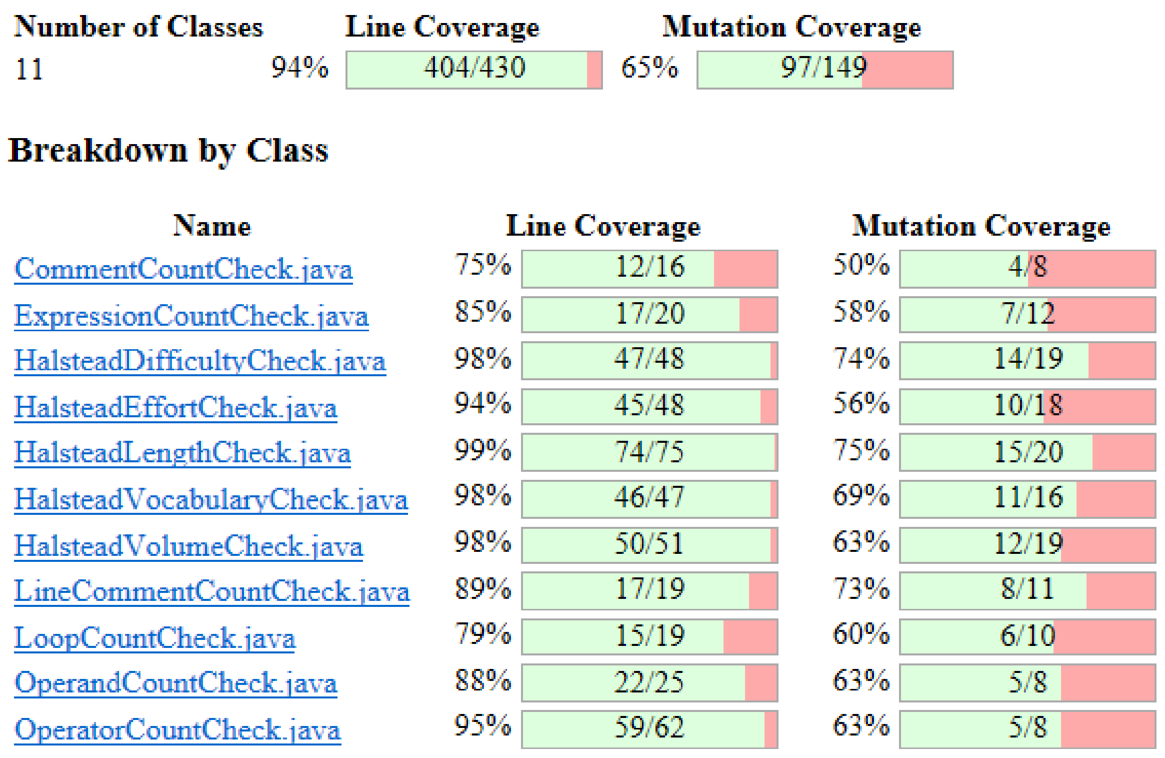
Fault Models

1. HalsteadLengthCheck
    1.1. Multiple unique operands and operators result in incorrect value
    1.2. No length can be calculated if there are no operators or operands
2. HalsteadVocabularyCheck
    2.1. Multiple operands and operators result in incorrect value
    2.2. Multiple unique operands and operators result in incorrect value
    2.3. No value can be calculated if there are no operators or operands
3. HalsteadVolumeCheck
    3.1. Multiple operands and operators result in incorrect value
    3.2. Multiple unique operands and operators result in incorrect value
    3.3. No operands/operators throws error because $\log_2(0)$ is negative infinity.
4. HalsteadEffortCheck
    4.1. Multiple operands and operators result in incorrect value
    4.2. Multiple unique operands and operators result in incorrect value
    4.3. No operands/operators throws error because $\log_2(0)$ is negative infinity.
5. HalsteadDifficultyCheck
    5.1. Multiple operands result in incorrect value
    5.2. Multiple unique operands and operators result in incorrect value
    5.3. No value can be calculated if there are no operators or operands
6. CommentCountCheck
    6.1. Block comments are counted as more than one
    6.2. Multiple backslashes are counted as more than one comment
    6.3. A single backslash is counted as a comment
    6.4. Nested line comments inside block comments are counted as multiple comments
7. LineCommentCountCheck
    7.1. Block comments do not register
    7.2. No comments throw null
    7.3. Inline comments do not register
    7.4. Multiple comments next to each other count as one
8. LoopCountCheck
    8.1. "Do while" counts as two loops
    8.2. Nested loops give incorrect count
    8.3. No loops return null
9. OperandCountCheck
    9.1. Operand count with unary operators incorrect
    9.2. Loops count operands multiple times
    9.3. No operands returns null
10. OperatorCountCheck
    10.1. Pairing operators such as ==, ++ and -- count as two operators
    10.2. Loops count operators multiple times
    10.3. No operators returns null
11. ExpressionCountCheck
    11.1. Count incorrect due to counting data type as expressions

11.2. Counts enumerators as expressions
11.3. Expressions in loops counted multiple times
11.4. No expressions returns null

Black Box PIT Mutation Testing

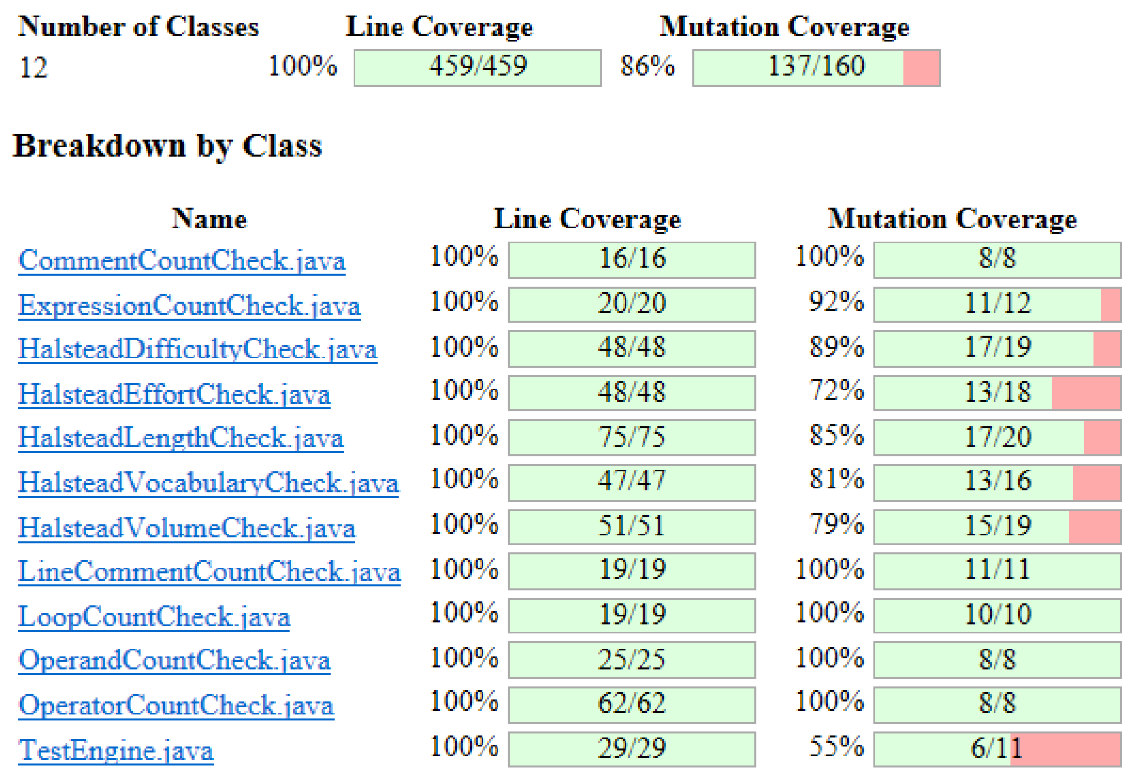


Number of Classes	Line Coverage		Mutation Coverage	
11	94%	404/430	65%	97/149

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
CommentCountCheck.java	75%	12/16	50%	4/8
ExpressionCountCheck.java	85%	17/20	58%	7/12
HalsteadDifficultyCheck.java	98%	47/48	74%	14/19
HalsteadEffortCheck.java	94%	45/48	56%	10/18
HalsteadLengthCheck.java	99%	74/75	75%	15/20
HalsteadVocabularyCheck.java	98%	46/47	69%	11/16
HalsteadVolumeCheck.java	98%	50/51	63%	12/19
LineCommentCountCheck.java	89%	17/19	73%	8/11
LoopCountCheck.java	79%	15/19	60%	6/10
OperandCountCheck.java	88%	22/25	63%	5/8
OperatorCountCheck.java	95%	59/62	63%	5/8

White Box PIT Mutation Testing



Number of Classes	Line Coverage		Mutation Coverage	
12	100%	459/459	86%	137/160

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
CommentCountCheck.java	100%	16/16	100%	8/8
ExpressionCountCheck.java	100%	20/20	92%	11/12
HalsteadDifficultyCheck.java	100%	48/48	89%	17/19
HalsteadEffortCheck.java	100%	48/48	72%	13/18
HalsteadLengthCheck.java	100%	75/75	85%	17/20
HalsteadVocabularyCheck.java	100%	47/47	81%	13/16
HalsteadVolumeCheck.java	100%	51/51	79%	15/19
LineCommentCountCheck.java	100%	19/19	100%	11/11
LoopCountCheck.java	100%	19/19	100%	10/10
OperandCountCheck.java	100%	25/25	100%	8/8
OperatorCountCheck.java	100%	62/62	100%	8/8
TestEngine.java	100%	29/29	55%	6/11

Class Based Testing

Class base testing accounts for inheritance and object instances and the state of private variables so it would likely require more test cases. Class base testing would make it easier to test certain functions like FinishTree() because it relies on functions from the DetailAST class. Class base testing could also uncover potential bugs that traditional testing could not. For example, in HalsteadLengthCheck I created a private Operand/Operator count variables that may contain bugs and could use additional testing.