Russell Heppell

422 Deliverable 3 Report

**Statement and Branch coverage of white box (from deliverable 2):**

Checkstyle coverage tool output:

A picture containing graphical user interface

Description automatically generated

HalsteadEffortCheck

91.2% coverage

11/12 branches covered. A branch wasn’t covered in the visitToken function, specifically the nested if/else if block inside visitToken in HalsteadEffortCheck.java. The main first if statement checks to be sure that the token passed in is a token that we care about, so the if/else if branch that’s inside the larger if doesn’t have a final branch that looks at tokens we don’t care about, since the encompassing one does that first. This is why the false branch on the final else if isn’t tested. Note that both cases of the large if statement are tested, so the case where we receive a token that is irrelevant is still tested.

HalsteadDifficultyCheck

93.2% coverage

8/8 branches covered

HalsteadVolumeCheck

90.3% coverage

2/2 branches covered

CommentCheck

73.4% coverage

No branches

CommentLinesCheck

82.8% coverage

6/6 branches covered

ExpressionsCheck

66.7% coverage

No branches

HalsteadLengthCheck

93.1% coverage

No branches

HalsteadVocabularyCheck

93.7% coverage

2/2 branches covered

LoopingStatementCheck

78.5% coverage

2/2 branches covered

OperandsCheck

74.6% coverage

No branches

OperatorsCheck

91.9% coverage

No branches

\*\*\*NOTE\*\*\*

For all the checks, finishTree was not tested. Mocking the log function was extra credit and calling the finishTree function in a test resulted in errors from the log function and so testing it was omitted completely. This is the cause of the lack of 100% coverage in all of the checks.

**Mutation Testing with Pitclipse**

Text

Description automatically generated

**Fault Models**

1. Operator / Operand

* 1.1: ‘-=’ and ‘+=’ getting counted twice because of & , and and .
* 1.2: getting counted twice because of and .
* 1.3: ‘{}’ Getting counted as two
* 1.4: function calls should be counted twice because of and . The token is an operator token and the is an operand token.

1. Comments

* 2.1: Single line comment ‘//’ inside a block comment ‘/\* \*/’ getting counted
* 2.2: Single line comment ‘//’ in another single line comment getting counted
* 2.3: block comment ‘/\* \*/’ inside a single line comment getting counted
* 2.4: block comment ‘/\* \*/’ inside another block comment getting counted
  + I tried testing this case, but java would read the nested blocks ‘\*/’ as the end of the outer block comment. Test case omitted.

1. Expression

* 3.1: Incrementing/decrementing (++, --) should count as an expression.
* 3.2: Variable definitions should count as an expression.
* 3.3: Variable assignment should count as an expression.
* 3.4: String concatenation should count as an expression.

Results from black box test cases based on fault models above:

Graphical user interface, text, application, Teams

Description automatically generated

* All tests passed. No fault model analysis resulted in any failed tests.

NOTES:

* Black box test engine located at src/main/java/ProjectPackage/TestEngine.java
* Black box test case java samples located in BlackBoxTestCases
* Test file that runs all black box files through checks with the engine located at src/test/java/ProjectPackage/BlackBoxTest.java

**Class Testing Discussion**

Applying class testing to our project doesn’t seem to be too beneficial, because we don’t have a large, multi-component system, just independent check classes that don’t depend on each other. The state of our checks doesn’t change drastically, and the state of them don’t depend on other classes in our system, rather just themselves. Class data slicing seems like it could be beneficial to our project because it tells us if our data members(check counts, token arrays( are correctly representing the state of the class (the check), as well as tells us if the member functions are manipulating the representation of the object. For the checks, though, the entire class deals with nearly all the same members and methods, so it seems we would have only a few data slices to test, making me wonder again if class testing is beneficial for the project.