Project7
CECS277 Fall 2021
Due October 25th 11:59 pm
Submit project7.zip folder before the deadline

Please make sure to follow the naming convention for your project. If your project does not run because of the naming issues, you won't receive any credit.
Project7.zip should include **project7 Package** and **project7.jar**
Project7 package will include:

- CourseGrades.java
- Essay.java
- FinalExam.java
- GradedActivity.java
- PassFailActivity.java
- PassFailExam.java
- Analyzable.java

Plagiarism/Academic Integrity Policy

Cheating and plagiarism will not be tolerated in this course. Students are to do their own assignments. Cases of copying, cheating, and plagiarism of assignments and/or tests, and any other violations, will be pursued to the maximum extent permitted by the University, which can include expulsion from the University. This applies equally to students who intentionally assist other students in academic dishonesty.

Any form of plagiarism or cheating will result in a failing grade on the assignment (at a minimum), and could result in a failing grade in the course or even university disciplinary action.

To learn more about the University policy on Cheating and Plagiarism, visit:

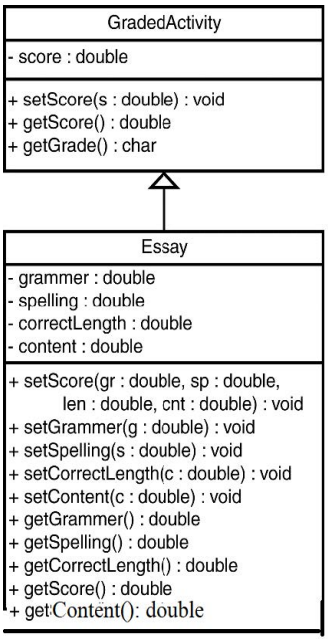
Academic Information and Regulations-Cheating and Plagiarism

Part 1:**Essay Class**

Design an Essay class that inherits from the **GradedActivity** class. The Essay class should determine the grade a student receives on an essay. The student's essay score can be up to 100 and is determined in the following manner:

```
Grammar: 30 points
Spelling: 20 points
Correct length: 20 points
Content: 30 points
```

Demonstrate the class in a simple program.




```
 Assign scores to the object.
 Grammer = 25 points, Spelling = 18 points,
 Length = 20 points, and Content = 25 points.
Essay termPaper = new Essay();
 termPaper.setScore(25.0, 18.0, 20.0, 25.0);
```

Sample output:

```
Term paper:
Grammar points: 25.0
Spelling points: 18.0
Length points: 20.0
Content points: 25.0
Total points: 88.0
Grade: B
```

Part 2 : **Course Grades**

In a course, a teacher gives the following tests and assignments:

- A **lab activity** that is observed by the teacher and assigned a numeric score.
- A **pass/fail exam** that has 10 questions. The minimum passing score is 70.
- An **essay** that is assigned a numeric score.
- A **final exam** that has 50 questions.

Write a class named `CourseGrades`. The class should have an array of `GradedActivity` objects as a field. The array should be named `grades`. The `grades` array should have four elements, one for each of the assignments previously described. The class should have the following methods:
Demonstrate the class in a program.

`setLab`	This method should accept a `GradedActivity` object as its argument. This object should already hold the student's score for the lab activity. Element 0 of the `grades` field should reference this object.
`setPassFailExam`	This method should accept a `PassFailExam` object as its argument. This object should already hold the student's score for the pass/fail exam. Element 1 of the `grades` field should reference this object.
`setEssay`	This method should accept an `Essay` object as its argument. (See Programming Challenge 4 for the `Essay` class. If you have not completed Programming Challenge 4, use a `GradedActivity` object instead.) This object should already hold the student's score for the essay. Element 2 of the `grades` field should reference this object.
`setFinalExam`	This method should accept a `FinalExam` object as its argument. This object should already hold the student's score for the final exam. Element 3 of the `grades` field should reference this object.
`toString`	This method should return a string that contains the numeric scores and grades for each element in the `grades` array.

Following classes extends from GradedActivity…

Essay class: please see the class description in step one.

FinalExam class:

Private Member Fields:

numQuestions: int
pointsEach: double
numMissed: int

Public methods:

Overloaded constructor: void
accepts as arguments the number of questions on the exam and the number of questions the student missed.

getPointsEach(): double
getNumMissed(): int

PassFailActivity class:

Private Member Fields:

minPassingScore: double // Minimum passing score

Public methods:

Overloaded constructor

Accept the minimum passing score as its argument

getGrade: char

The getGrade method returns a letter grade determined from the score field. This method overrides the superclass method.

PassFailExam class *extends* from PassFailActivity

Private Member Fields:

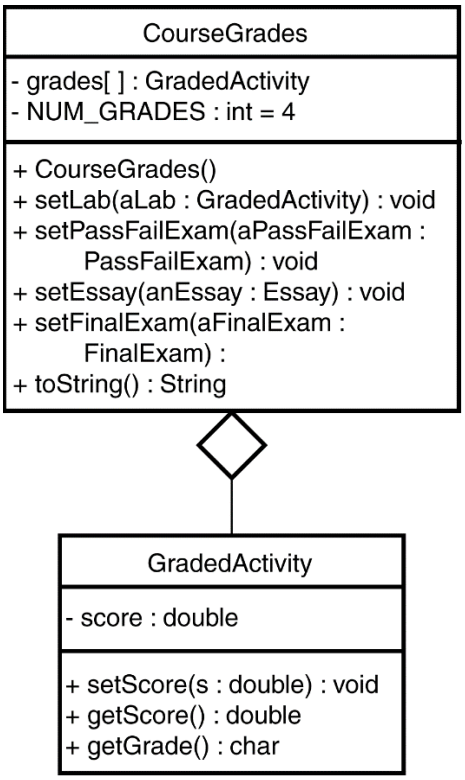
numQuestions: int
pointsEach: double
numMissed: int

Public methods:

Overloaded constructor: void

accepts as arguments the number of questions on the exam and the number of questions the student missed.

getPointsEach(): double
getNumMissed(): int



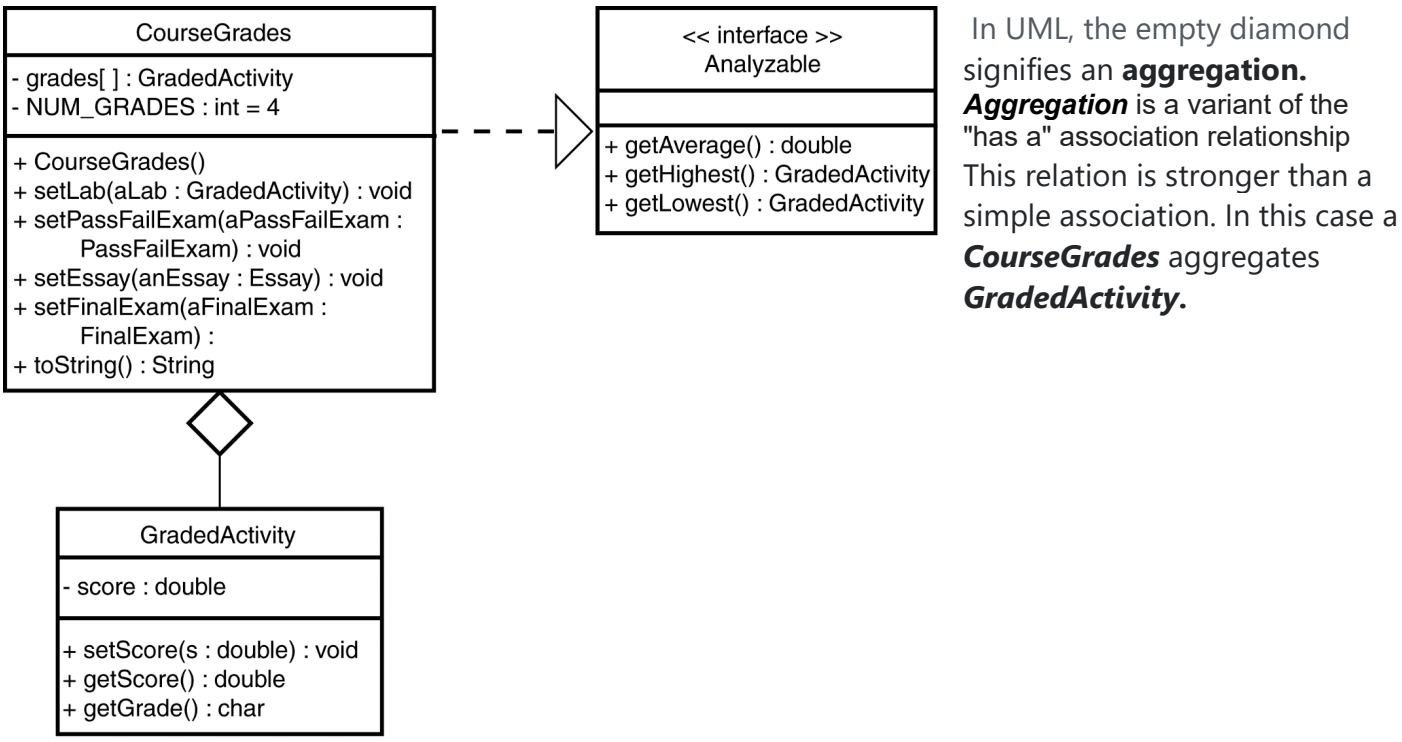
Sample output for these specific inputs:

```
Create an object for the lab grade and set the lab score to 85.
Create an object for the pass/fail exam.
20 total questions, 3 questions missed, minimum passing score is 70.
Create an object for the essay grade and Set the essay scores.
Grammer = 25, spelling = 18, length = 17, content = 20.
Create an object for the final exam. 50 questions, 10 missed.
```

```
Lab Score: 85.0 Grade: B
Pass/Fail Exam Score: 85.0      Grade: P
Essay Score: 80.0       Grade: B
Final Exam Score: 80.0  Grade: B
```

Part 3: **Analyzable Interface**

Modify the **CourseGrades** class you created in part2 so that it implements the following interface:
The **getAverage** method should return the average of the numeric scores stored in the grades array.
The **getHighest** method should return a reference to the element of the grades array that has the highest numeric score. The **getLowest** method should return a reference to the element of the grades array that has the lowest numeric score. Demonstrate the new methods in a complete program.



In UML, the empty diamond signifies an **aggregation.** ***Aggregation*** is a variant of the "has a" association relationship This relation is stronger than a simple association. In this case a ***CourseGrades*** aggregates ***GradedActivity.***

```
Lab Score: 85.0 Grade: B
Pass/Fail Exam Score: 85.0      Grade: P
Essay Score: 80.0       Grade: B
Final Exam Score: 80.0  Grade: B
Average score: 82.5
Highest score: 85.0
Lowest score: 80.0
```