School of Information Technology and Engineering

Winter Semester 2021-22

B.Tech (Information Technology)

CSE1007 Java Programming

Slot: L27 + L28

Name: Anubhav Chachra

Reg No: 20BIT0104

SOURCE CODE

```java
// Anubhav Chachra
// 20BIT0104

package DA1.src.com.anubhav_chachra;

import java.util.Arrays;
import java.util.Random;

public class Da1 {

    // Constants
    final static int minCredit = 12;
    final static int maxCredit = 24;

    // Declaring Student class
    static class Student {

        private String regNo;
        private String name;
        private String dob;
        private String address;
        private String mobileNo;
        private Course[] allotedCourses; // an array of type "Course".
        private int totalCredits; // total credits of all the courses.
        private int[] marks; // int array to store marks.
        private char[] grades; // char array to store grades.
```

```
        // Constructor for Student
        public Student(String regNo, String name, String dob, String address, String mobileNo)
                throws InstantiationException { // I have declared this exception to throw it in
case of invalid input.
            this.regNo = regNo;
            this.name = name;
            this.dob = dob;
            this.address = address;
            this.mobileNo = mobileNo;

        }

        // Mutator methods
        public void setRegNo(String regNo) {
            this.regNo = regNo;
        }

        public void setAddress(String address) {
            this.address = address;
        }

        public void setName(String name) {
            this.name = name;
        }

        public void setDob(String dob) {
            this.dob = dob;
        }

        public void setMobileNo(String mobileNo) {
            this.mobileNo = mobileNo;
        }

        // this method will set the courses for the respective student and also check if
        // total no of credits are invalid or not.
        public void setCourses(Course[] allotedCourses) throws InstantiationException {
            this.allotedCourses = allotedCourses;
            for (Course c : allotedCourses) {
                this.totalCredits += c.getCredits();
            }

            // if someone doesnt have 12 to 24 credits, then throw an exception.
            if (totalCredits < minCredit || totalCredits > maxCredit) {
                throw new InstantiationException(
                        this.name + " does not have valid number of credits! \nMinimum credits
required: "
                                + minCredit + "\nMaximum credits required: " + maxCredit);
            }
        }

        // Method to populate the marks array with random values between 0 and 100.
        public void setMarks() {
            Random rand = new Random();
            marks = new int[allotedCourses.length];
            for (int i = 0; i < marks.length; i++) {
                this.marks[i] = rand.nextInt(100);
            }
        }

        public void setGrades() {
            this.grades = new char[marks.length];
            for (int i = 0; i < marks.length; i++) {
```

```
            if (marks[i] >= 90) {
                grades[i] = 'S';
            } else if (marks[i] >= 80) {
                grades[i] = 'A';
            } else if (marks[i] >= 70) {
                grades[i] = 'B';
            } else if (marks[i] >= 60) {
                grades[i] = 'C';
            } else {
                grades[i] = 'D';
            }
        }
    }

    // Accessor methods

    public String getRegNo() {
        return regNo;
    }

    public String getAddress() {
        return address;
    }

    public String getName() {
        return name;
    }

    public String getDob() {
        return dob;
    }

    public String getMobileNo() {
        return mobileNo;
    }

    public Course[] getAllotedCourses() {
        return allotedCourses;
    }

    public String getAllotedCoursesString() {
        Course[] courses = getAllotedCourses();
        StringBuilder sb = new StringBuilder();
        for (Course c : courses) {
            sb.append(c.courseName + ",");
        }
        return sb.toString();
    }

    public int[] getMarks() {
        return marks;
    }

    public char[] getGrades() {
        return grades;
    }

    public int getTotalCredits() {
        return totalCredits;
    }

    // Method to calculate GPA.
```

```
        public String getGPA() {

            float gpa, sum = 0;
            for (int i = 0; i < this.getGrades().length; i++) {
                if (this.getGrades()[i] == 'S') {
                    sum += 10 * this.getAllotedCourses()[i].getCredits();
                } else if (this.getGrades()[i] == 'A') {
                    sum += 9 * this.getAllotedCourses()[i].getCredits();
                } else if (this.getGrades()[i] == 'B') {
                    sum += 8 * this.getAllotedCourses()[i].getCredits();
                } else if (this.getGrades()[i] == 'C') {
                    sum += 7 * this.getAllotedCourses()[i].getCredits();
                } else {
                    sum += 6 * this.getAllotedCourses()[i].getCredits();
                }
            }
            gpa = sum / this.totalCredits;
            String GPA = String.format("%.2f", gpa);
            return GPA;
        }

        // Overriden toString method to display Student details.
        @Override
        public String toString() {
            String courses = " ";
            for (Course c : allotedCourses) {
                courses += "Course Code:" + c.getCourseCode() + " , Course Name:" +
c.getCourseName() + " , Credits:"
                        + c.getCredits() + "\n\t\t  ";
            }
            return "Student Detail \n-----------------------------------\n" + "Name = " + name
+ "\n\nRegister No = "
                    + regNo
                    + "\n\nAlloted Courses ="
                    + courses + "\n-----------------------------------";

        }

    }

    // Declaring Course class
    static class Course {
        private String courseCode;
        private String courseName;
        private int credits;

        // Constructor for Course class.
        public Course(String courseCode, String courseName, int credits) {
            this.courseCode = courseCode;
            this.courseName = courseName;
            this.credits = credits;
        }

        // Mutator methods
        public void setCourseCode(String courseCode) {
            this.courseCode = courseCode;
        }

        public void setCourseName(String courseName) {
            this.courseName = courseName;
        }
```

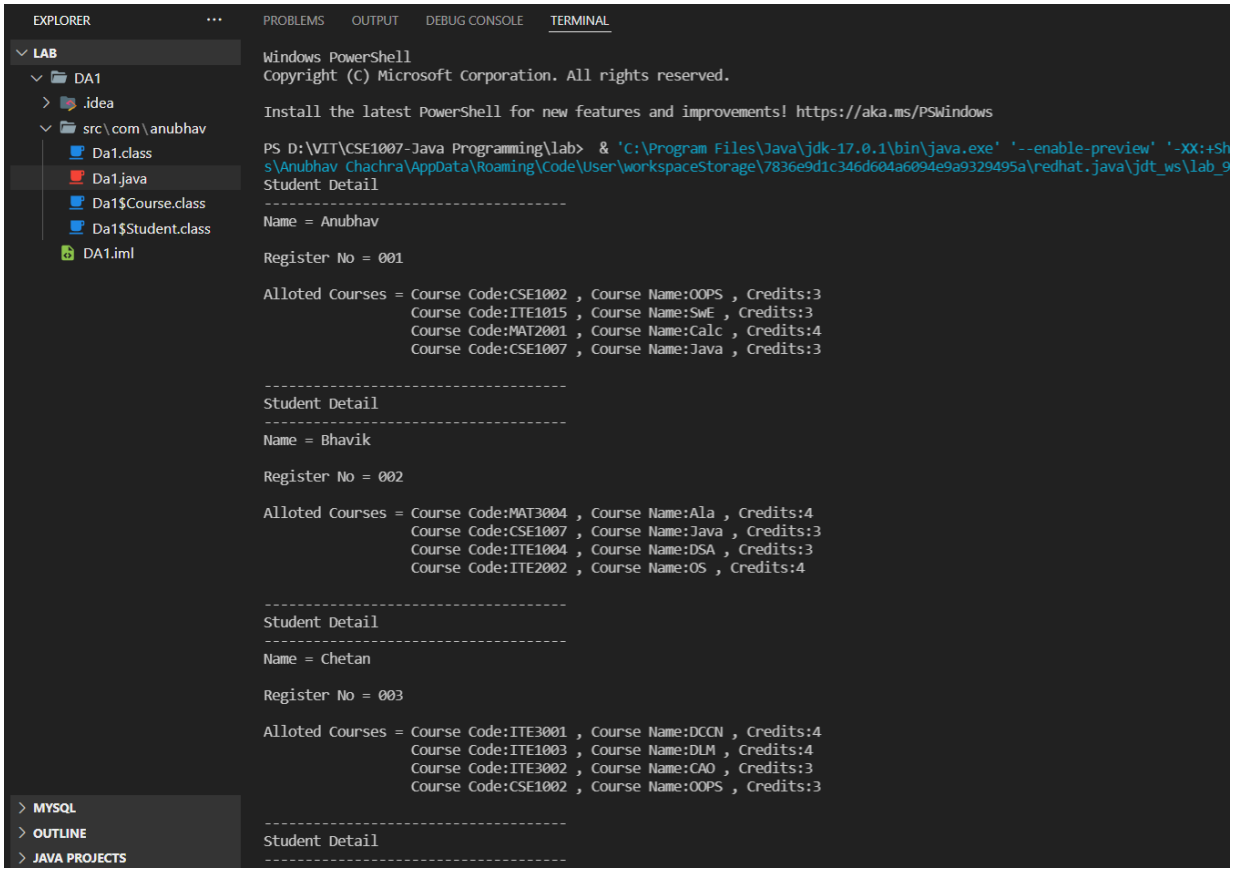
```
        public void setCredits(int credits) {
            this.credits = credits;
        }

        // Accessor Methods
        public String getCourseCode() {
            return courseCode;
        }

        public String getCourseName() {
            return courseName;
        }

        public int getCredits() {
            return credits;
        }

        // Overriden toString() method to display Course details.
        @Override
        public String toString() {
            return "Course Name = " + courseName + ", Course Code = " + courseCode + ", Credits
= " + credits;
        }

    }

    public static void main(String[] args) throws InstantiationException {

        // Declaring students and courses arrayss.
        Student[] students = new Student[10];
        Course[] courses = new Course[10];

        // Populating students array with student objects.
        students[0] = new Student("001", "Anubhav", "08/11/2001", "Faridabad", "7217818288");
        students[1] = new Student("002", "Bhavik", "2/2/2001", "Gujarat", "9876543210");
        students[2] = new Student("003", "Chetan", "3/3/2002", "Mumbai", "9451562948");
        students[3] = new Student("004", "Dheeraj", "4/4/2001", "Chennai", "7811556181");
        students[4] = new Student("005", "Eshan", "5/5/2002", "Delhi", "9894533210");
        students[5] = new Student("006", "Farhan", "6/6/2001", "Lucknow", "7893254160");
        students[6] = new Student("007", "Gauri", "7/7/2002", "Punjab", "9899955210");
        students[7] = new Student("008", "Harsh", "8/8/2001", "Kerela", "9786453021");
        students[8] = new Student("009", "Ishank", "9/9/2002", "Tamil Nadu", "8523697410");
        students[9] = new Student("010", "Jasmine", "10/10/2001", "West Bengal", "7987654321");

        // Populating courses array with course objects.
        courses[0] = new Course("CSE1002", "OOPS", 3);
        courses[1] = new Course("ITE1015", "SwE", 3);
        courses[2] = new Course("MAT2001", "Calc", 4);
        courses[3] = new Course("MAT3004", "Ala", 4);
        courses[4] = new Course("CSE1007", "Java", 3);
        courses[5] = new Course("ITE1004", "DSA", 3);
        courses[6] = new Course("ITE2002", "OS", 4);
        courses[7] = new Course("ITE3001", "DCCN", 4);
        courses[8] = new Course("ITE1003", "DLM", 4);
        courses[9] = new Course("ITE3002", "CAO", 3);

        // Allocating courses to students.
        students[0].setCourses(new Course[] { courses[0], courses[1], courses[2], courses[4] });
        students[1].setCourses(new Course[] { courses[3], courses[4], courses[5], courses[6] });
        students[2].setCourses(new Course[] { courses[7], courses[8], courses[9], courses[0] });
        students[3].setCourses(new Course[] { courses[1], courses[2], courses[3], courses[4] });
        students[4].setCourses(new Course[] { courses[5], courses[6], courses[7], courses[8] });
```

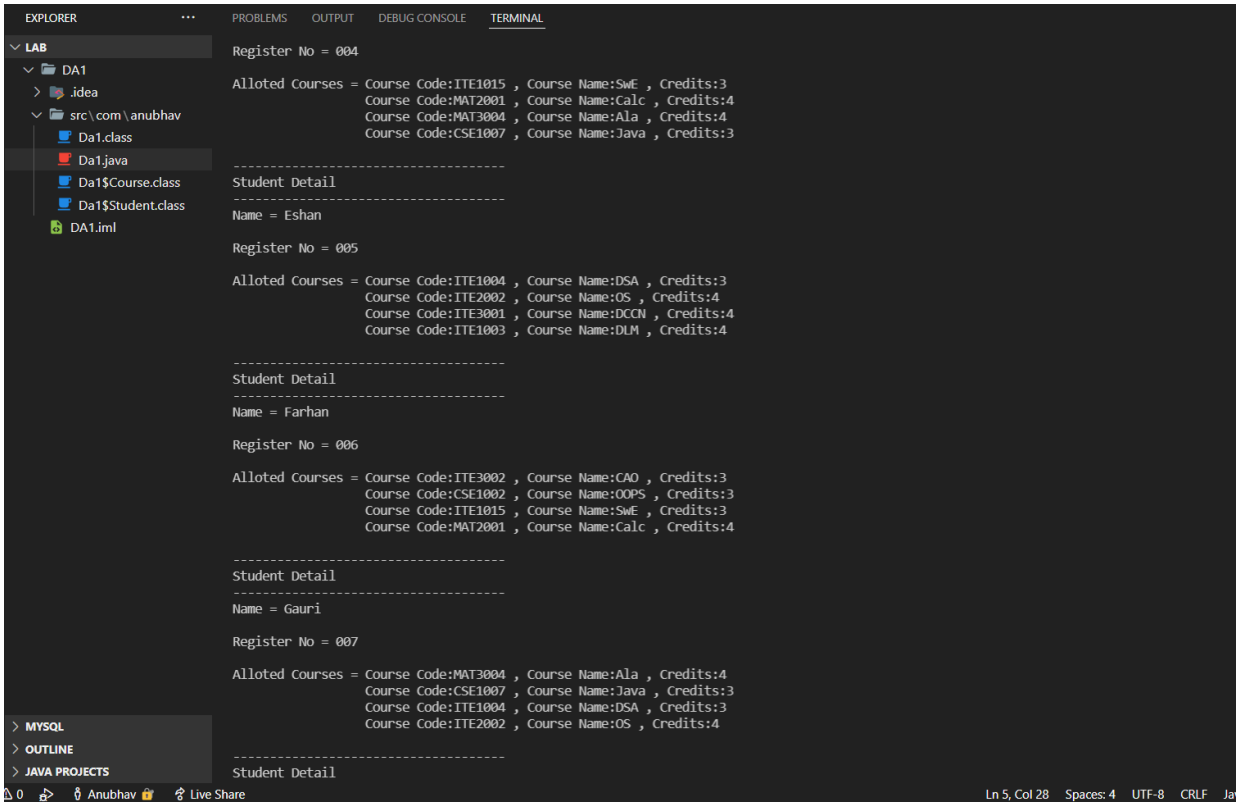
```
        students[5].setCourses(new Course[] { courses[9], courses[0], courses[1], courses[2] });
        students[6].setCourses(new Course[] { courses[3], courses[4], courses[5], courses[6] });
        students[7].setCourses(new Course[] { courses[7], courses[8], courses[9], courses[0] });
        students[8].setCourses(new Course[] { courses[1], courses[2], courses[3], courses[4] });
        students[9].setCourses(new Course[] { courses[5], courses[6], courses[7], courses[8] });

        // Displaying students details.
        printStudentDetails(students);

        // Students who have registered for the same courses.
        studentsWithSameCourse(students);

        // Results of students
        studentResult(students);

        // End of main method.
    }

    // Below are ALL the display methods compiled at one place.

    /*
     * This is the method to display the name of every student along with their
     * registration number and
     * the list of courses allocated
     * to every student.
     */

    static void printStudentDetails(Student[] students) {
        for (Student s : students) {
            System.out.println(s.toString());
        }
    }

    /*
     * Display the name and registration number of the students who registered for
     * the same list of courses.
     */

    static void studentsWithSameCourse(Student[] students) {
        int flag = 0;
        for (int i = 0; i < students.length; i++) {
            for (int j = i + 1; j < students.length; j++) {
                if (Arrays.equals(students[i].getAllotedCourses(),
students[j].getAllotedCourses())) {

                    flag = 1;
                    System.out.println("------------------------------------------------------------
---------");
                    System.out.println("For these courses: " +
(students[i].getAllotedCoursesString()) + "\n");
                    System.out.println("These students have registered : \n");
                    System.out.println("Name :" + students[i].getName() + ", Register No. :" +
students[i].getRegNo());
                    System.out.println("Name :" + students[j].getName() + ", Register No. :" +
students[j].getRegNo());
                    System.out.println("------------------------------------------------------------
---------");
                }

            }
        }
        // in case no students have common courses.
```

```
        if (flag == 0) {
            System.out.println("No students are having the same courses");

        }
    }

    /*
     * This is the method to display the student registration number and name (at
     * the top), the list of
     * courses along
     * with the grade obtained in the course (in the middle) and the GPA (at the
     * bottom).
     */

    static void studentResult(Student[] students) {
        for (Student s : students) {
            s.setMarks();
            s.setGrades();

            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < s.getAllotedCourses().length; i++) {
                sb.append(s.getAllotedCourses()[i].getCourseName() + "\t\t\t\t  " +
s.getGrades()[i] + "\n");
            }
            System.out.println("\t   Student Result\n\t   --------------\n\nRegister No: " +
s.getRegNo() + "\t\t"
                    + "Name : "
                    + s.getName() + "\n" + "\nSubject\t\t\t\tGrades\n-------------------------
--------------\n" + sb
                    + "----------------------------------------\n\n\t\tGPA: " + s.getGPA() +
"\n\n"
                    + "----------------------------------------");
        }
    }
}
```

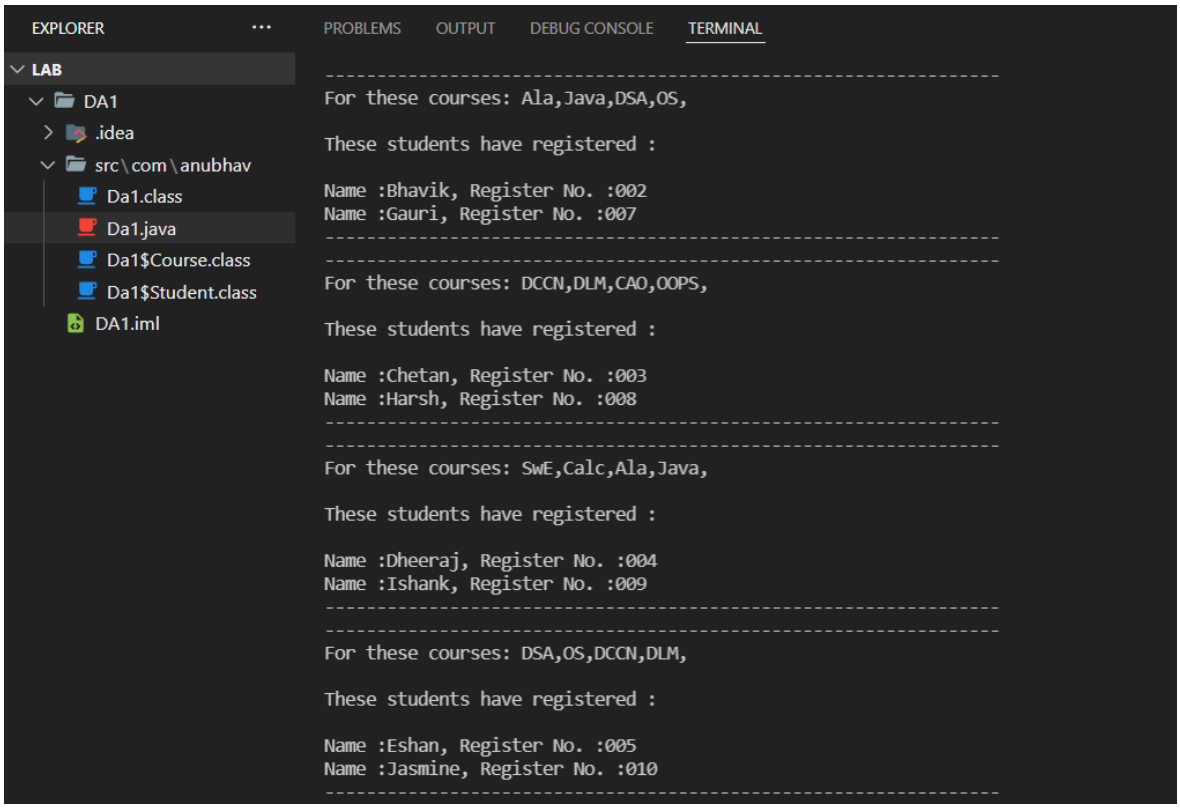
OUTPUT SCREENSHOTS

Printing student details

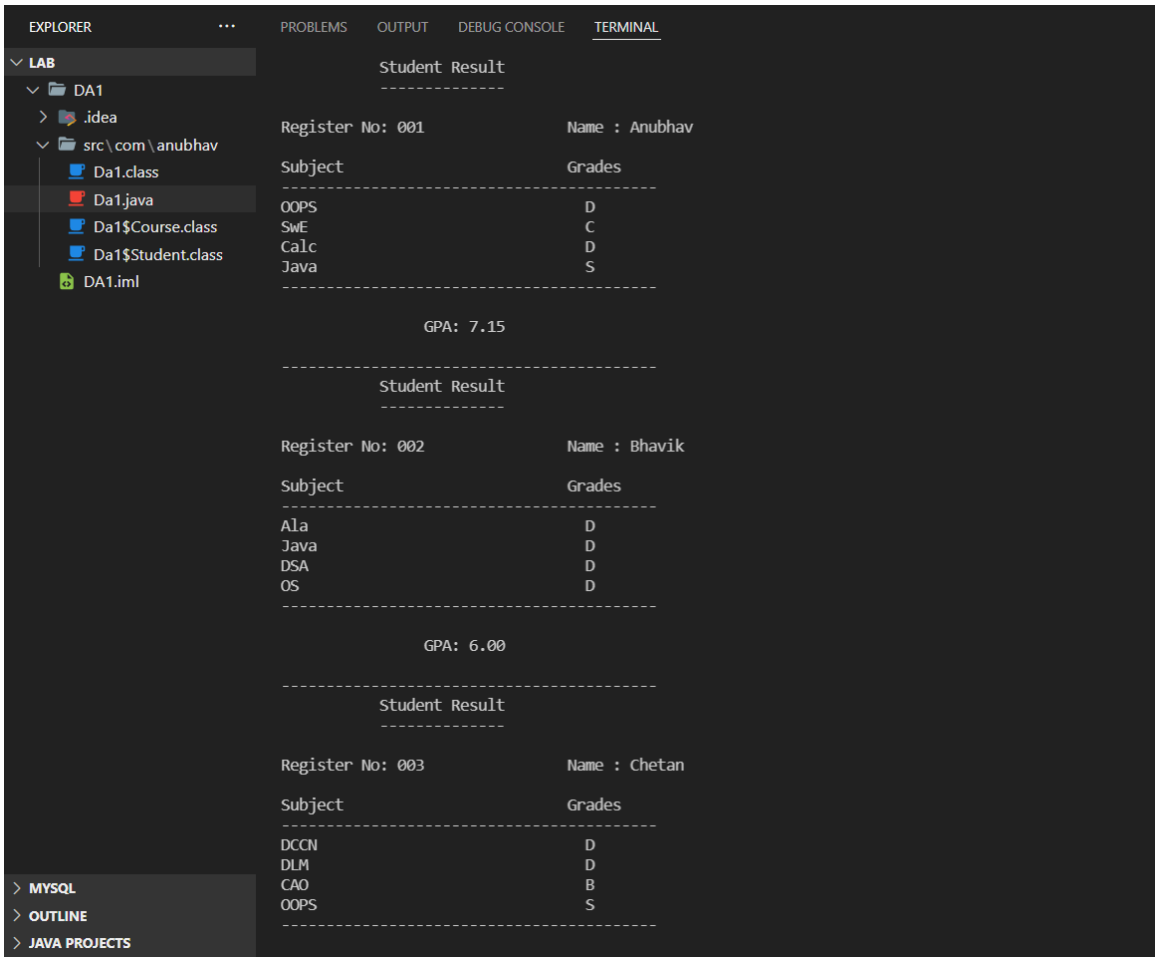


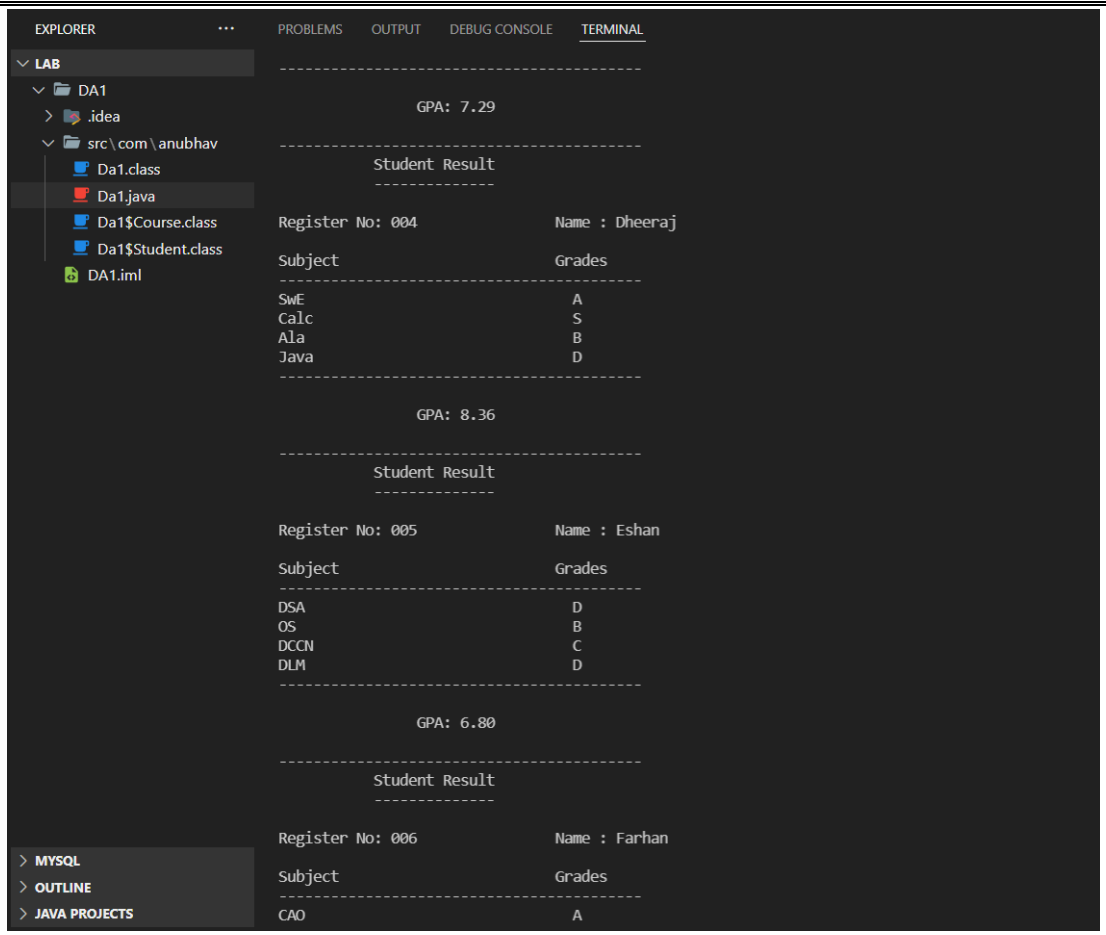


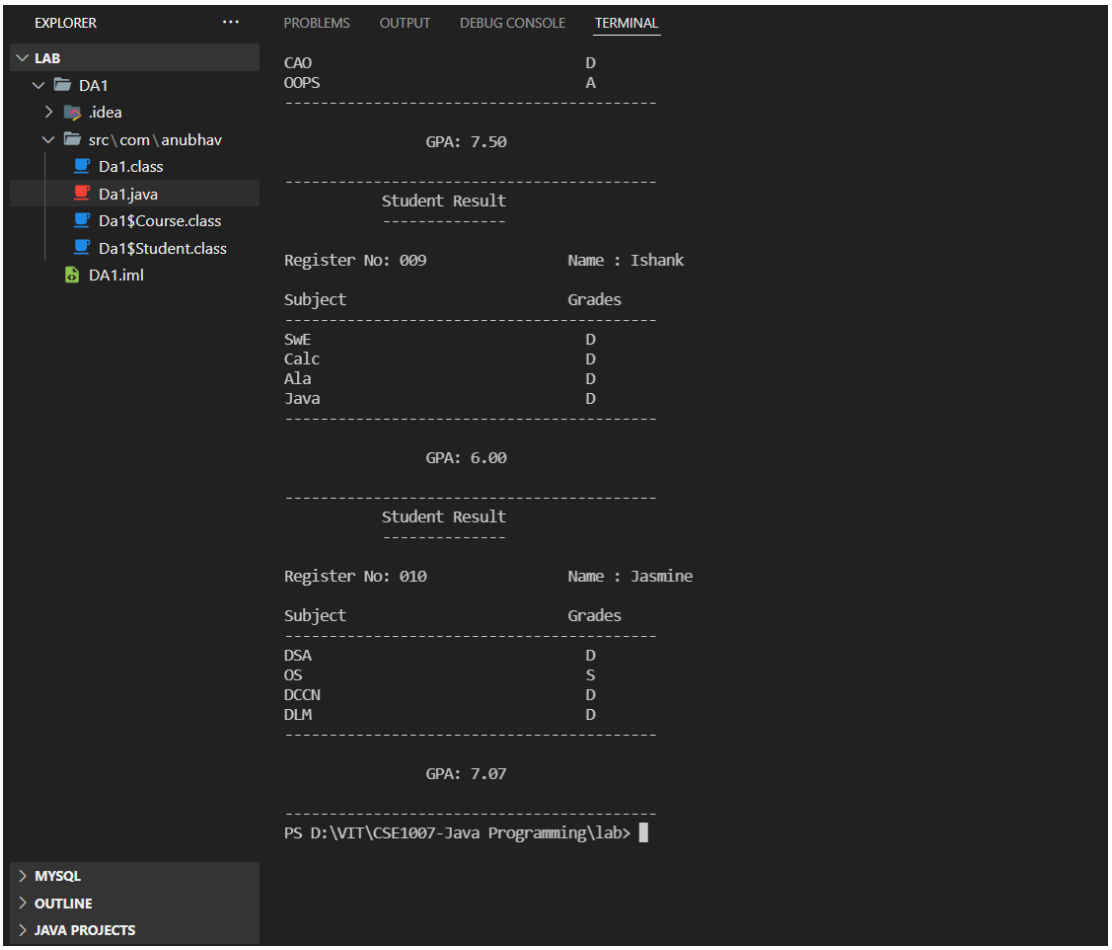
Students who have the same courses



Student Result with grades and GPA




EXPLORER
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
LAB
DA1
.idea
src\com\anubhav
Da1.class
Da1.java
Da1$Course.class
Da1$Student.class
DA1.iml
MYSQL
OUTLINE
JAVA PROJECTS
GPA: 7.29
Student Result
Register No: 004 Name : Dheeraj
Subject Grades
SwE A
Calc S
Ala B
Java D
GPA: 8.36
Student Result
Register No: 005 Name : Eshan
Subject Grades
DSA D
OS B
DCCN C
DLM D
GPA: 6.80
Student Result
Register No: 006 Name : Farhan
Subject Grades
CAO A


EXPLORER
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
LAB
DA1
.idea
src\com\anubhav
Da1.class
Da1.java
Da1$Course.class
Da1$Student.class
DA1.iml
MYSQL
OUTLINE
JAVA PROJECTS
CAO D
OOPS A
GPA: 7.50
Student Result
Register No: 009 Name : Ishank
Subject Grades
SwE D
Calc D
Ala D
Java D
GPA: 6.00
Student Result
Register No: 010 Name : Jasmine
Subject Grades
DSA D
OS S
DCCN D
DLM D
GPA: 7.07
PS D:\VIT\CSE1007-Java Programming\lab>