CENG 201
Object-Oriented Programming
HOMEWORK 3
Due Date:

In this homework, you will create an application for generating transcripts of students. In order to perform this, you should implement the following classes defined below. Please go over each item carefully and make sure that you have covered each of these requirements in your homework.

- You will create two packages in your homework, namely, `hw3.util` and `hw3.main`. The first package will contain the utility classes that will be used in your application. The second one, however, will contain the main classes that perform the generation of transcripts.
- Package `hw3.util` will contain the definition of the enum type `Grade`, and a test class for it, named `GradeTest`.
- Package `hw3.main` will contain the `CourseGrade`, `Transcript` and `GenerateTranscript` classes.
- The requirements for each class are specified below.

1. Enum Grade

1. The enum type `Grade` represents the grade of a student and it should define constants representing the grades of A, B, C, D, F.
2. The constants should be named `Grade.A, Grade.B, Grade.C, Grade.D, Grade.F`.
3. `Grade` should have two instance fields, `stringValue` and `numericValue`.
4. `stringValue` should contain a `String` representation of the letter grade, e.g., "C", "A". `numericValue` should contain the numeric grade corresponding to the letter grade, which are 4, 3, 2, 1, 0 for A, B, C, D, and F respectively.
5. Implement an appropriate `toString()` method for this class that will print the letter and numerical grade stored for a grade as in the following form (*for Grade.F*):

```
Grade F corresponds to numeric grade 0
```

2. GradeTest

1. In the package, `hw3.util`, write a test program `GradeTest` that uses the enhanced `for` loop, and the `values()` method of `Grade` to print out the following output:

```
Grade F corresponds to numeric grade 0.
Grade D corresponds to numeric grade 1.
.
.
.
Grade A corresponds to numeric grade of 4.
```

2. Use toString() method of Grade class during printing.

3. CourseGrade

1. The `CourseGrade` class will keep information about the department the course is offered, course code of the course, credit of the course and the grade taken for that course in the `courseDepartment`, `courseCode` and `courseCredit` and `gradeTaken` fields.
2. `courseDepartment` is a four letter acronym indicating the department, the class is offered.
3. The possible values of departments in your application are CENG, COMP, ECE, ME and MATH.
4. `courseCode` is a three digit code (e.g., 101, 200, or 590) for the course. This number should be between 100 and 599.
5. `courseCredit` is an integer that indicates how many credits the course is worth. The only valid values are 3 and 4.
6. `gradeTaken` is a `Grade` value that represents the grade obtained in this course.
7. Provide all appropriate get and set methods for the fields above. Set methods should perform a check on their arguments and set the field to a default value if the argument to the set method is invalid.
8. Default values for `courseCode` is 100, `courseCredit` is 4, `courseDepartment` is CENG and `gradeTaken` is `Grade.F`.
9. The set method for `gradeTaken` should be overloaded as specified below:

    - `setGradeTaken(double val)` should select the letter grade whose numeric value is closest to `val` if `numValue` is between 0 and 4.0 . Otherwise it should set `gradeTaken` to `Grade.F`
    - `setGradeTaken (Grade g)` should simply set the grade to the value given by `g`

10. Implement four overloaded constructors which will take either
    - only `courseDepartment`
    - `courseDepartment` and `courseCode`
    - `courseDepartment`, `courseCode` and `courseCredit`
    - `courseDepartment`, `courseCode` , `courseCredit` , and `gradeTaken`

    as input. They will fill in the missing parameters for the fields to the default values mentioned in item 8.

11. The first , second and third constructor will call the fourth constructor and that constructor should set all the fields using the set method of those variables.
12. Implement a `toString()` method for this class that will return a string value that contains all the information stored for a CourseGrade object in the following form:

```
Department: CENG CourseCode: 201 Credit: 5 Grade: F
```

4. Transcript

1. This class keeps the transcript of a student with id `studentID`.
2. `studentID` is an integer value storing the id of the student.

3. Each student transcript can contain grades of arbitrary number of courses, therefore, the class keeps an arbitrary number of `CourseGrade` objects that stands for the grades taken by the student throughout her education. Therefore, you should include an appropriate field in the class to keep these arbitrary number of `CourseGrade` objects.
4. The class also calculates and stores the `GPA` of the student as a double field which is the average of all grades taken by the student for all the courses. GPA is therefore a value between 0.0 and 4.0.
5. Class has a one-argument constructor that takes an integer containing the student's id ,sets `GPA` to 0.0 and initializes the list of `CourseGrade` objects.
6. It has an `addCourseTaken` method which will take a `CourseGrade` object as input and inserts `it` into the list of course grades stored in the Transcript. This method should first check that `CourseGrade` object given as parameter is not null. Otherwise, it should not perform the addition and print an error message. In addition, it should update the `GPA` of the student accordingly.
7. The class should have a `toString()` method that prints the student's transcript. This method should use the `toString()` method of `CourseGrade` class. The student's `GPA` together with each of the course grades should be printed in the following form.

```
Student ID: 1112234
Department: CENG Code: 201 Credit: 5 Grade: C
Department: CENG Code: 201 Credit: 5 Grade: A
Department: CENG Code: 201 Credit: 5 Grade: C
Department: CENG Code: 201 Credit: 5 Grade: A
Department: CENG Code: 201 Credit: 5 Grade: B
GPA:3.0
```

5. GenerateTranscript

1. Write a class named `GenerateTranscript` that will implement two methods.
2. The method takeInputFromUser takes input from the user to enter a number of classes and grades and adds them to a transcript object one by one using `addCourseTaken` method. When the user is done entering grades by pressing endoffile indicator, your program should print the transcript of the student whose grades have been entered. A sample execution of the program is given below.

```
Enter Student Id:
1112234
Enter Department: CENG
Enter Course Code: 201
Enter Credit: 5
Enter Grade: 4
Enter Department: CENG
Enter Course Code: 205
Enter Credit: 5
Enter Grade: 3
........
```

```
Student ID: 1112234
Department: CENG Code: 201 Credit: 5 Grade: C
Department: CENG Code: 201 Credit: 5 Grade: A
Department: CENG Code: 201 Credit: 5 Grade: C
Department: CENG Code: 201 Credit: 5 Grade: A
Department: CENG Code: 201 Credit: 5 Grade: B
GPA:3.0
```

3. Implement takeInputFromFile method which will take a text file name from the user as input. The text file will contain information about courses taken by a student.
4. A sample file would be as follows:

   120122
   ```
   CENG 201 5 3.5
   MATH 200 3 2.5
   ECE 121 5 2.2
   ```
   First line of the file will include the id of the student and each following line will include department, course code, credit and grade in order.
5. The method will ask the user to enter the name of the text file:
   ```
   Enter filename:
   ```

6. After filename is taken it will process the file and generate and print the student transcript as specified in item 2. The output will be printed to the console.