Computer Science 507
Software Engineering
The College of Saint Rose
Spring 2015

Lab 8: Static Analysis

Due: 6:00 PM, Monday, April 13, 2015

This week's lab exercise will introduce you to a tool that analyzes Java code for potential errors. Of course, it is impossible for a program to find all possible errors in another program. However, there are many common errors that can be found.

You may work alone or in groups of up to size 4 on this lab. Only one submission per group is needed.

Guest Speaker Reactions

Note: this portion must be done individually and is graded separately.

Participate in the discussion of this week's speakers, Tim Kelliher and Wes Turner, on the speaker blog. To earn full credit, you should make at least 3 comments on this week's post or replies to comments (at least 1 should be a reply, and at least 1 should be posted within the day following the talk). Comments may address any aspect of the speaker's presentation, including how it relates to topics from other parts of the class. You are welcome to start this process during the talk, if you wish. (5 points)

Lecture Assignment Questions

Note: these remain due one week after the assignment is made.

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

? LA Question 1:

Sommerville Exercise 17.1, p. 476 (3 points)

? LA Question 2:

Sommerville Exercise 17.2, p. 476 (4 points)

? LA Question 3:

Sommerville Exercise 17.4, p. 476 (3 points)

Static Analysis

For a quick introduction to static analysis and to the tool we will be using this week, read through the slides of a talk by William Pugh at the 2009 JavaOne conference. Don't worry too much about technical details of all examples in the talk – just go for the big picture ideas.

FindBugs Installation and Setup

The tool we will use this week is called FindBugs from the University of Maryland.

FindBugs finds bugs in Java programs, and is itself a Java program. That makes it very easy to install and use either standalone or as an Eclipse plugin. It is up to you how you decide to run FindBugs.

Follow the instructions in the FindBugs Manual to download, install, and run FindBugs on a computer of your choice.

A Simple Example

Copy the Java files from the directory `/home/cs433/examples/javageneric` on `mogul.strose.edu` to the computer where you installed FindBugs. Set up a project, add the directory containing the "javageneric" example code to the project and run an analysis.

Question 1:
What errors are reported by FindBugs? (2 points)

Question 2:
Are these errors legitimate errors? If not, why not? If so, how could they be fixed? (3 points)

A Larger Example

Next, we will analyze a larger Java example. If you have a Java program that you'd like to use (minimum 10 non-trival Java classes), please feel free to do so. Otherwise, you can use the "Structure package", the source code of which is available from here. To analyze this one, you'll add the "structure5" directory that is produced when you extract the gzip'ed tar file.

Question 3:
Use FindBugs to analyze your larger example (either structure or another package/program of your choosing). Summarize the results, and choose 3 of the reported problems and describe it in more detail. If it is a legitimate error, suggest a fix. If it is not, explain why it is not. (10 points)

Interaction with JUnit

Question 4:
Use FindBugs to analyze your JUnit tests from the previous lab. For each problem reported by FindBugs, describe it briefly. If it is a legitimate error, suggest a fix. If it is not, explain why it is not. (5 points)

Submitting

Before 6:00 PM, Monday, April 13, 2015, submit your lab for grading. Package up all required files into an appropriate archive format (`.tar.gz`, `.zip`, and `.7z` are acceptable) and upload a copy of the using Submission Box at `http://sb.teresco.org` under assignment "Lab8".

Grading

This assignment is worth 30 points, which are distributed as follows:

Feature	Value	Score
LA 1 (17.1)	3	
LA 1 (17.2)	4	
LA 1 (17.4)	3	
Question 1: "javageneric" FindBugs errors	2	
Question 2: "javageneric" FindBugs error analysis	3	
Question 3: FindBugs on a larger library	10	
Question 4: FindBuge with JUnit	5	
Total	30	