CSC148 Assigment 1

Hyungmo Gu

April 27, 2020

1) Get the starter code and read the documentation

1. Download the zip file that contains the starter code here a1.zip
2. Unzip the file and place the contents in pycharm in your a1 folder (remember to set your a1 folder as a sources root)
3. You should see the following files:
    - *course.py*
    - *criterion.py*
    - *grouper.py*
    - *survey.py*
    - *tests.py*
    - *example_tests.py*
    - *example_usage.py*
    - *example_course.json*
    - *example_survey.json*

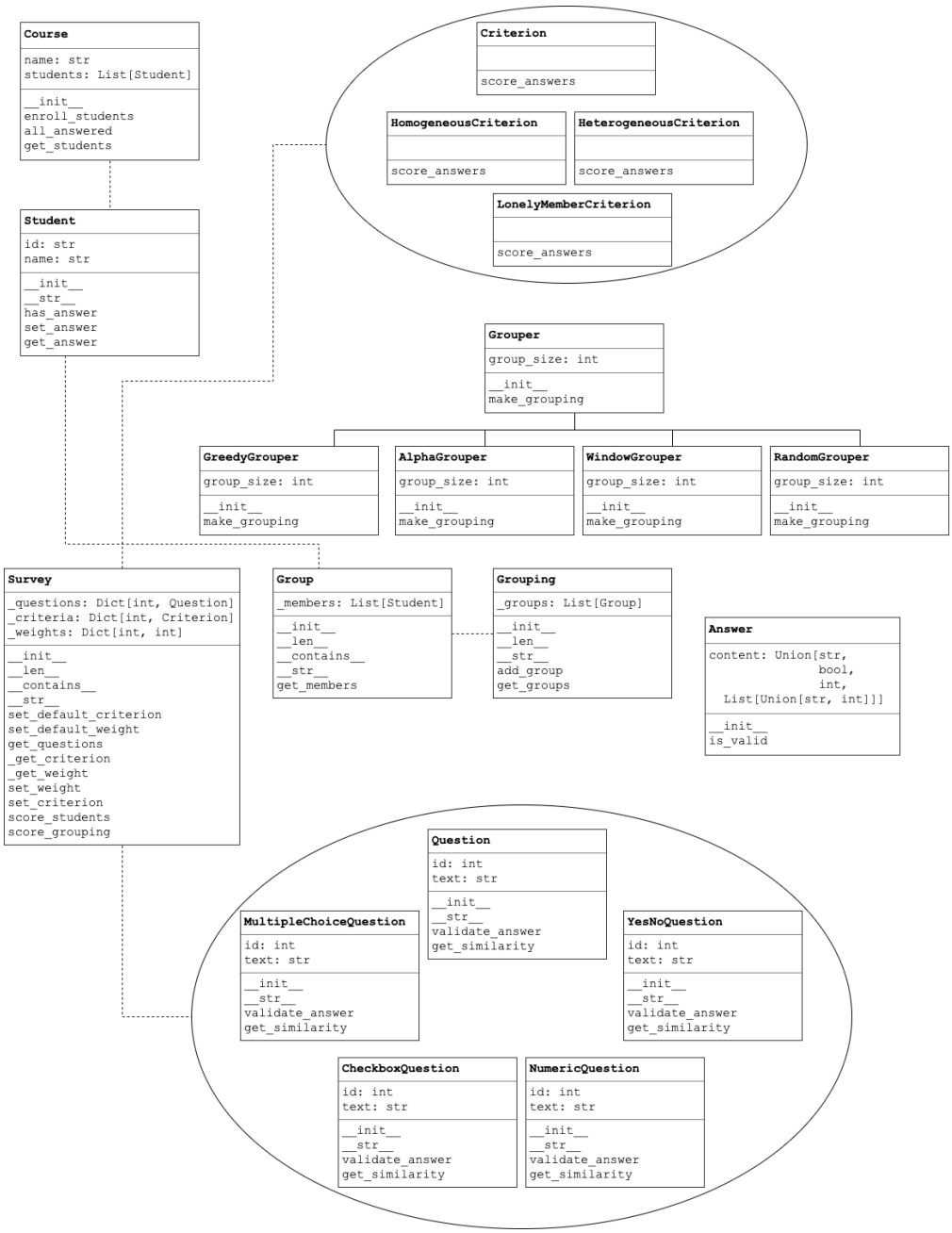
For this assignment, you will be required to edit and submit the following files only:

- course.py
- criterion.py
- grouper.py
- survey.py
- tests.py

If you look at these files you will notice that you have been given the signature and docstrings for all classes and methods. Read through these docstrings carefully; they describe how you are expected to implement these classes and methods.

A picture!

It might be difficult to imagine how all the classes defined in these files will interact before you start writing the code itself. To help you out, here is a diagram of all the classes you will be asked to contribute to for this assignment:



Note that the attributes and methods shown in this diagram are only the ones that we have given you in the starter code. You may need to define additional private attributes or private helper methods.

Legend:

- dashed lines indicate a composition relationship between classes
- solid lines indicate an inheritance relationship between classes
- a solid circle around a group of classes indicates that there exists an inheritance relationship between these classes but it is not defined (you get to decide!)

Test your code!

- Try running the example_tests.py file: all of the tests should fail because you haven't written any code yet!
- Try running example_usage.py file: you should get an error since you haven't written any code yet!
- Open up the tests.py file: it is empty! This is where you will be writing all of your tests for this assignment

Something to think about!

Unlike A0, you will be submitting code split across multiple files. Open up each of the files and look at which functions and classes are defined in each file. Why do you think the files were organized in this way? Is there a different way we could have organized these files?

2) Complete the Student Class

The Student class represents a student who can be enrolled in a university course.

The starter code for the *Student* class can be found in *course.py*. Open up this file and read through the docstrings for each of the the *Student* class's methods. Then, implement each of the methods in the *Student* class.

Remember: you may need to define additional private attributes or private helper methods!

Test your code!

- Write at least one unit test for each method in *Student*. You are not required to write tests for initializers.
- You should write these tests in the *tests.py* file.
- Once you have finished writing these tests, run all the tests in *test.py*. Make sure your code passes all your tests before moving on.
- Run the tests in *example_tests.py*, the tests in the *TestStudent* class should now pass.

Something to think about!

The *Student.has_answer* method asks you to check if a student has a valid answer to a given question. Do we have a way to determine if an answer is valid or not yet? Answer: no and we won't until we complete step 4. You may need to come back and finish this method after completing step 5.

3) Complete the Course Class

The *Course* class represents a university course.

The starter code for the *Course* class can be found in *course.py*. Open up this file and read through the docstrings for each of the the *Course* class's methods. Then, implement each of the methods in the *Course* class. You may find the function *sort_students* helpful.

Remember: you may need to define additional private attributes or private helper methods!

Test your code!

- Write at least one unit test for each method in *Course*. You are not required to write tests for initializers.
- You should write these tests in the *tests.py* file.
- Once you have finished writing these tests, run all the tests in *test.py*. Make sure your code passes all your tests before moving on.
- Run the tests in *example_tests.py*, the tests in the *TestCourse* class should now pass.
- Something to think about!
- The *Course.all_answered* method asks you to check if all students have a valid answer for every question in a *Survey*. Which steps do you need to complete before you can finish this method? You may have to come back later to finish the *Course.all_answered* method.

4) Complete the Question Classes

The file *survey.py* contains an abstract *Question* class, and the following classes for representing different types of questions that you might find on a survey:

- Question
- MultipleChoiceQuestion
- NumericQuestion
- YesNoQuestion

- CheckboxQuestion

As well as defining the text of the question itself, these classes also specify what are valid answers to these questions.

Open up *survey.py* and read through the docstrings for the methods in these question classes.

You might notice that we have not defined any inheritance hierarchy between these classes. You get to decide what it should be. However, in doing so you must follow these rules:

1. The abstract Question class should not inherit from any class other than object.
2. All other *Question* classes should inherit from the abstract *Question* class either directly or indirectly.
3. At least one non-abstract *Question* class should inherit from another non-abstract Question class.
4. There are many possible inheritance structures you could choose. Remember that one of the requirements for this assignment is to avoid writing duplicate code. Think about which sort of inheritance structure best lets you avoid duplicate code.

Implement each of the methods in the Question classes. You may remove a method that we included in the starter code for a child class if you wish to simply inherit the parent's method rather than to override it.
Remember: you may need to define additional private attributes or private helper methods!

Test your code!

- Write at least one unit test for each method in each of the Question classes. You do not need to write tests for abstract methods or initializers but you do need to write tests for inherited methods.
- For example, even if you structure your code so that a child class inherits its *validate_answer* method without modification from the parent class, you still need to write separate tests for the *validate_answer* method in the parent class and the child class.
- You should write these tests in the *tests.py* file.
- Once you have finished writing these tests, run all the tests in *test.py*. Make sure your code passes all your tests before moving on.
- Run the tests in *example_tests.py*, the tests in the *TestMultipleChoiceQuestion*, *TestNumericQuestion*, *TestYesNoQuestion*, and *TestCheckboxQuestion* class should now pass.

Something to think about!

The *validate_answer* methods ask you to check if an answer is a valid answer for this question. Do we have a enough information about the *Answer* class in order to complete this method now? You may need to come back and finish this method after completing step 4.

5) Complete the Answer Class

The *Answer* class represents an answer to one of the questions you wrote classes for in Step 3.

The starter code for the *Answer* class can be found in *survey.py*. Open up this file and read through the docstrings for each of the the *Answer* class's methods. Then, implement each of the methods in the *Answer* class.

Remember: you may need to define additional private attributes or private helper methods!

If you have not implemented the *validate_answer* methods in the *Question* classes, the *Course.all_answered* and the *Student.has_answer* methods yet, go back and finish them now.

Test your code!

- Write at least one unit test for each method in *Answer*. You are not required to write tests for initializers.
- You should write these tests in the *tests.py* file.
- Once you have finished writing these tests, run all the tests in *test.py*. Make sure your code passes all your tests before moving on.
- Run the tests in *example_tests.py*, the tests in the *TestAnswer* class should now pass.

Something to think about!

The *Answer* class is one of the simplest classes that we will implement in this assignment. What is the advantage of creating such a simple class? Are there any disadvantages?

6) Complete the Criterion Class

7) Complete the Group Class

8) Complete the Grouping Class

9) Complete the Survey Class

10) Complete the helper functions in *grouper.py*

11) Complete the Grouper Classes

12) Test the Code Again

13) Submit your work