Homework : Java-Based Student Semester Result

Note:
Any similarity between two projects will result in the grade being halved.
Any student who cannot explain their code during the review will receive a grade of zero.

The objective of the homework is to develop a Java program that calculates a student's semester results based on their grades.

- The program receives as input:
  - List of CourseUnits and courses for a semester;
  - grades in each course (exam, tutorials, and practicals).
- As output, the program displays the results of a semester (Average and credits earned) and allows the user to view detailed results (grades, Average , and credits earned for each CourseUnits and each course).

To build the program, we plan to implement a set of classes (`Semester`, `CourseUnit`, `Course`, `LectureCourse`, `LectureTutorialCourse`, `LectureTutorialPracticalCourse`) and an interface (`Evaluable`):

1. The `Evaluable` Interface

The `Evaluable` interface declares two abstract methods:

```
double calculateAverage();
int calculateCreditsEarned();
```

- **The `Course` Class** is characterized by:
  - The `title` (string);
  - A `credit` (integer);
  - A `coefficient` (integer);
  - An `examGrade` (real).

- The class `Course` implements the `Evaluable` interface.
- The `Course` class has three subclasses : `LectureCourse` (Course with lectures only), `LectureTutorialCourse` (Course with lectures and tutorials), and `LectureTutorialPracticalCourse` (Course with lectures, tutorials, and Practicals):

  ✓ **LectureCourse :**
  `average = examGrade`

  ✓ **LectureTutorialCourse**
  Includes a `tutorialGrade` (double)
  Constant: `COEF_TUTORIAL = 0.4`

```
average = examGrade * (1 - COEF_TUTORIAL) + tutorialGrade * COEF_TUTORIAL
```

  ✓ **LectureTutorialPracticalCourse**
  Includes `tutorialGrade` and `practicalGrade` (both doubles)
  Constants: `COEF_TUTORIAL = 0.2`, `COEF_PRACTICAL = 0.2`

```
average = examGrade * (1 - COEF_TUTORIAL - COEF_PRACTICAL) +
          tutorialGrade * COEF_TUTORIAL +
          practicalGrade * COEF_PRACTICAL
```

- The credits earned for a course are equal to its credit value if the average is >=10, and 0 otherwise.

- The class `Course` and its subclasses implement constructors that initialize all attributes.
- The class `Course` and its subclasses implement getter and setter methods for each attribute.
- The class `Course` and its subclasses implement a `toString()` methods that returns a string of the form:

```
-----------------------------------
Course Title: Object-Oriented Programming
Credits: 4
Coefficient: 3
Exam Grade: 10.0
Tutorial Grade: 10.0
Practical Grade: 10.0
Average: 10.0
Credits Earned: 4
```

2. The `CourseUnit` class

- A CourseUnit is characterized by:
  - A `code` (string);
  - A `type` (fundamental, methodological, discovery, or transversal);
  - A `listOfCourses` : a list (an object of the `ArrayList` class) of courses (objects of class `Course`) in the CourseUnit.
- The `CourseUnit` class implements a constructor that initializes its attributes. Initialization of the list of courses by the constructor is limited to creating an object of the `ArrayList` class (the courses will be added later).
- The `CourseUnit` class implements getter and setter methods for all attributes.
- The `CourseUnit` class declares the following methods:
  - `credit()` (returns the sum of credits of courses in its `listOfCourses`);
  - `coefficient()` (returns the sum of the coefficients of courses in its `listOfCourses`);

- The `CourseUnit` class declares a `toString()` method that returns a string in the following format:

```
============================================================
CourseUnit: UEF212 Credits: 13 coefficient: 8
 Grade: 11.69
 Credits earned: 13
 --------------------List of subjects--------------------
 Course: Object-Oriented Programming
 Credits: 4
...
------------------------------------------------------------
 Course: web development
 Credits: 4
...
```

- The `CourseUnit` class implements the `Evaluable` interface:

  - CourseUnit Average $= \frac{\sum a_i \times c_i}{\sum c_i}$

Where:

- ✓ a_i: Average of course i in `listOfCourses`
- ✓ c_i: coefficient of courses i in `listOfCourses`

- The credits earned for a `CourseUnit` equals to the sum of the credits for its courses if its average is >=10, and to the sum of the credits earned for its courses otherwise.

3. The `Semester` class

- A `Semester` is characterized by:
  - A field (Computer Science, Mathematics, etc.);
  - A year (1,2,3)
  - A code (L1,L2,L3,etc.);
  - A list `listOfCU` of CourseUnits (an `ArrayList` of objects of `CourseUnit` class).

- The `Semester` class implements a constructor that initializes all attributes.
- The `Semester` class implements getter and setter methods for all attributes.
- The `Semester` class declares the following methods:
  - `credit()` (returns the sum of credits of CourseUnits in `listOfCU`);
  - `coefficient()` (returns the sum of the coefficients of CourseUnits in `listOfCU`);
- The `Semester` class implements the **`Evaluable`** interface**:**

  - Semester Average $= \frac{\sum CUa_i \times c_i}{\sum c_i}$

  ✓ CUa_i: Average of `CourseUnit` i in `listOfCU`
  ✓ c_i: coefficient of `CourseUnit` i in `listOfCU`

  - The credits earned for a semester equals the sum of the CourseUnits credits if the semester average is >= 10, and to the sum of the earned credits of its CourseUnits otherwise.
  -

- The `Semester` class declares a `toString()` method that returns a string of the form:

```
----------------------------
Number:1
Field : Computer science
Year: 2
Code: L4
Average: 11.50
Credits earned: 30
```

I. Assignment

1. Implement the classes and interface described above
2. Declare a `MainProgram` class, which has no attributes and contains a single `main()` method that will allow the user to:
   - Enter a list of CourseUnits, along with the courses for each unit.
   - Enter a semester's information, then display their semester results (averages and credits earned), as well as their detailed results (averages and credits earned for each CourseUnit and each course).

Appendix

ArrayList

Completing this project requires working with lists using the **ArrayList** class from the `java.util` package. A detailed description of this class is available at:
https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Among the methods of the `ArrayList` class:

- `add(E e)`: adds an element to the list
- `get(int i)`: returns the element at index *i*
- `remove(int i)`: removes the element at index *i*
- `size()`: returns the number of elements in the list
- `clear()`: removes all elements from the list