

### TEST CASES:

I've carefully designed test cases to cover all the functions I've used in each check. This includes starting with the tree, where each counter resets, and working through abstractCheck functions like getAcceptableTokens, getRequiredTokens, and beginTree etc.

For checking expected and actual outputs, I've set up assert equals from JUnit. This helps catch errors in any calculations. Additionally, I've made use of Mockito, trying out 'when' and 'mock' for added testing flexibility.

SAMPLE from del1, I used halsteadLengthCheck and HalsteadVocab:

Finished after 0.836 seconds

Runs: 10/10

Errors: 0

Failures: 0

#### HalsteadLengthCheckTest [Runner: JUnit 5] (0.780 s)

- testIsOperator (0.000 s)
- testDefaultTokens (0.000 s)
- testIsOperand (0.000 s)
- testOperatorTokenHandling (0.777 s)
- testAcceptableTokens (0.000 s)
- testBeginTree (0.001 s)
- testCountOperatorsAndOperands (0.001 s)
- testFinishTree (0.000 s)
- testRequiredTokens (0.000 s)
- testVisitToken (0.001 s)

Finished after 0.896 seconds

Runs: 9/9

✖ Errors: 0

✖ Failures: 0

- ✓ HalsteadVocabularyCheckTest [Runner: JUnit 5] (0.837 s)
  - ✓ testIsOperator (0.000 s)
  - ✓ testDefaultTokens (0.000 s)
  - ✓ testIsOperand (0.000 s)
  - ✓ testAcceptableTokens (0.000 s)
  - ✓ testBeginTree (0.808 s)
  - ✓ testCountOperatorsAndOperands (0.027 s)
  - ✓ testFinishTree (0.001 s)
  - ✓ testRequiredTokens (0.000 s)
  - ✓ testVisitToken (0.001 s)

When I run the entire package as JUNIT: I do have some failures due to testing on zero that I need to rework but the rest are successful.

Finished after 0.907 seconds

Runs: 99/99

✖ Errors: 0










✖ Failures: 3

- > ✓ LoopingStatementCountCheckTest [Runner: JUnit 5] (0.783 s)
- > ✓ HalsteadVocabularyCheckTest [Runner: JUnit 5] (0.010 s)
- > ✓ CastCountCheckTest [Runner: JUnit 5] (0.004 s)
- > ✓ NumberOfCommentCheckTest [Runner: JUnit 5] (0.007 s)
- > ✖ OperatorCountCheckTest [Runner: JUnit 5] (0.011 s)
- > ✓ OperandCountCheckTest [Runner: JUnit 5] (0.004 s)
- > ✓ HalsteadEffortCheckTest [Runner: JUnit 5] (0.002 s)
- > ✓ LinesOfCommentTest [Runner: JUnit 5] (0.006 s)
- > ✓ HalsteadLengthCheckTest [Runner: JUnit 5] (0.004 s)
- > ✖ HalsteadVolumeCheckTest [Runner: JUnit 5] (0.005 s)
- > ✓ VariableDeclarationCheckTest [Runner: JUnit 5] (0.004 s)
- > ✖ HalsteadDifficultyCheckTest [Runner: JUnit 5] (0.004 s)
- > ✓ ExpressionCountCheckTest [Runner: JUnit 5] (0.004 s)

## TEST RESULTS:
































Since I am using all assert statements I didn't individually write out and test every possible expression or token but I've made use of what I was able to do from white box testing and my tests were passing (the majority was passing) since I still have some things to work out specifically with the tokens.

Here is coverage as junit test sample:

CastCountCheckTest.java		100.0 %	134	0	134
CastCountCheckTest		100.0 %	134	0	134
setUp()		100.0 %	6	0	6
testAcceptableTokens()		100.0 %	16	0	16
testBeginTree()		100.0 %	15	0	15
testDefaultTokens()		100.0 %	16	0	16
testMultipleCasts()		100.0 %	39	0	39
testRequiredTokens()		100.0 %	16	0	16
testSingleCast()		100.0 %	23	0	23

## COVERAGE:

Overall the coverage of my project was 95% however the coverage of each individual check varies. This was due to some miscalculations within the halstead properties for the given checks. My plan is to go back and double check these calculations and also add some more white box tests.

samplepluggin		95.8 %	4,623	201	4,824
src		95.8 %	4,623	201	4,824
MyPackage		95.8 %	4,623	201	4,824
AntiHungarianCheck.java		0.0 %	0	86	86
HalsteadVolumeCheck.java		88.1 %	171	23	194
VariableDeclarationCheck.java		52.6 %	20	18	38
HalsteadLengthCheck.java		87.9 %	123	17	140
HalsteadVocabularyCheck.java		88.7 %	134	17	151
HalsteadDifficultyCheckTest.java		95.1 %	254	13	267
HalsteadVolumeCheckTest.java		97.0 %	226	7	233
OperatorCountCheckTest.java		96.5 %	195	7	202
OperandCountCheck.java		94.7 %	89	5	94
LinesOfComment.java		95.5 %	64	3	67
CastCountCheck.java		97.1 %	33	1	34
ExpressionCountCheck.java		99.4 %	175	1	176
HalsteadEffortCheck.java		98.6 %	73	1	74
NumberOfCommentCheck.java		97.7 %	43	1	44
OperatorCountCheck.java		99.8 %	402	1	403
CastCountCheckTest.java		100.0 %	134	0	134
ExpressionCountCheckTest.java		100.0 %	210	0	210
HalsteadDifficultyCheck.java		100.0 %	182	0	182
HalsteadEffortCheckTest.java		100.0 %	132	0	132
HalsteadLengthCheckTest.java		100.0 %	217	0	217
HalsteadTokens.java		100.0 %	449	0	449
HalsteadVocabularyCheckTest.java		100.0 %	216	0	216
LinesOfCommentTest.java		100.0 %	213	0	213
LoopingStatementCountCheck.java		100.0 %	44	0	44
LoopingStatementCountCheckTest.java		100.0 %	237	0	237
NumberOfCommentCheckTest.java		100.0 %	211	0	211
OperandCountCheckTest.java		100.0 %	211	0	211
VariableDeclarationCheckTest.java		100.0 %	165	0	165