**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**Object Oriented Java Programming**

**(23CS3PCOOJ)**

***Submitted by***

**Gayathri S (24BECE417)**

***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 **Gayathri S (24BECS417),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

|  |  |
| --- | --- |
| Lab faculty Swathi Sridharan  Assistant Professor  Department of CSE, BMSCE | Dr. Joythi S Nayak  Professor & HOD  Department of CSE, BMSCE |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 01-10-2024 | Quadratic Solver | 4-5 |
| 2 | 08-10-2024 | Student SGPA Calculator | 6-8 |
| 3 | 15-10-2024 | Book Collection | 9-12 |
| 4 | 22-10-2024 | Shape Area Calculator | 12-15 |
| 5 | 29-10-2024 | Bank Account Manager | 15-19 |
| 6 | 13-11-2024 | Final Marks Calculator (Packages) | 19-22 |
| 7 | 19-11-2024 | Inheritance Exception Demo | 23-25 |
| 8 | 26-11-2024 | Thread Demo | 25-28 |

Github Link:

“https://github.com/GayathriS-CSE/OOJ”

Lab 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.

**CODE:**

import java.util.Scanner;

import java.lang.Math;

public class QuadraticEquationSolver {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter coefficient A:");

double a = scanner.nextDouble();

System.out.println("Enter coefficient B:");

double b = scanner.nextDouble();

System.out.println("Enter coefficient C:");

double c = scanner.nextDouble();

double discriminant = b \* b - 4 \* a \* c;

if (discriminant >= 0) {

double root1 = (-b + Math.sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.sqrt(discriminant)) / (2 \* a);

if (discriminant == 0) {

System.out.println("One real solution: " + root1);

} else {

System.out.println("Real solutions:");

System.out.println("Root 1: " + root1);

System.out.println("Root 2: " + root2);

}

} else {

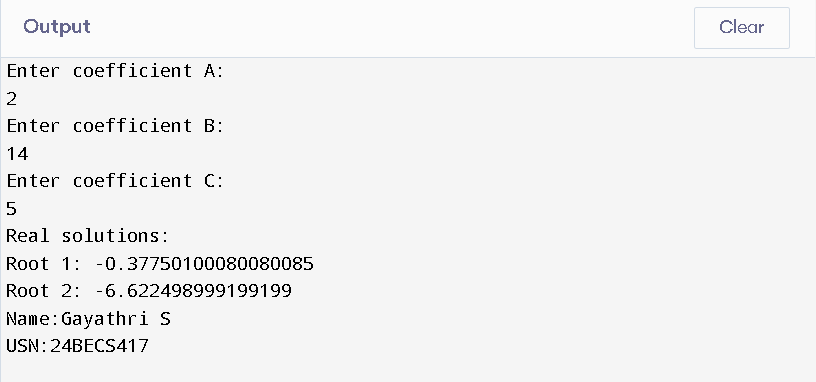
System.out.println("No real solutions.");

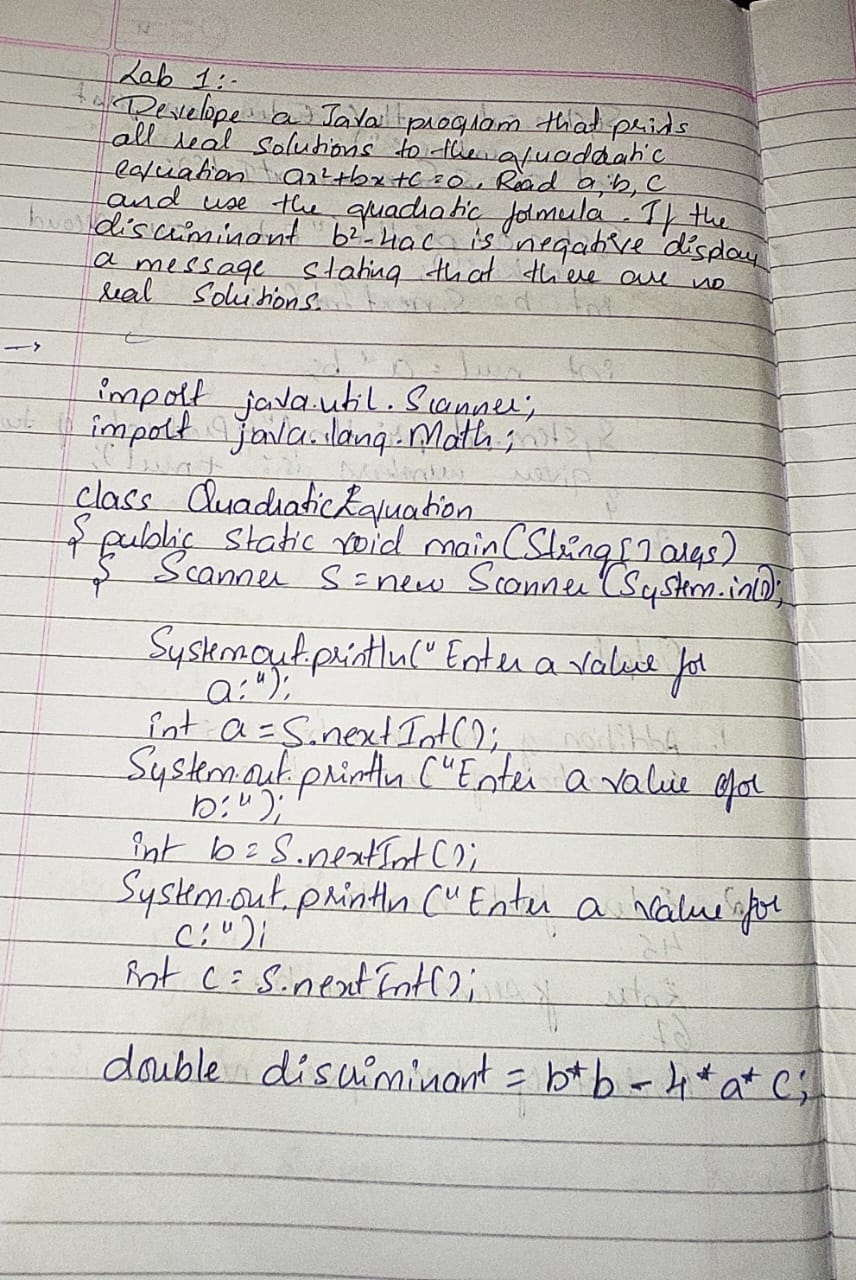
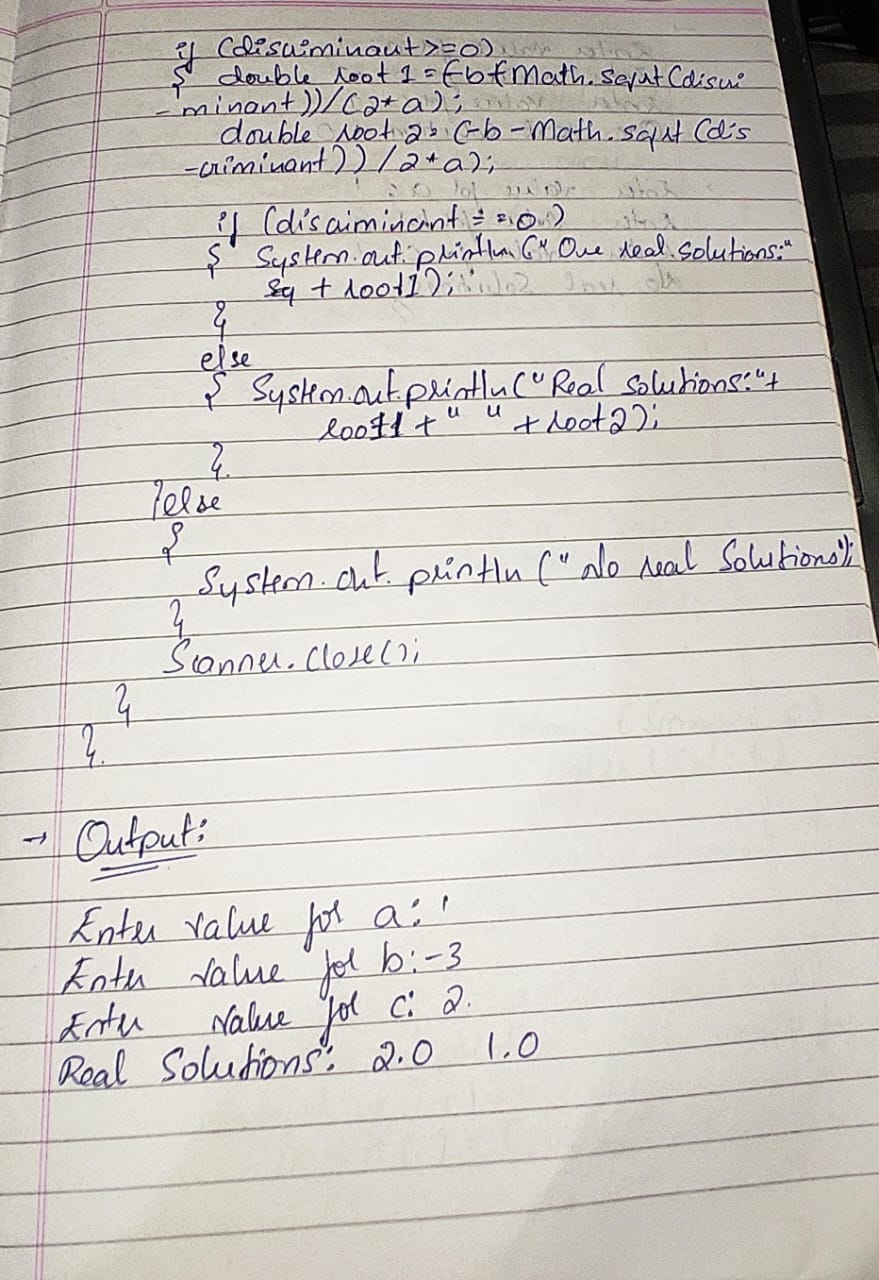
}

scanner.close();

}

}

OUTPUT: 



Lab 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.

**CODE:**

**import java.util.Scanner;**

**public class Student {**

**String usn, name;**

**int[] credits, marks;**

**public Student(int numSubjects) {**

**credits = new int[numSubjects];**

**marks = new int[numSubjects];**

**}**

**public void acceptDetails(Scanner scanner) {**

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

**usn = scanner.next();**

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

**name = scanner.next();**

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

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

**credits[i] = scanner.nextInt();**

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

**marks[i] = scanner.nextInt();**

**}**

**}**

**public void displayDetails() {**

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

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

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

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

**}**

**}**

**public double calculateSGPA() {**

**double totalCredits = 0, totalGradePoints = 0;**

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

**double gradePoint = marks[i] / 10;**

**totalCredits += credits[i];**

**totalGradePoints += gradePoint \* credits[i];**

**}**

**return totalGradePoints / totalCredits;**

**}**

**public static void main(String[] args) {**

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

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

**int numSubjects = scanner.nextInt();**

**Student student = new Student(numSubjects);**

**student.acceptDetails(scanner);**

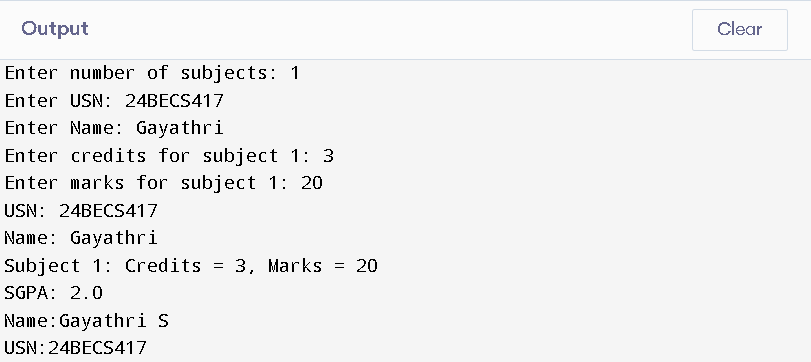
**student.displayDetails();**

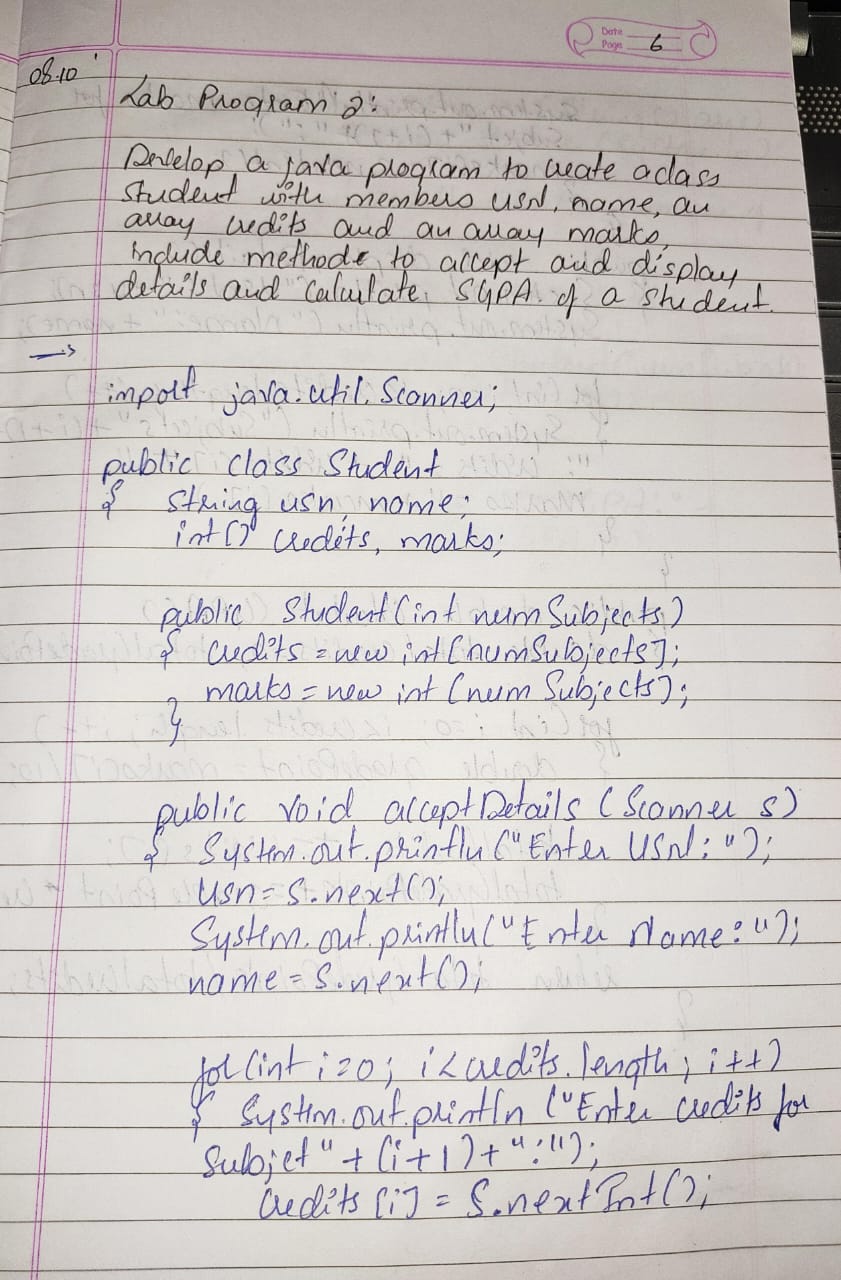
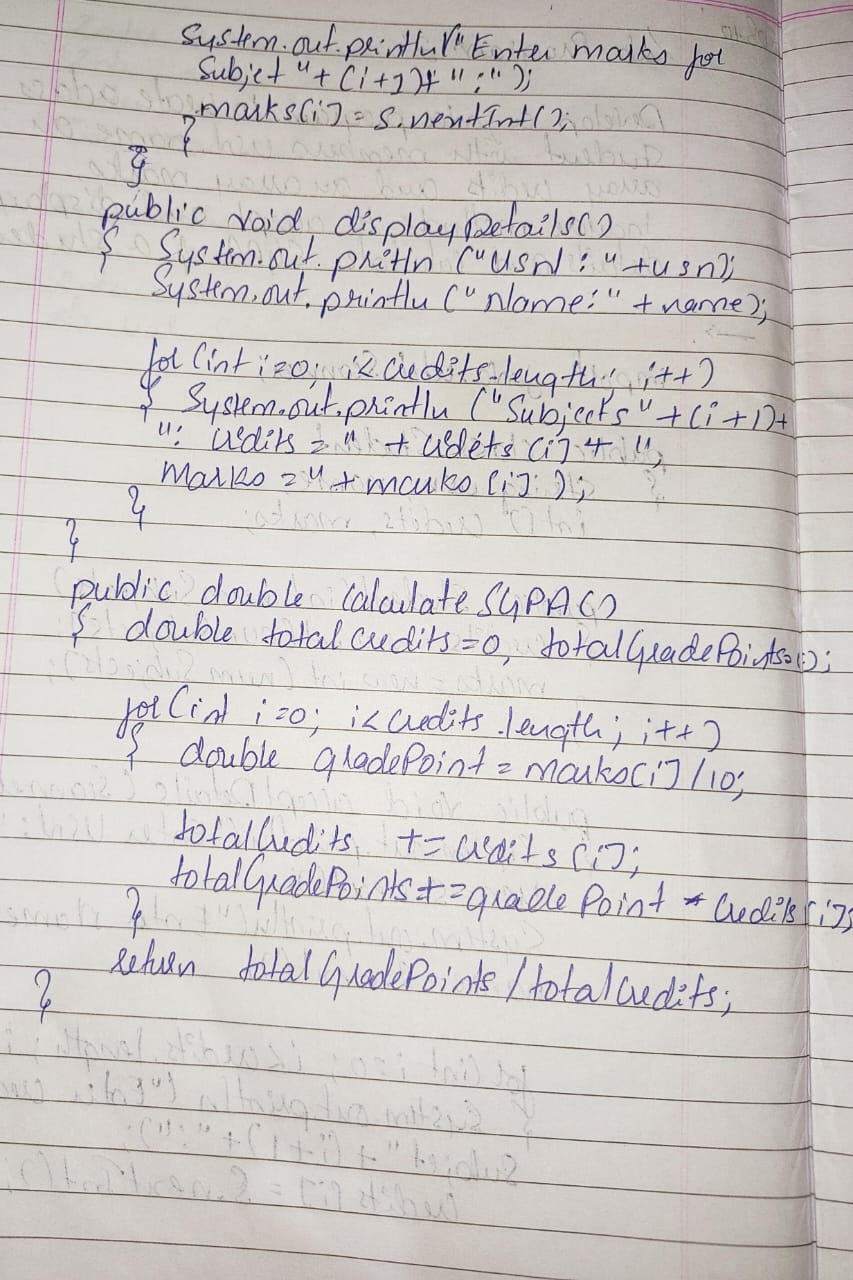
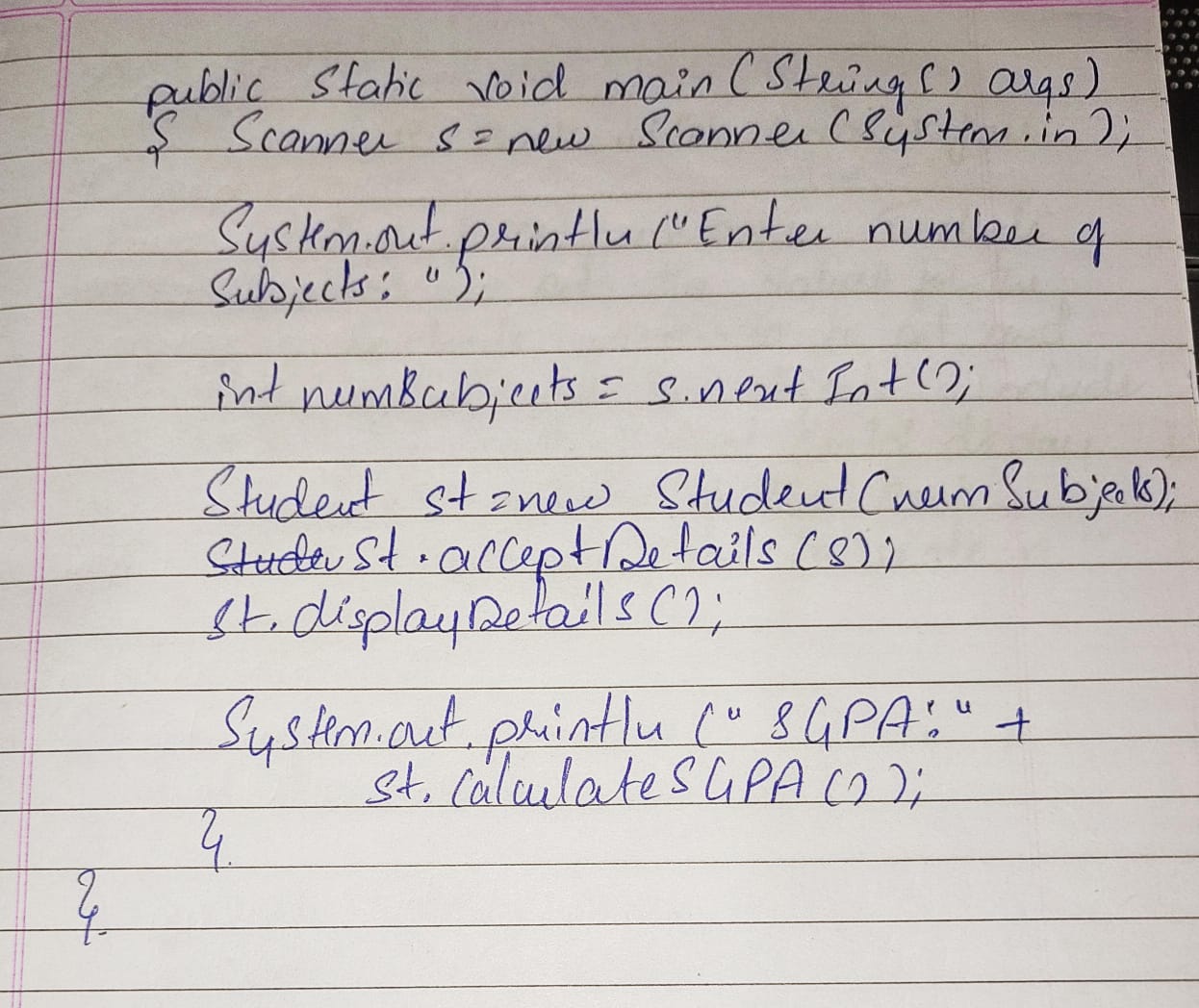
**System.out.println("SGPA: " + student.calculateSGPA());**

**}**

**}**

OUTPUT:





Lab 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.

**CODE:**

import java.util.Scanner;

class Book {

private String name;

private String author;

private double price;

private int numPages;

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

this.name = name;

this.author = author;

this.price = price;

this.numPages = numPages;

}

public void setName(String name) {

this.name = name;

}

public void setAuthor(String author) {

this.author = author;

}

public void setPrice(double price) {

this.price = price;

}

public void setNumPages(int numPages) {

this.numPages = numPages;

}

public String getName() {

return name;

}

public String getAuthor() {

return author;

}

public double getPrice() {

return price;

}

public int getNumPages() {

return numPages;

}

@Override

public String toString() {

return "Book Details:\n" +

"Name: " + name + "\n" +

"Author: " + author + "\n" +

"Price: " + price + "\n" +

"Number of Pages: " + numPages + "\n";

}

}

public class Test {

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("\nEnter details for book " + (i + 1) + ":");

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

String name = scanner.nextLine();

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

String author = scanner.nextLine();

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

double price = scanner.nextDouble();

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

int numPages = scanner.nextInt();

scanner.nextLine();

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

}

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

for (Book book : books) {

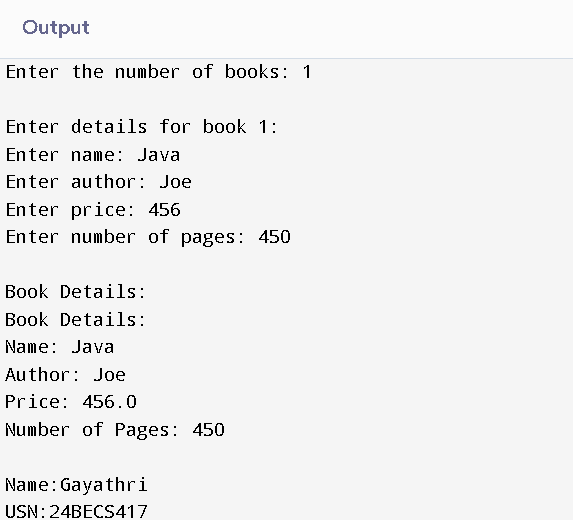
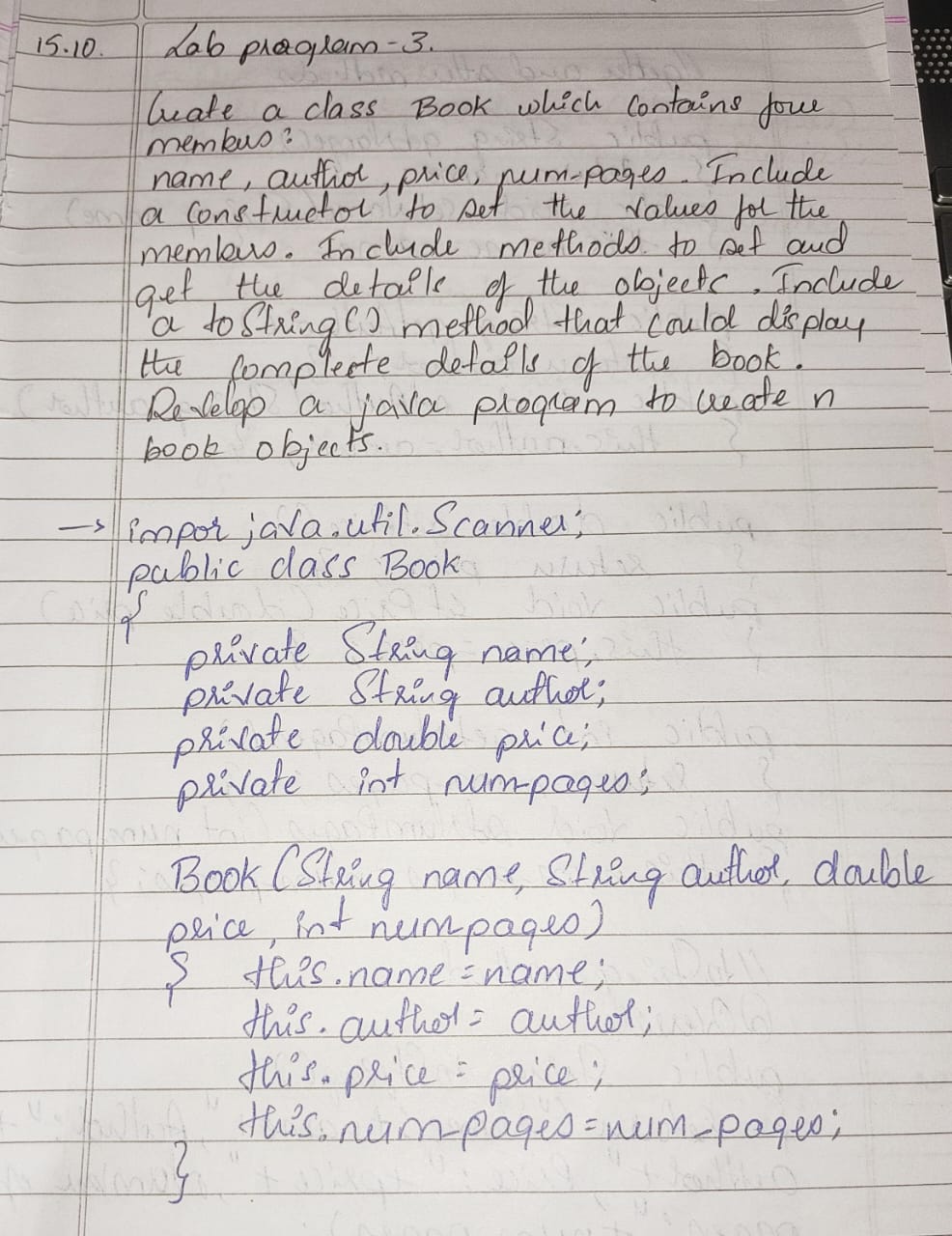
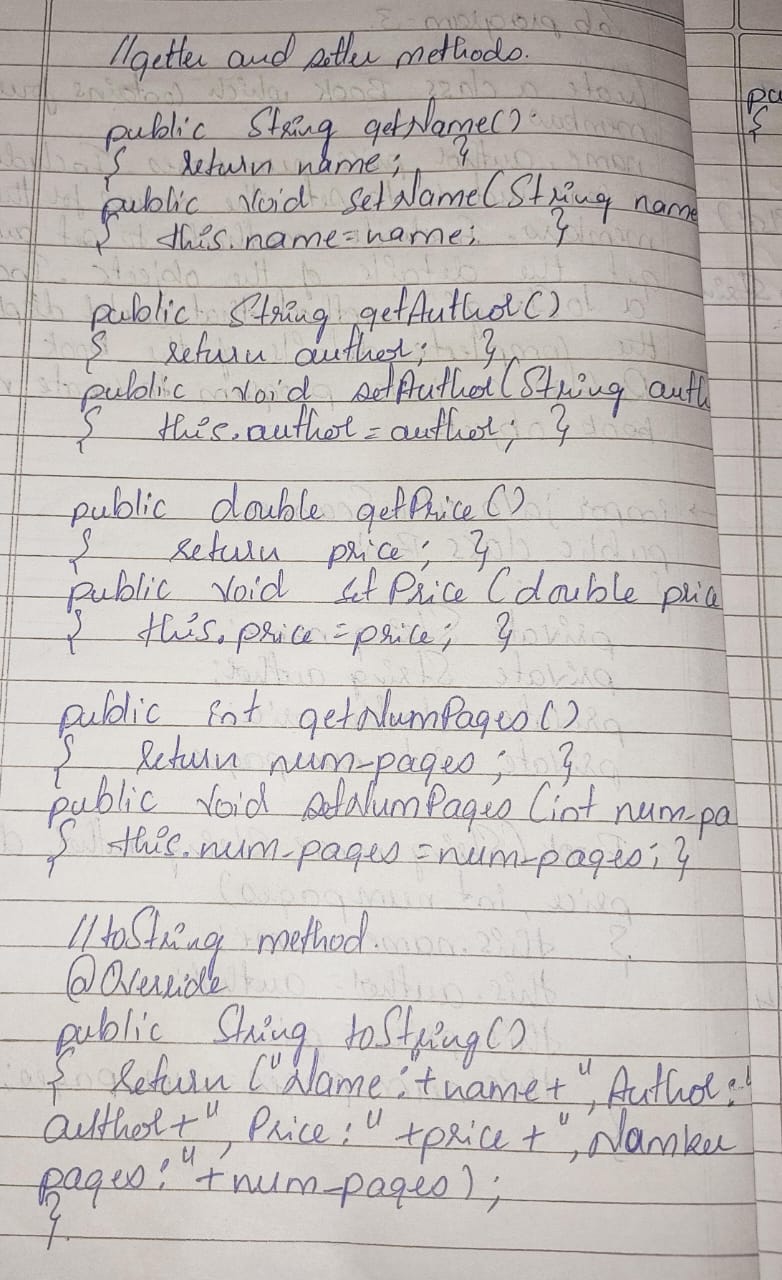
System.out.println(book);

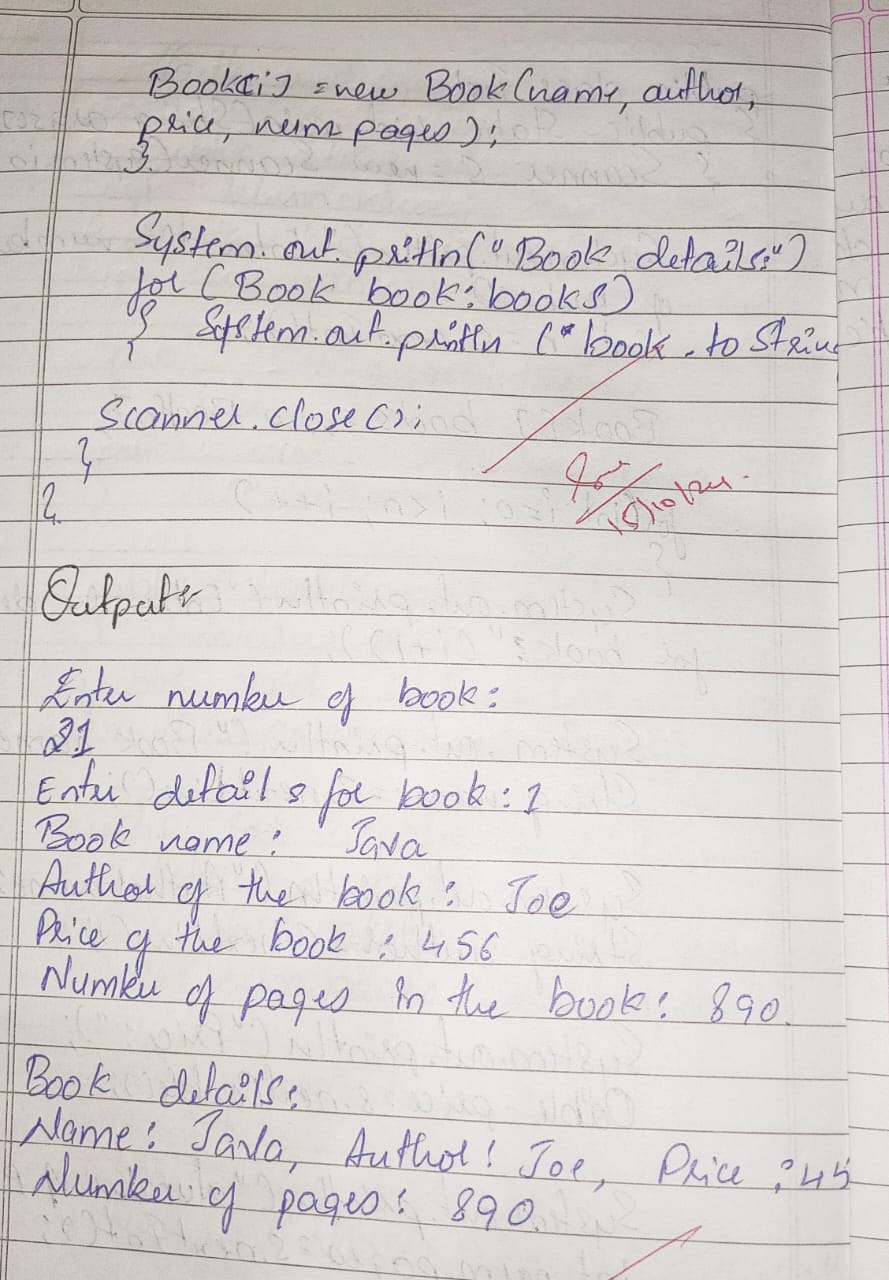
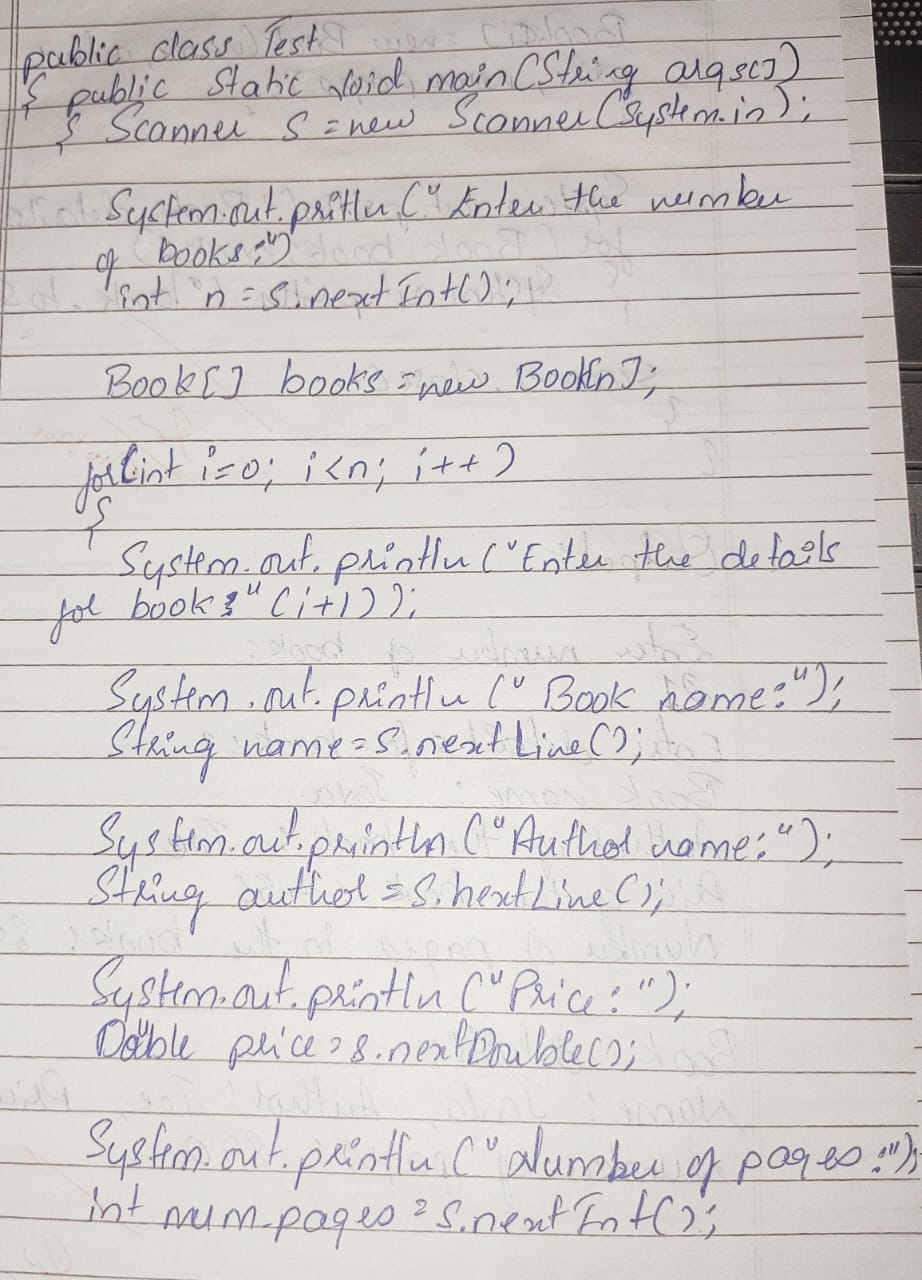
}

}

}

OUTPUT:





Lab 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.

**CODE:**

**import java.lang.Math;**

**abstract class Shape**

**{**

**protected int val1;**

**protected int val2;**

**Shape(int val1,int val2)**

**{**

**this.val1=val1;**

**this.val2=val2;**

**}**

**abstract void printArea();**

**}**

**class Rectangle extends Shape**

**{**

**int area;**

**Rectangle(int val1,int val2)**

**{**

**super(val1,val2);**

**}**

**void printArea()**

**{**

**area=val1\*val2;**

**System.out.println("Area of the rectangle:"+area);**

**}**

**}**

**class Triangle extends Shape**

**{**

**double area;**

**Triangle(int val1,int val2)**

**{**

**super(val1,val2);**

**}**

**void printArea()**

**{**

**area=0.5\*val1\*val2;**

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

**}**

**}**

**class Circle extends Shape**

**{**

**double area;**

**Circle(int val1,int val2)**

**{**

**super(val1,0);**

**}**

**void printArea()**

**{**

**area=Math.PI\*val1\*val1;**

**System.out.println("Area of the circle:"+area);**

**}**

**}**

**public class Test**

**{**

**public static void main(String args[])**

**{**

**Rectangle r=new Rectangle(12,6);**

**r.printArea();**

**Triangle t=new Triangle(34,6);**

**t.printArea();**

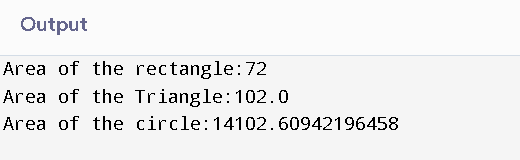
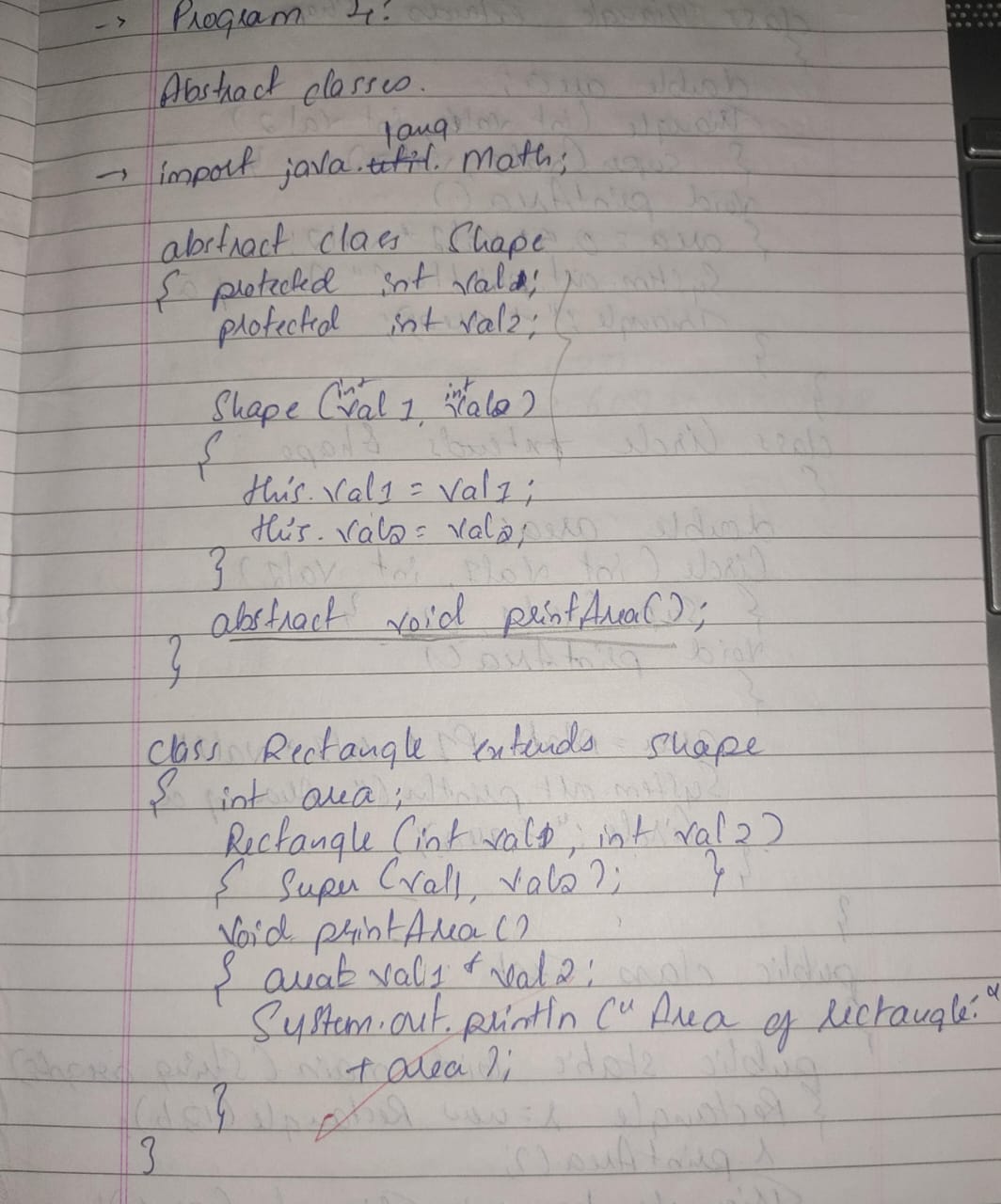
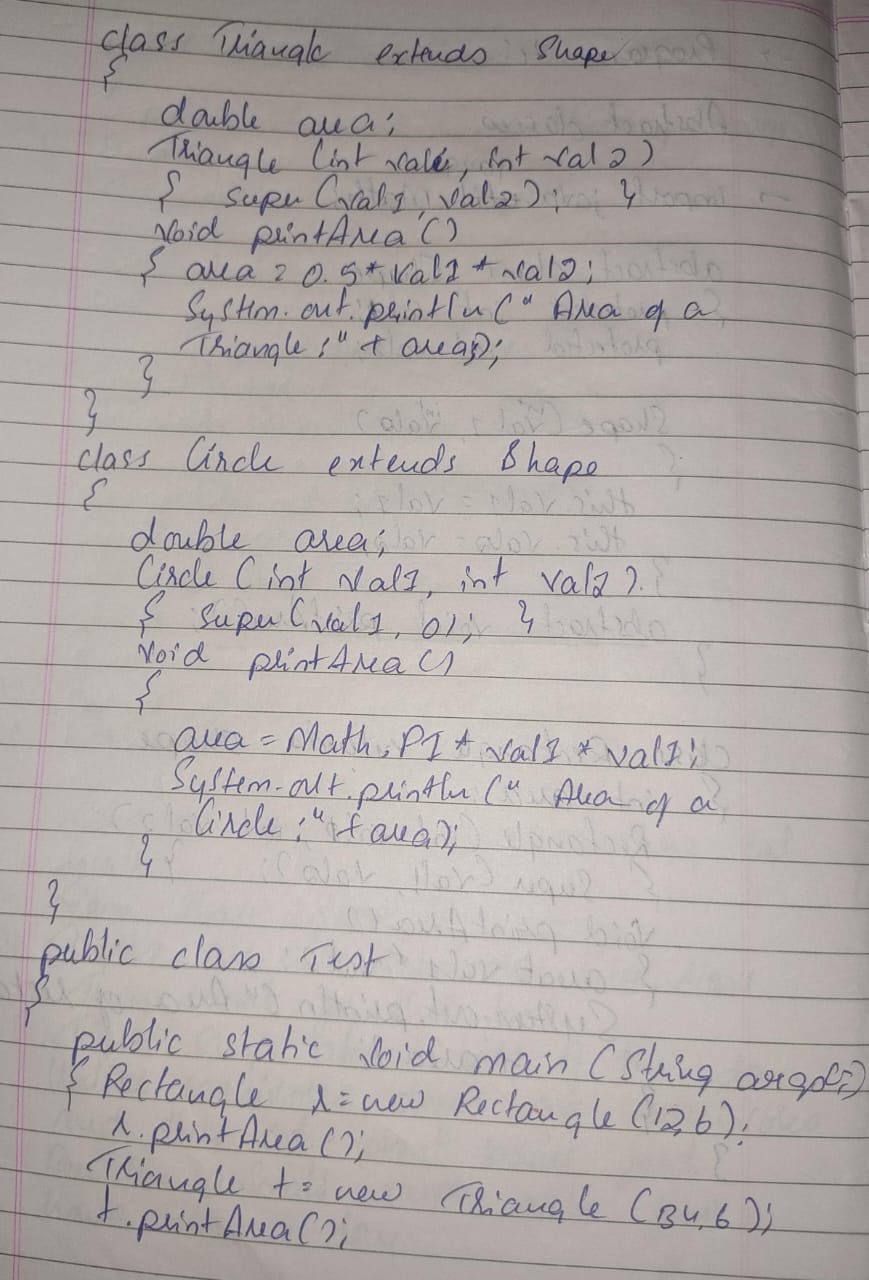
**Circle c=new Circle(67,0);**

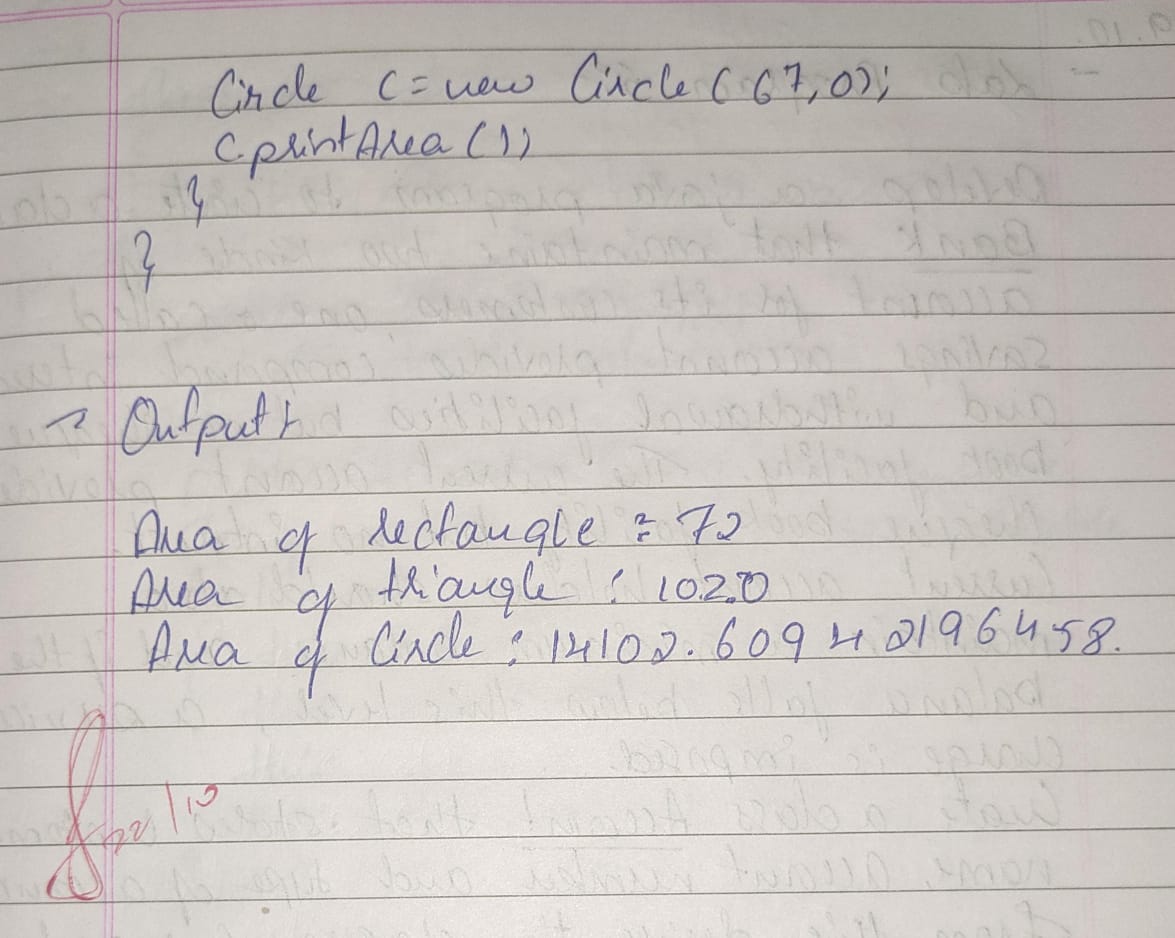
**c.printArea();**

**}**

**}**

OUTPUT:





Lab 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.

**CODE:**

class Account {

public String customerName;

public String accountNumber;

protected double balance;

public Account(String customerName, String accountNumber) {

this.customerName = customerName;

this.accountNumber = accountNumber;

this.balance = 0.0;

}

public void deposit(double amount) {

balance += amount;

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

}

public void displayBalance() {

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

}

public void withdraw(double amount) {

if (amount <= balance) {

balance -= amount;

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

} else {

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

}

}

protected double getBalance() {

return balance;

}

}

class SavAcct extends Account {

private double interestRate;

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

super(customerName, accountNumber);

this.interestRate = interestRate;

}

public void computeAndDepositInterest() {

double currentBalance = getBalance();

double interest = currentBalance \* interestRate / 100;

deposit(interest);

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

}

public String toString()

{ return "Customer Name: "+customerName+"\nAccount Number: "+accountNumber; }

}

class CurAcct extends Account {

private double minimumBalance;

private double serviceCharge;

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

super(customerName, accountNumber);

this.minimumBalance = minimumBalance;

this.serviceCharge = serviceCharge;

}

public void withdraw(double amount) {

if (getBalance() - amount < minimumBalance) {

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

deposit(-serviceCharge);

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

} else {

super.withdraw(amount);

}

}

public String toString()

{ return "Customer Name: "+customerName+"\nAccount Number: "+accountNumber; }

}

public class Bank {

public static void main(String[] args) {

SavAcct savingsAccount = new SavAcct("Alice", "S12345", 5.0);

System.out.println("Customer details:\n"+savingsAccount.toString());

System.out.println("\nTransaction details:");

savingsAccount.deposit(1000);

savingsAccount.computeAndDepositInterest();

savingsAccount.displayBalance();

savingsAccount.withdraw(500);

savingsAccount.displayBalance();

System.out.println();

CurAcct currentAccount = new CurAcct("Bob", "C12345", 1000, 50);

System.out.println("Customer details:\n"+currentAccount.toString());

System.out.println("\nTransaction details:");

currentAccount.deposit(2000);

currentAccount.displayBalance();

currentAccount.withdraw(1900);

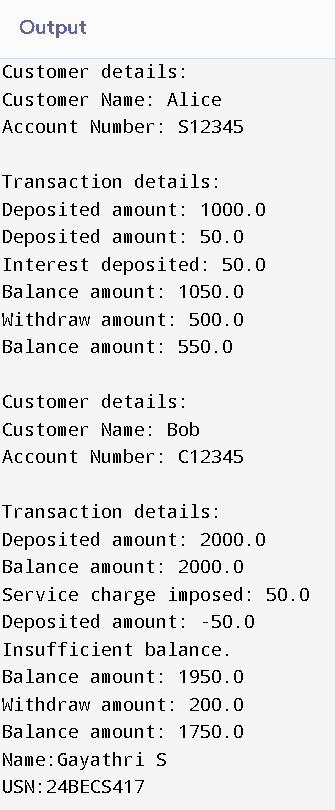
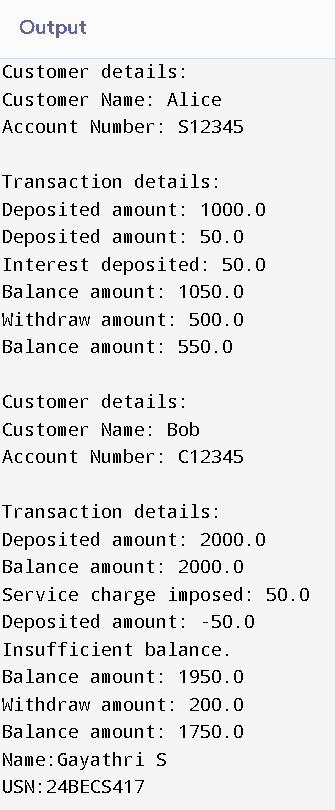
currentAccount.displayBalance();

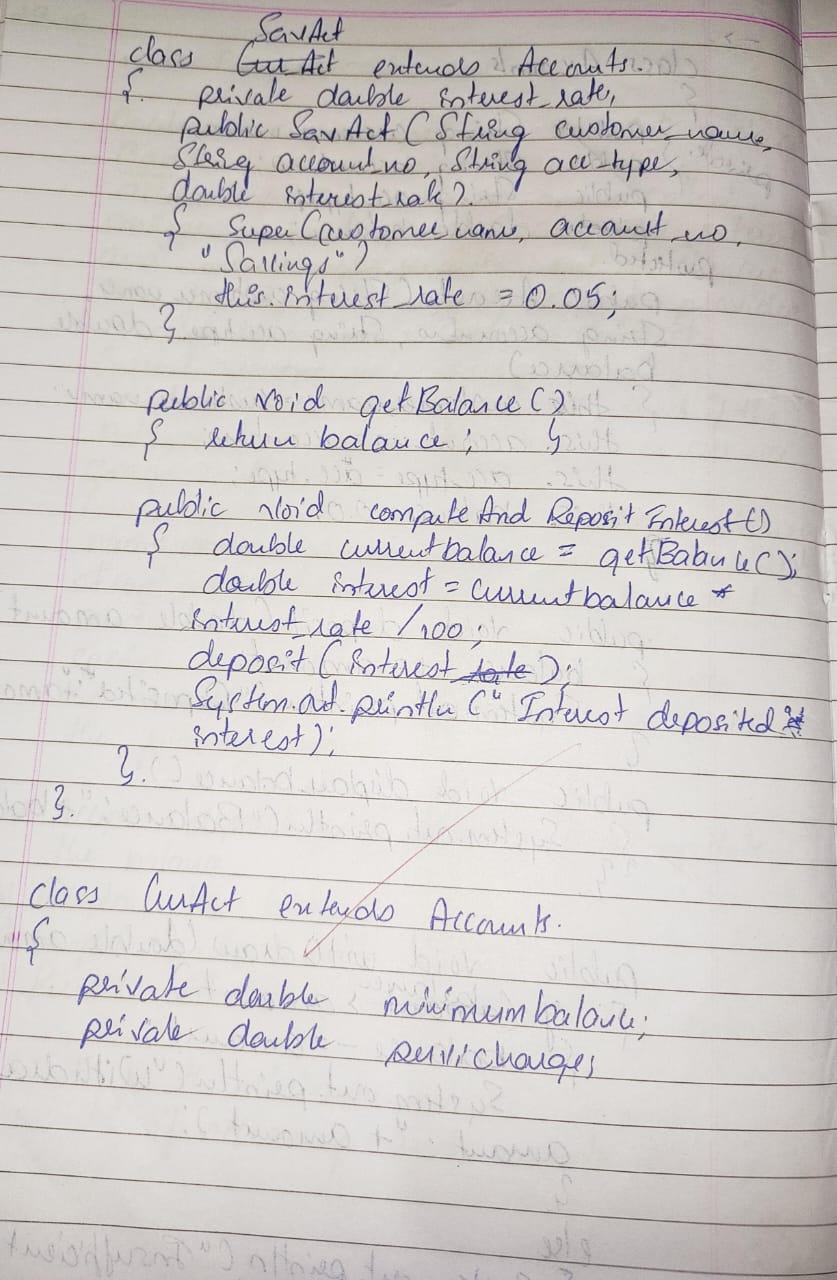
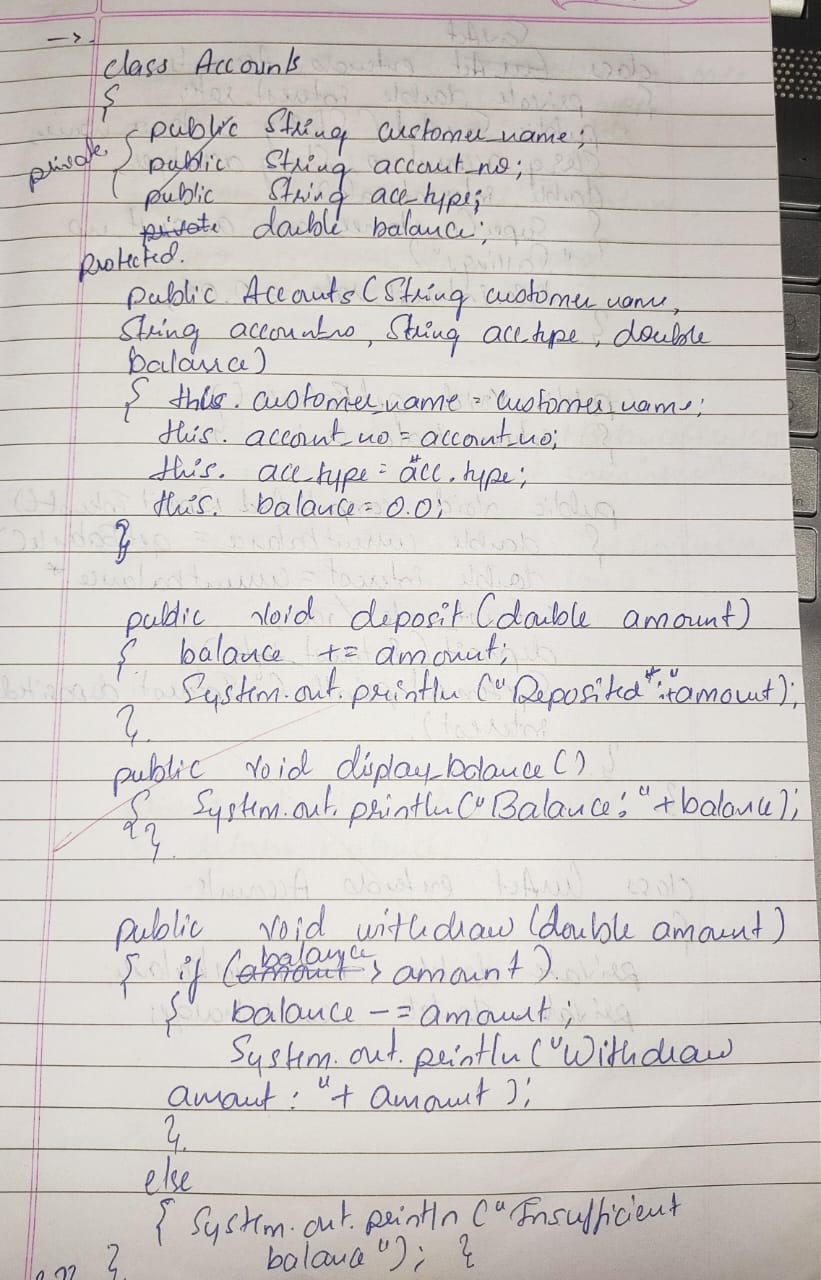
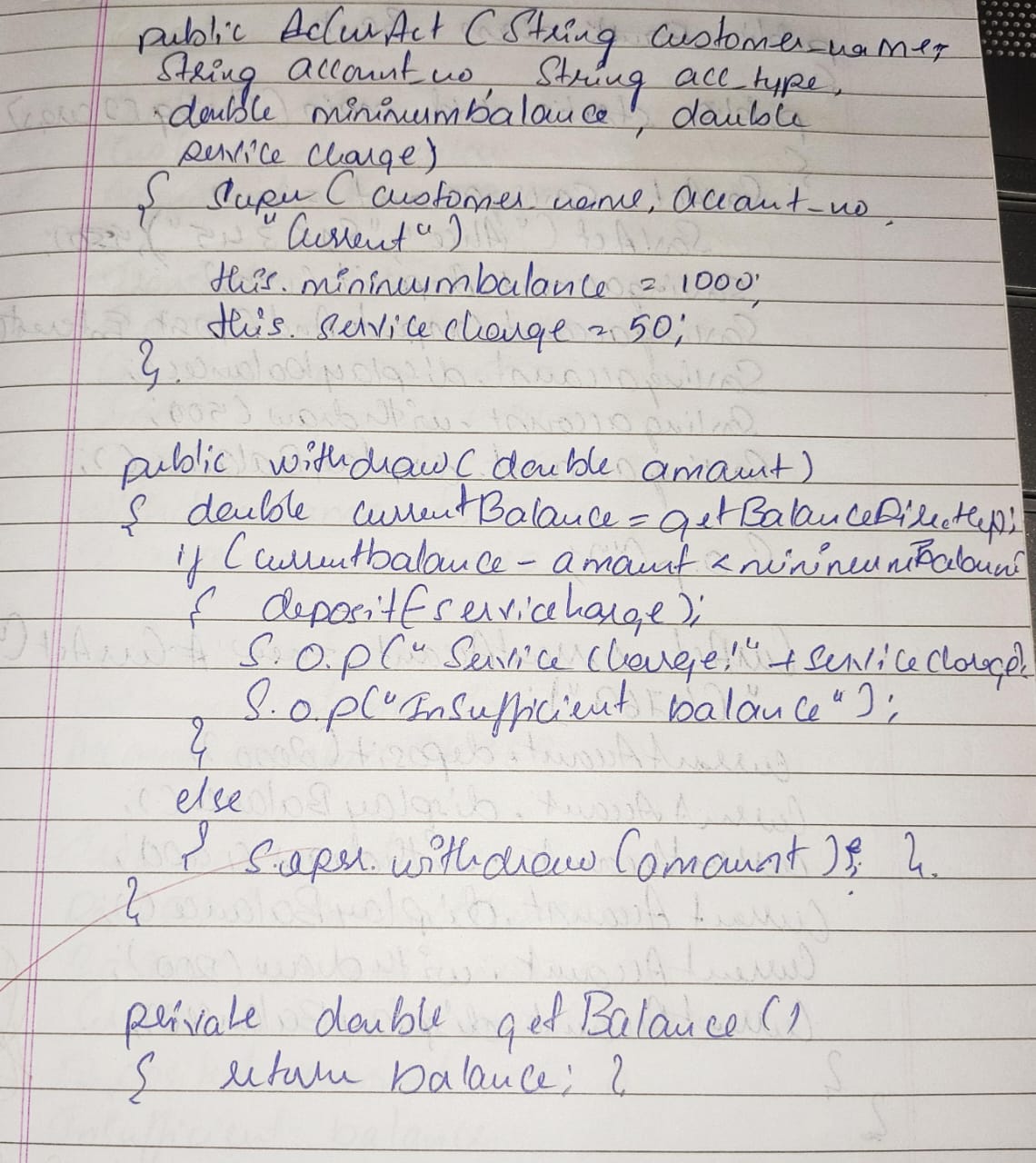
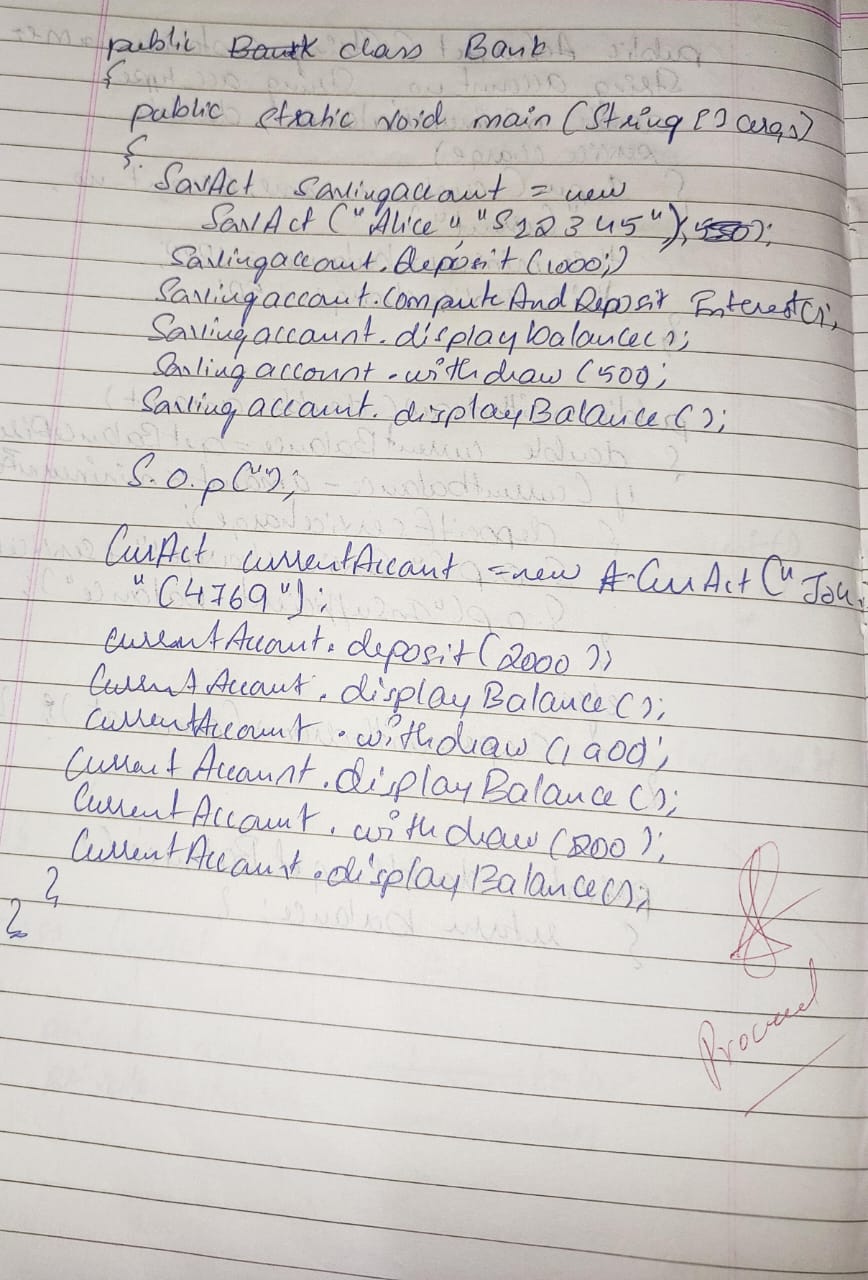
currentAccount.withdraw(200);

currentAccount.displayBalance();

}

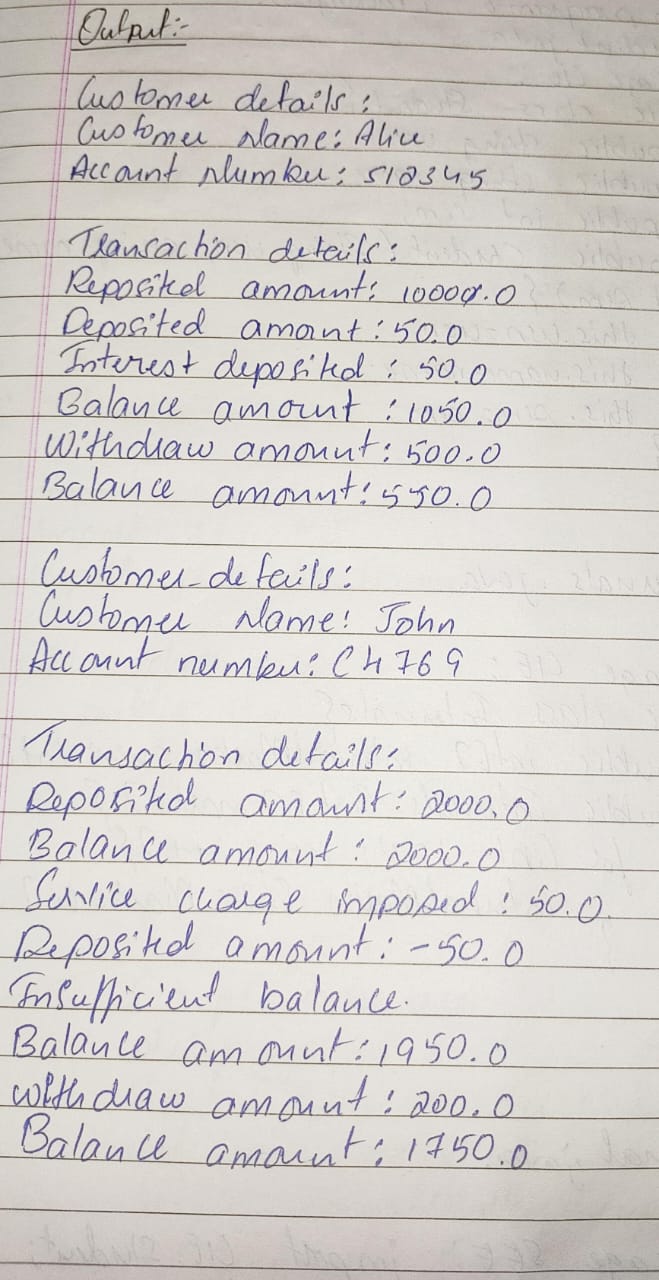
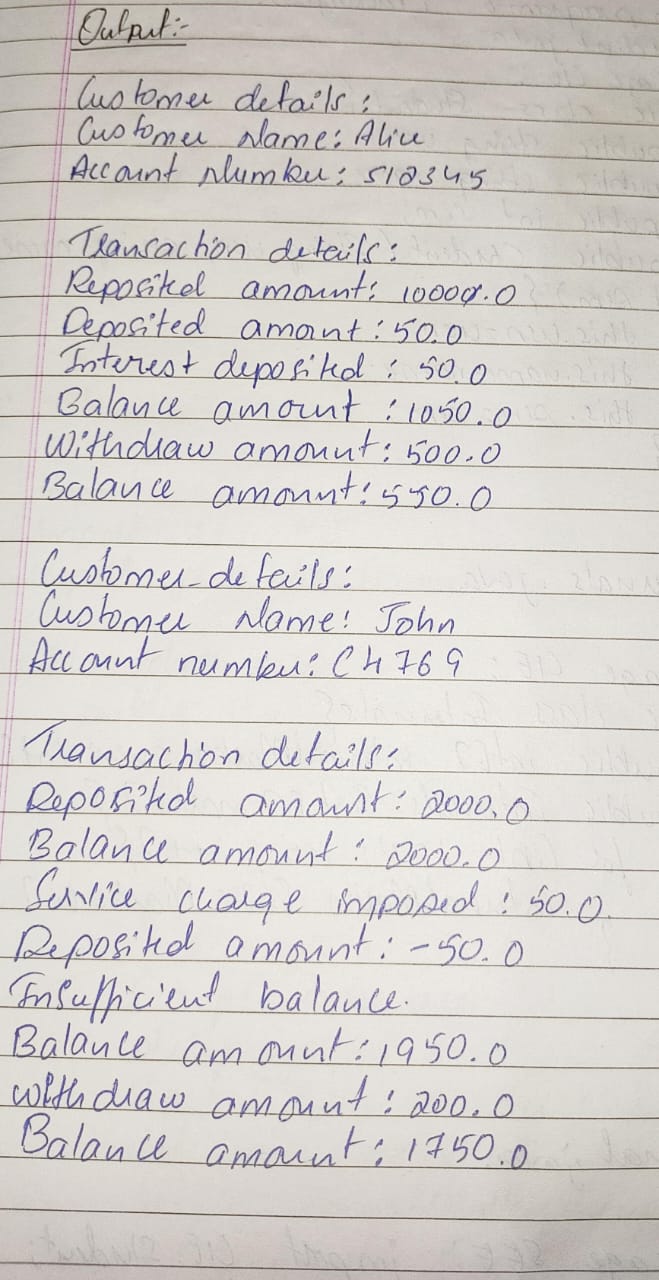
}

OUTPUT:



Lab program-6

Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

CODE:

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 class Internals {

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

public Internals(int[] marks) {

if (marks.length == 5) {

System.arraycopy(marks, 0, internalMarks, 0, 5);

} else {

throw new IllegalArgumentException("Exactly 5 marks are required for internal marks.");

}

}

}

// Package SEE: Contains External class

package SEE;

import CIE.Student;

public class External extends Student {

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

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

super(usn, name, sem);

if (marks.length == 5) {

System.arraycopy(marks, 0, externalMarks, 0, 5);

} else {

throw new IllegalArgumentException("Exactly 5 marks are required for SEE marks.");

}

}

}

// Main Program: Computes final marks

import CIE.\*;

import SEE.\*;

import java.util.Scanner;

public class FinalMarks {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

scanner.nextLine();

External[] students = new External[n];

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

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

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

String usn = scanner.nextLine();

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

String name = scanner.nextLine();

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

int sem = scanner.nextInt();

System.out.println("Enter Internal Marks (5 courses):");

int[] internalMarks = new int[5];

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

internalMarks[j] = scanner.nextInt();

}

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

int[] externalMarks = new int[5];

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

externalMarks[j] = scanner.nextInt();

}

scanner.nextLine();

Internals internals = new Internals(internalMarks);

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

System.out.println("\nCalculating final marks for student...");

}

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

for (External student : students) {

System.out.println("\nUSN: " + student.usn);

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

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

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

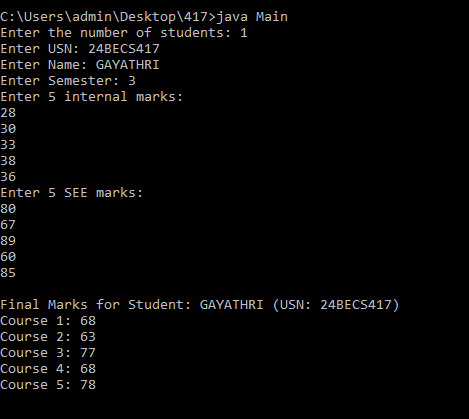
for (int j = 0; j < 5; j++) {

int finalMark = (student.externalMarks[j] / 2) + student.internalMarks[j];

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

}

}

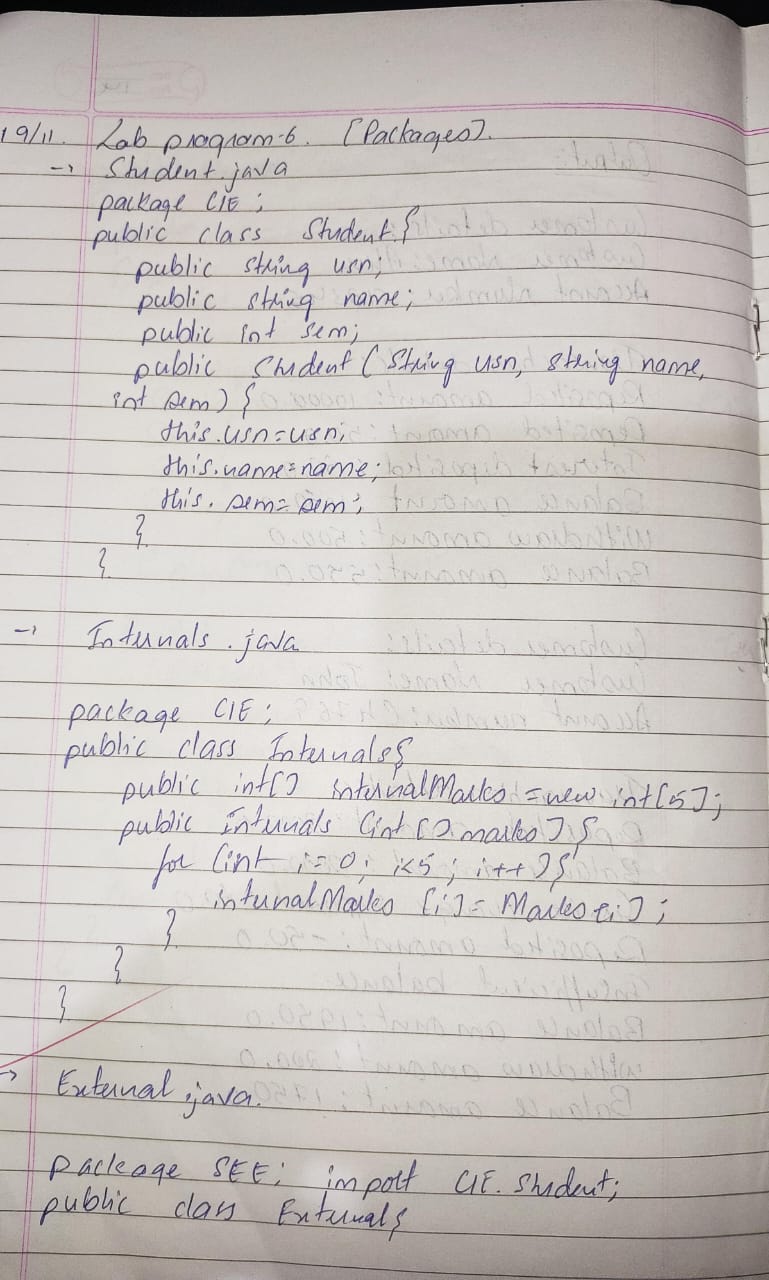
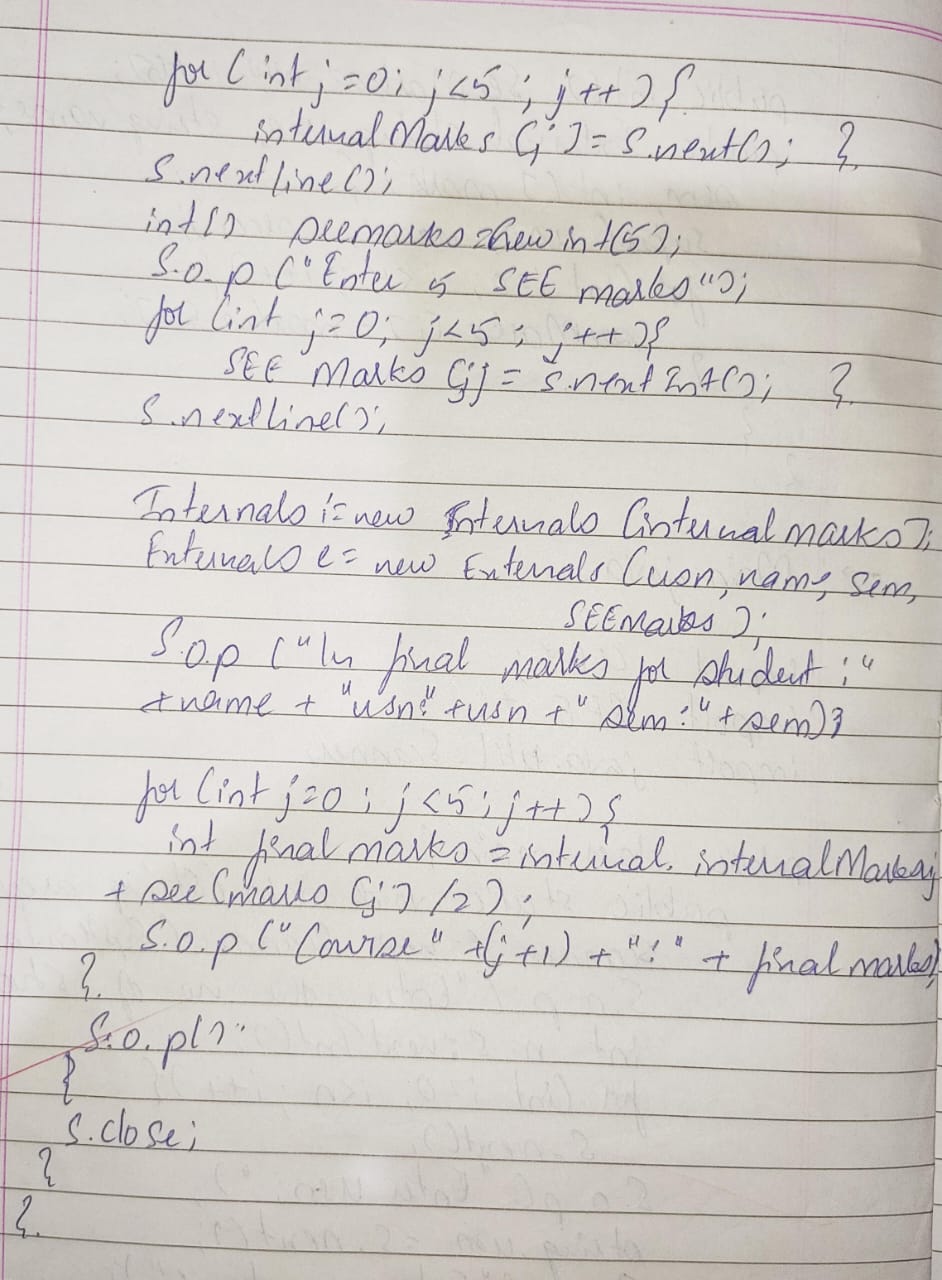
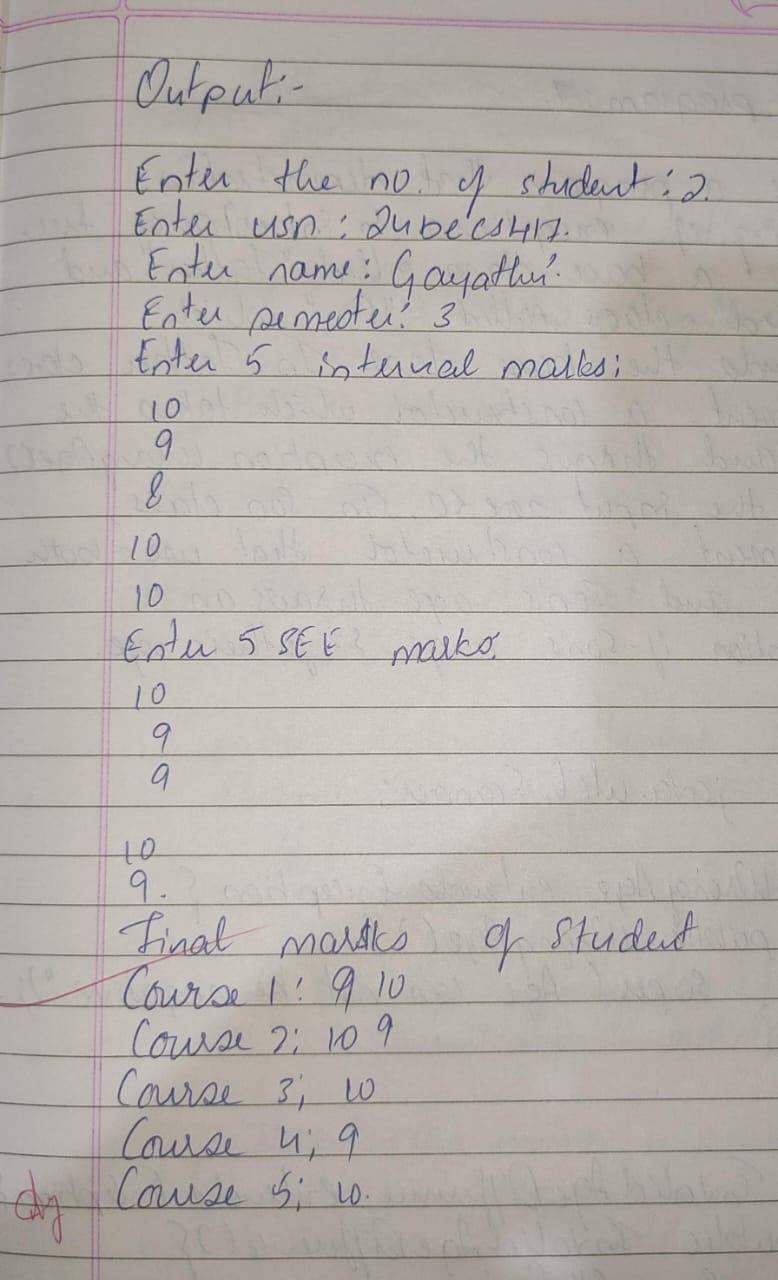
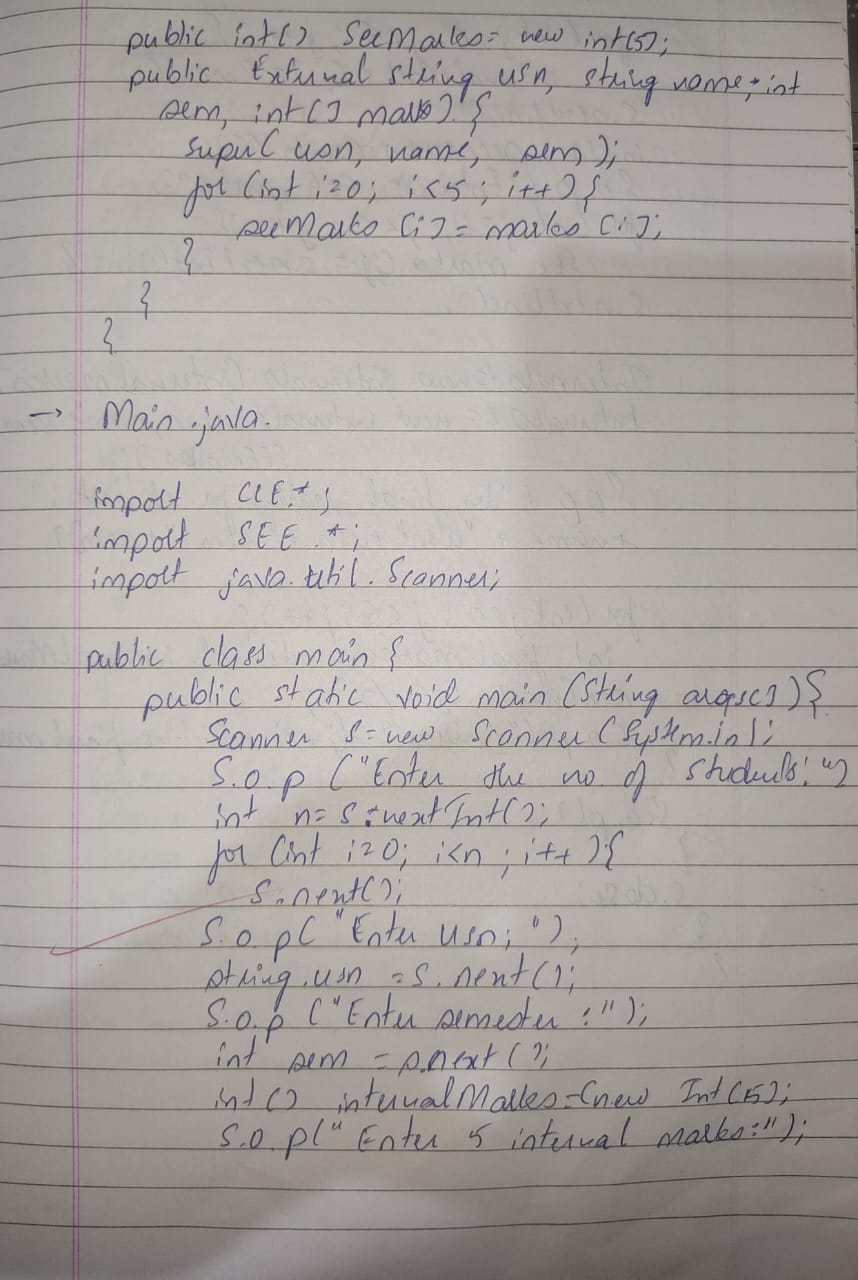


scanner.close();

}

}

OUTPUT:



Lab 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 Wrong Age( ) when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.

**CODE:**

import java.util.Scanner;

class WrongAge extends Exception {

public WrongAge() {

super("Age cannot be negative.");

}

}

class InvalidAgeDifference extends Exception {

public InvalidAgeDifference() {

super("Son's age cannot be greater than or equal to Father's age.");

}

}

class Father {

int age;

public Father(int age) throws WrongAge {

if (age < 0) {

throw new WrongAge();

}

this.age = age;

}

}

class Son extends Father {

int sonAge;

public Son(int fatherAge, int sonAge) throws WrongAge, InvalidAgeDifference {

super(fatherAge);

if (sonAge < 0) {

throw new WrongAge();

}

if (sonAge >= fatherAge) {

throw new InvalidAgeDifference();

}

this.sonAge = sonAge;

}

}

public class Main {

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);

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

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

} catch (WrongAge | InvalidAgeDifference e) {

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

} finally {

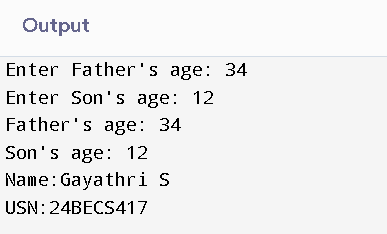
scanner.close();

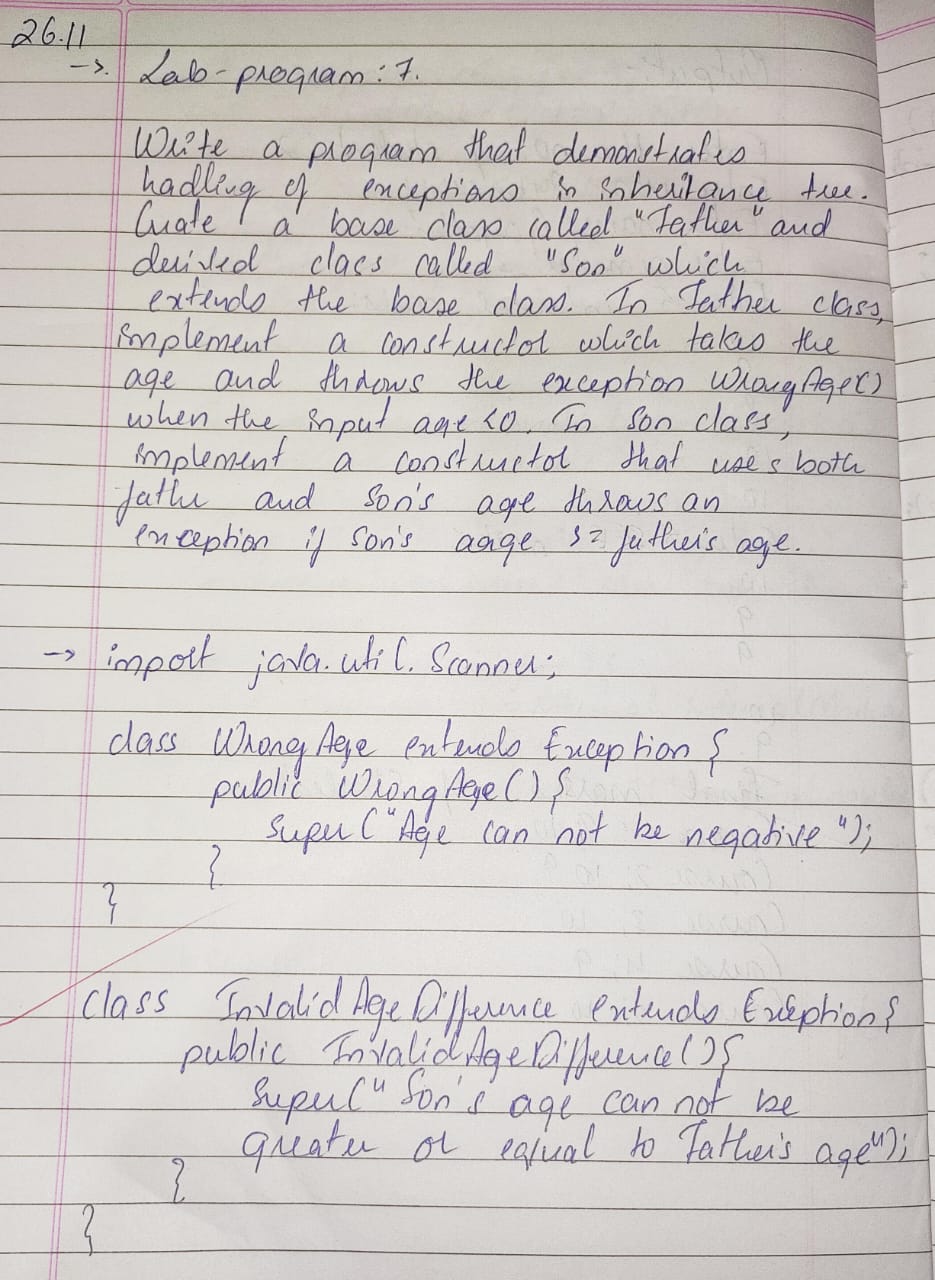
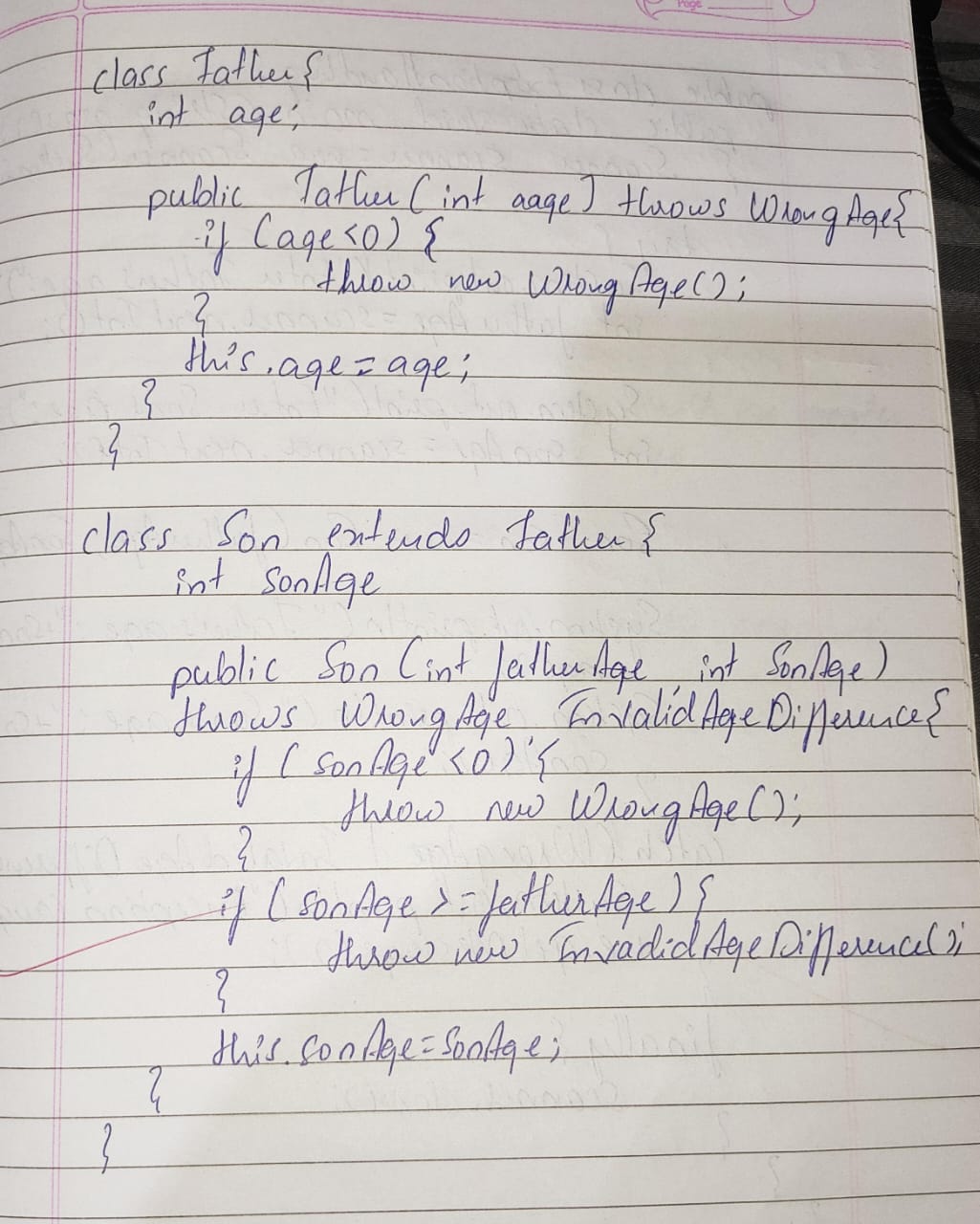
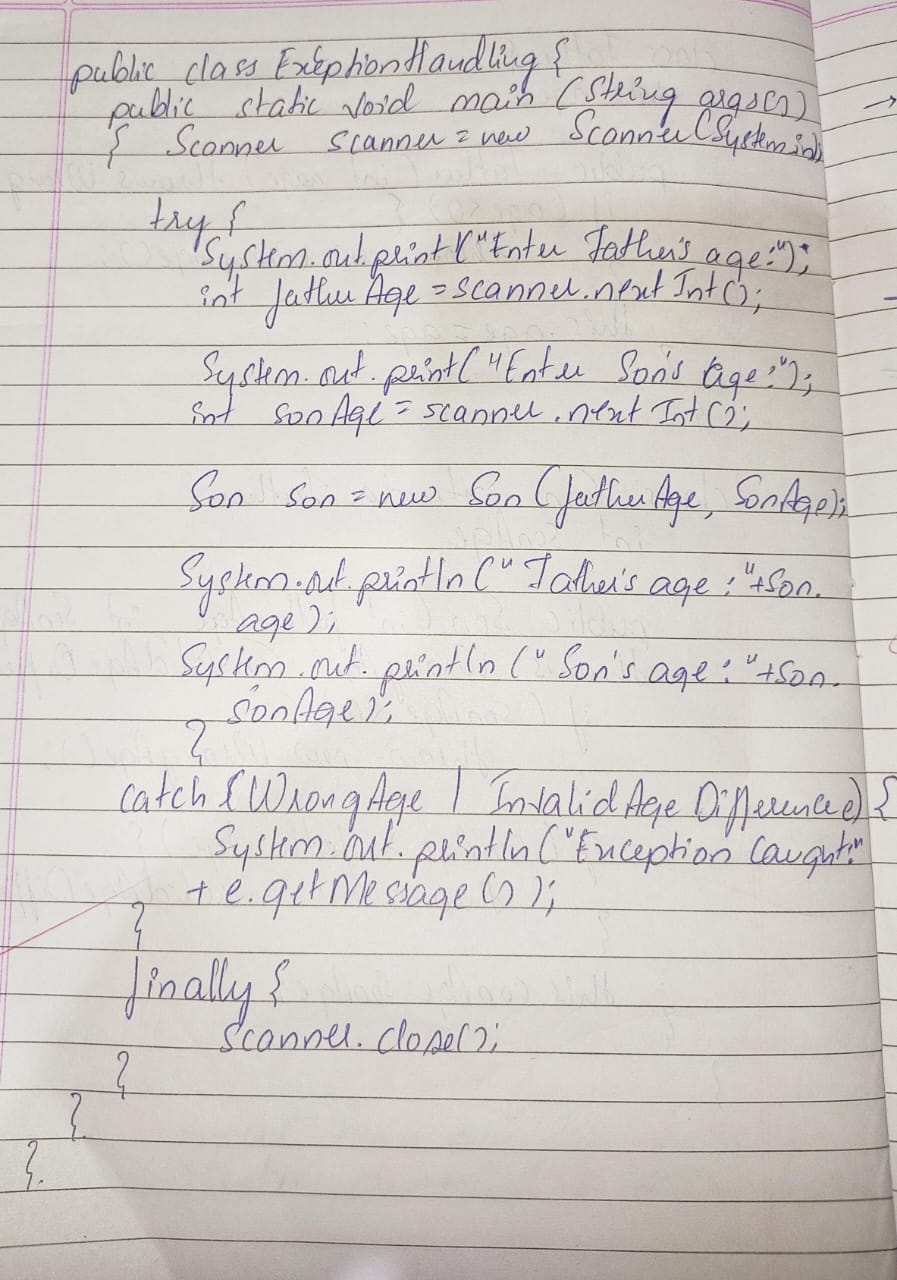
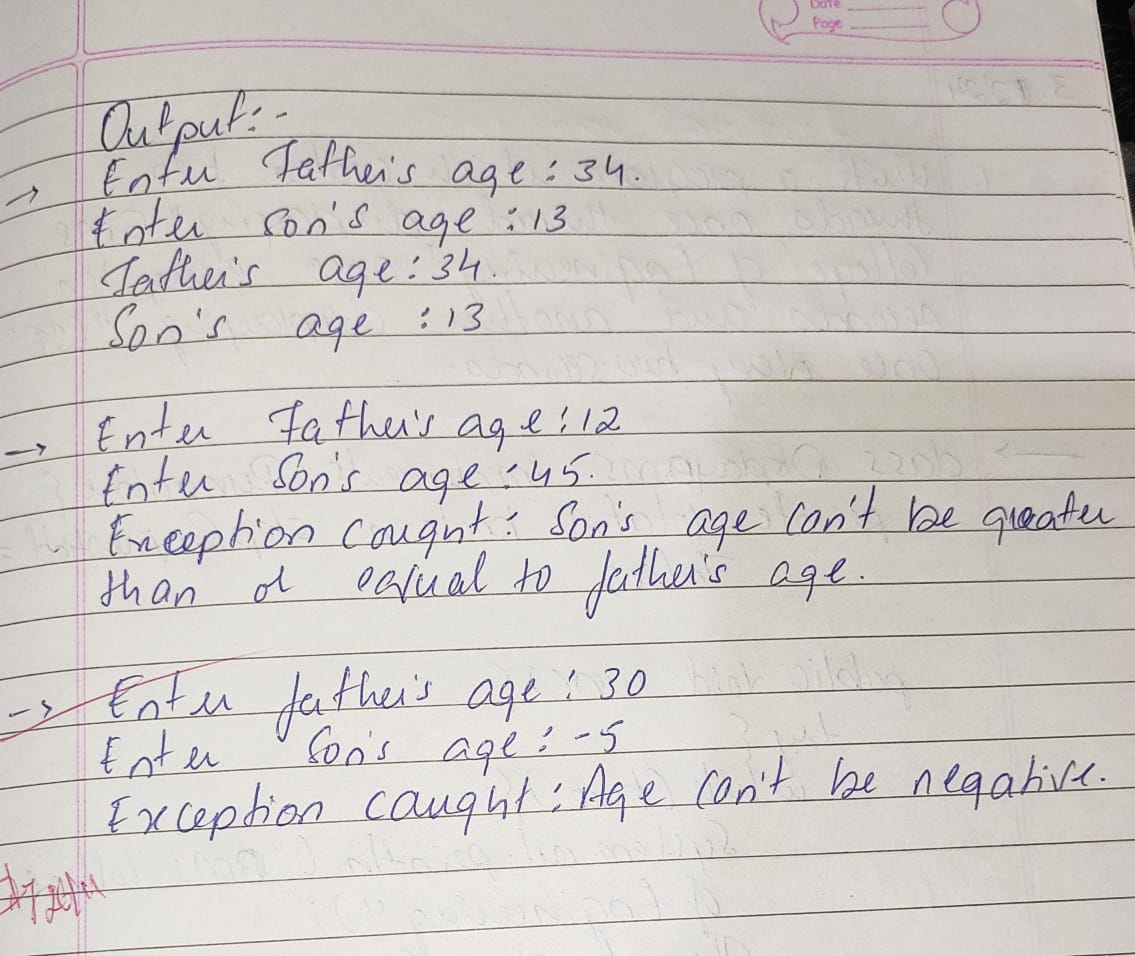
}

}

}

OUTPUT:





Lab 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.

**CODE:**

class DisplayBMS implements Runnable {

@Override

public void run() {

try {

while (!Thread.currentThread().isInterrupted()) {

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

Thread.sleep(10000);

}

} catch (InterruptedException e) {

System.out.println("BMS Thread Interrupted");

}

}

}

class DisplayCSE implements Runnable {

@Override

public void run() {

try {

while (!Thread.currentThread().isInterrupted()) {

System.out.println("CSE");

Thread.sleep(2000);

}

} catch (InterruptedException e) {

System.out.println("CSE Thread Interrupted");

}

}

}

public class CollegeThreads {

public static void main(String[] args) {

Thread thread1 = new Thread(new DisplayBMS());

Thread thread2 = new Thread(new DisplayCSE());

thread1.start();

thread2.start();

try {

Thread.sleep(20000);

thread1.interrupt();

thread2.interrupt();

thread1.join();

thread2.join();

} catch (InterruptedException e) {

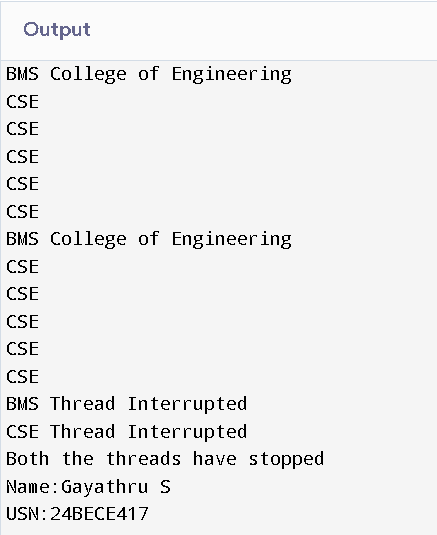
System.out.println("Main Thread Interrupted");

}

System.out.println("Both the threads have stopped")

}

}

OUTPUT:

