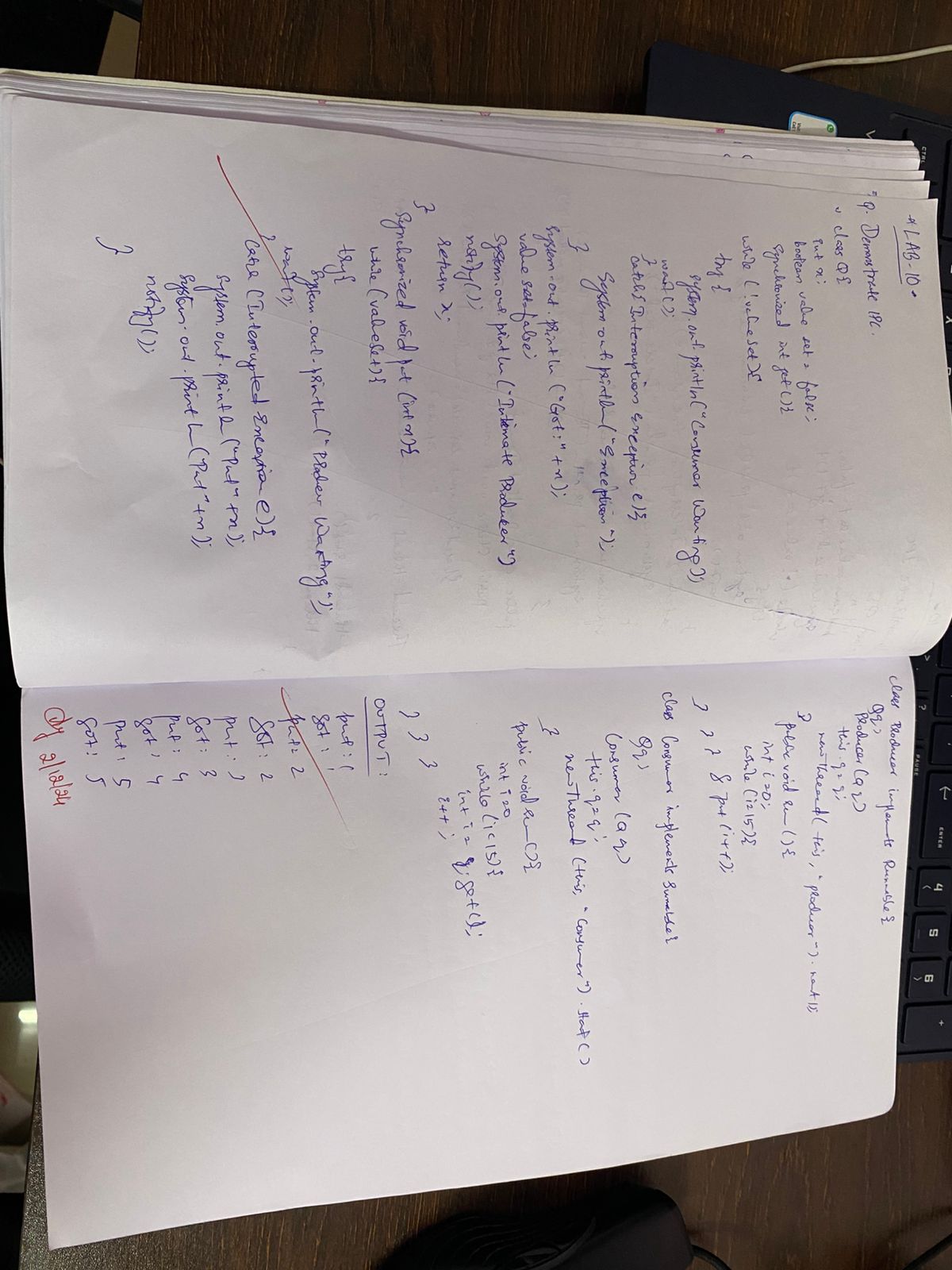
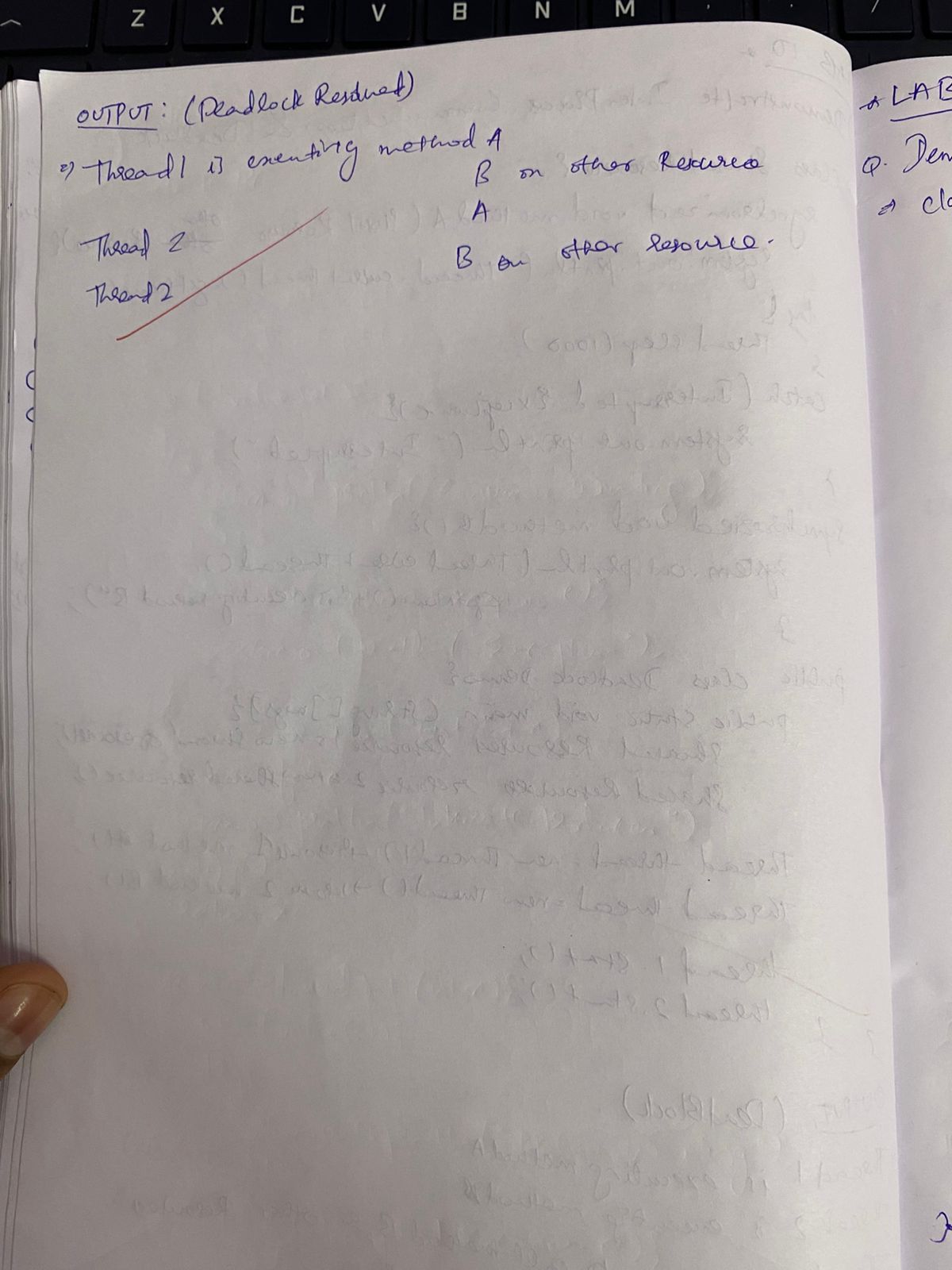
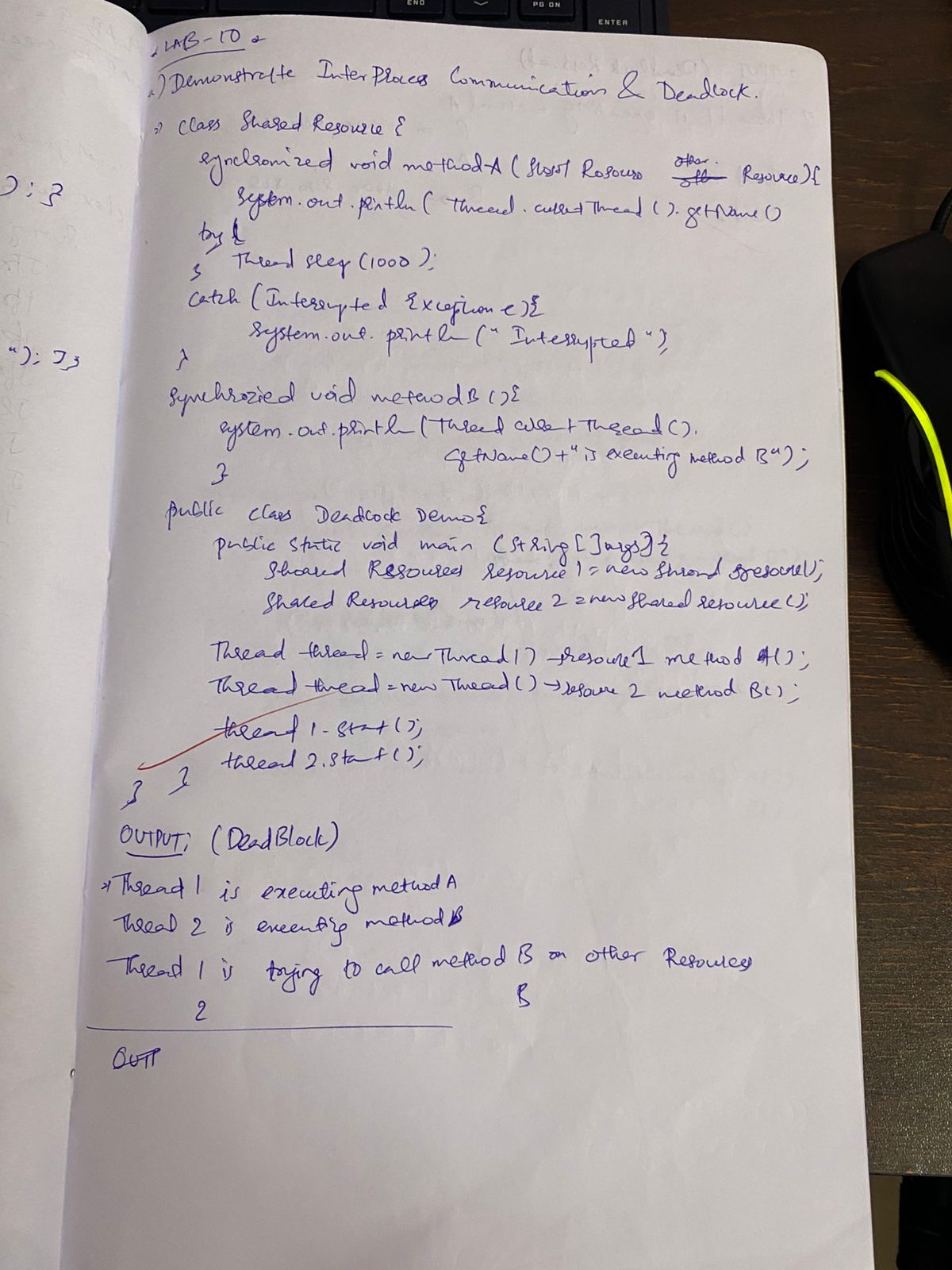
}

}



**PROGRAM-10:**

Demonstrate Inter process Communication and deadlock



CODE:

DEADLOCK DEMO

class Resource1 {

synchronized void lock(Resource2 res2) {

System.out.println(Thread.currentThread().getName() + " locked Resource1");

try {

Thread.sleep(100); // Simulate some work

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println(Thread.currentThread().getName() + " trying to lock Resource2...");

res2.method2(); // Try to lock Resource2

}

synchronized void method1() {

System.out.println(Thread.currentThread().getName() + " is working with Resource1");

}

}

class Resource2 {

synchronized void lock(Resource1 res1) {

System.out.println(Thread.currentThread().getName() + " locked Resource2");

try {

Thread.sleep(100); // Simulate some work

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println(Thread.currentThread().getName() + " trying to lock Resource1...");

res1.method1(); // Try to lock Resource1

}

synchronized void method2() {

System.out.println(Thread.currentThread().getName() + " is working with Resource2");

}

}

public class DeadlockDemo {

public static void main(String[] args) {

Resource1 res1 = new Resource1();

Resource2 res2 = new Resource2();

Thread t1 = new Thread(() -> res1.lock(res2), "Thread-1");

Thread t2 = new Thread(() -> res2.lock(res1), "Thread-2");

t1.start();

t2.start();

}

}

DEADLOCK RESOLVED

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

class SafeResource1 {

private final Lock lock = new ReentrantLock(); // Lock for the resource

public boolean tryLockBoth(SafeResource1 otherResource) {

while (true) {

boolean gotFirstLock = lock.tryLock(); // Try to acquire this resource's lock

boolean gotSecondLock = otherResource.tryLock(); // Try to acquire the other resource's lock

if (gotFirstLock && gotSecondLock) {

System.out.println(Thread.currentThread().getName() + " acquired both locks");

return true;

}

if (gotFirstLock) lock.unlock(); // Release if only first lock acquired

if (gotSecondLock) otherResource.unlock(); // Release if only second lock acquired

}

}

public boolean tryLock() {

return lock.tryLock();

}

public void unlock() {

lock.unlock();

}

}

public class DeadlockResolved {

public static void main(String[] args) {

SafeResource1 res1 = new SafeResource1();

SafeResource1 res2 = new SafeResource1();

Thread t1 = new Thread(() -> {

if (res1.tryLockBoth(res2)) {

System.out.println("Thread-1 completed safely");

res1.unlock();

res2.unlock();

}

}, "Thread-1");

Thread t2 = new Thread(() -> {

if (res2.tryLockBoth(res1)) {

System.out.println("Thread-2 completed safely");

res2.unlock();

res1.unlock();

}

}, "Thread-2");

t1.start();

t2.start();

}

}

IPC

class Q {

int n;

boolean valueSet = false;

synchronized int get() {

while (!valueSet) {

try {

System.out.println("\nConsumer waiting\n");

wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

return -1;

}

}

System.out.println("Got: " + n);

valueSet = false;

notify();

return n;

}

synchronized void put(int n) {

while (valueSet) {

try {

System.out.println("\nProducer waiting\n");

wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

return; }

}

this.n = n;

valueSet = true;

System.out.println("Put: " + 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();

if (r != -1) {

System.out.println("Consumed: " + r);

}

i++;

}

}

}

public class ProducerConsumerFix {

public static void main(String args[]) {

Q q = new Q();

new Producer(q);

new Consumer(q);

System.out.println("Press Control-C to stop.");

}

}

