Fault Models

1. BaseHalsteadCheck
- 1.1 Count paring operators as two, such as (), {}, [], etc.
- 1.2 Count operand in equation with unary operators wrong: a = -2 + 3.
- 1.3 Failure to ignore comments.
- 1.4 Failure to ignore identifier and function declaration.
- 1.5 Failure to count brackets, commas, and terminators as operators.
- 1.6 Improper counting of array variable.
- 1.7 One operand and operator.
- 1.8 All unique operands and operators.
- 1.9 Multiple same operands and operators.
- 1.10 Counting operand or operator within string object.
- 1.11 No operators or operands.

2. HalsteadDifficultyCheck
- 2.1 No operands or operators.
- 2.2 One or more operand and operator.

3. HalsteadEffortCheck
- 3.1 No operands or operators.
- 3.2 One or more operand and operator.

4. HalsteadLengthCheck
- 4.1 No operands or operators.
- 4.2 One or more operand and operator.

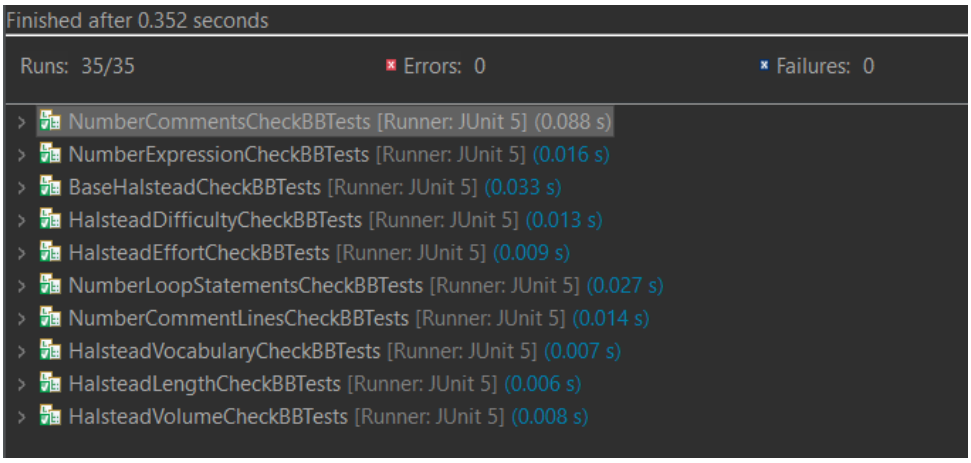
5. HalsteadVocabularyCheck
- 5.1 No operands or operators.
- 5.2 One or more operand and operator.

6. HalsteadVolumeCheck
- 6.1 No operands or operators.
- 6.2 One or more operand and operator.

7. NumberCommentLinesCheck
- 7.1 No comment lines.
- 7.2 One or more comment lines.
- 7.3 Comment lines and block comments.
- 7.4 Only block comments.
- 7.5 Counting '//' string.
- 7.6 Counting '/*' *or* '*/' string.

```
## 8. NumberCommentsCheck
- 8.1 No comments.
- 8.2 One or more comments.
- 8.3 Comments and block comments.
- 8.4 Only block comments.
- 8.5 Counting block comments as multiple comments.
- 8.6 Counting '//' string.
- 8.7 Counting '/*' or '*/' string.

## 9. NumberExpressionsCheck
- 9.1 No expressions.
- 9.2 One expression.
- 9.3 Multiple expressions.
- 9.4 Counting expression in string.
- 9.5 Counting expression in comment.

## 10. NumberLoopStatementsCheck
- 10.1 No loops.
- 10.2 One loop.
- 10.3 Multiple loops.
- 10.4 Counting loop in string.
- 10.5 Counting loop in comment.
- 10.6 Not counting nested loops seperately.

## 11. OperandAmountCheck
- covered by BaseHalsteadCheck tests.

## 12. OperatorAmountCheck
- covered by BaseHalsteadCheck tests.
```

Black Box Junit Test Results:



White Box Mutation Testing:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
13	95%	192/202	79%	112/141	85%	112/132

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
checks	12	95%	173/182	85%	83/98	93%	83/89
util	1	95%	19/20	67%	29/43	67%	29/43

Report generated by PIT 1.6.8

Black Box Mutation Testing:

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
13	81%	163/202	68%	96/141	79%	96/122

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
checks	12	82%	149/182	69%	68/98	86%	68/79
util	1	70%	14/20	65%	28/43	65%	28/43

Report generated by PIT 1.6.8

Results:

For white box testing, all relevant check classes were covered therefore the white box tests are sufficient. Black box results show less line coverage, though this is since some classes were not covered since it was not necessary too. HalsteadMath, OperandAmountCheck, and OperatorAmountCheck classes all were omitted from black box testing due to other classes covering their functionality. Operand and operator check classes simply returned operator and operand amount derived from BaseHalsteadCheck. The black box tests for the base class already cover potential faults on counting operand and operator amounts. HalsteadMath is a utility class containing the Halstead metric equations used in the relevant check class. The faults for this are covered by the appropriate check class, for instance HalsteadLength. The faults for HalsteadLength also covers the faults for the length equation in

HalsteadMath. Checking for no operators or operands handles checking for potential divide by 0 errors. Overall, since the BaseHalsteadCheck class covers the common faults for Halstead metric checks, the sub classes have less fault models to account for.

Class Testing:

The main differences that would occur with class testing would be the addition of integration testing, and more abstraction layers to unit testing. Furthermore, with class testing, more fault models are created, though the only new one relevant to this project would be dynamic binding and type errors. The way that Halstead check classes inherit from the base Halstead class does not change the value of attributes from the base class. The subclasses only read the attributes but do not modify them, therefore they are pure extensions.

Overall, the main classes that would be impacted by class testing are HalsteadMath and each Halstead metric check class. For HalsteadMath this would involve additional tests for type casting. Potentially, a value casted to int or double could be passed into a math function, in turn resulting in errors. For the Halstead checks, interaction between the base class would need to be handled since they depend on attributes from the base class.