Assignment 2

Create a new Eclipse workspace named "**OOP**" on the desktop of your computer. Create a Java Project named "Assignment2". Download the **assignment2-sample-code-for-student.zip** from iSpace and unzip it. Next, we will import the sample code to you project (right click the src -> Import… -> File System -> locate to the unzipped folder -> import all the questions to your project). Answer all the questions below. At the end, create a ZIP archive of all the src. In Eclipse, right click src -> Show In -> System Explorer, then zip up all folders inside the src folder. Rename your src.zip to "**Assignment2_1234567890.zip**" (replace **1234567890** with your student ID number). Upload the ZIP file onto AutoLab. For more details, refer to the file ***How to Use Eclipse and submit to Autolab for student.pdf*** on iSpace. Note: you **must** use the sample code, so AutoLab can work correctly.

Here are a few extra instructions:

- Give meaningful names to your variables so we can easily know what each variable is used for in your program.
- Put comments in your code (in English!) to explain WHAT your code is doing and also to explain HOW your program is doing it.
- Make sure all your code is properly indented (formatted). Your code should be beautiful to read.

Failure to follow these instructions will result in you losing points.

question 1

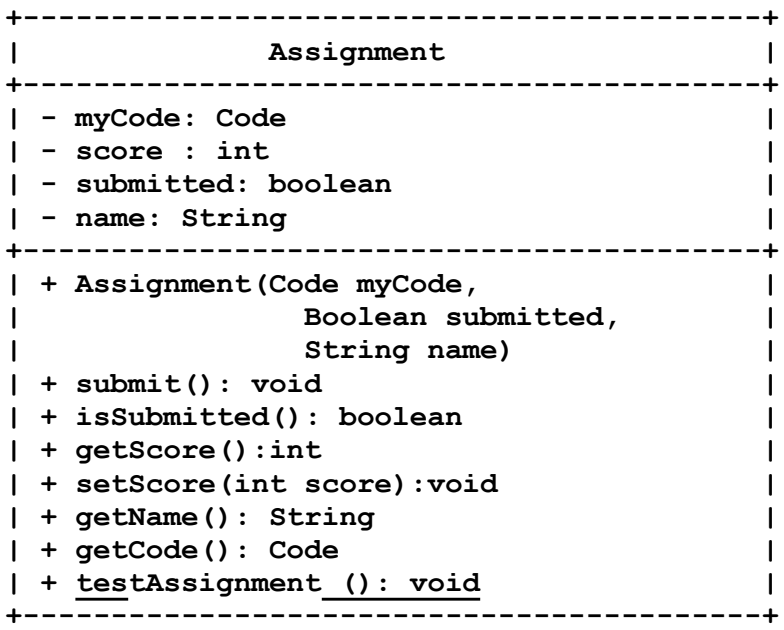
Create four classes with the following UML diagrams.

```
+-----------------------------------------+
|              Code                       |
+-----------------------------------------+
| - canRun: boolean                       |
| - canCompile : boolean                  |
| - lines: int                            |
+-----------------------------------------+
| + Code()                                |
| + Code(boolean canCompile, boolean canRun,|
|        int lines)                       |
| + compile(): boolean                    |
| + run(): boolean                        |
| + debug(): void                         |
| + coding(int lines): void               |
| + countLines(): int                     |
| + testCode(): void                      |
+-----------------------------------------+
```

where:

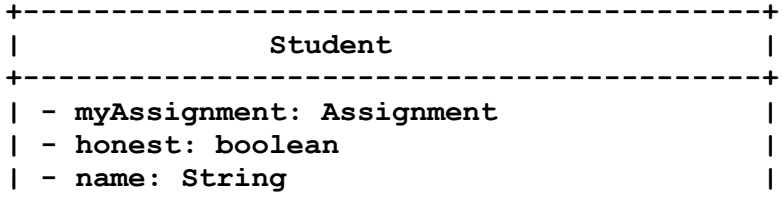
- **canRun** is a private instance variable describing whether the code can run correctly, the default value is false.
- **canCompile** is a private instance variable describing whether the code can be compiled correctly, the default value is false.
- **lines** is a private instance variable describing the number of lines in the code, the default value is 0.
- **compile()** is a public method that returns as result whether the code can be compiled correctly.
- **run()** is a public method that returns as result whether the code can be run correctly.

- **debug()** is a public method that corrects all errors and makes the code compile and run correctly.
- **coding()** is a public method that write the code to a certain number of lines.
- **countLines()** is a public method that returns the number of lines in the code.
- The class has two overloaded constructors, pay attention to their differences.



Where

- **myCode** is a private instance variable the contains a Code object.
- **score** is a private instance variable describing the score of the current assignment.
- **submitted** is a private instance variable describing whether the assignment is submitted on time before deadline.
- **name** is a private instance variable describing the student's name who finished this assignment.
- **submit()** is a public method that submits the assignment on time.
- **isSubmitted()** is a public method that returns whether the assignment is submitted.
- **setScore()** is a public method that set the score of the current assignment
- **getScore()** is a public method that get the score of the current assignment.
- **getName ()** is a public method that returns the name variable.



```
+-----------------------------------------+
| + Student(String name,Boolean honest)    |
| + getAssignment(): Assignment            |
| + writeAssignment(Assignment a): void    |
| + copyAssignemnt(Assignment a): void     |
| + getName(): String                      |
| + testStudent (): void                   |
+-----------------------------------------+
```

- **name** is a private instance variable the contains the student's name.
- **myAssigment** is a private instance variable the contains an Assignment object, the default value is NULL.
- **honest** is a private instance variable describing whether the student is honest or not. An honest student will write assignment by him/herself, otherwise will copy from others. The default value for honest is true.
- **getAssignment()** is a public method that return the student's assignment.
- **writeAssignment()** is a public method that accepts an assignment and assign it to **myAssignment** variable.
- **copyAssignemnt()** is a public method that copy an existing assignment a from someone else, and assign it to **myAssignment** variable. Pay attention when copying from others, the name variable of the current student object should NOT be equal with the name variable in the copied assignment object.
- **getName ()** is a public method that returns the name variable of the student.

```
+-----------------------------------------+
|              Teacher                    |
+-----------------------------------------+
| - name: String                          |
+-----------------------------------------+
| + Teacher(String name)                  |
| + grading(Student s): int               |
| + testTeacher(): void                   |
+-----------------------------------------+
```

- **name** is a private instance variable the contains the teacher's name.
- **grading()** is a public method that gives score to the assignment. The rule of grading is as follows:
  - **If the assignment is not submit on time, the score is 0;**
  - **If the assignment is submitted on time, but cannot compile, the score is 0;**
  - **If the assignment is submitted on time, can compile, but cannot run, the score is 50;**
  - **If the assignment is submitted on time, can compile, can run, but the line number is less than 100, the score is 80;**
  - **If the assignment is submitted on time, can compile, can run, but the line number is equal or larger than 100, the score is 100;**
  - **If the assignment's name (the student who finished the assignment) is not equal to the name of the student who submitted the assignment, the score is 0;**

  Before the student s argument is passed into the grading method, you need to make sure there is at least an assignment object assigned to the student's myAssignment instance variable.
- **testTeacher()** is the test function for class Teacher. In order to test the class completely, you need to create and grade at least six students with different assignment scenarios according to the six rules in the grading() method.

Note:

if you want to compare constant strings, you can use the == operator to do that in Java, but if you want to compare dynamic strings (strings that have been created using the + operator while the program is running) then you must use the "equals" method, the == operator will not work in that case. For example, if s1 and s2 are two string variables, then you can compare the two strings by writing: s1.equals(s2) The result of the "equals" method is a boolean.

Add to your software a **Start** class with a main method that calls the test method of each of the four classes, to test everything in your software