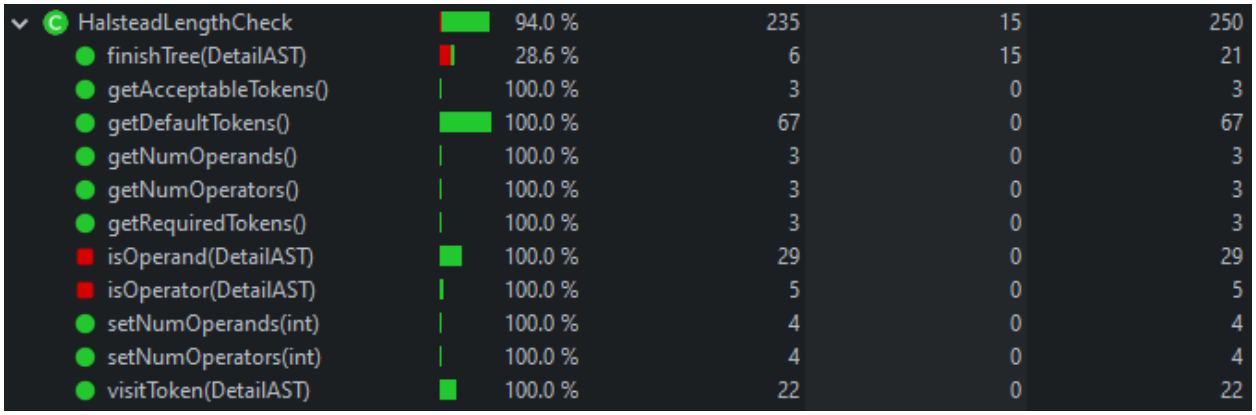
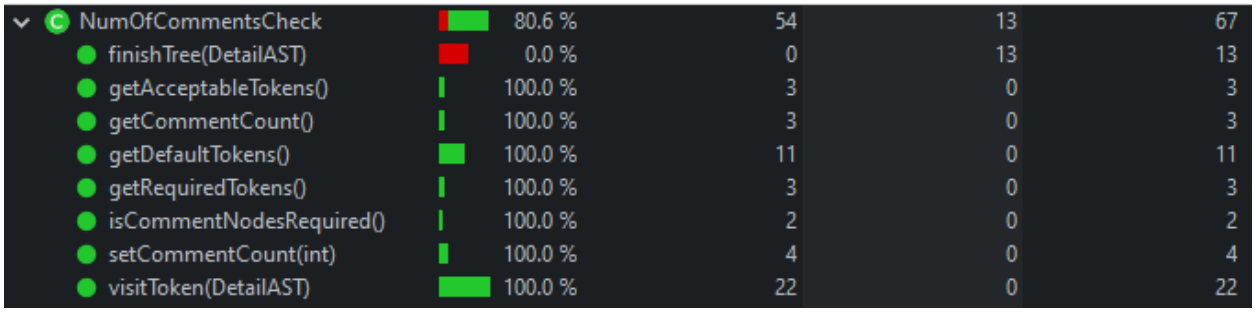
HalsteadLengthCheckTest



HalsteadLengthCheck	94.0 %	235	15	250
finishTree(DetailAST)	28.6 %	6	15	21
getAcceptableTokens()	100.0 %	3	0	3
getDefaultTokens()	100.0 %	67	0	67
getNumOperands()	100.0 %	3	0	3
getNumOperators()	100.0 %	3	0	3
getRequiredTokens()	100.0 %	3	0	3
isOperand(DetailAST)	100.0 %	29	0	29
isOperator(DetailAST)	100.0 %	5	0	5
setNumOperands(int)	100.0 %	4	0	4
setNumOperators(int)	100.0 %	4	0	4
visitToken(DetailAST)	100.0 %	22	0	22

Testing the finishTree method is challenging because it uses Checkstyle's logging system, which produces side effects rather than a testable return value. This makes verifying logs difficult, as tools like Mockito struggle with internal framework methods. Additionally, reaching finishTree requires simulating a full code tree, complicating direct output verification in a unit test.

NumOfCommentsCheck



NumOfCommentsCheck	80.6 %	54	13	67
finishTree(DetailAST)	0.0 %	0	13	13
getAcceptableTokens()	100.0 %	3	0	3
getCommentCount()	100.0 %	3	0	3
getDefaultTokens()	100.0 %	11	0	11
getRequiredTokens()	100.0 %	3	0	3
isCommentNodesRequired()	100.0 %	2	0	2
setCommentCount(int)	100.0 %	4	0	4
visitToken(DetailAST)	100.0 %	22	0	22

Testing finishTree is difficult due to its reliance on Checkstyle’s protected log method in the AbstractCheck superclass. Standard mocking methods, like using a spy or stubbing log, can disrupt class behavior and reduce coverage. Additionally, finishTree resets commentCount after logging, making isolated testing challenging. This issue often arises when testing framework-specific logging that’s hard to decouple.