Writing Good Test Cases

Good test cases let you catch problems in your code early and help you pinpoint exactly where things are going wrong which saves you lots of time and energy later.

This is a list of common mistakes programmers make and how to test for them. The list was inspired by *Code Complete 2nd Edition* written by Steve McConnell.

Let's say you have the problem and solution:

Write a method that adds two ints if they are $\geq 0$. If either int is $> 100$ or $< 0$, return -1;

```java
public int funkyMath(int a, int b) {
    if (a > 100 || b > 100) {
        return -1;
    }
    if (a < 0 || b < 0) {
        return -1;
    }
    return a + b;
}
```

**Test each requirement**

We'd write three tests, one for each of the requirements listed in the problem:

- `funkyMath(1, 1); // adds two ints if they are $\geq 0$
- `funkyMath(1, -1); // should be -1
- `funkyMath(1, 1000); // $< 0$

**Make sure each line of code is in at least one test**

In this example, the three tests above each of the `if`s and `return`s are tested, so each line is run at least once.

**Test for the maximum and minimum allowed values**

In this example, we'd want to test the numbers on the edge of the range which is $[0, 100]$. 
@Test
public void testFunkyMath1() {
    assertEquals("unexpected result", 101, funkyMath(1,100));
}

@Test
public void testFunkyMath1() {
    assertEquals("unexpected result", 101, funkyMath(1, 0));
}

(from here on in, we will ignore the JUnitTest syntax)

Test for non allowed values

These values are those right outside and far outside the range:

funkyMath(1,101);       // should be -1
funkyMath(1, -1);       // should be -1
funkyMath(1, 1000);     // should be -1

Test combinations of things that shouldn’t be allowed.

He’re well test for both numbers being outside the range:

funkyMath(2000, -200);  // both outside (positive + negative)
funkyMath(-2000, 200);  // both outside (positive + negative)
funkyMath(2000, 200);   // both outside (positive)
funkyMath(-200, -200); // both outside (negative)

Note that even when you create tests, you can’t be sure that your code is 100% correct but it will help you with 90% of the common problems.

Prepared 2015-09-30 - Joseph Lewis joseph@josephlewis.net MIT License