Mutation Test

A Way to Learn How Good Your Unit Test is

Joseph Yao
Who I am?

- Agile Coach at Odd-e
- Father, Husband...
- Software Craftsman
Mutant

Code Changes
Mutation Test

Mutant causes any test failed?

- **Yes**: Killed Mutant
- **No**: Survived Mutant
WHY???
WHY GOD WHY!!!
Expectation of Automated Test

Any Possible “Bad” Code Change can be Detected
int foo (int x, int y)
{
    int z = 0;
    if ((x>0) && (y>0)) {
        z = x;
    }
    return z;
}

Unit Tests
assertEquals(2, foo(2, 2))
assertEquals(0, foo(2, -1))
assertEquals(0, foo(-1, 2))

To
assertEquals(2, foo(2, 2))
assertEquals(0, foo(2, 0))
assertEquals(0, foo(-1, 2))
int foo (int x, int y) {
    int z = 0;
    if ((x>0) && (y>0)) {
        z = x;
    }
    sideEffect(z);
    return z;
}

Either missing a test to prove the removed code is needed

Or the removed code is truly Redundant
Three Rules of TDD

- You are not allowed to write any production code unless it is to make a failing unit test pass.

- You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.

- You are not allowed to write any more production code than is sufficient to pass the one failing unit test.
<table>
<thead>
<tr>
<th>Hand Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight Flush</td>
<td></td>
</tr>
<tr>
<td>Four of a Kind</td>
<td></td>
</tr>
<tr>
<td>Full House</td>
<td></td>
</tr>
<tr>
<td>Flush</td>
<td></td>
</tr>
<tr>
<td>Straight</td>
<td></td>
</tr>
<tr>
<td>Three of a Kind</td>
<td></td>
</tr>
<tr>
<td>Two Pair</td>
<td></td>
</tr>
<tr>
<td>One Pair</td>
<td></td>
</tr>
<tr>
<td>High Cards</td>
<td></td>
</tr>
</tbody>
</table>
Survived Mutant

```java
private List<Integer> getPairCardRanks(List<Integer> cardRanks) {
    List<Integer> result = new ArrayList<>();

    for (int index = 0; index < CARD_COUNT - 1; index++)
        if (isTwoNeighborCardRanksEquals(index, cardRanks))
            result.add(cardRanks.get(index));

    return result;
}
```
Problematic Test

```java
@Test
public void pair_compare_to_pair_by_pair_card_rank() {
    assertPokerHandsSmallerThan("2S 3H 5S 6C 6D", "2S 3H 5S 7C 7D");
    assertPokerHandsLargerThan("2S 3H 5S 8C 8D", "2S 3H 5S 7C 7D");
    assertPokerHandsLargerThan("2S 3H 8S 8C 9D", "2S 7H 7S 8C 9D");
}
```
Mutants

- Conditionals Boundary
- Negate Conditionals
- Math
- Increments
- Invert Negatives

- Inline Constant
- Return Values
- Void Method Calls
- Non Void Method Calls
- Constructor Calls

More detail at http://pitest.org/quickstart/mutators/
Equivalent Mutant

\[ \geq = \rightarrow \geq \]
Mutation Test Tool
protected Integer getThreeOfAKindCardRank(List<Integer> cardRanks) {
    for (int index = 0; index < CARD_COUNT - 2; index++)
        if (isThreeNeighborCardRanksEquals(index, cardRanks))
            return cardRanks.get(index);

    throw new IllegalStateException();
Q & A

Email: joseph.yao.ruozhou@gmail.com
Kata Video: http://tudou.com/home/yaoruozhou
Github: https://github.com/JosephYao