Dylan Gyori
Check test case report
12/8/2024

WHITEBOX TESTS:

CommentLineCountCheck Test Cases: all passed with 100% branch and line coverage.

- **isCommentNodesRequiredTest:** tests if comment nodes required method returns true.
- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array with one COMMENT_CONTENT token and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets comment line count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method correctly counts number of lines a multiline comment and single line comment take up.

NumCommentsCheck Test Cases: all passed with 100% branch and line coverage.

- **isCommentNodesRequiredTest:** tests if comment nodes required method returns true.
- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array with one COMMENT_CONTENT token and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets comment count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

ExpresssionCountCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array with one EXPR token and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

LoopCountCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of LITERAL_FOR, DO_WHILE, and LITERAL_WHILE tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

NumOperandsCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operand tokens and getRequiredTokens returns an empty array.

- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

NumOperatorsCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

HalsteadLengthCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator and operand tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count on token visit.

HalsteadVocabularyCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator and operand tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets count to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method.
- **visitTokenTest:** tests if visitToken method increments count only when unique tokens are visited.

HalsteadVolumeCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator and operand tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets halstead vocabulary and length to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method when vocabulary is equal to zero or one.
- **visitTokenTest:** tests if visitToken method increments vocabulary once but length multiple times on visit of duplicate tokens.

HalsteadDifficultyCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator and operand tokens and getRequiredTokens returns an empty array.

- **beginTreeTest:** tests if beginTree method sets unique operators, unique operands, and total operands to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method when no or one operand is counted.
- **visitTokenTest:** tests if visitToken method increments unique operators once when duplicate operators are visited, then it checks if unique operands are incremented once but total operands are incremented multiple times when duplicate operands are visited.

HalsteadEffortCheck Test Cases: all passed with 100% branch and line coverage.

- **getTokensTest:** tests if getDefaultTokens and getAcceptableTokens methods return an array of operator and operand tokens and getRequiredTokens returns an empty array.
- **beginTreeTest:** tests if beginTree method sets unique operators, unique operands, total operators, and total operands to 0.
- **finishTreeTest:** tests if frinishTreeTest passes correct message to AbstractCheck's log method when no or one operand is counted.
- **visitTokenTest:** tests if visitToken method increments unique operators once but total operators multiple times when duplicate operators are visited, then it checks if unique operands are incremented once but total operands are incremented multiple times when duplicate operands are visited.

MUTATION TESTS RESULT:

CommentLineCountCheck Test Cases: 8/9 mutants killed.

The surviving mutant was functionally equivalent.

NumCommentsCheck Test Cases:

ExpresssionCountCheck Test Cases: 6/6 mutants killed.

LoopCountCheck Test Cases:

NumOperandsCheck Test Cases:

NumOperatorsCheck Test Cases:

HalsteadLengthCheck Test Cases: 6/6 mutants killed

HalsteadVocabularyCheck Test Cases:

HalsteadVolumeCheck Test Cases:

HalsteadDifficultyCheck Test Cases: 12/15 mutants killed

HalsteadEffortCheck Test Cases: 16/23 mutants killed

BLACKBOX TESTS

Fault models (test case for fault model):

1. Halstead Length (HalsteadLengthBBTestCase.java)

- Counting brackets as two operators e.g. {}, [], and ()
- Not counting unary operators e.g. ++k, --k, -k, k++, k—
- Not counting different types of assignment operators e.g. a += b, a -=b, a /= b, a %=b
- Miscounting operands of unary operators
- Miscounting function calls as operands e.g. Max(3, 4)

2. Halstead Vocabulary (HalsteadVocabularyBBTestCase.java)

- Miscounting different variable names as one unique operand.
- Miscounting different function names as one unique operator.
- Counting brackets as two unique operators e.g. {}, [], ()
- Miscounting different loops as one unique operator e.g. for, while, do while

3. Halstead Volume (HalsteadVolumeBBTestCase.java)

- Same faults as Halstead Vocabulary
- Same faults as Halstead Length

4. Halstead Difficulty (HalsteadDifficultyBBTestCase.java)

- Same faults as Halstead Vocabulary

5. Halstead Effort (HalsteadEffortBBTestCase.java)

- Same faults as Halstead Difficulty
- Same faults as Halstead Volume

6. Number of comments (NumCommentsBBTestCase.java)

- Miscounting comments inside strings e.g. "//"
- Miscounting comments between code e.g. int test /* comment */ = 2
- Counting comments that contain // and /* as separate comments
- Not counting single line comments at end of file

7. Number of lines of comments (CommentLineCountBBTestCase.java)

- Miscounting multiline comments as a single line comment
- Miscounting multiple comments on the same line
- Miscounting single line comment at end of file
- Miscounting comments inside strings e.g. "//"
- Miscounting comments between code e.g. int test /* comment */ = 2

8. Number of looping statements (LoopCountBBTestCase.java)

- Counting conditions in a for loop as multiple loops
- Miscounting Do while loops

9. Number of operators (NumOperatorsBBTestCase.java)

- Counting brackets as two operators e.g. {}, [], and ()
- Not counting unary operators e.g. ++k, --k, -k, k++, k—
- Not counting different types of assignment operators e.g. a += b, a -=b, a /= b, a %=b

10. Number of operands (NumOperandsBBTestCase.java)

- Miscounting operands of unary operators
- Miscounting function calls as operands e.g. Max(3, 4)

11. Number of expressions (ExpressionCountBBTestCase.java)

- Miscounting expressions inside comments
- Miscounting expressions containing an assignment operator e.g. 5 + 2*(var = 2)
- Miscounting expression with multiple terms count as multiple expressions e.g. 5 + 2 + 3

CLASS TESTING

If Class Testing was performed instead of unit testing. Then the whitebox tests would include different combinations of the Check methods as individual tests. This will be done in such a way to get the check class into certain states. These tests would fail if the check class fails to go into a state it was supposed to. The tests could also fail if the check class goes into a state it's not supposed to.