**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

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

***Submitted by***

**SIDDHANT SAHARE (1BM23CS326)**

***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 **SIDDHANT SAHARE (1BM23CS326),** 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.

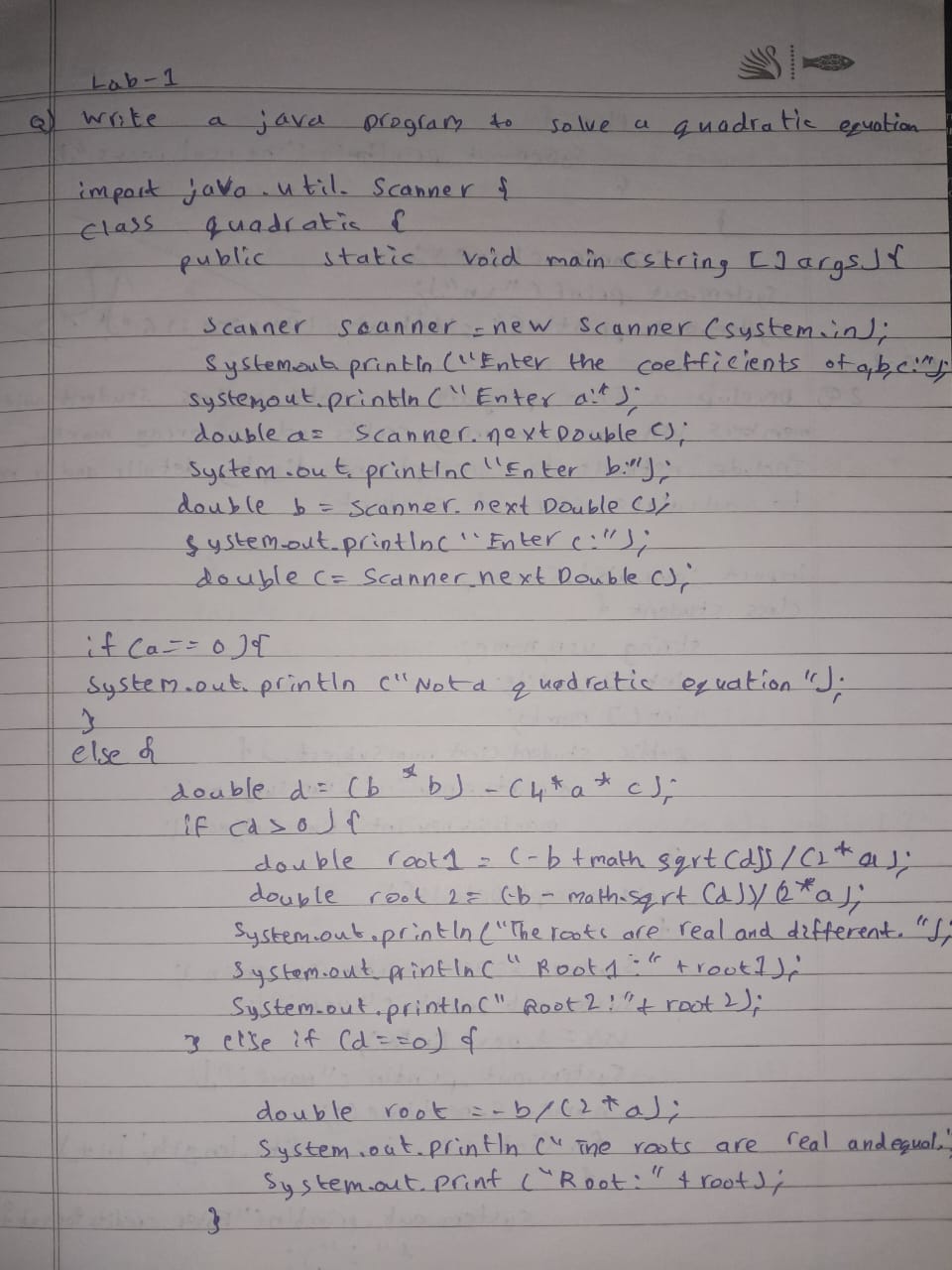
|  |  |
| --- | --- |
| Dr. Seema Patil  Assistant Professor  Department of CSE, BMSCE | Dr. Jyothi S Nayak  Professor & HOD  Department of CSE, BMSCE |

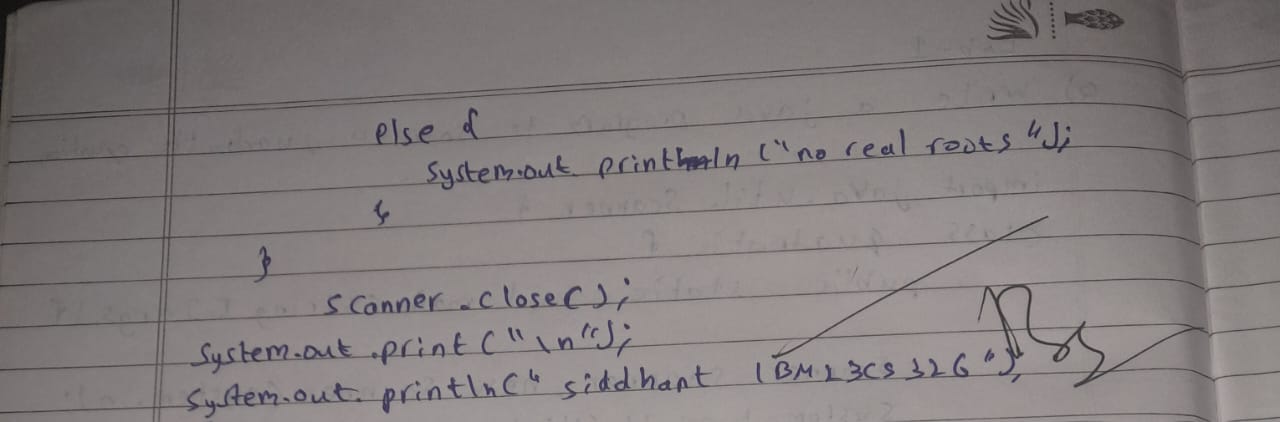
**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 11/10/24 | QUADRATIC EQUATION | 4 |
| 2 | 18/10/24 | SGPA CALCULATION | 7 |
| 3 | 24/10/24 | TOSTRING METHOD | 12 |
| 4 | 24/10/24 | ABSTRACT SHAPES | 16 |
| 5 | 07/11/24 | BANK ACCOUNT | 20 |
| 6 | 021/11/24 | PACKAGES | 30 |
| 7 | 21/11/24 | EXCEPTION | 35 |
| 8 | 28/11/24 | THREADS | 40 |
| 9 | 24/11/24 | GRAPHICS | 43 |
| 10 | 24/11/24 | IPC AND DEADLOCK | 47 |

**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 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.");

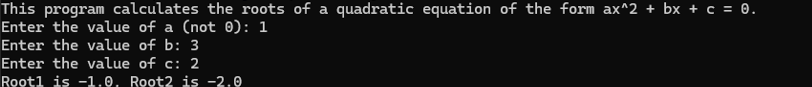
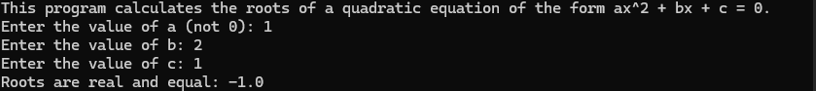
}

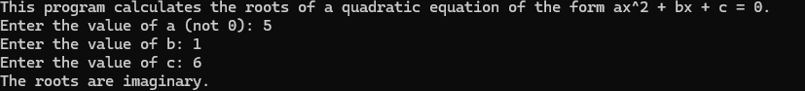
}

}

Quadratic q = new Quadratic();

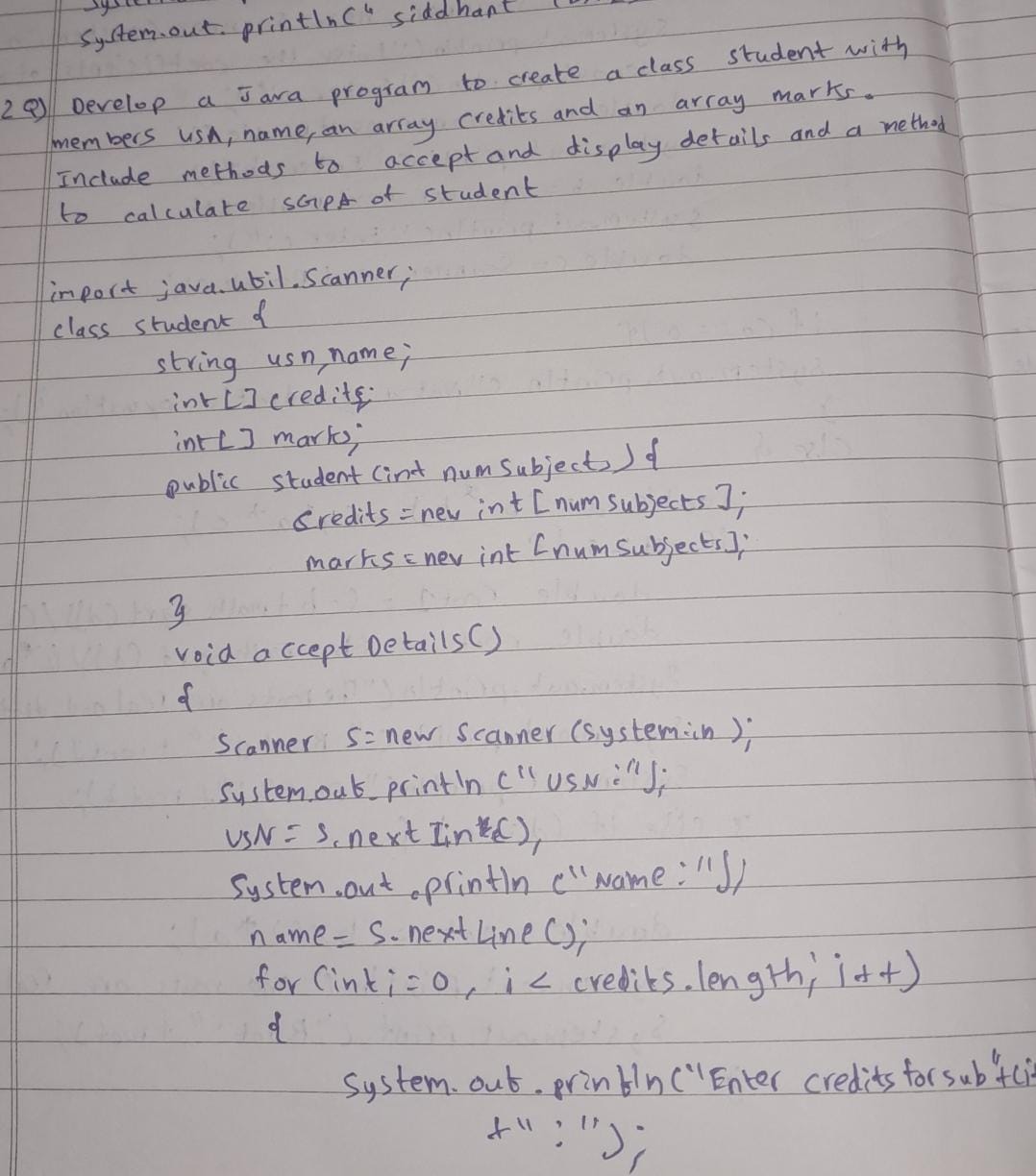
}

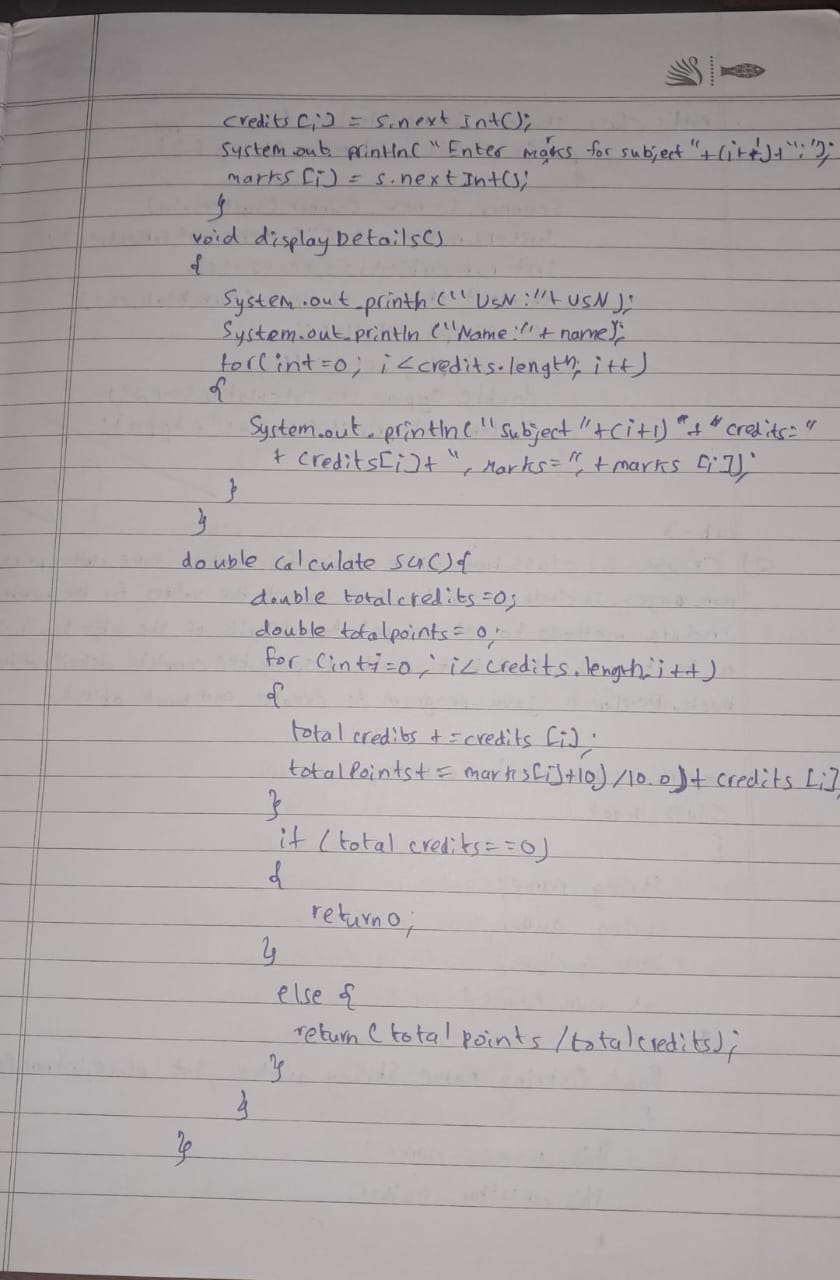
}  
  


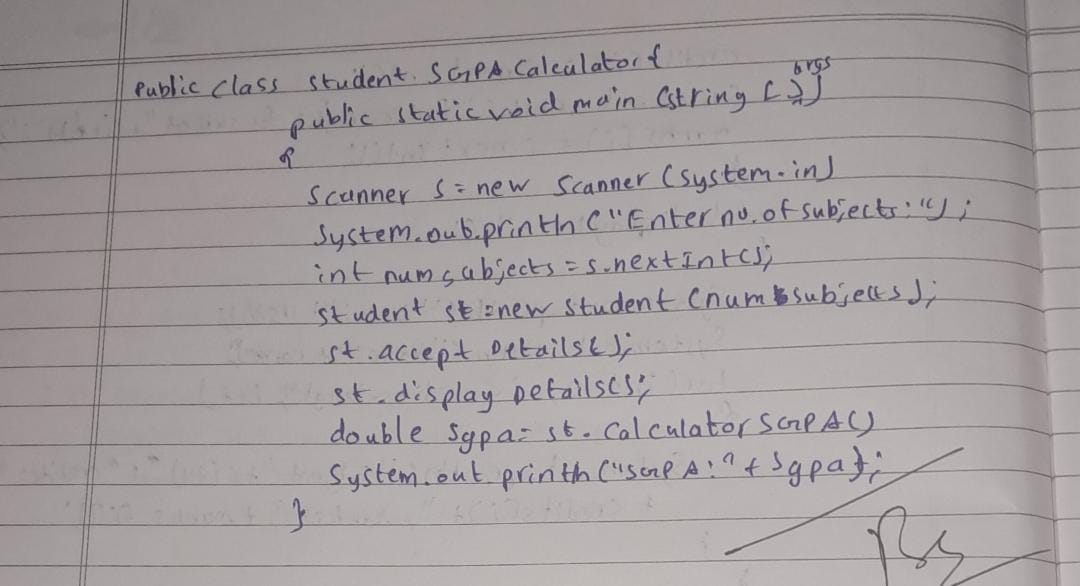


**Program 2**Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student

Algorithm:







Code:  
import java.util.Scanner;

public class Main {

public static void main(String[] args) {

class Subject {

int subjectMarks, credits, grade;

void calculateGrade() {

if (subjectMarks < 40) {

grade = 0;

} else {

grade = subjectMarks / 10 + 1;

}

}

}

class Student {

String name;

String usn;

double SGPA;

Subject subject[];

Scanner s;

Student() {

subject = new Subject[8];

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

subject[i] = new Subject();

}

s = new Scanner(System.in);

}

void getStudentDetails() {

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

name = s.nextLine();

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

usn = s.nextLine();

}

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("Marks should not exceed 100");

}

}

}

void computeSGPA() {

int totalCredits = 0, totalPoints = 0;

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

totalCredits += subject[i].credits;

totalPoints += subject[i].grade\*subject[i].credits;

}

SGPA = (double)totalPoints / totalCredits;

}

void displayResult() {

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

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

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

}

}

Student s1 = new Student();

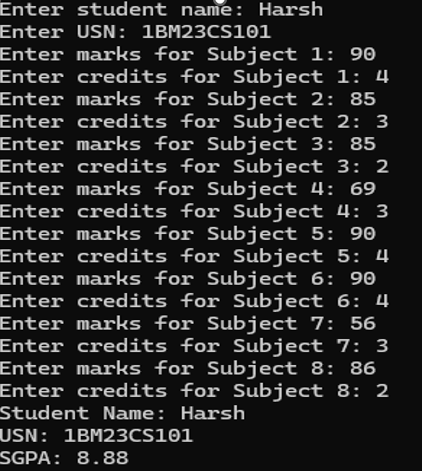
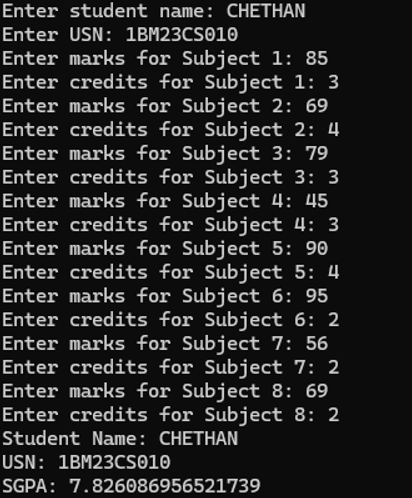
s1.getStudentDetails();

s1.getMarks();

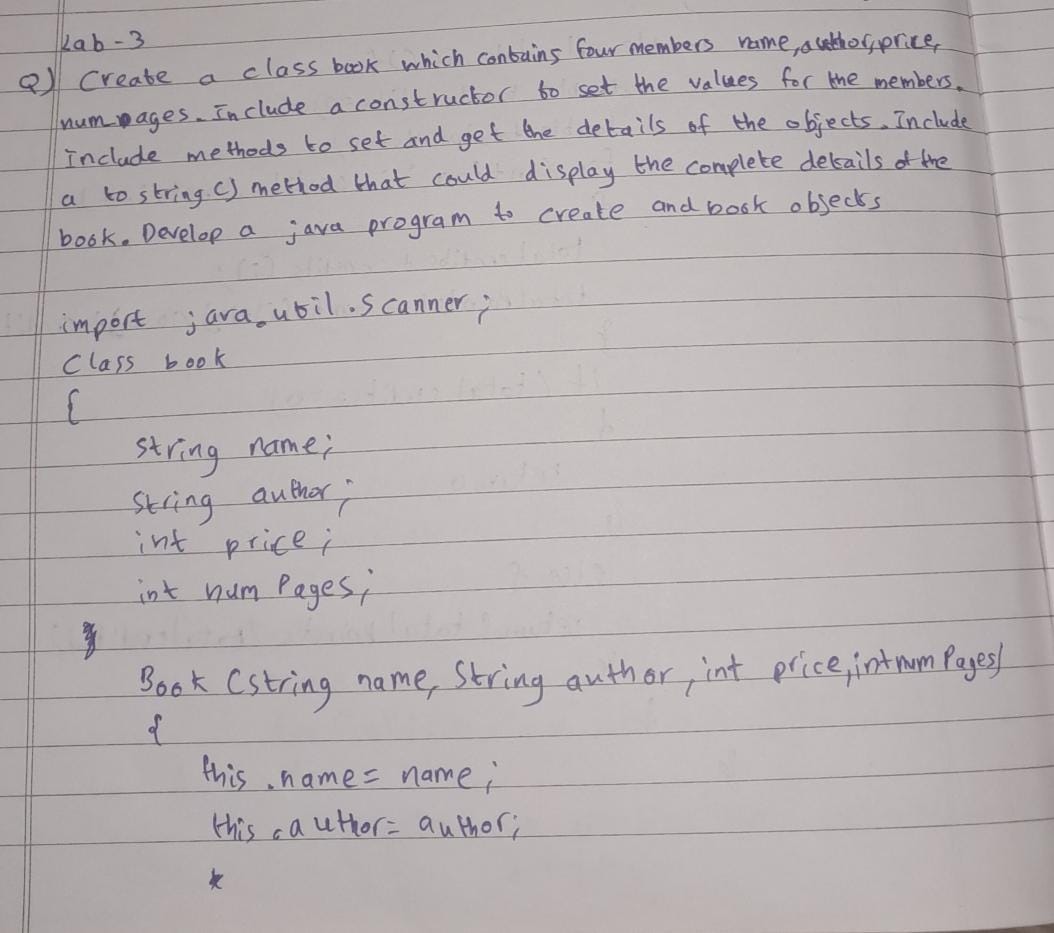
s1.computeSGPA();

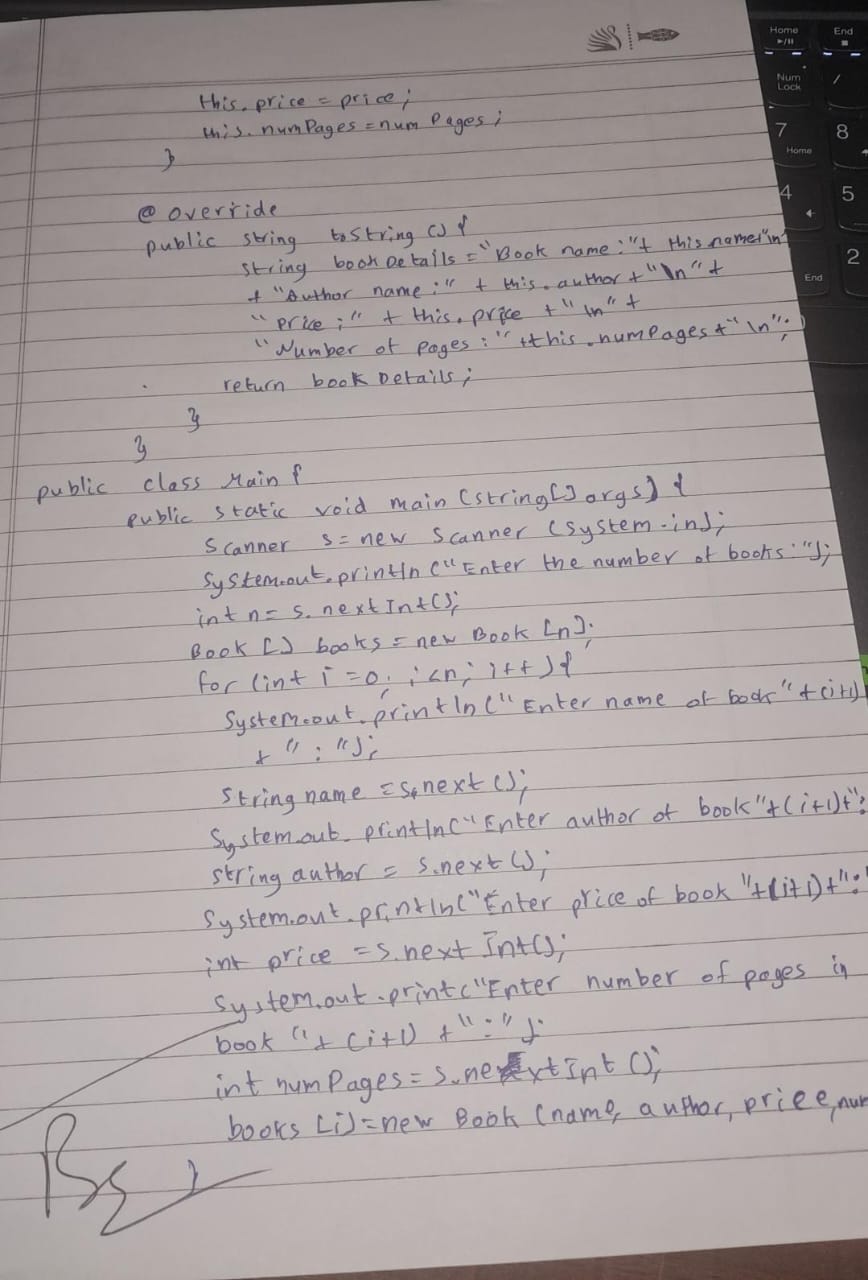
s1.displayResult();

}

}  
  
  
**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:





Code:  
import java.util.Scanner ;

public class Main{

public static void main(String args[]){

int n ;

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

Scanner sc = new Scanner(System.in) ; n = sc.nextInt() ;

sc.nextLine() ;

Book books[] = new Book[n];

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

System.out.print("Enter the book name: ") ;

String name = sc.nextLine() ;

System.out.print("Enter the author name:") ;

String author = sc.nextLine() ;

System.out.println("Enter the price of the book:") ;

int price = sc.nextInt() ;

System.out.println("Enter the number of pages in the book:") ;

int numPages = sc.nextInt() ;

sc.nextLine() ;

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

}

System.out.println("");

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

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

}

sc.close();

}

}

class Book{

String name , author ;

int price , numPages ;

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" ;

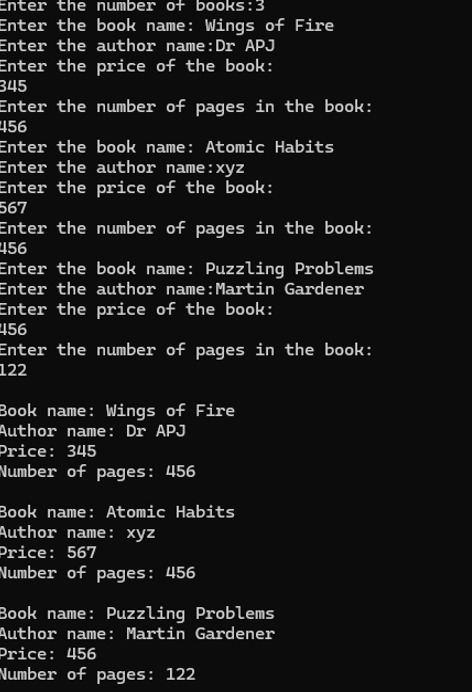
price = "Price: " + this.price + "\n" ;

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

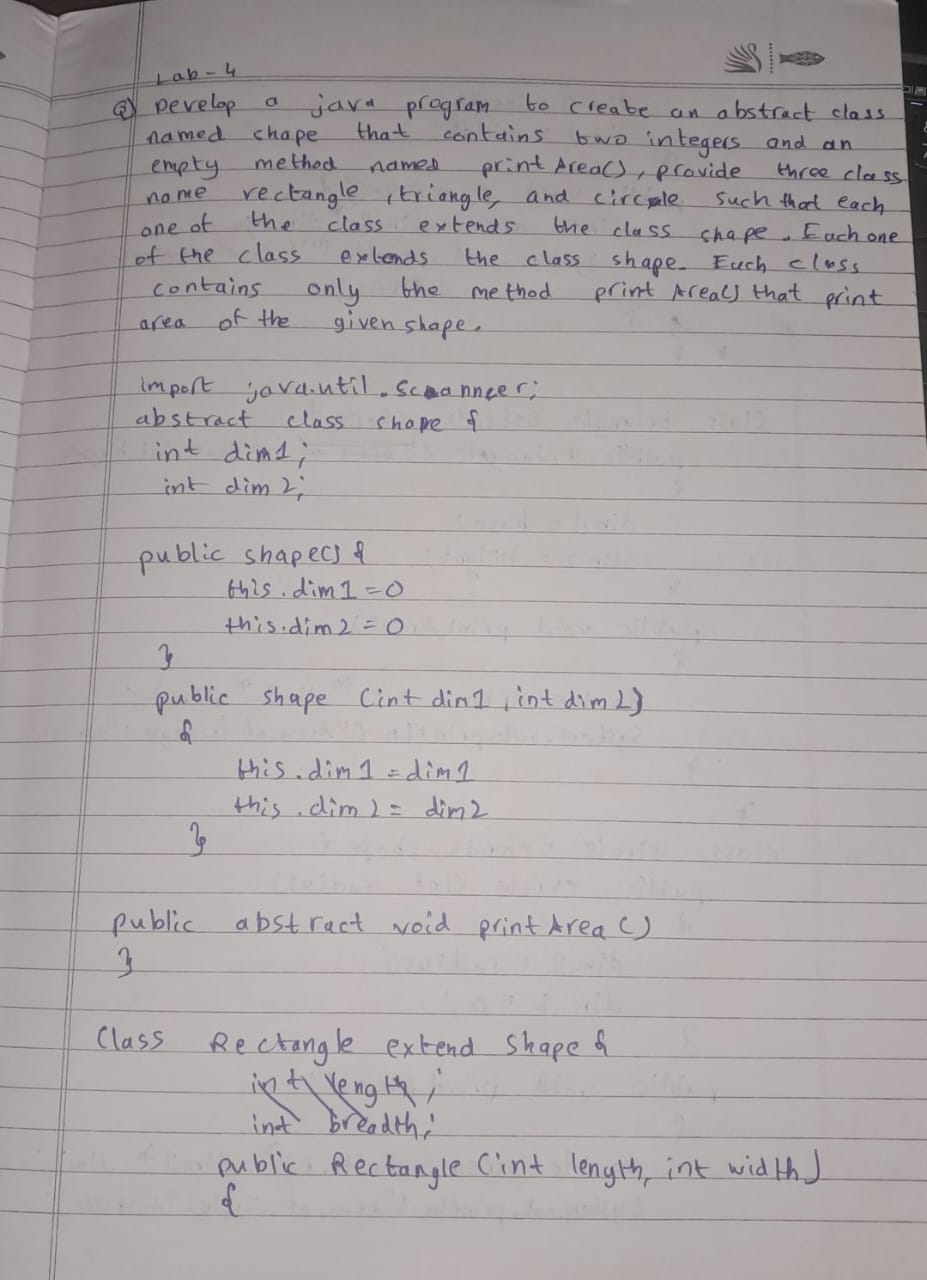
return name + author + price + numPages ;

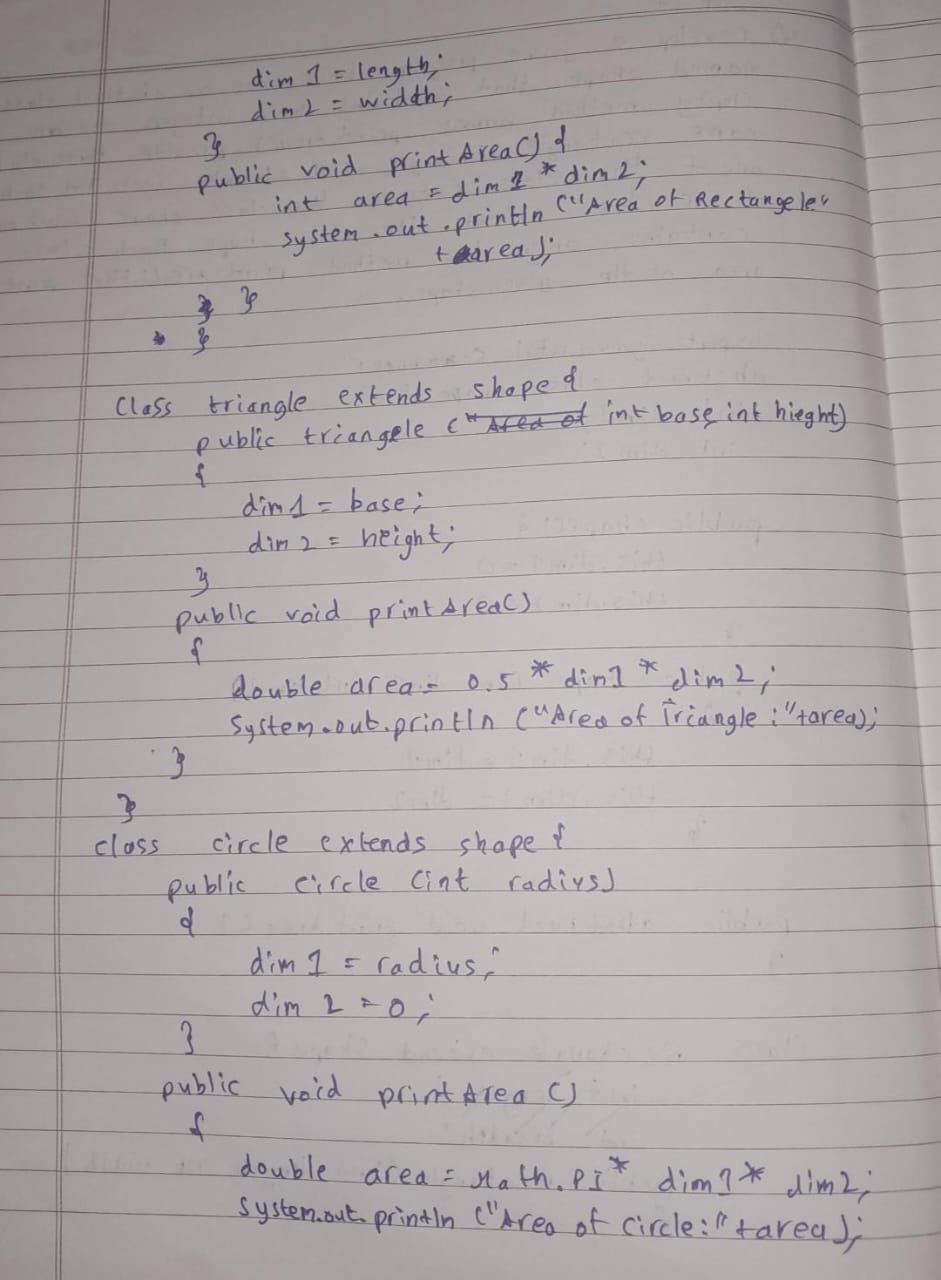
}

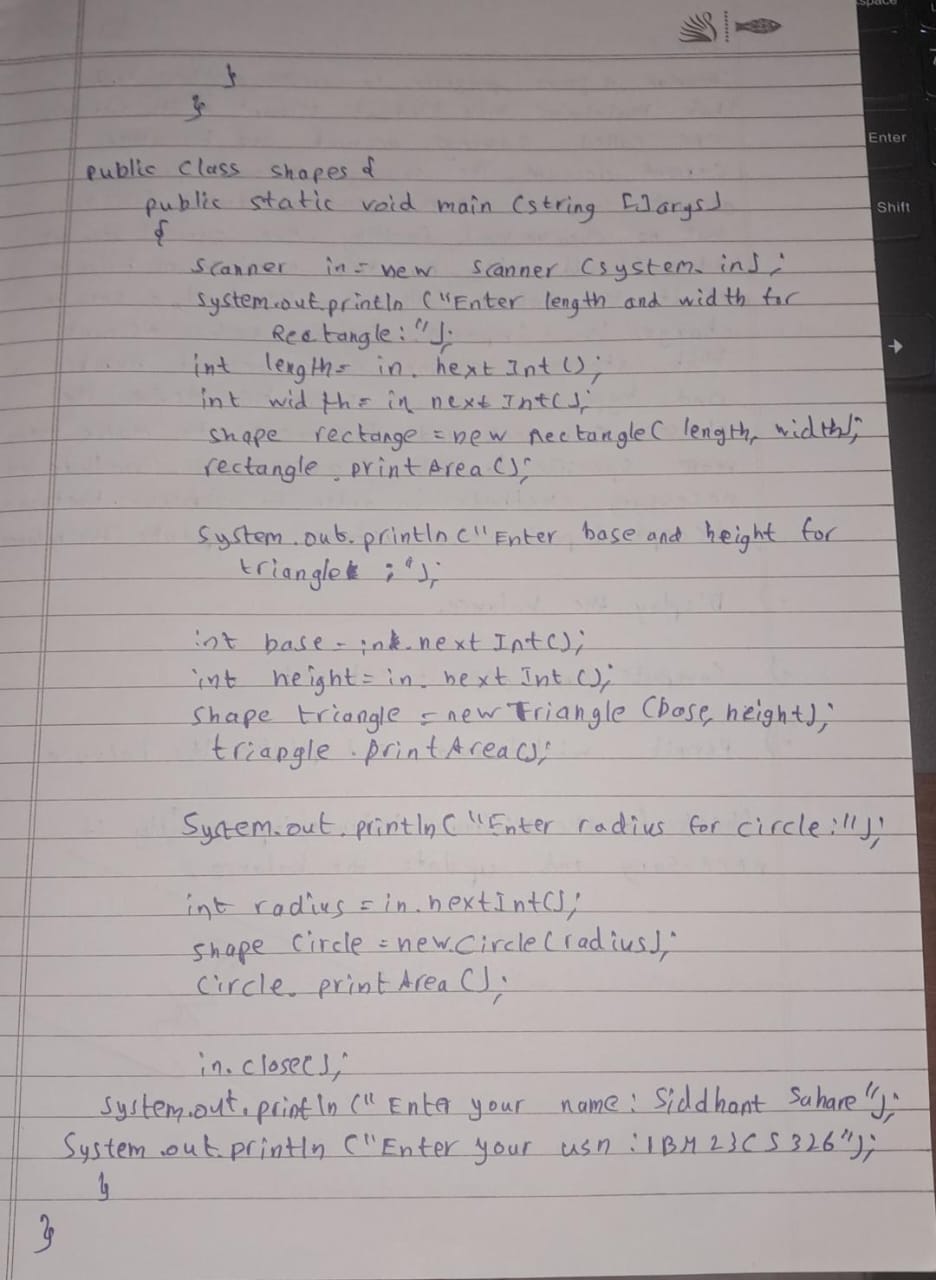
}



**Program 4**Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape..

Algorithm:  






Code:  
import java.util.Scanner ;

public class Main{

public static void main(String[] args){

Rectangle ob2 = new Rectangle() ;

Triangle ob1 = new Triangle() ;

Circle ob3 = new Circle() ;

ob2.printArea() ;

ob1.printArea() ;

ob3.printArea() ;

}

}

abstract class Shape{

Scanner sc = new Scanner(System.in) ;

int dimension1 , dimension2 ;

abstract void printArea();

}

class Rectangle extends Shape{

Rectangle(){

System.out.println("Enter the dimensions of the rectangle(Length and Breadth): " ) ;

dimension1 = sc.nextInt() ;

dimension2 = sc.nextInt() ;

}

void printArea(){

System.out.print("The area of the rectangle is = ") ;

System.out.println(dimension1\*dimension2) ;

}

}

class Triangle extends Shape{

Triangle(){

System.out.println("Enter the dimensions of the triangle(base and height): " ) ;

dimension1 = sc.nextInt() ;

dimension2 = sc.nextInt() ;

}

void printArea(){

System.out.print("The area of the Triangle is = ") ;

System.out.println(0.5\*dimension1\*dimension2) ;

}

}

class Circle extends Shape{

Circle(){

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

dimension1 = sc.nextInt() ;

}

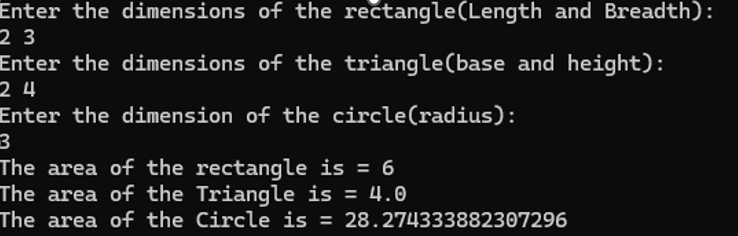
void printArea(){

System.out.print("The area of the Circle is = ") ;

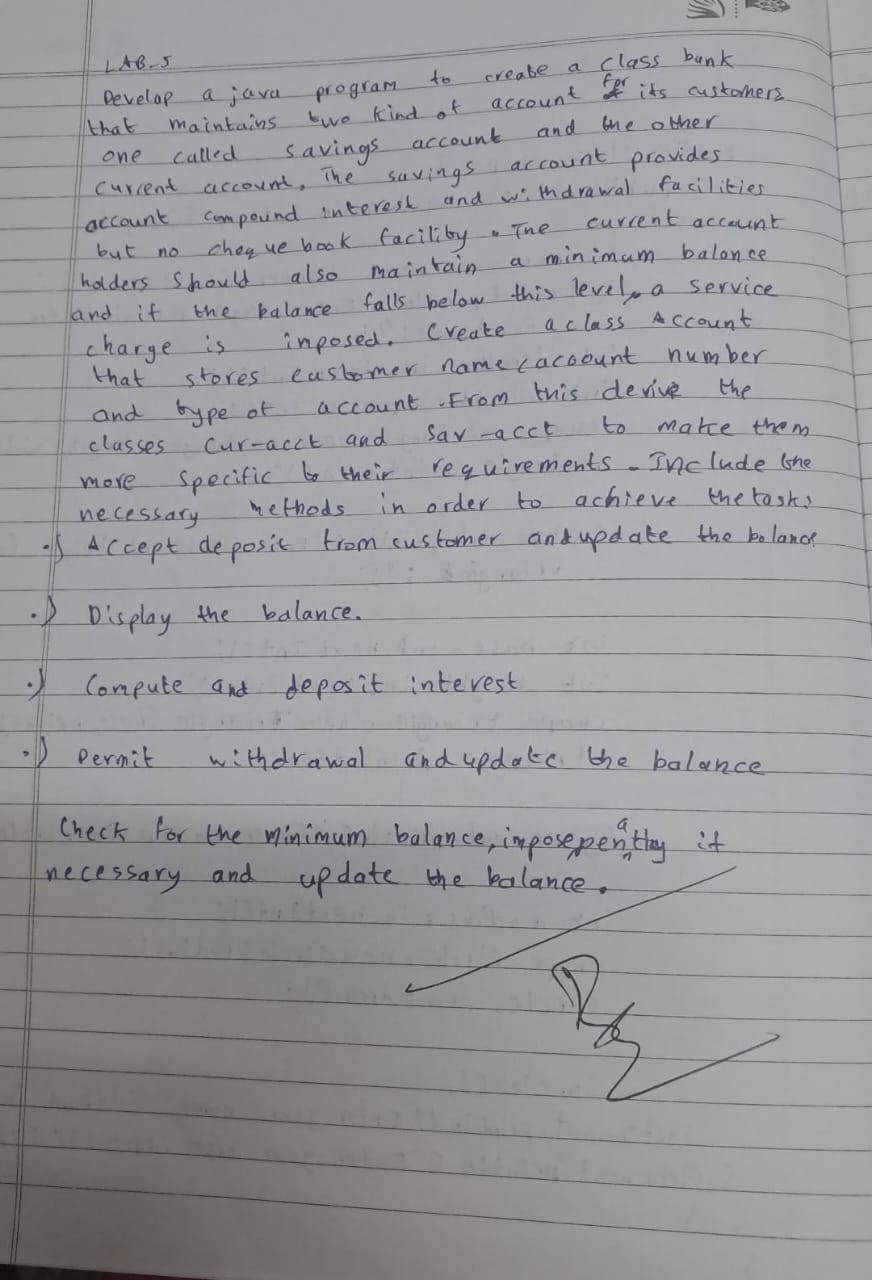
System.out.println(3.1415926535897\*dimension1\*dimension1) ;

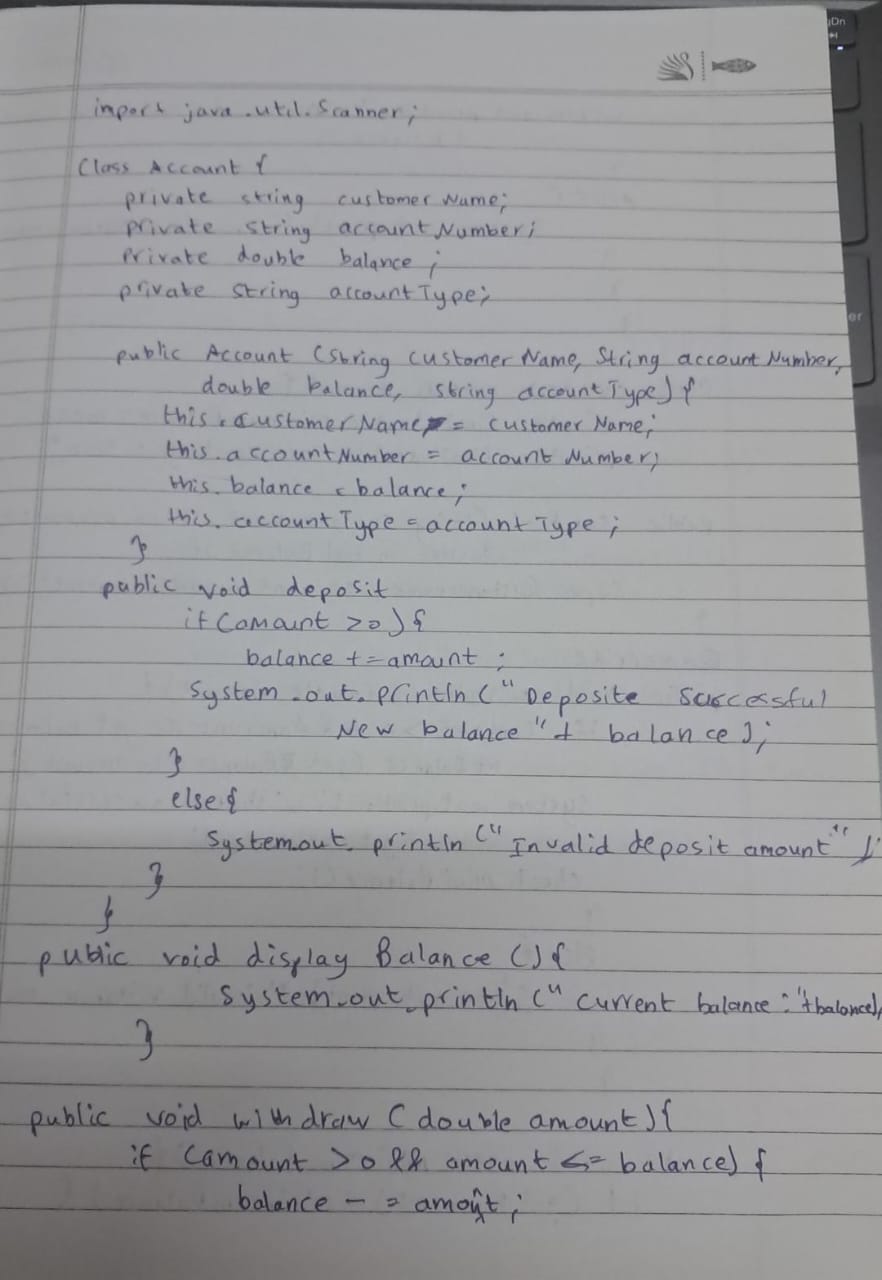
}

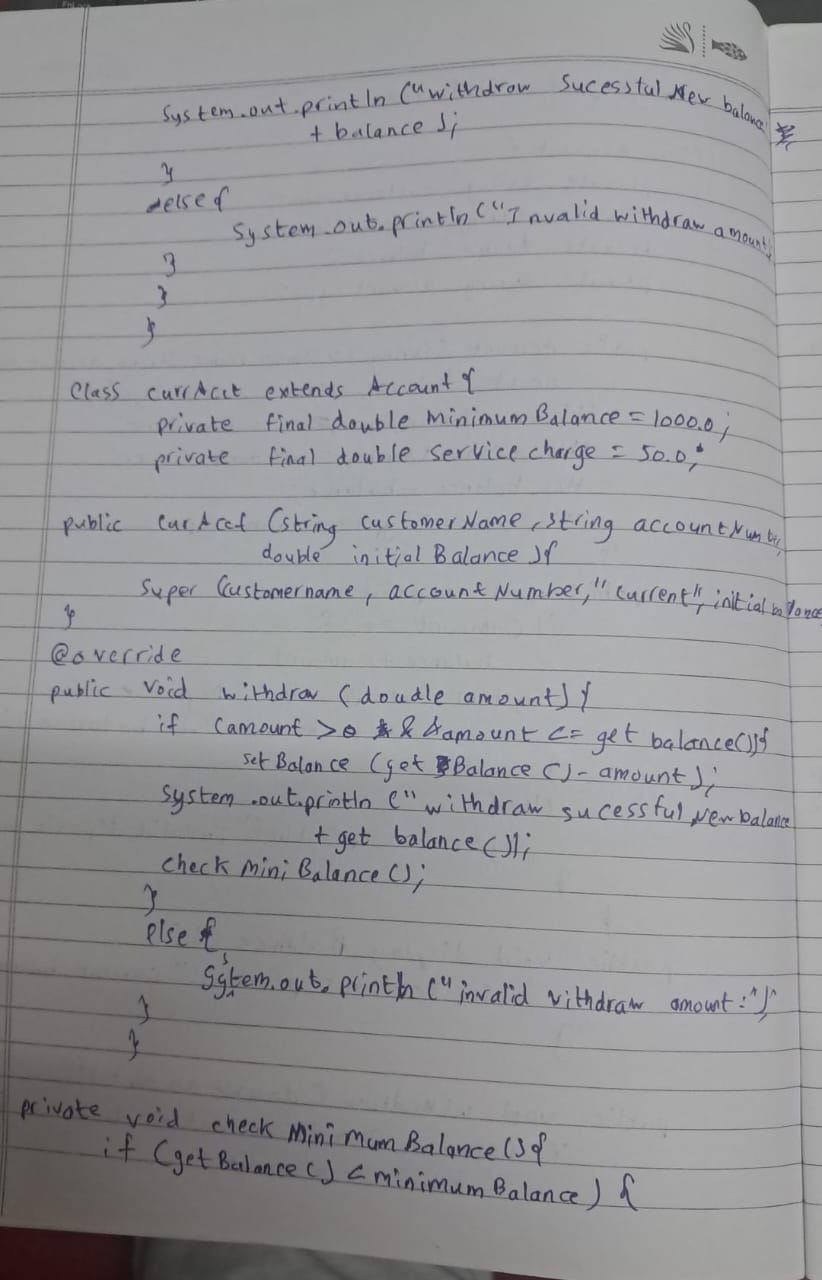
}

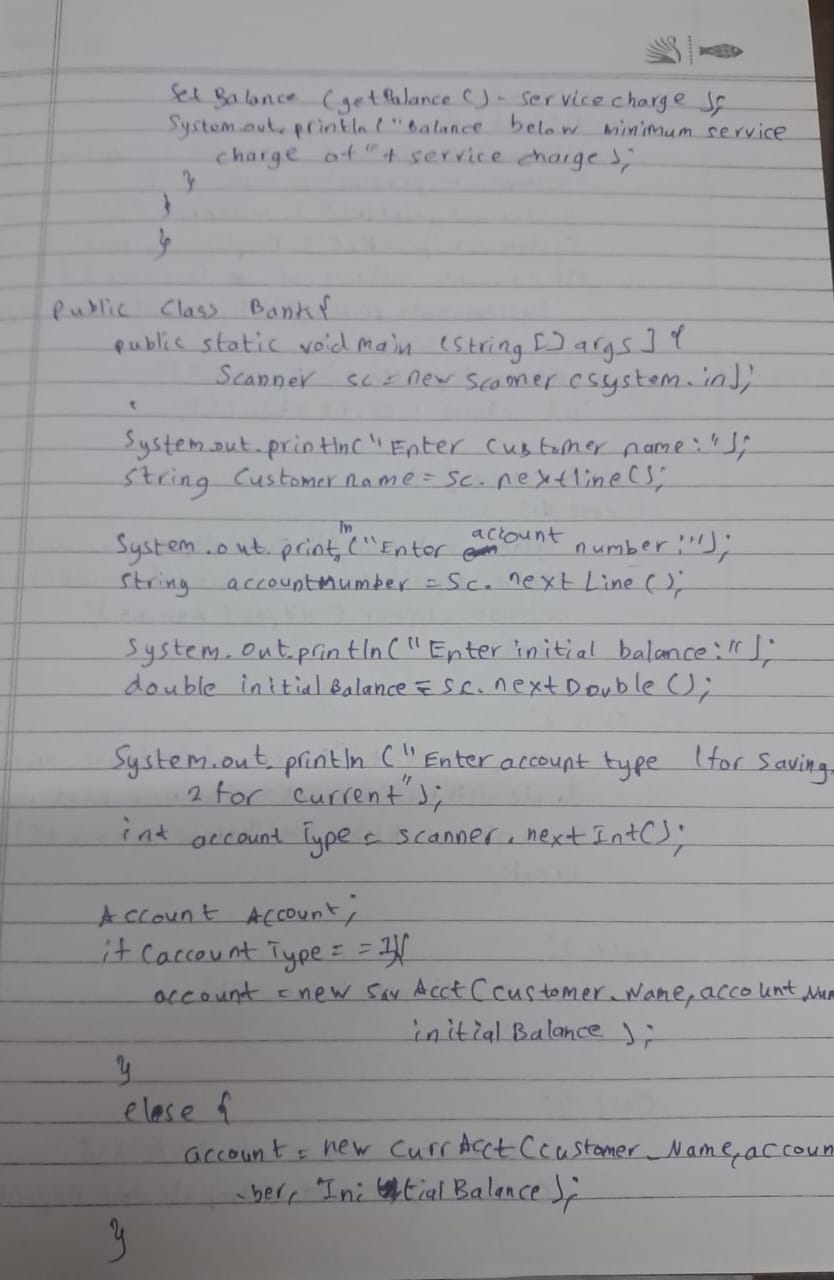


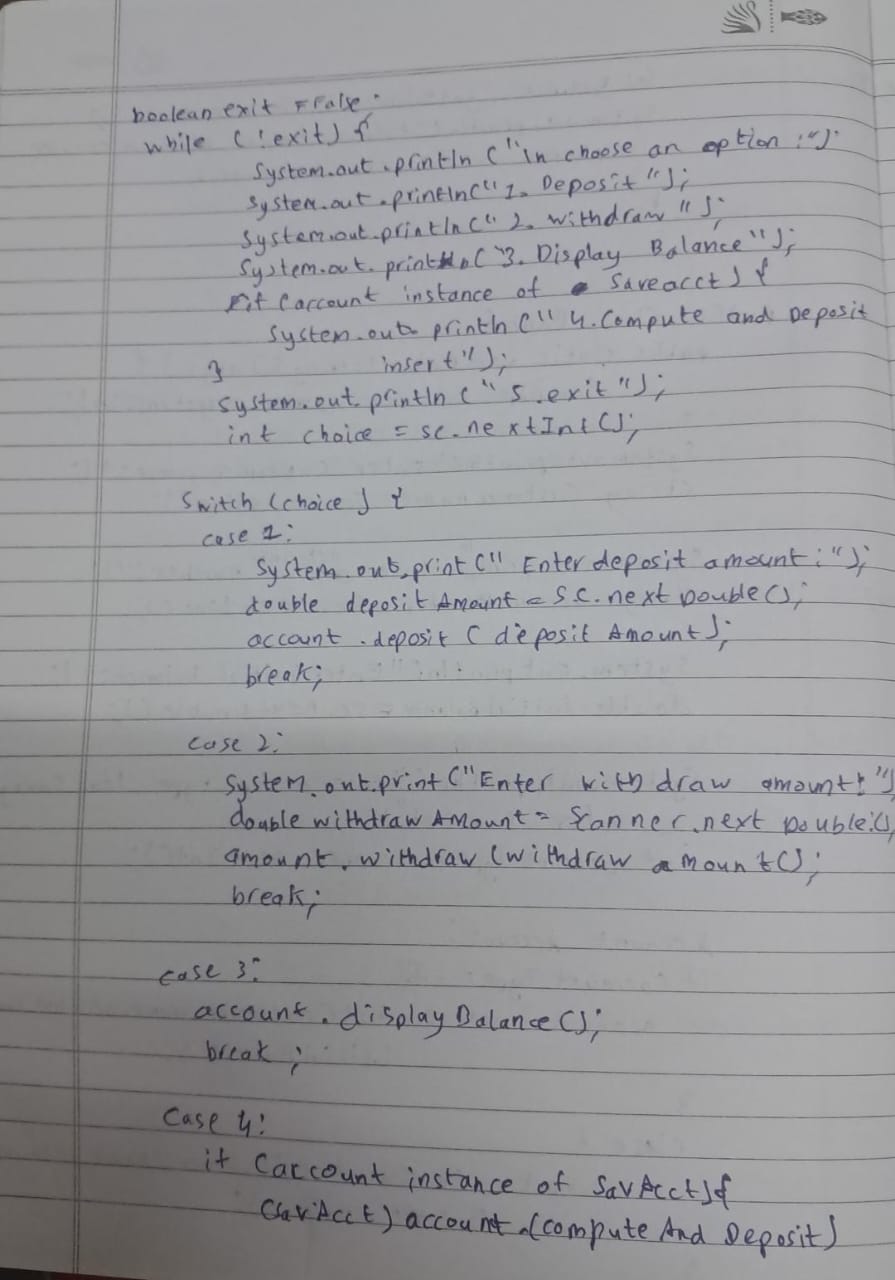
**Program 5**Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides 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.

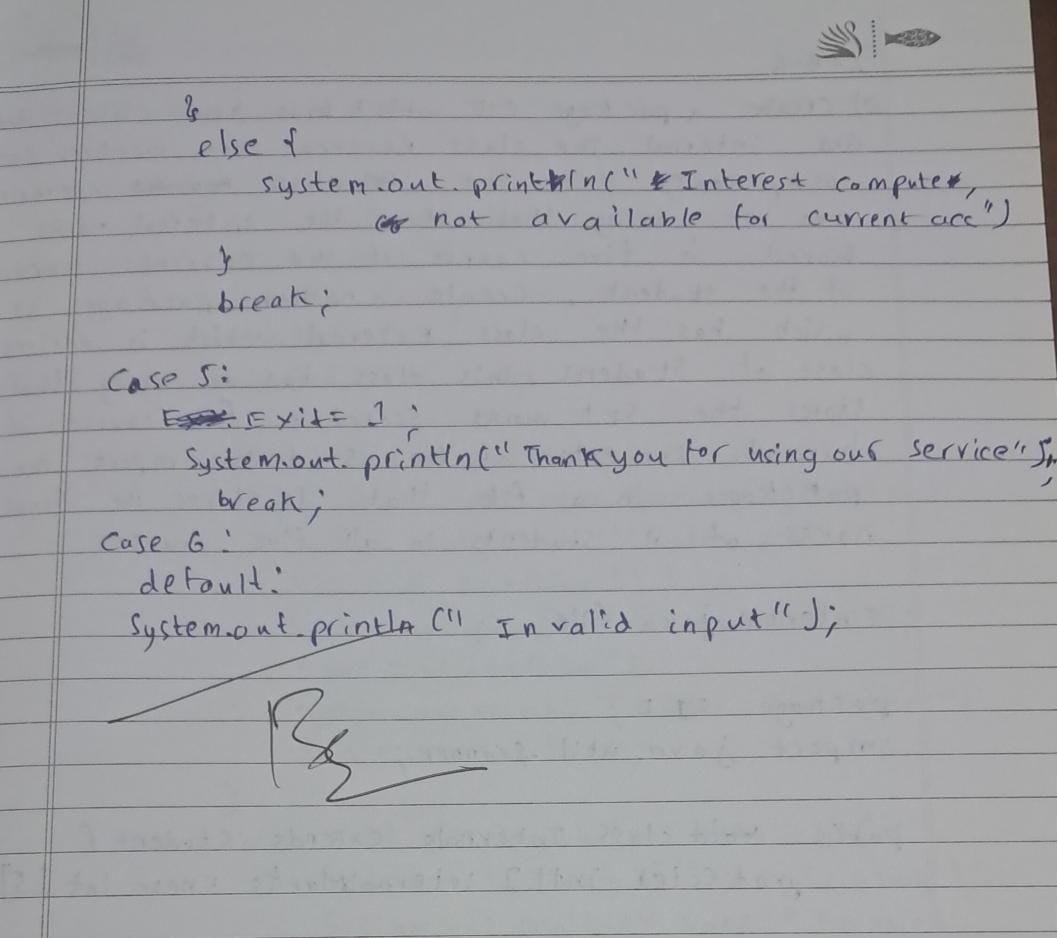
Algorithm:  












Code:

import java.util.Scanner;

public class Bank {

static Scanner sc = new Scanner(System.in);

Account ob1;

void createAccount() {

String customer;

int account;

String type;

int initBal;

System.out.print("Enter the customer name: ");

customer = sc.nextLine();

System.out.print("Enter account Number: ");

account = sc.nextInt();

sc.nextLine(); // Consume the newline

System.out.print("Enter Account type (Savings or Current): ");

type = sc.nextLine();

System.out.print("Enter the initial Balance: ");

initBal = sc.nextInt();

if (type.equals("Savings")) {

ob1 = new Savings(customer, account, initBal);

} else {

ob1 = new Current(customer, account, initBal);

}

}

public static void main(String[] args) {

Bank bank = new Bank();

bank.createAccount();

while (true) {

System.out.println("-------------------MENU-----------------");

System.out.println("1. Deposit 2. Withdraw");

System.out.println("3. Compute interest");

System.out.println("4. Display account details");

System.out.println("5. exit " ) ;

int choice = sc.nextInt();

switch (choice) {

case 1:

bank.ob1.deposit();

break;

case 2:

bank.ob1.withdraw();

break;

case 3:

if (bank.ob1 instanceof Savings) {

((Savings) bank.ob1).computeInterest();

} else {

System.out.println("Interest computation is only available for Savings accounts.");

}

break;

case 4:

bank.ob1.display();

break;

case 5:

break ;

default:

System.out.println("Invalid choice. Please try again.");

}

if(choice == 5) break ;

}

}

}

class Account {

String customerName;

int accountNumber;

int balance;

Account(String customer, int accountNum, int bal) {

customerName = customer;

accountNumber = accountNum;

balance = bal;

}

void deposit() {

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

int amt = Bank.sc.nextInt();

balance += amt;

System.out.println("Deposited: " + amt + ", New Balance: " + balance);

}

void withdraw() {

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

int amt = Bank.sc.nextInt();

if (balance - amt < 0) {

System.out.println("Insufficient Balance to withdraw the given amount.");

} else {

balance -= amt;

System.out.println("Amount of " + amt + " withdrawn successfully. Current Balance is " + balance);

}

}

void display() {

System.out.println("The Balance in the account is " + balance);

}

}

class Savings extends Account {

double interestPercent;

Savings(String customer, int accountNum, int bal) {

super(customer, accountNum, bal);

System.out.print("Enter the interest percentage on the account: ");

interestPercent = Bank.sc.nextDouble();

}

void computeInterest() {

balance += balance \* (interestPercent / 100);

System.out.println("Amount after applying interest is: " + balance);

}

}

class Current extends Account {

int minBalance = 1000;

Current(String customer, int accountNum, int bal) {

super(customer, accountNum, bal);

}

void withdraw() {

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

int amt = Bank.sc.nextInt();

if (balance - amt < minBalance) {

System.out.println("Insufficient Balance to maintain the minimum required.");

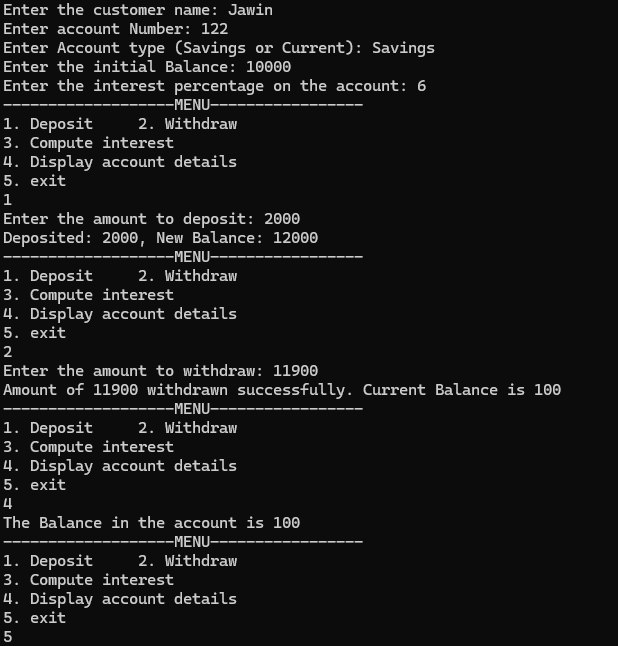
} else {

balance -= amt;

System.out.println("Amount of " + amt + " withdrawn successfully. Current Balance is " + balance);

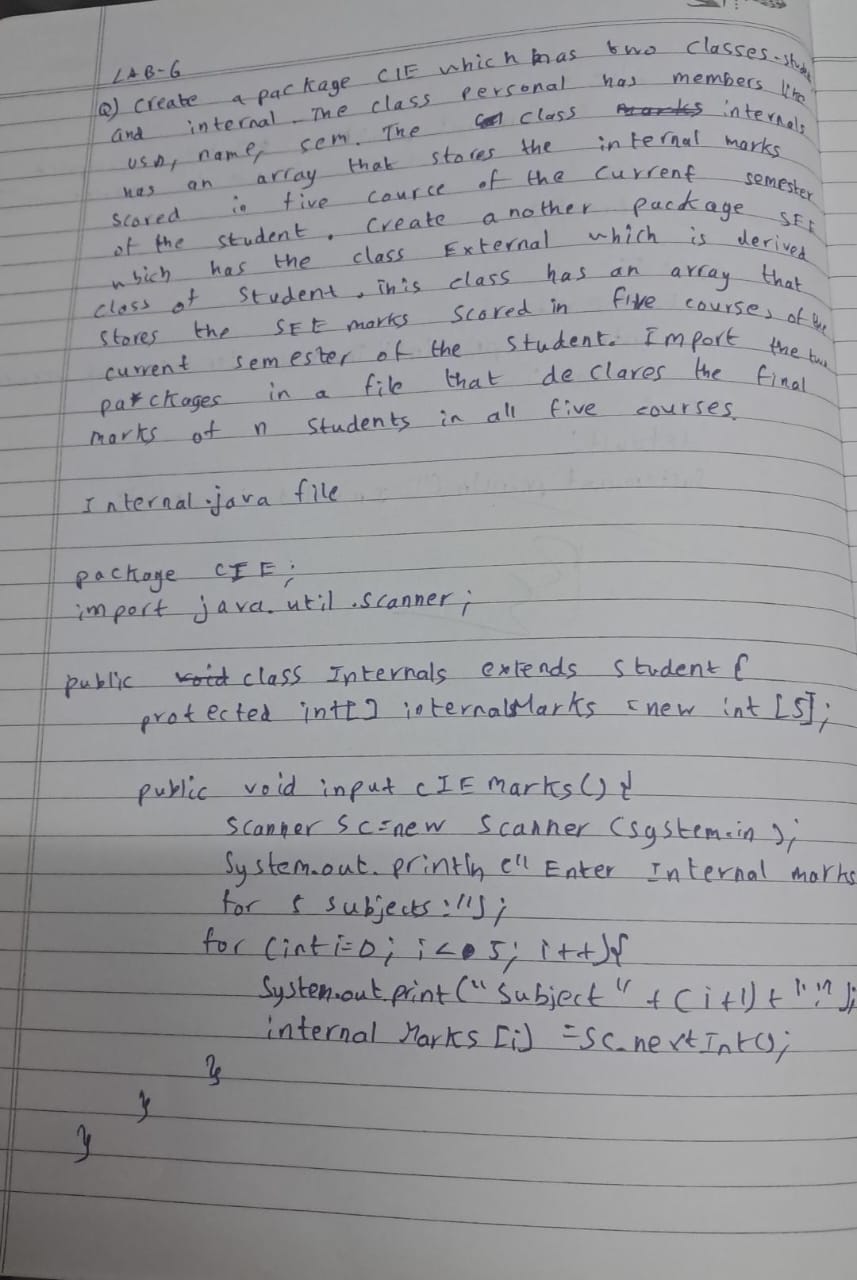
}

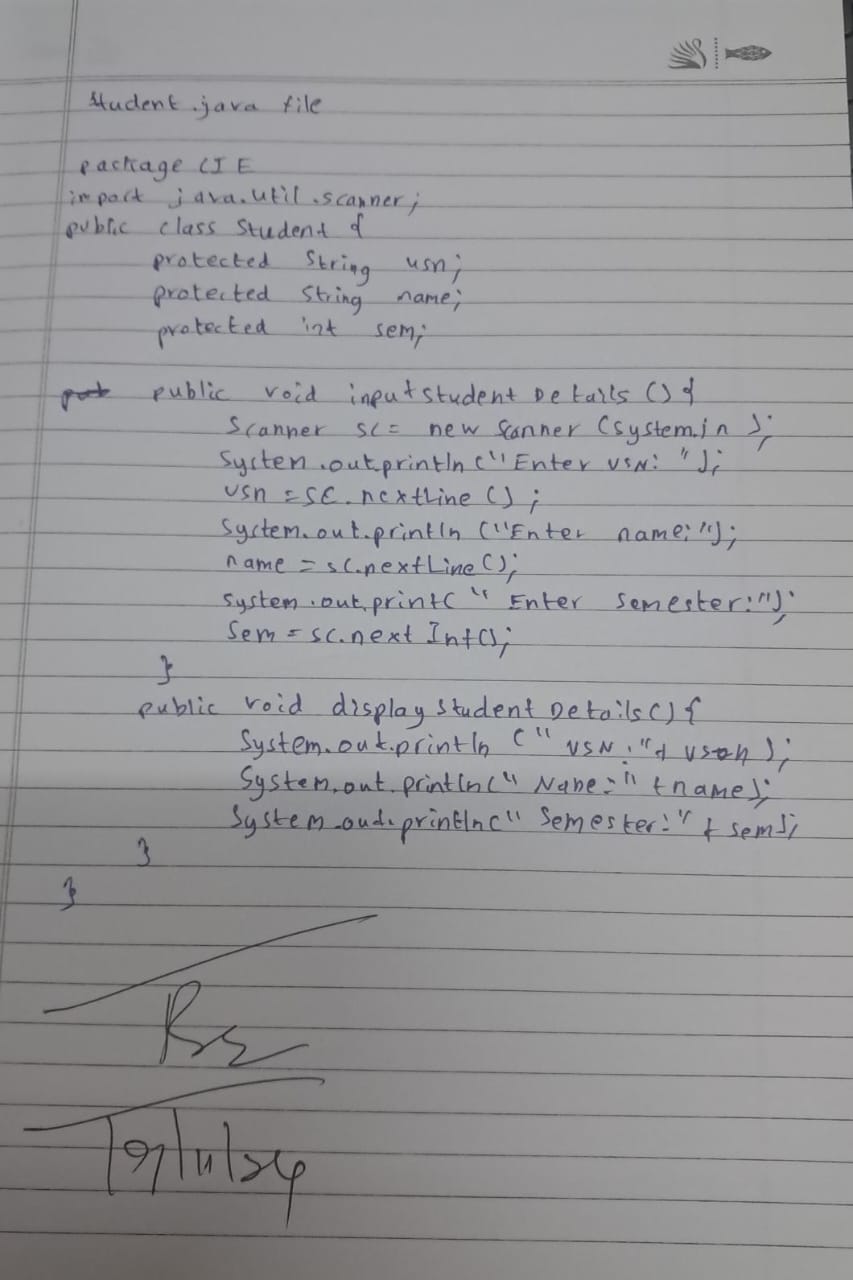
}

}  


**Program 6**Create a package CIE which has two classes - Personal 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 Personal. 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:





Code:  
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();

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();

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();

}

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

int[] seeMarks = new int[5];

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

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

seeMarks[j] = scanner.nextInt();

}

externals[i] = new External(usn, name, sem, seeMarks);

}

System.out.println("\nFinal Marks of Students:");

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

internals[i].display();

internals[i].displayInternalMarks();

externals[i].displaySeeMarks();

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

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

int finalMarks = internals[i].internalMarks[j] + externals[i].seeMarks[j];

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

}

System.out.println();

}

scanner.close();

}

}  
package CIE;

public class Student {

public String usn;

public String name;

public 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 extends Student {

public int[] internalMarks;

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

super(usn, name, sem);

this.internalMarks = internalMarks;

}

public void displayInternalMarks() {

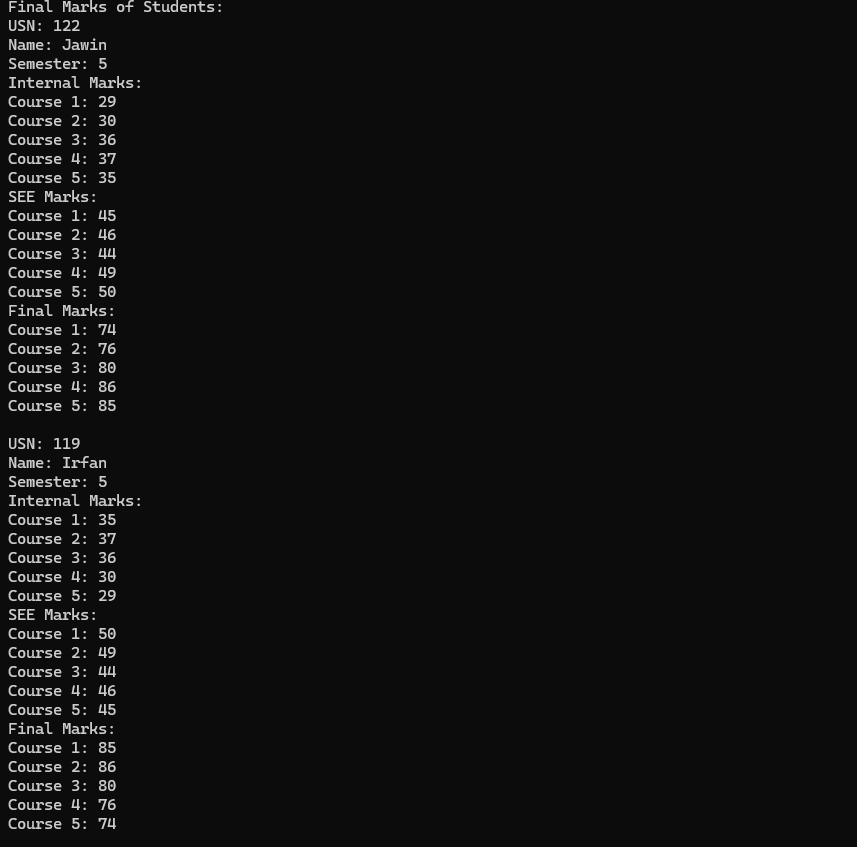
System.out.println("Internal Marks: ");

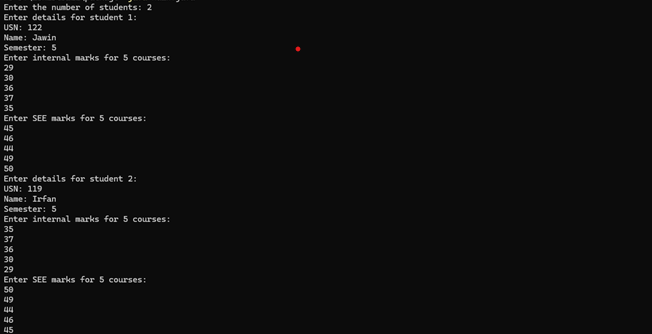
for (int i = 0; i < internalMarks.length; i++) {

System.out.println("Course " + (i + 1) + ": " + internalMarks[i]);

}

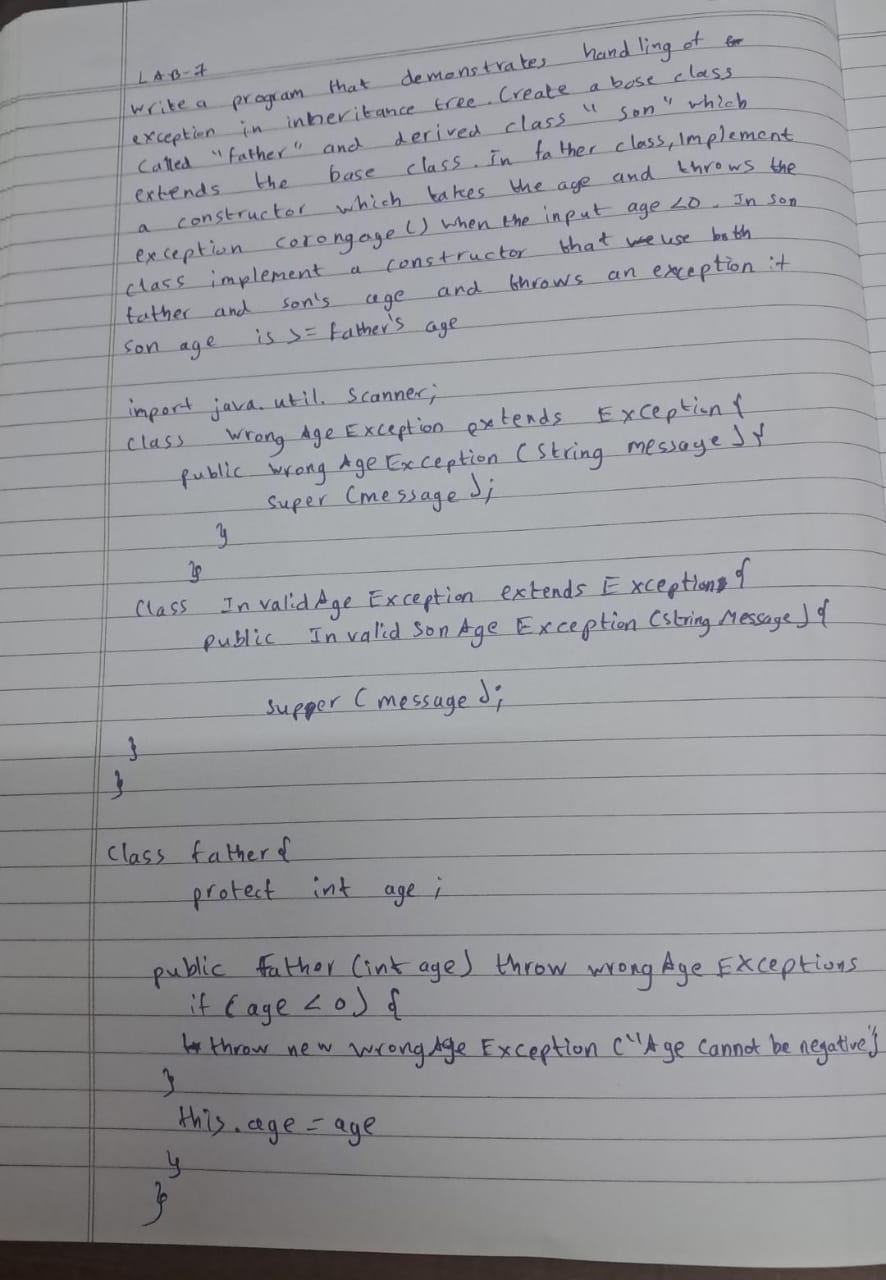
}

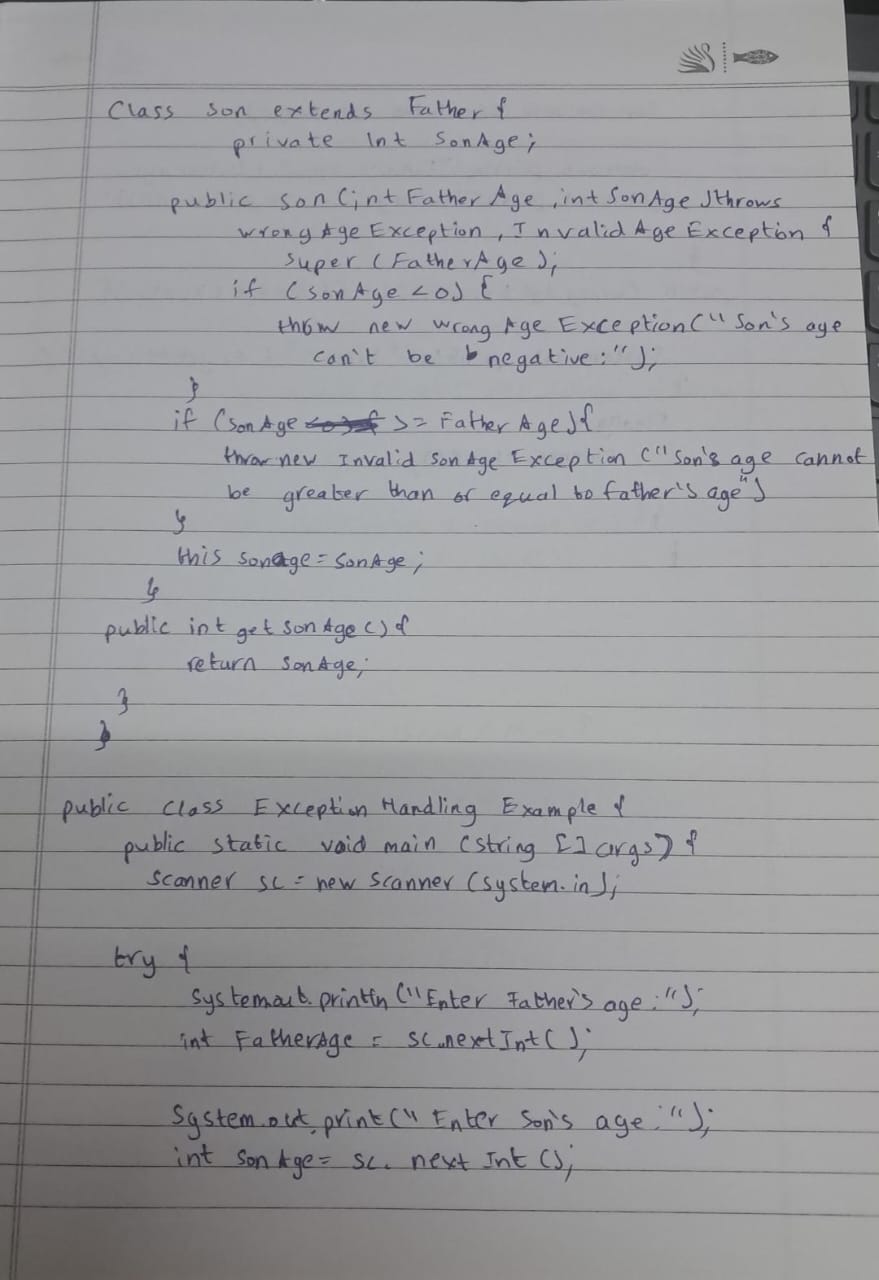
}  


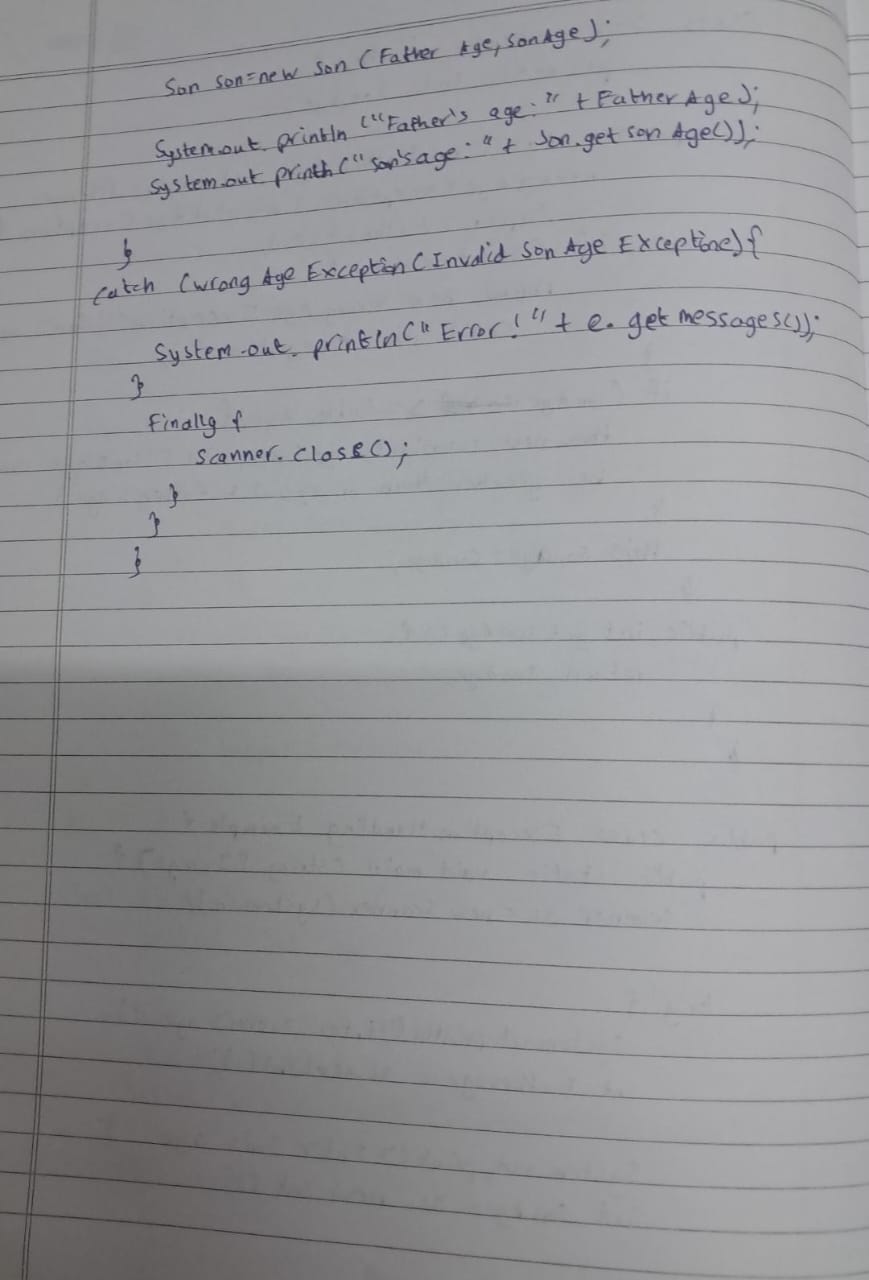


**Program 7**Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age=father’s age.

Algorithm:







Code:

import java.util.Scanner;

class WrongAgeException extends Exception {

public WrongAgeException(String message) {

super(message);

}

}

class Father {

protected int fatherAge;

public Father(int age) throws WrongAgeException {

if (age < 0) {

throw new WrongAgeException("Father's age cannot be negative!");

}

this.fatherAge = age;

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

}

}

class Son extends Father {

private int sonAge;

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

super(fatherAge);

if (sonAge < 0) {

throw new WrongAgeException("Son's age cannot be negative!");

}

if (sonAge >= fatherAge) {

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

}

this.sonAge = sonAge;

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

}

}

public class ExceptionInInheritance {

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);

} catch (WrongAgeException e) {

System.err.println("Exception: " + e.getMessage());

} catch (Exception e) {

System.err.println("Invalid input! Please enter integers.");

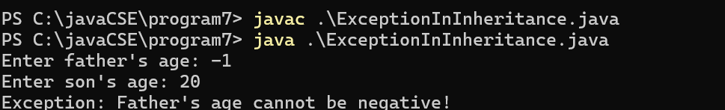
} finally {

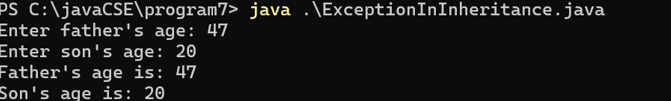
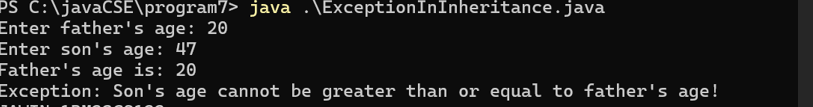
scanner.close();

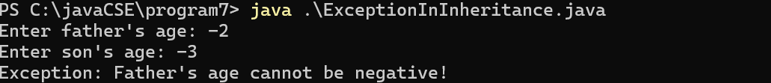
}

}

}

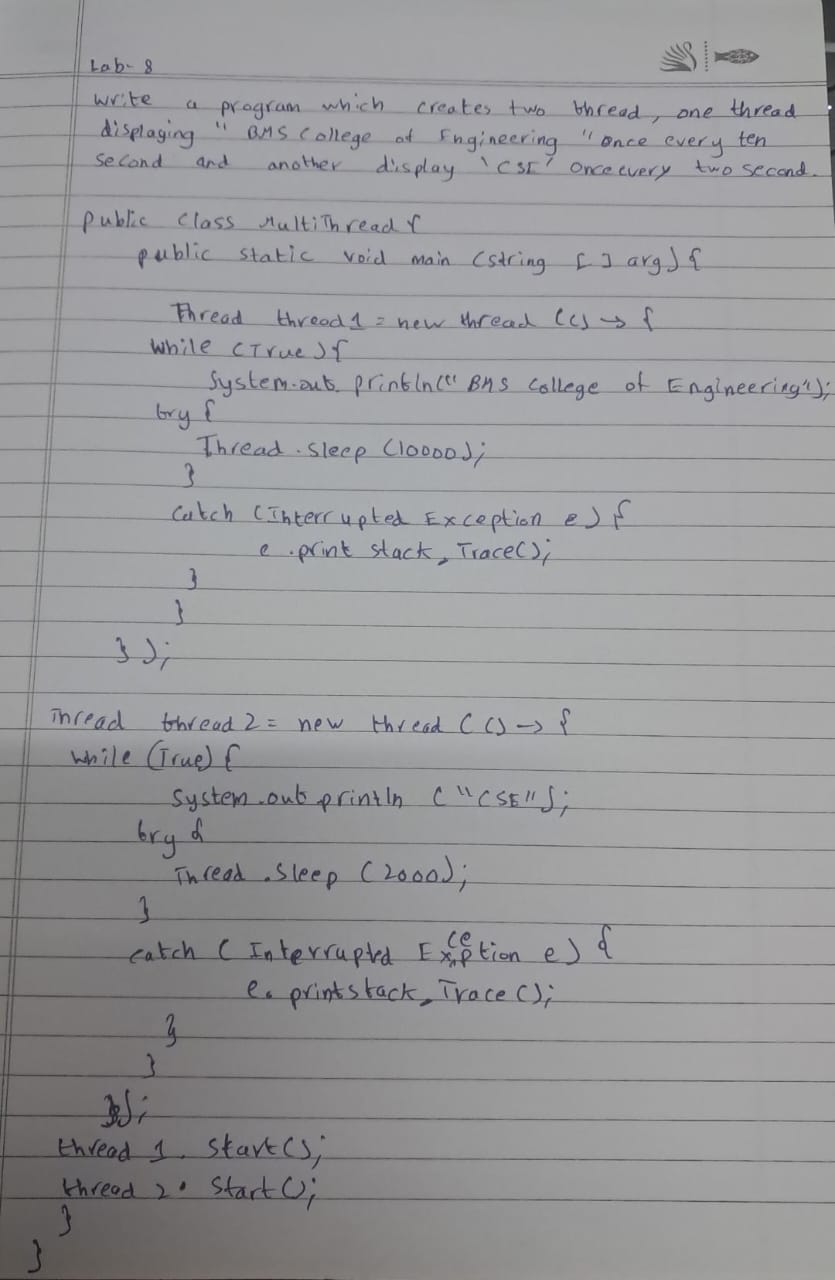




****

**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:



Code:  
class DisplayMessage extends Thread {

private String message;

private int delay;

public DisplayMessage(String message, int delay) {

this.message = message;

this.delay = delay;

}

public void run() {

try {

while (true) {

System.out.println(message);

Thread.sleep(delay);

}

} catch (InterruptedException e) {

System.err.println("Thread interrupted: " + e.getMessage());

}

}

}

public class MultiThreadDisplay {

public static void main(String[] args) {

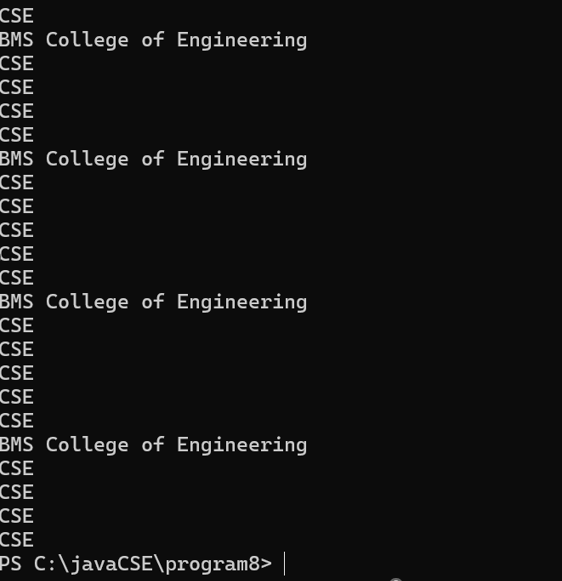
DisplayMessage thread1 = new DisplayMessage("BMS College of Engineering", 10000);

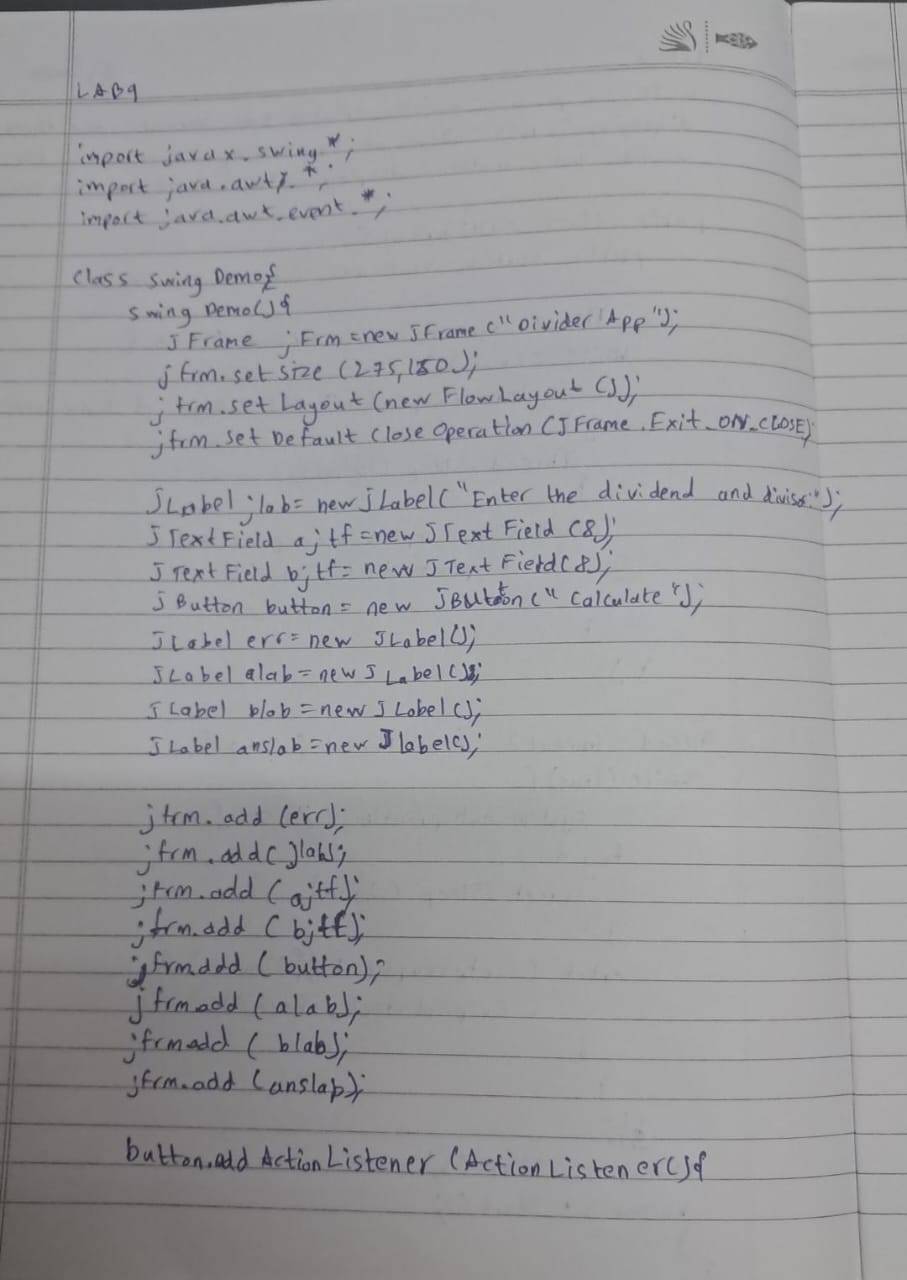
DisplayMessage thread2 = new DisplayMessage("CSE", 2000);

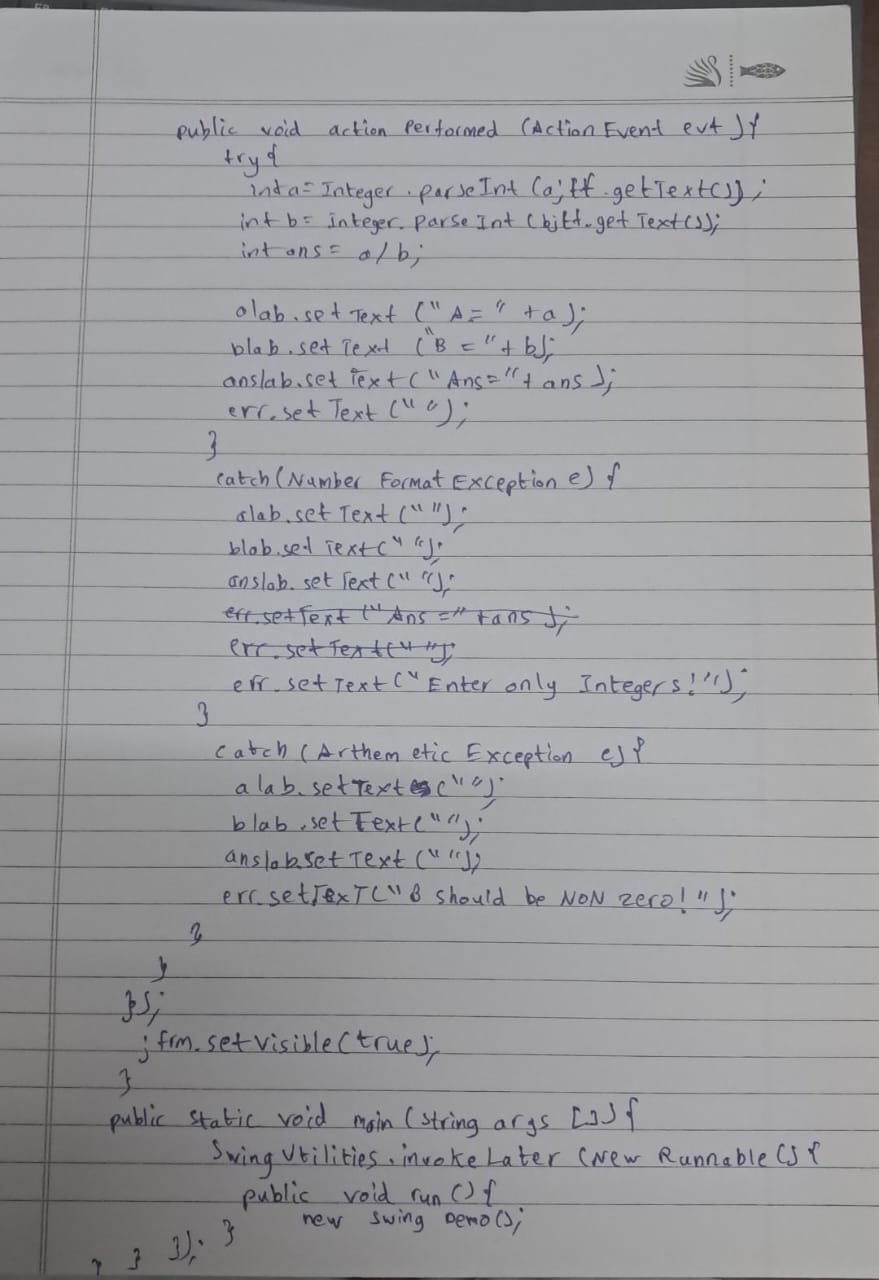
thread1.start();

thread2.start();

}



**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 boxAlgorithm:  




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 dividend and divisor:");

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);

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("A = " + a);

blab.setText("B = " + b);

anslab.setText("Ans = " + ans);

err.setText("");

} 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() {

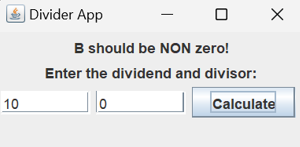
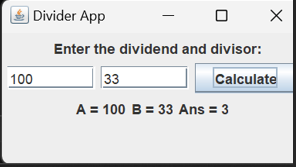
public void run() {

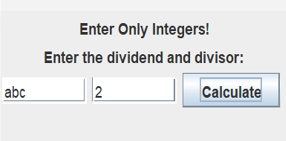
new SwingDemo();

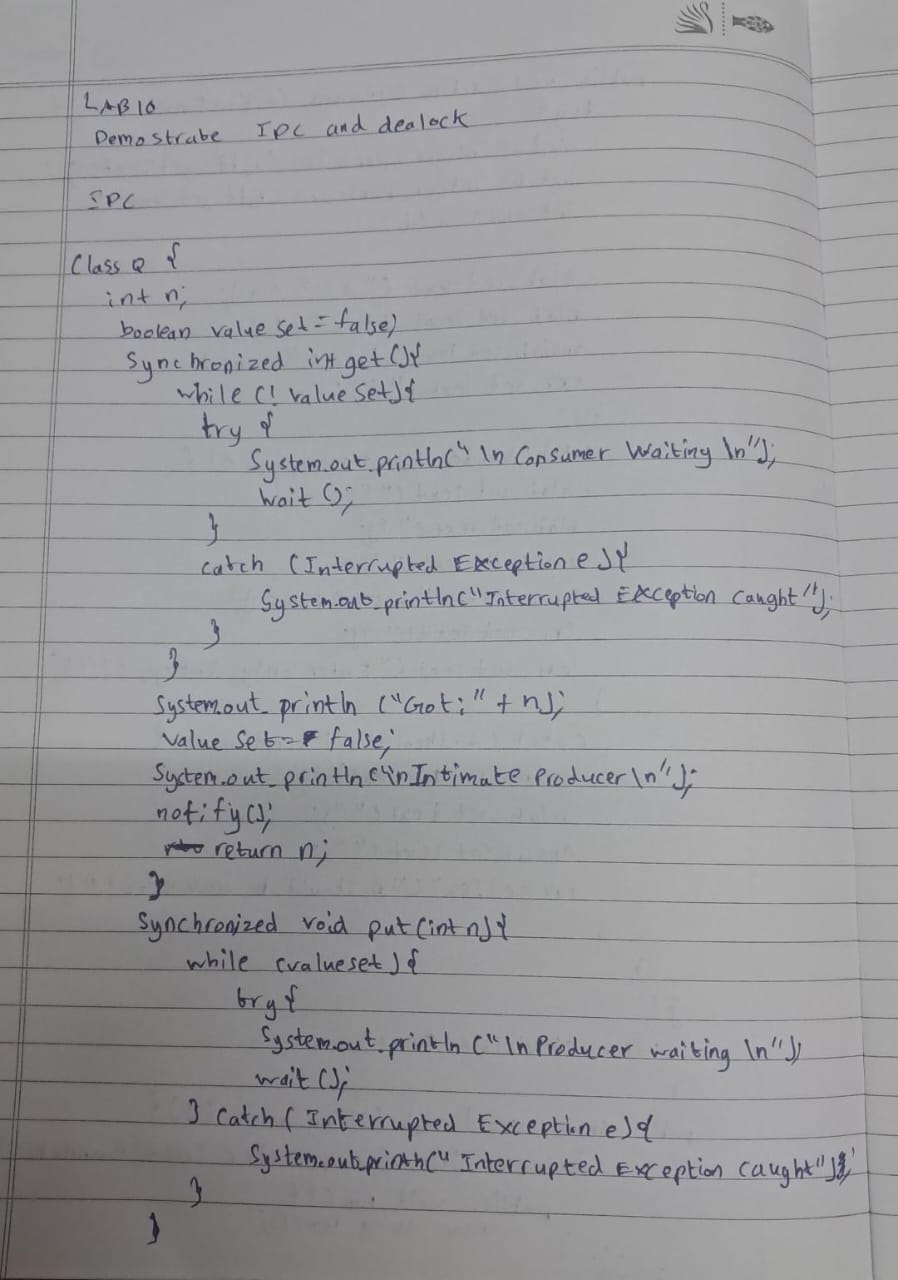
}

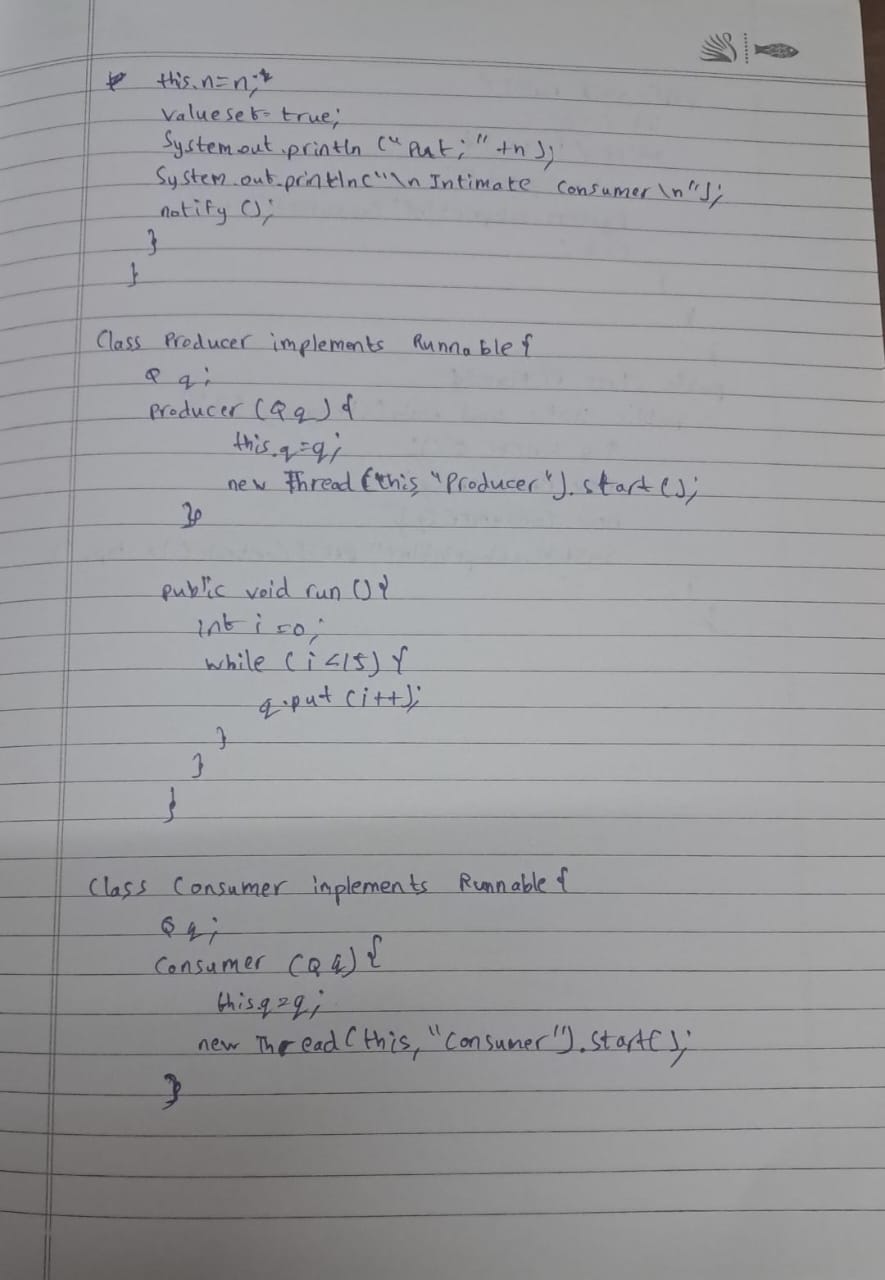
});

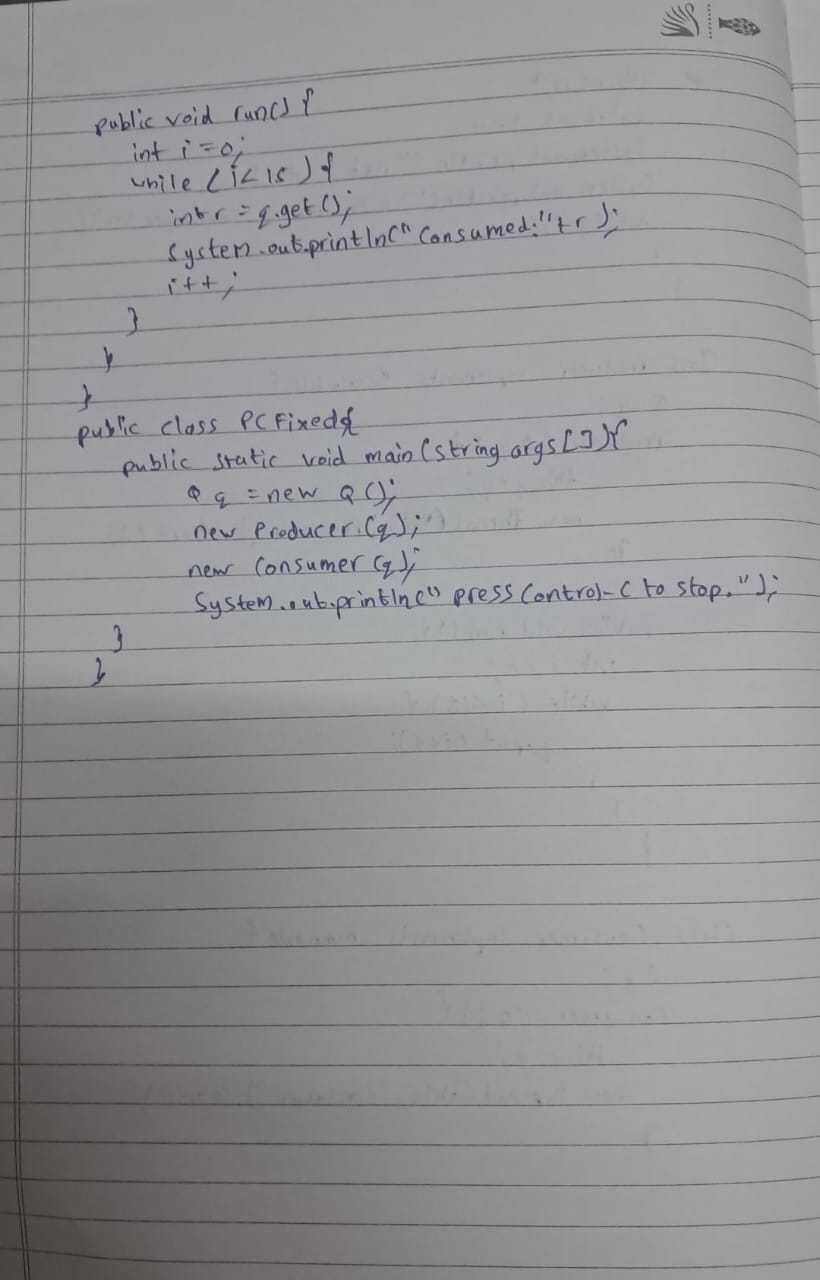
}

}  




**Program 10**Demonstrate Inter process Communication and deadlock  
IPC  
Algorithm:  






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++;

}

}

}

public class PCFixed {

public static void main(String args[]) {

Q q = new Q();

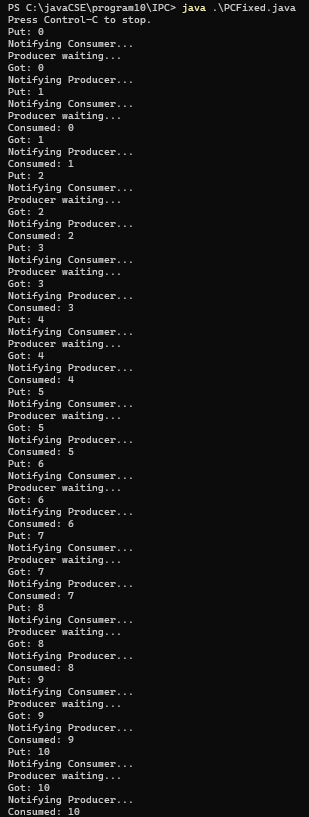
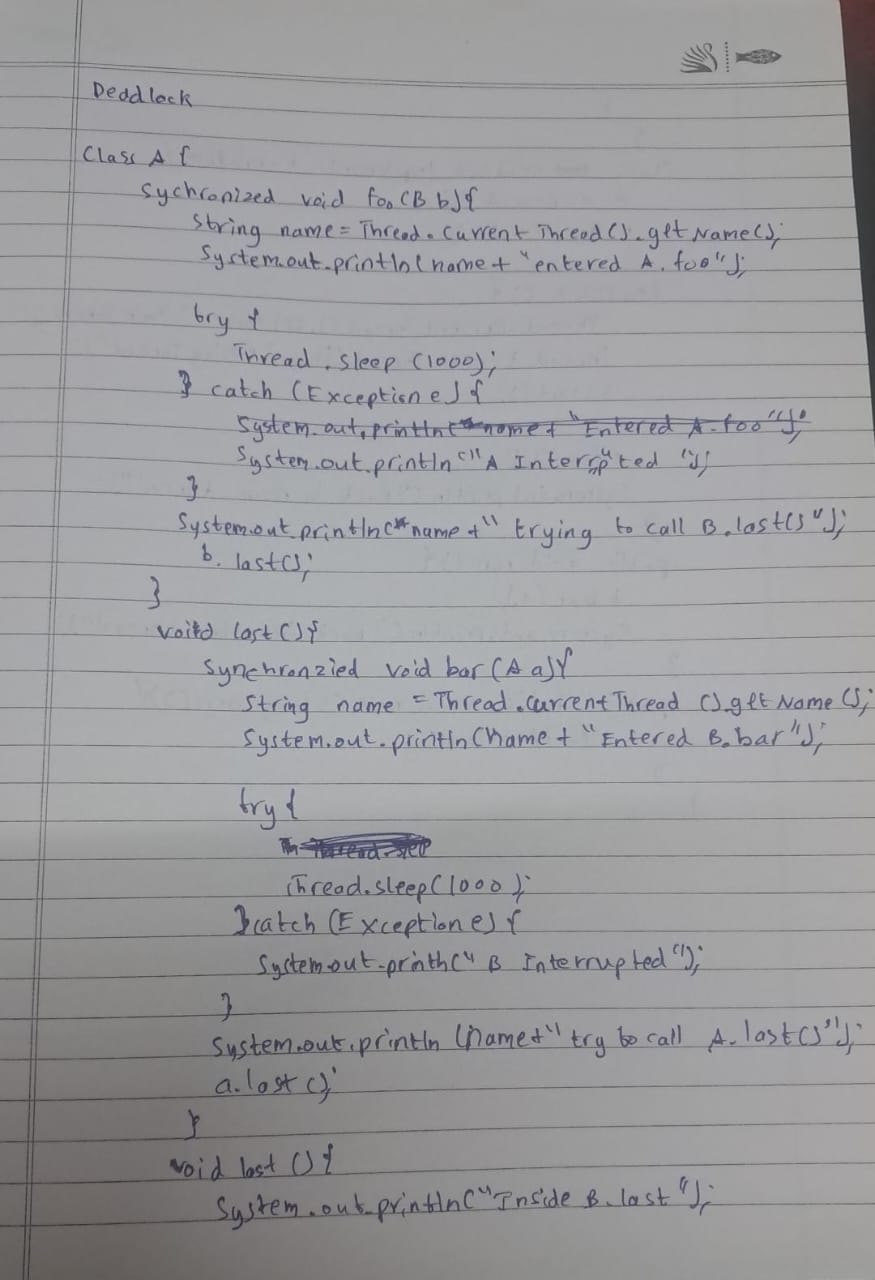
new Producer(q);

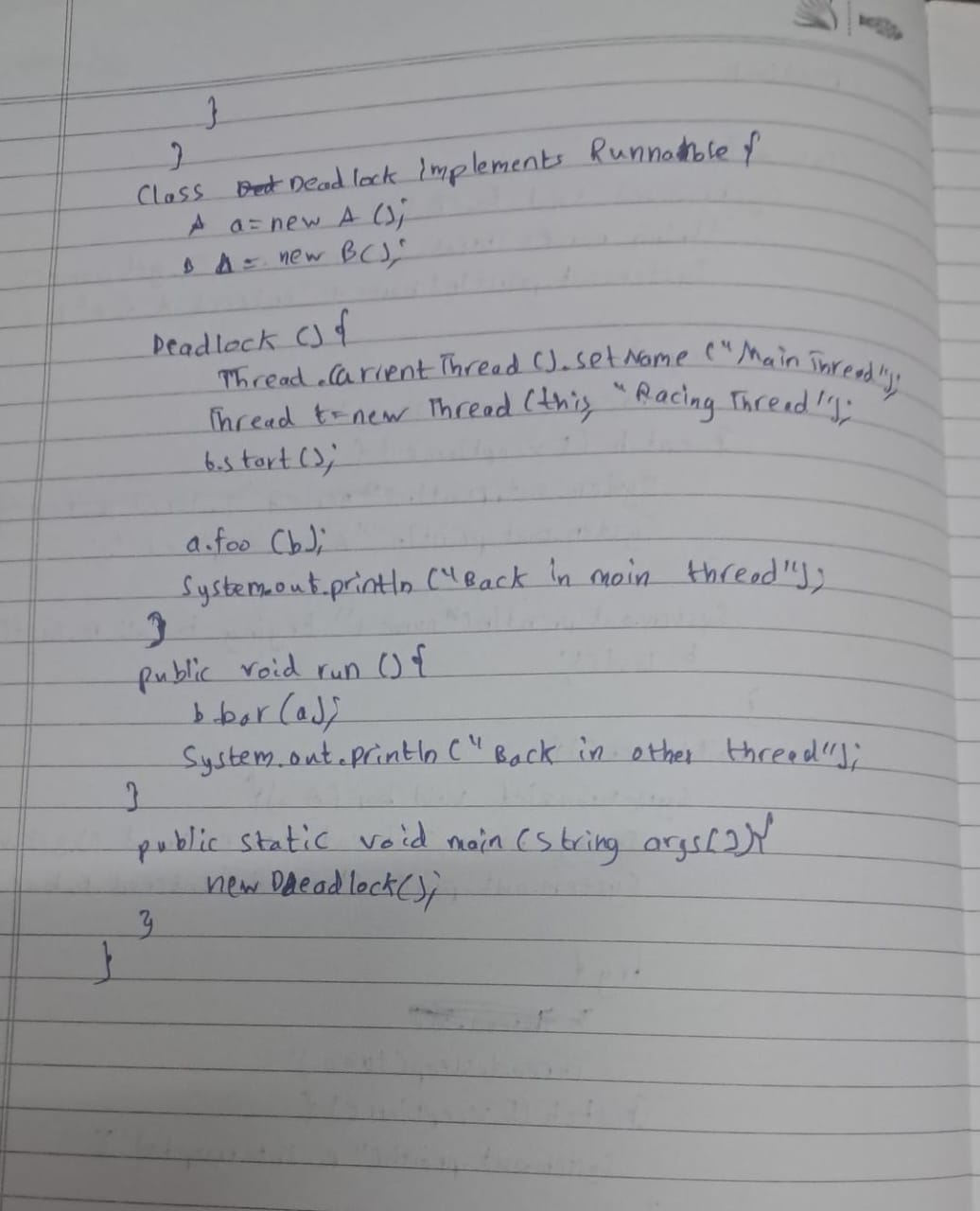
new Consumer(q);

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

}

}

  
DeadLock  
Algorithm:  




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();

}

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();

}

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);

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

}

public void run() {

b.bar(a);

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

}

public static void main(String args[]) {

new Deadlock();

}

}  
