Milestone 2

Arlo Jones

So far, I’ve implemented eleven checks, one for each item in categories A and B. Each check has six public functions to test: getDefaultTokens(), getAcceptableTokens(), getRequiredTokens(), beginTree(), visitToken(), and finishTree(). All other functions the checks use are private functions. Each constructor is also indirectly tested because each check is instantiated in each test that concerns it. Thus, there are sixty-six functions to unit test.

For ExpressionsCheck, I assert that all three get\_\_\_Tokens() functions return an integer array containing Token-Type.EXPR. I check beginTree(), visitToken(), and finishTree() each by calling the function and asserting that it ran once, as is expected with void functions. There is no branching logic in any of these functions, so by running each once, both 100% branch and 100% line coverage are achieved. There were no errors in calling any of the functions, so all tests passed.

This was the case for nearly every check. Halstead length, operators, operands, and vocabulary, the number and lines of comments, and the number of looping statements all had no branching logic and were 100% covered with identical tests that all passed. The only big differences being what Token-Types were checked in each array, shown below:

Length, Vocabulary, Difficulty, and effort shared the token set: ARRAY\_DECLARATOR, ASSIGN, BAND, BAND\_ASSIGN, BNOT, BOR, BOR\_ASSIGN, BSR, BSR\_ASSIGN, BXOR, BXOR\_ASSIGN, COLON, COMMA, CTOR\_CALL, DEC, DIV, DIV\_ASSIGN, DOT, DOUBLE\_COLON, EQUAL, GE, GT, INC, INDEX\_OP, INSTANCE\_INIT, LAMBDA, LAND, LCURLY, LE, LITERAL\_ASSERT, LITERAL\_BREAK, LITERAL\_CASE, LITERAL\_CATCH, LITERAL\_CONTINUE, LITERAL\_DEFAULT, LITERAL\_DO, LITERAL\_FINALLY, LITERAL\_FOR, LITERAL\_IF, LITERAL\_INSTANCEOF, LITERAL\_RETURN, LITERAL\_SWITCH, LITERAL\_SYNCHRONIZED, LITERAL\_THROW, LITERAL\_TRY, LITERAL\_WHILE, LNOT, LOR, LPAREN, LT, METHOD\_CALL, MINUS, MINUS\_ASSIGN, MOD, MOD\_ASSIGN, NOT\_EQUAL, PLUS, PLUS\_ASSIGN, POST\_DEC, POST\_INC, QUESTION, SEMI, SL, SL\_ASSIGN, SR, SR\_ASSIGN, STAR, STAR\_ASSIGN, SUPER\_CTOR\_CALL, TYPE\_EXTENSION\_AND, TYPECAST, UNARY\_MINUS, UNARY\_PLUS, CHAR\_LITERAL, EMPTY\_STAT, EXPR, FOR\_CONDITION, FOR\_INIT, FOR\_ITERATOR, IDENT, LABELED\_STAT, METHOD\_REF, NUM\_DOUBLE, NUM\_FLOAT, NUM\_INT, NUM\_LONG, RESOURCE, STRING\_LITERAL

Operators:

ARRAY\_DECLARATOR, ASSIGN, BAND, BAND\_ASSIGN, BNOT, BOR, BOR\_ASSIGN, BSR, BSR\_ASSIGN, BXOR, BXOR\_ASSIGN, COLON, COMMA, CTOR\_CALL, DEC, DIV, DIV\_ASSIGN, DOT, DOUBLE\_COLON, EQUAL, GE, GT, INC, INDEX\_OP, INSTANCE\_INIT, LAMBDA, LAND, LCURLY, LE, LITERAL\_ASSERT, LITERAL\_BREAK, LITERAL\_CASE, LITERAL\_CATCH, LITERAL\_CONTINUE, LITERAL\_DEFAULT, LITERAL\_DO, LITERAL\_FINALLY, LITERAL\_FOR, LITERAL\_IF, LITERAL\_INSTANCEOF, LITERAL\_RETURN, LITERAL\_SWITCH, LITERAL\_SYNCHRONIZED, LITERAL\_THROW, LITERAL\_TRY, LITERAL\_WHILE, LNOT, LOR, LPAREN, LT, METHOD\_CALL, MINUS, MINUS\_ASSIGN, MOD, MOD\_ASSIGN, NOT\_EQUAL, PLUS, PLUS\_ASSIGN, POST\_DEC, POST\_INC, QUESTION, SEMI, SL, SL\_ASSIGN, SR, SR\_ASSIGN, STAR, STAR\_ASSIGN, SUPER\_CTOR\_CALL, TYPE\_EXTENSION\_AND, TYPECAST, UNARY\_MINUS, UNARY\_PLUS

Operands:

CHAR\_LITERAL, EMPTY\_STAT, EXPR, FOR\_CONDITION, FOR\_INIT, FOR\_ITERATOR, IDENT, LABELED\_STAT, METHOD\_REF, NUM\_DOUBLE, NUM\_FLOAT, NUM\_INT, NUM\_LONG, RESOURCE, STRING\_LITERAL

Number of comments:

BLOCK\_COMMENT\_BEGIN, SINGLE\_LINE\_COMMENT

Lines of comments:

COMMENT\_CONTENT

Looping statements:

LITERAL\_DO, LITERAL\_FOR, LITERAL\_WHILE

Halstead Volume contained a while loop to calculate Log base 2 of a value in the finishTree() function. The tests used previously did not cover the inside of the while loop, but with slight modification, were able to get 100% branch and line coverage.

Likewise, Halstead Difficulty and Effort contained an if statement, that required values of a certain type to execute one path in their visitToken() function. Using when() then return(), the value was mocked and the branches covered.

In summary, every check passed all tests with 100% branch and line coverage.