EECE 334: Programming Language Design and Implementation

Assignment 2

Background

Code coverage analysis is a quality assurance task that aims at identifying the program elements that get executed during testing. This allows developers to identify untested sections and perform further testing accordingly. Different types of program elements can be used for this purpose. In this assignment, we consider the following: methods, method call pairs, basic blocks, and basic block edges.

Description

You are provided with a skeleton to perform the 4 types of coverage, a class to be tested, and 4 JUnit tests to validate your code. **You are required to complete the missing parts of *InstrumenterClassVisitor.java* and *Profiler.java (look for “EECE 334: INSERT CODE HERE”)*.** In addition, you need to:

1. **Display the dynamic call graph as a directed graph whose edges are labeled with the counts of each call.** To do that, you might want to use the graph description language DOT (<https://www.graphviz.org/doc/info/lang.html>). In such case, you can run the DOT code online to generate the graph (e.g. <http://www.webgraphviz.com/>).
2. **Print the percentage of the methods that were covered**
3. **Print the percentage of the basic-blocks that were covered**
4. **Display the dynamic CFG. The edges should be labeled with the counts of edge executions.**
5. **Display the static CFG overlaid by the dynamic CFG (use a different color for the exercised nodes/edges).**

The details of the classes provided in the skeleton are presented below:

*Class1.java*: The class to be tested

*test/Class1Test.java*: The test class containing the 4 JUnit tests

*MethodCall.java*: A class representing method calls. Note that a method profile element is represented as a string having the following form: *className*|*methodName*|*methodSignature*

*BasicBlockEdge.java*: A class representing basic block edges. Note that a basic block profile element is represented as a string having the following form: *className*|*methodName*|*leaderIndex* where *leaderIndex* is the index of the leader instruction in the list of instructions of the corresponding method

*Instrumenter.java*: The class that initiates instrumentation

*CoverageHelper.java*: A helper class that is used by the profiler to keep track of the covered program elements

*BasicBlockGeneratorMethodVisitor.java*: This class is used to identify the basic blocks of each method.

Tool Tips

1. Use Eclipse IDE for Java Developers

<https://www.eclipse.org/downloads/packages/>

1. Make sure you add the following to your classpath:
   * JUnit
   * asm
   * asm-util
   * asm-tree