Instructions

You are required to develop and test code that handles academic information related to professors, courses, and students. The steps to be done are as follows:

1. Using Eclipse, create a new java project and add to it a package named **midterm**
2. **[7 pts]** Define an abstract class **Person** as follows:
    a. It has a field of type **int** named **sin** (social insurance number) and another field of type **String** named **name**. Both fields are private.
    b. It has a public constructor that takes one parameter of type **int** and another one of type **String**, and initializes **sin** and **name** accordingly.
    c. It overrides **equals**. Two objects of type **Person** are equal if and only if their **sin** values are equal.
    d. It overrides **hashCode**.
    e. It overrides **toString** by returning **name**
    f. It has a public method **compareName** that takes an object of type **Person** as an argument and returns the result of comparing the name of the calling object with that of the argument. This method should throw an **IllegalArgumentException** if the argument is **null**. Note that you can compare the names using the **compareTo** method of class **String**.
    g. It should be defined such that it would not be possible to instantiate it using the **new** operator.
3. **[5 pts]** Define class **Professor** as follows:
    a. It inherits from **Person.**
    b. It doesn't have any fields.
    c. It has a public constructor that takes one parameter of type **int** and another one of type **String**, and initializes **sin** and **name** accordingly.
    d. It overrides **toString** by returning the name of the professor as **"Prof. [name]"**. For example, if the name is "X", the returned value would be "Prof. X"
4. **[10 pts]** Define class Student as follows:
    a. It inherits from **Person**.
    b. It has four package-private fields
        i. **cgpa** of type **double** (representing the cumulative grade points average)
        ii. **inCSCPOSt** of type **boolean** (indicating whether the student in enrolled in a CSC subject POSt)
        iii. **passedCSCA48** of type **boolean** (indicating whether the student has passed CSCA48)

        iv. **passedCSC207** of type **boolean** (indicating whether the student has passed CSC207)

   c. It has a public constructor that takes the following arguments and initializes the fields accordingly: **int** (to initialize **sin**), **String** (to initialize **name**), **double** (to initialize **cgpa**), **boolean** (to initialize **inCSCPOSt**), **boolean** (to initialize **passedCSCA48**), **boolean** (to initialize **passedCSC207**)

   d. It overrides **toString** by returning the name and cgpa information as **"[name], cgpa: [cgpa]"**. For example, if the name is "Sam" and the cgpa is 3.6, the returned value would be "Sam, cgpa: 3.6"

   e. It implements **Comparable**. Students should be ordered by **name** and if the names are equal, they should be ordered based on the **cgpa** values (lowest first).

5. **[12 pts]** Define class **Course** as follows:

   a. It has three package-private fields

      i. **code** of type **String** (representing the code of the course, e.g. "CSCB07")

      ii. **professor** of type **Professor** (representing the instructor of the course)

      iii. **students** of type **List<Student>** (representing the students registered in the course)

   b. It has a public constructor that takes one parameter of type **String** and another one of type **Professor**, and initializes **code** and **professor** accordingly. It also initializes **students** to an empty **ArrayList<Student>**

   c. It overrides **equals**. Two objects of type **Course** are equal if and only if their **code** values are equal.

   d. It overrides **hashCode**

   e. It has a public method named **isEligibile** that takes an object of type **Student** as an argument and returns a **boolean** value indicating whether the student is eligible to be registered in the course or not. To implement this method, you need to use the CSCB07 eligibility criteria. The student must satisfy the following three conditions:

      i. Passed CSCA48

      ii. cgpa>=3.5 or enrolled in a CSC subject POSt

      iii. Did not pass CSC207

   f. It has a public void method named **addStudent** that takes an object of type **Student** as an argument and adds it to **students** if it satisfies the following conditions:

      i. The student is eligible to be registered in the course.

      ii. The student is not already added to the list of students. You can use method **contains** of class **ArrayList** to check if that is the case.

   g. It has a public void method named **displayInfo** that takes no arguments and displays the course information in the following order: code, professor, students. Note that the list of students should be sorted before being displayed (you can use **students.sort(null);** to achieve that).

6. [8 pts] Define class **Administration** as follows:
    a. It should not be possible to have more than one instance of this class
    b. It has three fields whose types ensure that no duplicates are allowed and that the elements are stored in the same order they are added
        i. **professors** (a collection of all the instantiated professors)
        ii. **students** (a collection of all the instantiated students)
        iii. **courses** (a collection of all the instantiated courses)
    c. It has three methods of type **void**
        i. **addProfessor** that takes an object of type **Professor** as an argument and adds it to the collection of professors
        ii. **addStudent** that takes an object of type **Student** as an argument and adds it to the collection of students
        iii. **addCourse** that takes an object of type **Course** as an argument and adds it to the collection of courses
    d. Every time an object of type **Professor**, **Student**, or **Course** is instantiated in the code, it should be added to the appropriate collection. Note that you might need to modify other classes to achieve that.
7. [8 pts] The problem with method **isEligible** in class **Course** is that it is does not account for courses other than CSCB07. Even if it is to be used solely for CSCB07, modifying the eligibility conditions later on would require modifying class **Course**. As such, you are required to make the necessary changes so that any future modifications to the eligibility criteria could be done without modifying class **Course**.
8. **[20 pts]** Write JUnit tests to validate your code according to the following guidelines:
    a. Each test method should validate a specific behavior and should not contain code that is not relevant to the assertion. Test methods that violate any of these conditions will be discarded.
    b. One of the test methods should validate the behavior of **compareName** of class **Person** when a **null** value is passed as an argument. That is, the test method should validate that an **IllegalArgumentException** is thrown.
    c. A substantial part of the testing grade will be based on the percentage of branch coverage achieved by your test suite.

Submission

- Upload **all the source code files (.java files) as a single archive file** to “Term Test - Programming” on Quercus by 19:00 at the latest. Submitting the code after the deadline will be subject to a 5% penalty per minute.
- It is your responsibility to submit the proper files. Failing to do so (e.g. submitting .class files instead of .java files) will result in a grade of zero on the programming part of the test.