Fault Models

Model that describes the likely Faults that might occur in the system. IN our case, I will list the fault models in the 11 checkstyle checks that I developed.

- ***Halstead Length***
  - Incorrect length be calculator due to wrong count of operators or operands.
  - No length be calculated if operands or operators are missing.
- ***Halstead Vocabulary***
  - Incorrect uniqueness count of operands or operators resulting in wrong vocabulary.
  - Vocabulary not being counted if operands or operators are not initialized.
- ***Halstead Volume***
  - If internal java log2() does not calculate the correct value.
  - If Halstead vocabulary isn't initialized and results in null value.
  - If Halstead length isn't initialized and results in null value.
- ***Halstead Difficulty***
  - Incorrect gathering of variables for halstead difficulty.
  - Operators or operands not present in the equation resulting in a null.
- ***Halstead Effort***
  - Volume or difficulty calculated wrongly.
  - Non initialized variables in the equation results in null value in the resulting Halstead Effort.
- ***Number of comments***
  - The block comments may be counted as more than one comment.
- ***Number of lines of comments***
  - The starting "'/*" or ending "*/" could not be accounted for.
  - The contents of the block comment may not be accounted for.
- ***Number of looping statements***
  - *The "do while" may just be counted as a while loop or give error.*
- ***Number of operators***
  - *Wrong tokens are accepted and counted as operators.*
- ***Number of operands***
  - If Halstead vocabulary isn't initialized and results in null value.
- ***Number of expressions***
  - *Counts data type as an expression.*

Test Results

Black Box PIT Mutation Testing

Number of Classes	Line Coverage		Mutation Coverage	
11	93%	369/396	66%	99/149

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
HalsteadDifficultyCheck.java	98%	51/52	70%	14/20
HalsteadEffortCheck.java	94%	50/53	63%	12/19
HalsteadLengthCheck.java	98%	47/48	65%	11/17
HalsteadVocabularyCheck.java	98%	50/51	71%	12/17
HalsteadVolumeCheck.java	98%	55/56	75%	15/20
LinesOfCommentCheck.java	95%	18/19	80%	8/10
NumberOfCommetsCheck.java	83%	10/12	57%	4/7
NumberOfExpressionsCheck.java	81%	17/21	58%	7/12
NumberOfLoopsCheck.java	68%	19/28	50%	5/10
NumberOfOperandsCheck.java	91%	21/23	63%	5/8
NumberOfOperatorCheck.java	94%	31/33	67%	6/9

Mutation Score

Mutation Score = 100 * D / (N -E)
- D = Dead mutants
- N = Number of mutants
- E = Number of equivalent mutants

Mutation Score = 100 * D / (N -E)
Mutation Score = 100*99 /(149-99) = 66%

Black Box and White Box PIT Mutation Testing

Package Summary

Checks

Number of Classes	Line Coverage		Mutation Coverage	
11	99%	392/396	79%	117/149

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
HalsteadDifficultyCheck.java	100%	52/52	80%	16/20
HalsteadEffortCheck.java	100%	53/53	79%	15/19
HalsteadLengthCheck.java	100%	48/48	76%	13/17
HalsteadVocabularyCheck.java	100%	51/51	76%	13/17
HalsteadVolumeCheck.java	100%	56/56	80%	16/20
LinesOfCommentCheck.java	100%	19/19	90%	9/10
NumberOfCommetsCheck.java	100%	12/12	86%	6/7
NumberOfExpressionsCheck.java	90%	19/21	75%	9/12
NumberOfLoopsCheck.java	100%	28/28	70%	7/10
NumberOfOperandsCheck.java	96%	22/23	75%	6/8
NumberOfOperatorCheck.java	97%	32/33	78%	7/9

Class Based Testing

Class base testing can account for *inheritance and object instances*, and I believe that would have helped me in testing the parent class. For example , in the finishtree() , we need to test the log() which works with the DetailAST but not it and without class testing this was made difficult. Integration testing among classes will solve this.

Also, Class testing *tests the state* of private variables and this would have benefited me when I was testing any Halstead check that I extended. For example in developing the Halstead Length check , I created a private instance of the operator and operand check and in the visitToken() method, I increment the operand or operator count based on the token given. Class testing can make this testing more useful, without class testing, I would have to mock it and can't test the flow of the object.