Testing on the Toilet

Google

What Makes a Good Test?

Unit tests are important tools for verifying that our code is correct. But **writing good tests is about much more than just verifying correctness** — a good unit test should exhibit several other properties in order to be readable and maintainable.

One property of a good test is clarity. **Clarity means that a test should serve as readable documentation for humans, describing the code being tested in terms of its public APIs.** Tests shouldn't refer directly to implementation details. The names of a class's tests should say everything the class does, and the tests themselves should serve as examples for how to use the class.

Two more important properties are completeness and conciseness. **A test is complete when its body contains all of the information you need to understand it, and concise when it doesn't contain any other distracting information.** This test fails on both counts:

```
@Test public void shouldPerformAddition() {
  Calculator calculator = new Calculator(new RoundingStrategy(),
      "unused", ENABLE_COSIN_FEATURE, 0.01, calculusEngine, false);
  int result = calculator.doComputation(makeTestComputation());
  assertEquals(5, result); // Where did this number come from?
}
```

Lots of distracting information is being passed to the constructor, and the important parts are hidden off in a helper method. The test can be made more complete by clarifying the purpose of the helper method, and more concise by using another helper to hide the irrelevant details of constructing the calculator:

```
@Test public void shouldPerformAddition() {
  Calculator calculator = newCalculator();
  int result = calculator.doComputation(makeAdditionComputation(2, 3));
  assertEquals(5, result);
}
```

One final property of a good test is resilience. Once written, **a resilient test doesn't have to change unless the purpose or behavior of the class being tested changes.** Adding new behavior should only require adding new tests, not changing old ones. The original test above isn't resilient since you'll have to update it (and probably dozens of other tests!) whenever you add a new irrelevant constructor parameter. Moving these details into the helper method solved this problem.

More information, discussion, and archives:

http://googletesting.blogspot.com