CSCI 260 Assignment 5. Due 5:00pm November 23

In solving these problems, consult only inanimate references. You may work together on any technical problems you encounter with Eclipse in the lab. E-mail me if you have questions (allen@bowdoin.edu), and e-mail me your completed answers. This assignment serves also as Test #2.

Some of these questions require you to run Eclipse with the workspace `workspace260Reg`. You should copy this directory from the course web site `www.bowdoin.edu/~ allen/courses/cs260/` to your desktop before starting Eclipse. A summary of running Eclipse appears at the end of this document.

1. Suggest a different way to implement the method `Max(a, b)` in `lec1110.ppt` without changing its specifications.

2. Suggest how the method `Factorial` in `lec1110.ppt` can be implemented recursively without changing its specifications.

3. Consider the inner class `Node` defined in the file `Queue.java`. Define new methods for `Node` so that any client class (like `Queue`) would have no need to access the private instance variables `val`, `prior`, and `next`.

4. Write complete JML specifications for class `Node`; that is, give a class invariant and pre- and postconditions for its new methods and its constructor.

5. Consider the class `Queue` defined in the project `QueueTest`.

   (a) Alter its code so that it makes no reference to any private instance variables of class `Node`.

   (b) Test these changes using the simple driver program `QueueTest.java`, along with the commands `javac` and `java` for compiling and running.

   (c) Alternatively, set up the `Queue` and `QueueTest` classes as an Eclipse/Java project and run your tests inside that environment.

6. Alter the JML specifications for the class `Queue` so that they become consistent with the changes you made in the previous two questions. Test your JML changes in Eclipse to be sure that they are correct.

7. Suggest a simpler, more efficient implementation for the `Queue` method `size`. Would that method's specifications change when you make this change to the code? Why or why not?

8. Consider the ten classes that come out of the use cases in our StressFreev1.3 design. These classes were generated by Poseidon UML and are now located in the Eclipse project `StressFreeJava`. Recalling the use case requirements for *StressFree*, write a JML class invariant for the class `Course` that includes all the constraints that apply to every course during registration.

9. Consider the use case `enrollInCourses` for a single student, along with its included use cases addACourse and addALab. Write JML pre- and postconditions that would give a complete specification for this method that will become part of the Student class.

10. Consider the class TimeSlot in the *StressFreeJava* project. This class now has two new methods added to it, `conflictsWith`, and `timeOverlap`.

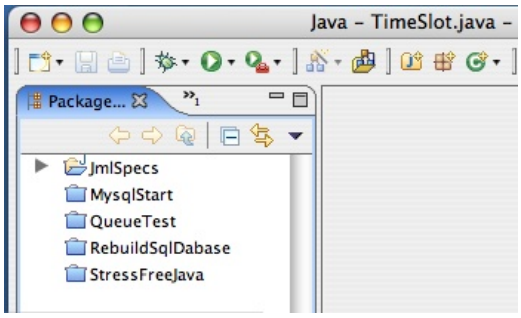
    (a) Consider the `TimeSlot t = F 08:30-11:25`. Find another valid TimeSlot x, different from t, for which `t.conflictsWith(x)` is false and `t.timeOverlap(x)` is true. To see a complete list of all the valid TimeSlots, run the project `MysqlStart` in Eclipse. Find another x for which both `t.conflictsWith(x)` and `t.timeOverlap(x)` are true. Is there a TimeSlot x for which `t.conflictsWith(x)` is true and `t.timeOverlap(x)` is false? Explain.

    (b) The method `conflictsWith()` contains a brief English-language description of its goal. Formalize this description by writing a complete JML `ensures` clause for it.

    (c) Test your JML specifications in part (b) by running "Check JML" in the JML menu.

Running Eclipse in the Lab

Look for the *Applications* folder `eclipse3.0`, and start the application `Eclipse`. Use the workspace `workspace260Reg` that should be on your desktop. If Eclipse opens properly, you should see the following projects in the left-hand window:



To run a project, you first need to open it (select the project in this list and then select Open in the Project menu). Then you need to select its run configuration from the Run menu. For example, to run `QueueTest`, first open it, then select Run in the Run menu, and then select QueueTest in the left-hand menu. (Notice that the input arguments '5 1 2 3 4 5' are provided as program Arguments for input to the run. Finally, the *Run* button will launch your run, and its output will appear in the Console window. Subsequent runs of the same project will use the same configuration unless you change it.