**CPTS 422 - Deliverable 3**

**Elisha Aguilera – SID #011745359 – 2021-12-11**

**Coverage metric defined as statement coverage.**

**Halstead Checks**

|  |  |  |
| --- | --- | --- |
| **Tested Class Name** | **Coverage Percentage** | **Reason** |
| *HalsteadLengthCheck* | 70.8% | *FinishTree* not covered, due to *log* function |
| *HalsteadVolumeCheck* | 93.3% | *FinishTree* not covered, due to *log* function |
| *HalsteadDifficultyCheck* | 89.8% | *FinishTree* not covered, due to *log* function |
| *HalsteadVocabularyCheck* | 67.5% | *FinishTree* not covered, due to *log* function |
| *HalsteadEffortCheck* | 68.4% | *FinishTree* not covered, due to *log* function |
| *Sum Total* | 81% | *Above rationale, along with adjustments.* |

**Tokens**

|  |  |  |
| --- | --- | --- |
| **Tested Class Name** | **Coverage Percentage** | **Reason** |
| *Operands* | 94.8% | Class is never instanced. |
| *Operators* | 98.4% | Class is never instanced. |

**B-list Checks**

|  |  |  |
| --- | --- | --- |
| **Tested Class Name** | **Coverage Percentage** | **Reason** |
| *ExpressionCheck* | 0.0% | No idea; Junit refuses to run test covering this check, and no error codes are logged. |
| *LoopingStatementsCheck* | 0.0% | No idea; Junit refuses to run test covering this check, and no error codes are logged. I attempted the Maven Fix, but it still refuses to run. |
| *GetOperatorsCheck* | 61.5% | *FinishTree* not covered, due to *log* function |
| *GetOperandsCheck* | 63.6% | *FinishTree* not covered, due to *log* function |
| *GetCommentsCheck* | 67.0% | *FinishTree* not covered, due to *log* function |

**Extras Checks**

|  |  |  |
| --- | --- | --- |
| **Tested Class Name** | **Coverage Percentage** | **Reason** |
| *VariableDeclarationsCheck* | 60.4% | *FinishTree* not covered, due to *log* function |

**Pitclipse Results**

Halstead Batch 1:

Graphical user interface

Description automatically generated

Variable Declarations Batch 1:

Table

Description automatically generated

As stated above, classes that cannot be run by Junit cannot be run on Pitclipse. The ‘extras’ package only have ‘ExpressionCheck’ and ‘VariableDeclarations’, and for unknown reasons Junit will not run ‘ExpressionsCheck’. This is incredibly bizarre, as the black-box test engine will run them without issue.

Get-Operands-Check test.

Graphical user interface

Description automatically generated

Get-Operators-Check fails to run with Pitclipse, despite running a green suite with Junit.

After some minor modifications to the test after black-box testing,

Halstead Batch 2:

Table

Description automatically generated

Variable Declaration Batch 2:

Table

Description automatically generated with low confidence

Unfortunately, I believe that since I cannot run all test-cases at once, and the mutations are applied across all the different packages, the coverage is really low. If I could have resolved the strange errors that are preventing some Junit test from running, or perplexingly Pitclipse mutation test not running for test that have passed, there would likely be far more coverage and fewer surviving mutations.

**Summary**

All instances of *FinishTree* are ignored due to the complexities of the *AbstractCheck.log* function; the remainder of the classes tested are fully covered. Testing checks are particularly difficult since they operate within an Eclipse environment on source code, compared to other, less recursed classes. The test and checks likely contain errors not caught due to this complexity.

**Fault Models and Black-box Testing results (copied here from the README):**

1. Halstead-Difficulty
   1. Operator is for unknown reason ignored.  
      - Operators are accounted for, added to hashtable
   2. Operand is for unknown reason ignored  
      - Operands are accounted for, added to hashtable
   3. If unique operands total to 0, might have divide-by-zero error  
      - Caught error wherein operators, operands iterators were looping to maximum integer value  
      - Caught error wherein integer division was distorting results, required casting
2. Halstead-Volume
   1. Operator is ignored for unknown reasons, not added to hashtable
   2. Operand is ignored for unknown reasons, not added to hashtable
   3. Volume for unknown reason computed as either 0 or Not-A-Number.  
      - Caught error where value was not reset between calls, causing unexpected output
3. Halstead-Length
   1. Valid token is not counted for unknown reason  
      - Caught error wherein some operands were ignored despite being valid,
4. Halstead-Effort
   1. Effort not computed correctly  
      - Caught issue where the value was not computed correctly
5. Halstead-Vocabulary
   1. Sum of unique operators, operands not computed correctly  
      - Caught iterator over set error wherein would iterate near endlessly
6. Number-Expressions  
   - Truly bizarre error, will run check with black-box test engine, but not the Junit white-box coverage test
   1. Expression token is ignored for unknown reasons  
      - All expression tokens counted
   2. Count is not incremented properly  
      - Count is properly incremented
7. Get-Comments  
   - Unfortunately, could not figure out appropriate configuration and context for comments, test engine wouldn’t run this particular black-box
   1. Comment tokens are ignored for unknown reasons
   2. Block comment line numbers not computed correctly
8. Get-Operators
   1. Operator token is ignored for unknown reason
   2. Operator count is not computed correctly  
      - Operates computed correctly, parentheses included changed count
9. Get-Operands
   1. Operand token is ignored for unknown reason
   2. Operands count is not computed correctly  
      - Operands computed correctly
10. Looping-Statements  
    10.1 Looping statements count not computed correctly  
     - The SLIST token along with the ‘getChildCount’ was misapplied. Fixed with exclusion of some token-types
11. Variable-Declarations
    1. Variable definition token not counted for unknown reasons
    2. Variable definition count not computed correctly for unknown reasons  
       - Both issues resolved by test, evidently any ‘x=y’ is counted as a declaration, even if it’s already been ‘declared’ according to Checkstyle

**Class Testing Differences**

Were class testing to be applied on the project, each Check as a class would have a sequence of methods to run in a test set. For example, Halstead Volume has the standard ‘beginTree’, ‘visitToken’, and ‘finishTree’ functions, along with various other functions such as ‘getVolume’. The difference between this versus standard unit testing is that Classes (or Objects) have sequences of methods that may mutate the state. As such, we can generate a state-chart and a state-sequence-tree from that, and make assertions on what we expect the state to be after a series of method calls. In this case, if we feed some test code through the engine, we could expect the state value to be in the ‘finish-tree’ state whenever the file is finished parsing. If the class is in a different state for some reason, we have found a fault with the class, in that our expected state given input is misaligned from the actual state.