## CST316 Software Enterprise: Construction and Transition Spring 2016 Online

## Practice: Static Analysis due 2/15/16 11:59:00pm

**Objectives:**

1. Understand how a static analysis tool helps discover potential defects and other quality issues.

Please complete individually. Submit by 11:59:00pm to Blackboard (submission instructions are on the next page).

You will use the same given code from the Unit Testing lab. Yes that code has some defects but that is OK.

**Setup:** For this lab you will use the same unit test “given” source code you used earlier in the course. Please do not use your solution, use the given (even though it is buggy) so you get results consistent with our grading guide.

**Task 1: Style checking using Checkstyle**

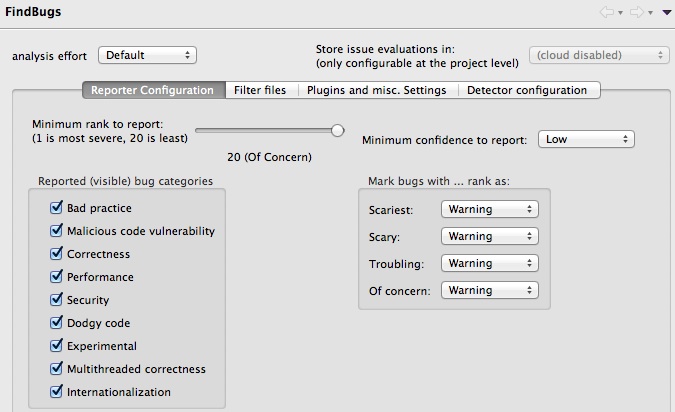
1. Install the Checkstyle plugin into Eclipse from the Eclipse marketplace: <http://marketplace.eclipse.org/content/checkstyle-plug>
2. Make sure you have downloaded the Checkstyle configuration XML file we provided for you on Blackboard. Know where you saved it on your filesystem, you will need it in step 4.
3. After restarting Eclipse, right-click on the project and click “Activate Checkstyle”. Then go to the Eclipse Preferences (Mac: Eclipse 🡪 Preferences; Windows: Window 🡪 Preferences) and find the Checkstyle configuration.
4. Click the “New” button. In the dialog, enter the name “CST316 SA lab”. Then click Import and navigate to XML settings file provided for you on Blackboard.
5. You need to rebuild your Eclipse project. Uncheck Project 🡪 Build Automatically, then do Project 🡪 Build Project.
6. In the bottom pane bring up the Checkstyle Violations view (you’ll find it under Window 🡪 Show View 🡪 Other). You should see a big pie chart. Please take a capture of the pie chart (you can right-click on the pie chart and do “Save As…” and copy it into your Word document under heading “Task 1, step 6”. If you are having trouble getting the Checkstyle output, try right-clicking on your ‘src’ folder, and doing Checkstyle 🡪 Check code with Checkstyle.
7. Look for these 3 entries (on mine they showed in the upper right part of the pie) out of all of the entries:
   1. ‘X’ construct must use ‘{}’s
   2. ‘X’ at column X should be alone on a line.
   3. Line is longer than X characters (found X).

In your Word document, record the defects found for each of these categories. You simply click on the proper wedge, and it will change the view to a listing of the defects. You can either transcribe the report from Eclipse or do a mini screen capture of that part of the view.

1. Now rectify all of the style violations from the 3 categories in the code. On the line above the start of your fix, add the comment “CST316 TASK 1 CHECKSTYLE FIX” in all CAPS just like that.
2. Rerun Checkstyle. Go view your new Checkstyle Violations Pie Chart, and show that the style violations are gone!

**Task 2: Static Analysis using FindBugs**

1. Install the FindBugs 3.0.1 plugin in your Eclipse environment. Instructions are here: http://findbugs.sourceforge.net/manual/eclipse.html
2. I also recommend doing the standalone FindBugs install. http://findbugs.sourceforge.net/manual/installing.html
3. Using the Menu (down caret) in the FindBugs Explorer, go to Configure Filters. You can also find this under Eclipse Preferences 🡪 Java 🡪 Findbugs. Set the Report Configuration as you see in the image below. Click OK.



1. Right-click on the project in the Package Explorer view, and from the context menu select FindBugs 🡪 Find Bugs. You may get a warning dialog, just click OK.
2. FindBugs should return with a few bugs in the project. Bring up the context menu again and choose “Save XML”. Save the exported XML as <asurite>.findbugs2-5.xml.
3. Go through the code and correct all the bugs. On the line above the start of your fix, add the comment “CST316 TASK 2 FINDBUGS FIX” in all CAPS just like that.
4. Rerun Findbugs (make sure it rebuilds, I have found it to be a bit wonky). Save an export of your XML again under the file name <asurite>.findbugs.2-7.xml

Note: Findbugs comes with its own Perspective that you can switch to (Window 🡪 Open Perspective) or you can open individual Views (Window 🡪 Show View 🡪 Other).

**SUBMISSION INSTRUCTIONS**

*Your submission should be a single zipfile that should be named <asurite>.StaticAnalysisLab.zip and include:*

1. A Word, OpenOffice, or Text file named <asurite>.StaticAnalysisLab.[docx|odt|txt]. Please do not give us a different format like a .pages file. In the document should be:
   1. A pie chart for Task 1, step 6.
   2. A listing (or screen captures) from Task 1, step 7.
   3. A pie chart for Task 1, step 9.
2. 2 XML files, <asurite>.findbugs2-5.xml and <asurite>.findbugs.2-7.xml
3. The entire src directory of your Eclipse project. If you forgot where this is on your computer, in Eclipse right-click on the src folder and display the properties. It will show you where on your computer the folder resides. Make sure the code has the commented fixes as indicated in Tasks 1 and 2.
4. Submit to Blackboard by Monday 2/15 11:59:00pm to avoid late penalty, late deadline Thursday 2/18 11:59:00pm
5. Incorporate static analysis tools into your project in Sprint 3.

*Please follow these instructions. It has been very difficult for the graders and I when you do not follow directions, and we will start deducting points for not following directions. There will be no allowed submissions after Thursday 2/18 at 11:59:00pm. Even with the late policy I am getting some emails. If you have a difficulty completing the lab, I suggest 2 remedies: 1) submit an incomplete version before the deadline. Remember Blackboard will let you submit a newer version over top of the previous version, so there is no reason to not submit anything! and 2) if there is some dire situation that you think excuses you from even the late policy, I should know about it* ***before*** *the deadline not after.*

**Going forward: put Findbugs and Checkstyle in Ant and use in Continuous Integration**

Both Findbugs and Checkstyle are runnable from Ant, I will provide a build.xml example soon. The instructions for doing this yourself are at http://findbugs.sourceforge.net/manual/anttask.html and <http://checkstyle.sourceforge.net/anttask.html> if you do not want to wait. I highly recommend you create your own local, standalone install of these 2 tools (outside of Eclipse). These require you to extend your ant installation to incorporate these new tasks. I did this for FindBugs by copying its findbugs-ant.jar to $ANT\_HOME/lib, and doing likewise for Checkstyle. As they are in ant, you can configure a CI job to run those Ant tasks on your projects.

As customary, your team should now consider how, when, and where applying static analysis makes sense in your project, and modify your quality policy on your team’s Wiki to indicate as such. Consider how each of these tools may be customized as well. You now have a number of quality techniques at your disposal by the completion of this lab – black-box and white-box unit testing, formal and informal code reviews, and now static analysis. Your quality policy should reflect the interplay of these techniques, not just be a checklist of “do it because Dr. Gary says”.