Testing on the Toilet

Keep Cause and Effect Clear

Can you tell if this test is correct?

```
208: @Test public void testIncrement_existingKey() {
209:   assertEquals(9, tally.get("key1"));
210: }
```

It's impossible to know without seeing how the `tally` object is set up:

```
1:   private final Tally tally = new Tally();
2:   @Before public void setUp() {
3:      tally.increment("key1", 8);
4:      tally.increment("key2", 100);
5:      tally.increment("key1", 0);
6:      tally.increment("key1", 1);
7:   }
// 200 lines away
208: @Test public void testIncrement_existingKey() {
209:   assertEquals(9, tally.get("key1"));
210: }
```

The problem is that the modification of `key1`'s values *occurs 200+ lines away from the assertion.* Otherwise put, **the *cause* is hidden far away from the *effect***.

Instead, **write tests where the effects immediately follow the causes**. It's how we speak in natural language: "If you drive over the speed limit *(cause)*, you'll get a traffic ticket *(effect)*." Once we group the two chunks of code, we easily see what's going on:

```
1:   private final Tally tally = new Tally();
2:   @Test public void testIncrement_newKey() {
3:     tally.increment("key", 100);
5:     assertEquals(100, tally.get("key"));
6:   }
7:   @Test public void testIncrement_existingKey() {
8:     tally.increment("key", 8);
9:     tally.increment("key", 1);
10:    assertEquals(9, tally.get("key"));
11:  }
12:  @Test public void testIncrement_incrementByZeroDoesNothing() {
13:    tally.increment("key", 8);
14:    tally.increment("key", 0);
15:    assertEquals(8, tally.get("key"));
16:  }
```

This style may require a bit more code. Each test sets its own input and verifies its own expected output. **The payback is in more readable code and lower maintenance costs.**

More information, discussion, and archives:
testing.googleblog.com