CENG 443 - Intr. to Object Oriented Programming Languages and Systems

Fall 2015 - Homework 2

Register, if you can!

Selim Temizer

Feedback : Between December 21st and December 24th, 2015
Due date : December 27th, 2015 (Submission through COW by 23:55)

In this homework, we will build an application to simulate the registration process that all students go through at the beginning of each semester. We will model each student as a separate thread, and start the threads to see how they progress.

In order to register, each student is supposed to take the following steps:

1. Go to the department laboratory and find an available computer (if necessary, wait until a computer becomes available)
2. Randomly take 3 distinct courses
3. Visit her/his advisor and ask the advisor to check the courses
4. If the advisor approves all courses, then go and join the after-registration party
5. If the advisor does not approve the courses, go to step 1 and redo everything again
6. Once all students finish registration, the party starts, and everyone starts chatting

Each student is either a BS student, or an MS student or a PhD student. Similarly each course has an associated level (BS, MS or PhD). When the students visit their advisors, the advisors suggest BS students not to take PhD level courses (since it might be too advanced), and they also suggest PhD students not to take BS level courses (since it would be less useful). If any of the courses taken by a student is not approved by the advisor, the student needs to randomly take 3 courses and come back for approval again.

Part One – Multi-Threading and Reflection (80 points)

Fill in the parts marked with ellipses ("…") in the files ***Barrier.java***, ***Instructor.java*** and ***Student.java*** which are provided among the template code files. The other files in the bundle are fully implemented and you should ***not*** modify them at all. During your implementation, also observe the following rules:

- Do not add any extra classes.
- Do not add any extra fields or methods to any class.
- Do not change the modifiers of any fields or methods of any class. If you need to reach a non-public field or method, you may do so by using reflection.

- Do not use high level multi-threading constructs, just use threading primitives (***wait***, ***notify***, ***notifyAll***, ***synchronized***, etc.).
- Do not use ***toString*** method of any entity (neither explicitly, nor implicitly) in order to figure out the value of the ***level*** field or any other fields. If you need to access field values, you may do so by using reflection. The ***toString*** methods are meant to be used for just printing out informative representations of entities.

You are always encouraged to enhance the homework scenarios if you want. If you would like to add extra features to your design for bonus points (such as keeping statistics, providing GUIs, etc.) and if those extra features require you to violate the rules stated above, talk to the staff members beforehand to get approval for your enhancement ideas.

Part Two – Analysis of Time and Space Complexities (30 points)

Suppose that we define the following parameters:

i : number of instructors in the department
s : number of students in the department
c : number of courses in the department
m : number of computers (machines) in the department
p : number of physical processors (cores) that will run your student threads concurrently

Thoroughly, carefully and clearly analyze the ***time*** and ***space complexities*** of your design in terms of the above parameters, providing every detail. You may introduce other parameters, create plots, and collect statistics by running an instrumented implementation in order to support your discussions. Make sure you report all your work and rationale.

What to submit? (Use ***only ASCII characters*** when naming all of your files and folders)

1. Analysis report and any other documentation that you would like to add (in a directory named "***Docs***").
2. Your source code (in a single directory named "***Source***"). Just put the files you filled in (no need to include the fully implemented files, we will copy them over anyway). Do NOT use packages. Do NOT submit binary code or IDE created project files. Just submit your ".java" files. I should be able to compile and run your code simply by typing the following:

```
C:\...\Source>  javac  *.java
C:\...\Source>  java  Register  <possible documented parameters after the required parameters>
```

Zip the 2 items above together, give the name <ID>_<FullName>.zip to your zip file (tar also works, but I prefer Windows zip format if possible), and submit it through COW. For example:

e1234567_SelimTemizer.zip

There are a number of design decisions and opportunities for visual improvements and creative extensions that are deliberately left open-ended in this homework specification. There will be bonuses awarded for all types of extra effort. Late submissions will NOT be accepted, therefore, try to have at least a working baseline system by the deadline. Good luck.