Assignment 2 — JUnit test generation and edge-pair coverage

Due Date — 1PM on 14 March 2013

1 Goal

Understand automated test generation, instrumentation, and coverage calculation.

2 Requirements

1. Extended assignment 1 to support invocations of class methods (INVOKESTATIC in bytecode).
2. Write a test generator that generates JUnit tests for all static class methods in the given class.
3. Write code to instrument every basic block and add a function call.
4. Write a coverage calculator that takes the test class and a directory containing JUnit tests. The coverage calculator should calculate edge pair coverage achieved using your generated CFG for the given tests.

3 Details

3.1 Support for method calls

Use a dummy node to provide a common exit point in your CFG. At every callsite (INVOKESTATIC), insert the CFG of the called function (do not worry about recursive functions). So if a function is called twice, its CFG is inserted twice.

3.2 Test suite generation

For all static class methods, generate JUnit tests. Assume that the parameters of these methods would be of one of the types listed in the following `Domain` class.

```
public class Domain {
    public static final int[] INT = new int[]{ -1, 0, 1 };
    public static final boolean[] BOOLEAN = new boolean[]{ false, true };
    public static final String[] OBJECT = new String[]{ "null", "new Object()" };
    public static final String[] STRING =
        new String[]{ null, "", "0", "Hello" };
}
```

The constant arrays in this class give you the domain of values to try for a particular type. If you have multiple arguments, you have to try all combinations of values. For example, if a method takes an `int` and a `bool`, then you should generate six JUnit tests with the following arguments to the method under test:
`(-1,false), (-1,true), (0,false), (0,true), (1,false), (1,true)`

3.3 Instrumentation

Use BECL (`http://commons.apache.org/bcel/manual.html`) to instrument your code. You will use the `InstructionList.append` and `InstructionFactory.createInvoke` methods to instrument the test code with additional calls to your own functions that associate `Node`s in your `CFG` with JUnit tests that exercise those nodes.

3.4 Coverage

For the last part of the assignment, calculate the edge-pair coverage for all JUnit tests in the directory passed. Use JUnit libraries for running the tests.

4 What to turn in

Source and class files for two Java programs. First program takes class C as input and generates a test suite T. Seconds program takes the class C and a test suite T' (which can be different from T) as input and outputs the edge-pair coverage of T' for class C.

This assignment is longer than the first one. Start early.
Best of luck!