Introduction

In this lab, I am going to ensure that the tests designed in one of our past labs will achieve full mutation coverage, or add tests until it does. This will ensure that the test suite will fail if almost any small part of the behavior is changed in the source file.

Mutation scores:

All initially live mutants:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Line # | Code segment | Mutation description | Analysis | After |
| 44 | **if** (name.length() <= 0) | changed conditional boundary | LIVE | STUBBORN Will still throw exception, but with slightly different error message |
| 63 | **if** (age <= 0) | changed conditional boundary | LIVE | SQUASHED w/testSingleAgeZero() |
| 95 | **if** (name.length() < 0) | changed conditional boundary | LIVE | STUBBORN Will still throw exception, but with slightly different error message |
| 118 | **if** (spouseAge <= 0) | changed conditional boundary | LIVE | SQUASHED w/testSpouseAgeZero() |
| 156 | } **else** **if** ((age < 65) | changed conditional boundary | EQUIVALENT | EQUIVALENT |
| 157 | (**this**.spouseAge >= 65) | changed conditional boundary | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 159 | } **else** **if** ((age >= 65) | changed conditional boundary | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 160 | (**this**.spouseAge < 65) | changed conditional boundary | EQUIVALENT | EQUIVALENT |
| 260 | **if** (**this**.filingStatus == TaxCalculatorInterface.***MARRIED\_FILING\_JOINTLY***) { | removed conditional | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 263 | } **else** **if** (((age < 65) | changed conditional boundary | EQUIVALENT | EQUIVALENT |
| 263 | negated conditional | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 264 | (spouseAge >= 65)) | changed conditional boundary | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 265 | ((age >= 65) | changed conditional boundary | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 266 | (spouseAge < 65) | changed conditional boundary | EQUIVALENT | EQUIVALENT |
| 266 | (spouseAge < 65) | negated conditional | LIVE | SQUASHED w/testMarriedJointAgeBounds() |
| 323 | retVal = **this**.grossIncome - **this**.getStandardDeduction(); | replaced double sub with add | EQUIVALENT | EQUIVALENT |
| 325 | **if** (**this**.grossIncome - **this**.getStandardDeduction() > 0) { | replaced double sub with add | LIVE | SQUASHED w/testTaxableIncomeBounds() |
| 326 | changed conditional boundary | EQUIVALENT | EQUIVALENT |
| 328 | } **else** **if** (**this**.grossIncome - **this**.getStandardDeduction() < 0) { | replaced double sub with add | LIVE | SQUASHED w/testTaxableIncomeBounds() |
| 329 | changed conditional boundary | LIVE | SQUASHED w/testTaxableIncomeBounds() |
| 329 | negated conditional | EQUIVALENT | EQUIVALENT |
| 346 | **if** (grossIncome < 0) { | changed conditional boundary | LIVE | SQUASHED w/testGrossIncomeReset() |
| 349 | } **else** **if** (grossIncome == 0.0) | removed conditional | EQUIVALENT | EQUIVALENT |
| 402 | **if** (remainingTaxableSalary > taxTable[startingOffset + index]) { | changed conditional boundary | EQUIVALENT | EQUIVALENT |

* Equivalent Identification
  + We were able to determine if a mutant was equivalent or not through a few processes. The first pass would involve looking over the mutants, and finding any obviously equivalent mutants. Then, pick a cluster of mutants in one function and write an automated test to try to eliminate some right away. We would look at the remaining mutants in the cluster and consider them more closely for equivalency. For mutants in complicated if-else structures with multi-clause Boolean arithmetic, I would reduce the expression to pure logic and see if the certain mutant could be shown to be logically equivalent to the pre-mutated code. For other mutants, I would look to see if the program code could possibly be caused to take a different path than it was meant to with the mutant. These methods proved to be effective in finding equivalent mutants. We found 9 equivalent mutants out of 24 live mutants.
  + I noticed that the same pattern of mutants appeared twice, in the age and spouse age if-else structure; most specifically where a clause was constructed to try to find if one of the members was above 65. The clause could be simplified to (!AB)+(A!B), or (!A+!B)(A+B). This appeared in an if-else-if structure, after the clause (AB) had already been checked for. So it could be assumed that !(AB), or (!A + !B), was already true, allowing the secondary expression in the if-else-if structure to be simplified to (A+B). This was a handy little proof. To sum it all up in one expression: Assuming !(AB), (!AB)+(A!B) 🡪 (A+B).
* Test Construction
  + To construct a test, I would first consider what state would cause the certain mutant to exhibit different behavior, aka, introduce an infection. Then I would construct a test to create an object with that state, and run the object through the motions to call the function in which the target mutant is. In many cases, I could group multiple mutants together in the same function, meaning we ended up creating only 3 new methods.