Software Testing, Quality Assurance & Maintenance (ECE453/CS447/SE465): Assignment 4 v4

Patrick Lam

Due: March 17, 2010 (♣)

You may discuss the assignment with others, but I expect each of you to do the assignment independently. I will follow UW's Policy 71 if I discover any cases of plagiarism.

Question 1 (10 points)

Consider the file `Diff.java`, from the Java Application Generator (jag)[1]. I've made it available at

`http://patricklam.ca/stqam/files/Diff.java`

(a, 6 points) Why is this class hard to unit test[2]? Modify the class to make it easier to unit test. (b, 4 points) Download the rest of the code, examine it, and discuss the implications of your modifications[3].

Question 2 (20 points)

Prepare a suite of JUnit tests which ensure Correlated Active Clause Coverage on your modified version of the `getDiffLines()` method. (Explain how you know that this suite of tests meets CACC.)

[1] `http://jag.sourceforge.net`
[2] Does it, perhaps, have some dependencies?
[3] Which other classes are coupled to `Diff`?

Question 3 (40 points)

In this question, you will create unit tests, using mock objects (with a mock object library of your choice), for the `setWallAtEnd()` method of the `com.eteks.sweethome3d.model.Wall` class from SweetHome3D. (We'll talk about the private method, which the public method calls with second parameter `true`.) These tests must kill the following mutants:

```
private void setWallAtEnd(Wall wallAtEnd, boolean detachJoinedWallAtEnd) {
 if (wallAtEnd != this.wallAtEnd) { // m1: != becomes ==
  Wall oldWallAtEnd = this.wallAtEnd; // m2: RHS becomes null
  this.wallAtEnd = wallAtEnd; // m3, m4: suggest 2 mutations to this line
  clearPointsCache(); // m5: drop this call
  this.propertyChangeSupport.firePropertyChange(Property.WALL_AT_END.name(),
     oldWallAtEnd, wallAtEnd); // m6: drop this call; m7: change parameters

  if (detachJoinedWallAtEnd) {
    detachJoinedWall(oldWallAtEnd);
  }
 }
}
```

Write down the mutations you suggest for `m3`, `m4` and `m7`. Don't use trivial mutants.

(Note that our testing is sort of analogous to Test-Driven Development, except backwards.)

Here are some additional hints:

1. Be aware of the EasyMock factory method `notNull()` and class `Capture<T>`.
2. The only mock object I created was the mock PropertyChangeListener.
3. In a test class, it's OK to use reflection to read private fields.