WEEK 3 Class Fields and Methods
Hands-on Coding Exercise

Group Leader (LN, FN MI.):	1	SORIANO FRANCE HEDRICH O		**Score:**	
Group Members (LN, FN MI.): In alphabetical Order	2	CLASIN, MARYJE			
	3	GALLO, SHEENA MAE D.			
	4	LANSANGAN, JOYCE E			
	5	LOBERIANO, MICHELLE J			
	6	MIRANDA, KIAN ANDREI		**Date:**	**09/15/23**
	7	REFREA MARJOH			
	8	SULLA, JOHN PATRICK			
	9	TOLENTINO, KRISTINE MAE T.			
	10	VILLENA KEN ANTHONY J.			
Section:	**2IT-D**	**Instructor:**	Prof. Janus Raymond C. Tan	**Sem./ A.Y.**	1st/ 23-24
Project Link:	*https://onlinegdb.com/njq4tdi-z*				

CRITERIA	GRADING SCALE				WEIGHT	SCORE
	NEEDS IMPROVEMENT 1	FAIR 2	GOOD 3	EXCELLENT 4		
CODE PROFICIENCY	Code shows minimal understanding of programming concepts, with solutions that are ineffective or incorrect.	Code demonstrates a basic understanding of concepts, but solutions may lack efficiency or elegance.	Code shows a solid grasp of programming concepts, and solutions are effective.	Code demonstrates exceptional understanding and mastery of programming concepts. Efficient and elegant solutions are consistently employed.	30%	
FUNCTIONALITY	The program does not meet essential requirements and has significant functionality problems or errors.	The program partially meets requirements, but functionality may be limited or inconsistent. Several bugs or issues affect program behavior.	The program meets most specified requirements and functions correctly in typical scenarios. Minor bugs or issues may be present.	The program fully meets all specified requirements and functions flawlessly under various scenarios.	30%	
DOCUMENTATION	Documentation is absent or provides little to no insight into code logic or usage.	Documentation is minimal and lacks clarity in explaining code logic or usage. Important details are missing.	Documentation is present and adequately explains code logic and usage. Some details may be lacking.	Comprehensive and well-organized documentation is provided, including clear explanations of code logic, usage, and any assumptions made.	20%	
CODE READABILITY	Code lacks organization and readability due to unclear variable names, inadequate comments, or messy format	Code is somewhat organized, but variable names, comments, or formatting may be inconsistent or unclear.	Code is well-organized, with clear variable names and comments. Formatting is consistent, aiding in understanding.	Code is exceptionally well-organized, with meaningful variable names, consistent formatting, and clear comments. Easy to understand and maintain.	20%	
				TOTAL	100%	

INSTRUCTION:

Create a Java class Student to model a student's information and grades. Implement methods to calculate the student's average grade and determine their letter grade based on a grading scale. The Student class should have the

following functionalities:

1. Initialize student's attributes: name (String) and an array of grades (double[]), five grades respectively.
2. Implement setter methods to assign values to the student's name and grades.
3. Implement a method CalculateAverage() that calculates and returns the average grade of the student. 4. Implement a method GetLetterGrade() that determines and returns the letter grade based on the following scale:

   A: 90 - 100
   B: 80 - 89
   C: 70 - 79
   D: 60 - 69
   F: 0 - 59

5. Your task is to create the Student class and demonstrate its functionality using the following steps:
   - Create several Student objects with different names and arrays of grades.
   - Calculate and display the average grade for each student.
   - Determine and display the letter grade for each student based on their average grade.

6. Remember to provide clear instructions for interaction and output in your code.

PROGRAM CODE


```java
public class Student {

    private String name;
    private double[] grades;

    // Constructor
    public  Student(String name, double[] grades) {
        this.name = name;
        this.grades = grades;
    }

    // Getter method for name
    public String getName() {
        return name;
    }

    // Setter method for name
    public void setName(String name) {
        this.name = name;
    }

    // Getter method for grades
    public double[] getGrades() {
        return grades;
    }

    // Setter method for grades
    public void setGrades(double[] grades) {
        this.grades = grades;
    }


    // Calculate average grade
    public double calculateAverage() {
        double sum = 0;
        for (double grade : grades) {
            sum += grade;
        }
        return sum / grades.length;
    }

    // Get letter grade
    public String getLetterGrade() {
        double averageGrade = calculateAverage();
```

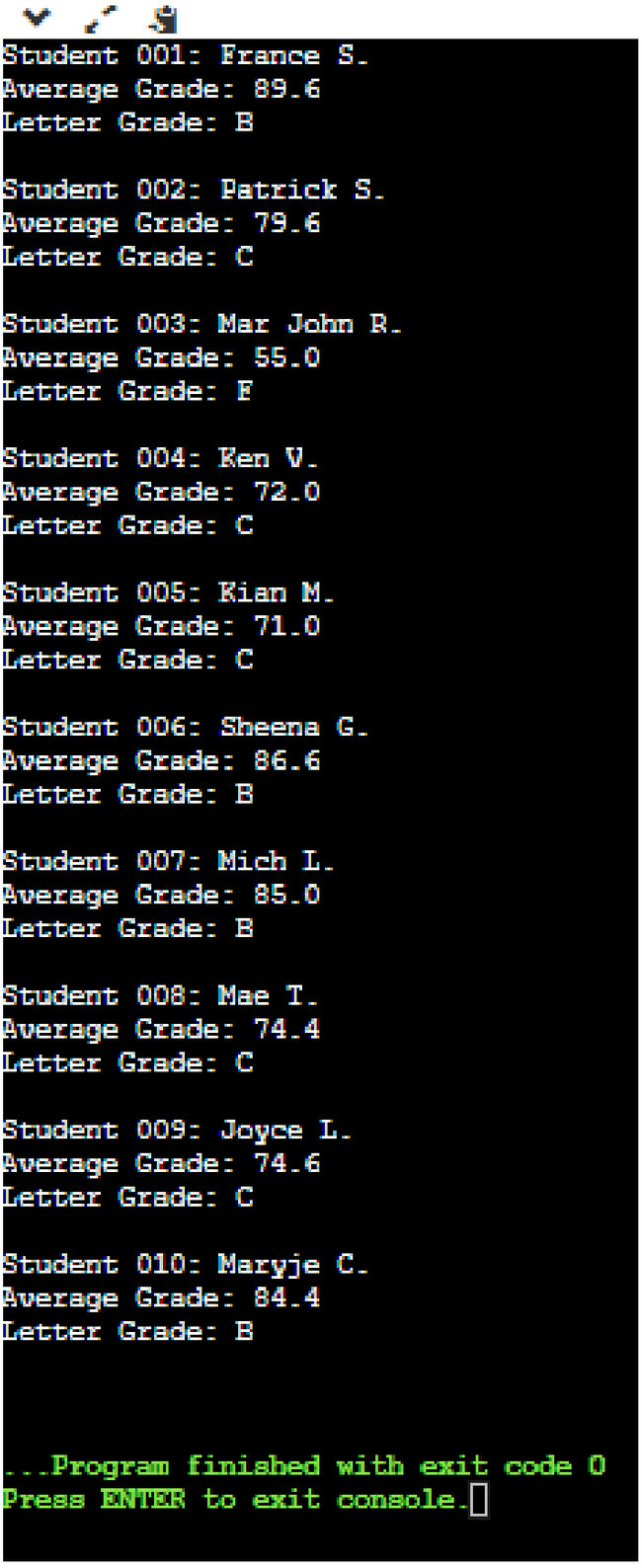
```
        double averageGrade = calculateAverage();
        if (averageGrade >= 90 && averageGrade <= 100) {
            return "A";
        } else if (averageGrade >= 80 && averageGrade < 90) {
            return "B";
        } else if (averageGrade >= 70 && averageGrade < 80) {
            return "C";
        } else if (averageGrade >= 60 && averageGrade < 70) {
            return "D";
        } else {
            return "F";
    }

  }

}
```

input

Student.java | Main.java

```
public class Main {
    public static void main(String[] args) {
        // Create student objects for three students
            Student student1 = new Student("France S.", new double[]{85, 92, 93, 90, 88});
            Student student2 = new Student("Patrick S.", new double[]{75, 79, 72, 90, 82});
            Student student3 = new Student("Mar John R.", new double[]{55, 62, 48, 70, 40});
            Student student4 = new Student("Ken V.", new double[]{50, 63, 78, 80, 89});
            Student student5 = new Student("Kian M.", new double[]{70, 83, 48, 75, 79});
            Student student6 = new Student("Sheena G.", new double[]{85, 92, 78, 90, 88});
            Student student7 = new Student("Mich L.", new double[]{75, 95, 83, 90, 82});
            Student student8 = new Student("Mae T.", new double[]{55, 88, 48, 92, 89});
            Student student9 = new Student("Joyce L.", new double[]{50, 91, 78, 65, 89});
            Student student10 = new Student("Maryje C.", new double[]{88, 83, 87, 85, 79});

            // Calculate and display average grade and letter grade for each student
            System.out.println("Student 001: " + student1.getName());
            System.out.println("Average Grade: " + student1.calculateAverage());
            System.out.println("Letter Grade: " + student1.getLetterGrade());
            System.out.println();

            System.out.println("Student 002: " + student2.getName());
            System.out.println("Average Grade: " + student2.calculateAverage());
            System.out.println("Letter Grade: " + student2.getLetterGrade() + "\n");

            System.out.println("Student 003: " + student3.getName());
            System.out.println("Average Grade: " + student3.calculateAverage());
            System.out.println("Letter Grade: " + student3.getLetterGrade() + "\n");

            System.out.println("Student 004: " + student4.getName());
            System.out.println("Average Grade: " + student4.calculateAverage());
            System.out.println("Letter Grade: " + student4.getLetterGrade() + "\n");

            System.out.println("Student 005: " + student5.getName());
            System.out.println("Average Grade: " + student5.calculateAverage());
            System.out.println("Letter Grade: " + student5.getLetterGrade() + "\n");

            System.out.println("Student 006: " + student6.getName());
            System.out.println("Average Grade: " + student6.calculateAverage());
            System.out.println("Letter Grade: " + student6.getLetterGrade() + "\n");

            System.out.println("Student 007: " + student7.getName());
            System.out.println("Average Grade: " + student7.calculateAverage());
            System.out.println("Letter Grade: " + student7.getLetterGrade() + "\n");

            System.out.println("Student 008: " + student8.getName());
            System.out.println("Average Grade: " + student8.calculateAverage());
            System.out.println("Letter Grade: " + student8.getLetterGrade() + "\n");

            System.out.println("Student 009: " + student9.getName());
            System.out.println("Average Grade: " + student9.calculateAverage());
            System.out.println("Letter Grade: " + student9.getLetterGrade() + "\n");

            System.out.println("Student 010: " + student10.getName());
            System.out.println("Average Grade: " + student10.calculateAverage());
            System.out.println("Letter Grade: " + student10.getLetterGrade() + "\n");

    }}
```

SAMPLE OUTPUT



WHAT HAVE YOU LEARNED?

This code demonstrates how to create a simple class containing attributes, a constructor, methods, and encapsulation—basic OOP ideas. It also shows how to use conditional statements and reuse code in OOP. In this task, we first defined a class called Studentinformation, which taught me a lot. A class in object-oriented programming (OOP) is a template for building objects with defined properties and actions. These classes have two private properties that indicate the names of the students and a range of grades. A constructor with parameters was also made. In addition, we employ getter and setter methods. This activity taught us a lot, and we will be able to apply it later.

Attach your own PDF file **(every group member should turn in his/ her own copy)** as your submission in the assigned task upon turning in using your own account in our LMSfollowing the given template. **(your PDF file should begin with the first page of the given template)**