Using the SpecChecker

For most assignments we will provide you a tool called a SpecChecker. The main purpose of the SpecChecker is to ensure that your code conforms to the assignment specification – things like the names of classes and their packages, public method names, the types of the arguments, and so on. For each assignment, part or all of your grade will be based on the results of automated tests, and it is impossible to run the automated tests unless your code exactly conforms to the assignment specification. If the tests don't run, you'll get no points!

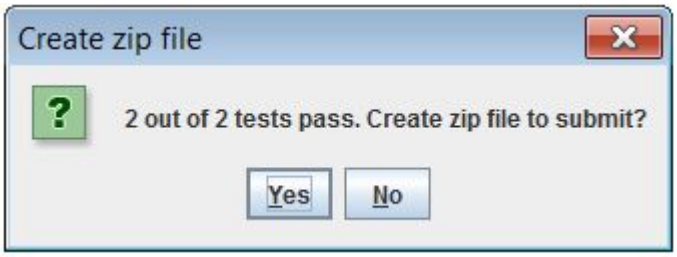
Note that in general there are two kinds of homework: regular *assignments*, and *miniassignments*. For the miniassignments, the SpecChecker will additionally run some simple functional tests of your code and will create a zip file for you to submit. For miniassignments, 100% of the points come from these automated tests. (For the regular assignments, 30% to 40% of the points come from automated tests and the remainder from reading and evaluating your code.)

Before you begin: make sure your code is in a new project that doesn't already contain another specchecker from lab or from any other assignment. If necessary, create a new, empty project, create the necessary packages and classes, and copy/paste your code from the existing project.

1. Download the SpecChecker for the assignment or miniassignment. It will normally be named something like **`speccheck_hw1.jar`**. (The ".jar" extension indicates that it is a "Java ARchive", that is, it contains Java classes.)
2. Import the jar file into your Eclipse project:
    a. Right-click on the project name. (Be sure you choose the *project*, not some other directory or file inside the project. The projects are the top-level items in the Eclipse Project Explorer.)
    b. Choose Import -> General -> File System.
    c. Browse to the directory where you downloaded the jar file.
    d. Check the box next to the jar file name and click Finish.
    e. The jar file should be visible in the project now. Right-click on it.
    f. Choose Build Path -> Add to Build Path. (The jar file should now appear under "Referenced Libraries".)
3. Add the JUnit 4 library to your project:
    a. Right-click on the project.
    b. Choose Build Path -> Configure Build Path ->Libraries Tab->Add Library...
    c. In the Add Library dialog, highlight JUnit and click Next.
    d. In the next dialog, select JUnit 4 from the dropdown menu, and click Finish.
    e. Click OK to close the project Properties dialog.
4. Run the jar file to execute the tests:

a. In the Package Explorer, right-click on the jar file under "Referenced Libraries"
b. Choose Run As -> Java Application
c. Results appear in the Console window. If there are errors the SpecChecker will attempt to tell you what's wrong.
d. Fix your code and run the tests again as often as you wish.

5. For *miniassignments* (and for the lab exercise), after running the tests you will also see a dialog giving you the option to create a zip file to submit, like the following:



a. If you're not ready to submit a zip file, click No. If you are satisfied with the results, click Yes, and you'll see a typical file chooser asking you to select a *directory* in which to store the zip file.
b. The zip file will automatically be named something like SUBMIT_THIS_lab1.zip.
c. You can run the tests as often as you wish before submitting the zip file.
d. Submission of the zip file is through Blackboard. For reference, see Part 1 of the assignment submission HOWTO posted at the top of the Assignments page on Blackboard.

Acknowledgment: The SpecChecker was invented and provided for our use by former faculty member Chris Johnson (currently in the Computer Science department at University of Wisconsin, Eau Claire).