Spring 2015 COM S 417 (SE 417) Assignment 2

Assigned: February 7, 2015
Due: February 19, 2015, 12:39pm

Part 1: Black Box, Follow-up

1. Download and install CodeCover, EclEmma, or another code coverage tool for Eclipse. Download `QueueFaulty.java` from Blackboard.
2. Run the coverage tool on QueueFaulty using your original JUnit test suite from Assignment 1 and report the coverage metrics (branch and statement) *per method* for this faulty version of the code. Retain a faulty version for Part 3.
3. Fix the faults in QueueFaulty. Now, re-run the test suite, again measuring the coverage. If not all the test cases pass, make the necessary modifications to the test suite.
4. In your report:
   (a) Report the coverage levels for the faulty and fixed versions of the code. This includes method coverage as well as statement coverage (all in the output of the tool).
   (b) Describe the faults. Comment how the faults you fixed may or may not be different from what you observed with Assignment 1.
   (c) Describe what changes, if any, you had to make to your test suite to make all test cases pass on the fixed QueueFaulty class. Justify those changes. Was the test invalid? Did you make an invalid assumption about how the code was supposed to work? If no changes were necessary, explain what you did to make sure the tests were *maintainable.*

Part 2: Graph-based Analysis

- Draw a CFG for the method, `toString()`
- Identify test requirements to achieve Edge-Pair coverage, but not Prime-Path coverage. Show which test cases from part 1 map to these test paths. Are any test requirements not covered?
- Identify the tests requirements to achieve Prime-Path coverage. Show which test cases from part 1 map to these test paths. Are any test requirements not covered?
- Generate new JUnit tests to traverse each identified test paths, or explain why its impossible.
- In your report:
  1. Include the CFG and test paths for both levels of coverage
  2. Clearly identify which new JUnit tests map to which test requirements, and note any that are missed

Part 3: Randoop

1. Download and install the Randoop random testing tool https://code.google.com/p/randoop/
2. With the faulty version of `QueueFaulty.java`, use Randoop to generate unit tests. You can limit the output of Randoop to 10,000 tests.
3. Run the coverage tool on the Randoop tests for each class.
4. In your report:
    (a) In your own words, describe how Randoop works. How does it know what the *correct* answer is.
    (b) Does Randoop identify any faults you missed? Which?
    (c) How does the coverage achieved by Randoop differ from the coverage achieved by your test suites? If Randoop failed to achieve 100% coverage, explain why. Is there unreachable code? Can you modify the Randoop test suite to achieve 100% coverage?
    (d) Are the Randoop test suites more or less maintainable than your test suites? Sound or unsound? More or less complete? More or less feasible? Explain.

Grade Calculation

This assignment is graded out of 150 points.

50pt Part 1

- **10pt** Reporting coverage levels before the changes were fixed
- **10pt** Reporting coverage levels after the changes were fixed
- **10pt** Fault fixes and descriptions and how they potentially differ from what you saw in Assignment 1
- **10pt** Describing changes to test suite
- **10pt** New test suite passes on fixed version of the code

50pt Part 2

- **20pt** Correct CFG
- **10pt** Edge-pair coverage test requirements and mappings to test cases
- **10pt** Prime-path coverage test requirements and mappings to test cases
- **10pt** New JUnit tests (or illustration why its unnecessary)

50pt Part 3

- **10pt** Explanation of Randoop
- **10pt** Reporting of coverage and faults by Randoop and explaining how it's different from your tests
- **30pt** Discussion on the maintainability, soundness, completeness, and feasibility of Randoop.

Submission Instructions

Submit the fixed Java code, your JUnit tests (the ones you wrote, not the ones from Randoop), any other files required to run your test suites, and your report (pdf format only) in a .zip folder using Blackboard.