

Assignment 3: 46 points

Build a Class hierarchy: College Course

Your goal is to create **class hierarchy** to implement a college **course**. The course has a **list of assessments**: coding assignments, participation quizzes, and exams. You will be making polymorphic calls for each type of assessment to obtain its respective information. Each level of class hierarchy will be responsible for its own data and for adding it to the assessment info. You should also write a demo program that tests your college course implementation.

Your project should have the following:

- An interface `IAssessment` to access the assessment hierarchy. It should have methods for generating the assessment info, and for setting and getting the due date for the assessment. (1 point)
- An enumeration for the assessment type. The possible types of assessments are participation quiz, coding assignment, midterm exam, and final exam. (1 point)
- An abstract class `Assessment` that will implement the `IAssessment` interface and will be at the root of the assessment class hierarchy. This class should have:
 - Three immutable fields for the name, the description, and the maximum number of credit points for the assessment. (1.5 points)
 - A mutable field for the due date (.5 points)
 - A constructor that sets all four fields. (1 points)
 - Getters for all fields and a setter for the mutable field. (2 points)
 - An implementation of the `IAssessment` interface. The method for generating the assessment info `generateAssessmentInfo()` needs to create a `StringBuilder` and call its overloaded version that takes a `StringBuilder` as an argument. (2 points)
 - This overloaded method needs to append the assessment type, name, description, maximum points, and due date to the `StringBuilder`. (2 points)
 - In order to get the assessment type it needs to call an abstract method `getType()` (defined in this class) that will be implemented in each subclass representing different assessment types. (1 point)
- An abstract class `Quiz` that extends `Assessment` and will be a common subclass for different types of quizzes. This class should have:

- Two immutable fields for the number of minutes to complete the quiz and for the class material covered by the quiz. (1 point)
- A constructor that constructs the superclass and then sets these two fields. (1 point)
- Getters for the fields (1 point)
- A method that overrides the superclass's `generateAssessmentInfo` method taking a `StringBuilder`, calls the superclass's overridden implementation and then adds the class material and the time limit to the `StringBuilder`. (2 points)
- Three subclasses for the `Quiz` class: `ParticipationQuiz`, `MidTermExam`, and `FinalExam`. All three classes need to have their own static immutable fields for the assessment type and the assessment description. The assessment type is one of the enumerated values you defined earlier and the description is the text describing the assessment. (3 points each)
 - All three subclasses will pass their description references to the superclass constructor so that they could be set in the `Assessment` class
 - All three subclasses will implement the abstract method `getType()` defined in the `Assessment` class and return their respective types.
- A `CodingAssignment` class that extends `Assessment`.
 - Similarly to the three `Quiz` subclasses, it needs to have its own static immutable fields for the assessment type and the assessment description. The assessment type is one of the enumerated values you defined earlier and the description is the text describing the assessment. (2 points)
 - Similarly to the three `Quiz` subclasses, it also needs to implement the abstract method `getType()` defined in the `Assessment` class and return its type. (1 point)
 - It should also have an enumerated immutable field for the assignment difficulty and a getter for this field. (1 point)
 - Its constructor should set the difficulty after constructing the superclass. (1 point)
 - Similarly to the `Quiz` class, but unlike the three `Quiz` subclasses, it should have a method that overrides its superclass's `generateAssessmentInfo` method taking a `StringBuilder`, calls the superclass's overridden implementation and then adds the assignment difficulty to the `StringBuilder`. (2 points)
- An enumeration for the assignment difficulty. The possible difficulty values are easy, medium, and hard. (1 point)

- A `Course` class. This class should have:
 - An immutable name, a mutable semester, and an immutable `ArrayList` of assessments. Note that although the field for assessments is immutable the `ArrayList` it is referencing is mutable. The type of references in the `ArrayList` should be `IAssessment`. (1.5 points)
 - A constructor that sets the name field. (1 point)
 - Getters for the name and the semester (but not the assessments) and a setter for the semester. (1.5 points)
 - A method to add an assessment to the list of assessments. (1 point)
 - An overridden `toString()` method that creates a `StringBuilder`, adds the course name to it, and then iterates over the list of assignment making a polymorphic call to `generateAssessmentInfo` and adding the result to the `StringBuilder`. (2 points)
- A `CourseDemo` class with the `main()` method. The `main()` method should:
 - Create a `Course` object and set its semester. (1 point)
 - Create one or two assessments of **each** defined type and add it to the course. (3 points)
 - Generate and print the course info. (1 points)

Your output should look similar to this:

```

Programming Fundamentals III
Fall 2021
Assessments:
    Type: PARTICIPATION
    Name: Week1 Participation Quiz
    Description: Participation Quizzes test weekly reading and are worth 10%
of the grade
    Points: 10
    Due: Aug 28
    Material: Chapter 6
    Time limit: 20 min

    Type: MIDTERM
    Name: Exam 1
    Description: There are two midterms worth 15% each
    Points: 100
    Due: Sep 27
    Material: Chapters 6-11, 15
    Time limit: 75 min

    Type: MIDTERM
    Name: Exam 2
    Description: There are two midterms worth 15% each
    Points: 100
  
```

Due: Nov 8
Material: Chapters 16-19
Time limit: 75 min

Type: FINAL
Name: Final Exam
Description: Comprehensive Final covering all class material and worth 20% of the grade
Points: 100
Due: Dec 6
Material: Chapters 6-11, 15-19
Time limit: 120 min

Type: CODING
Name: Assignment 1c: Marble Fun Facts
Description: Weekly coding assignments are worth 40% of the grade
Points: 10
Due: Aug 28
Difficulty: EASY

Type: CODING
Name: Assignment 2: Cash Register
Description: Weekly coding assignments are worth 40% of the grade
Points: 24
Due: Sep 4
Difficulty: MEDIUM

Type: CODING
Name: Assignment 3: College Course
Description: Weekly coding assignments are worth 40% of the grade
Points: 46
Due: Sep 11
Difficulty: HARD

Notes:

- Your classes and methods should have javadocs. Uncommented code is an automatic **1-point** deduction.
- Follow the Assignment documentation format (published in Canvas). Incorrectly formatted submissions will have an automatic **1-point** deduction.
- **Never** edit the program output. Program output inconsistent with the code is an automatic **0** for the assignment.