1. Introduction

This document presents a comparative analysis of two lexical analyzers developed for the Bili programming language:

- **Manual Scanner** (ManualScanner.java): A hand-coded implementation
- **JFlex Scanner** (Scanner.flex → Lexer.java): An automatically generated implementation

Both scanners were tested against identical .bili test files, and their outputs were systematically compared using a difference checker to evaluate functional equivalence and performance characteristics. Link

2. Output Comparison Analysis

2.1 Testing Methodology

Identical test files were executed for both scanner implementations. The outputs were compared using a differential analysis tool to identify discrepancies.

2.2 Observations

The comparative analysis revealed the following:

- **Token types**: Identical across both implementations.
- **Lexemes**: Identical across both implementations.
- **Error messages**: Identical across both implementations.
- **Differences**: Limited exclusively to line and column numbers.

Example Output Comparison:

Manual Scanner	JFlex Scanner
<DECLARE, "declare", Line: 3, Col: 1>	<DECLARE, "declare", Line: 4, Col: 1>
<IDENTIFIER, "X", Line: 3, Col: 9>	<IDENTIFIER, "X", Line: 4, Col: 9>

The analysis confirmed that no logical token differences exist between the two implementations. Discrepancies are confined to positional metadata (line and column numbers).

3. Explanation of Differences

The difference is due to how each scanner handles preprocessing and whitespace:

3.1 Manual Scanner

- Performs a **preprocessing step**
- Removes extra whitespace
- Collapses multiple spaces
- Removes empty lines
- Skips comments before scanning
- Updates the line and column manually

Because preprocessing modifies the original source layout, line and column numbers shift.

3.2 JFlex Scanner

- Does **not preprocess input**
- Uses %line and %column directives
- Counts line/column directly from the original file
- Skips whitespace and comments using rules, but does not modify the input layout

Therefore, JFlex reports positions based on the original file structure.

3.3 Conclusion About Differences

The differences in line and column numbers are not logical errors. They occur because the manual scanner preprocesses and alters the layout, while the JFlex Scanner scans raw input without altering it. Hence, differences appear in positional metadata while token recognition behavior remains functionally equivalent.

4. Functional Comparison

Feature	Manual Scanner	JFlex Scanner
Implementation	Hand-coded DFA logic	Regex-based specification
Preprocessing	Yes	No

Token Recognition	Correct	Correct
Error Handling	Custom logic	Regex-based rules
Unicode Escape Handling	Manual validation	Pattern matching
Maintainability	Higher complexity	Easier to modify
Specification Clarity	Implicit in code	Declarative rules

5. Performance Analysis

Performance evaluation focuses on execution speed, memory efficiency, and scalability.

Both scanners operate in linear time complexity O(n), as each character of the input is processed once. However, the Manual Scanner performs two linear passes over the input (preprocessing followed by tokenization), which introduces additional overhead and temporary storage for the preprocessed input.

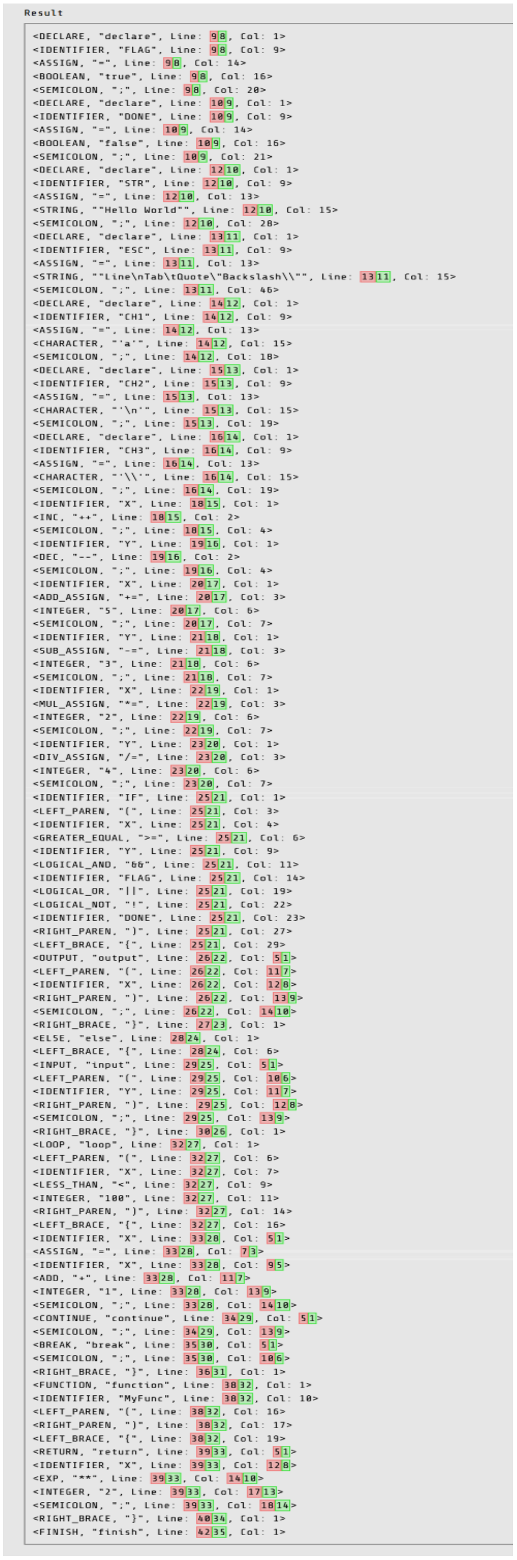
In contrast, the JFlex Scanner generates an optimized Deterministic Finite Automaton (DFA) at compile time. It performs tokenization in a single pass using efficient state transitions per character, eliminating the need for a separate preprocessing phase.

Due to its single-pass architecture and optimized DFA transitions, JFlex generally demonstrates better scalability and slightly improved performance for large input files. However, for small test cases, the performance difference between the two implementations remains negligible.

6. Conclusion

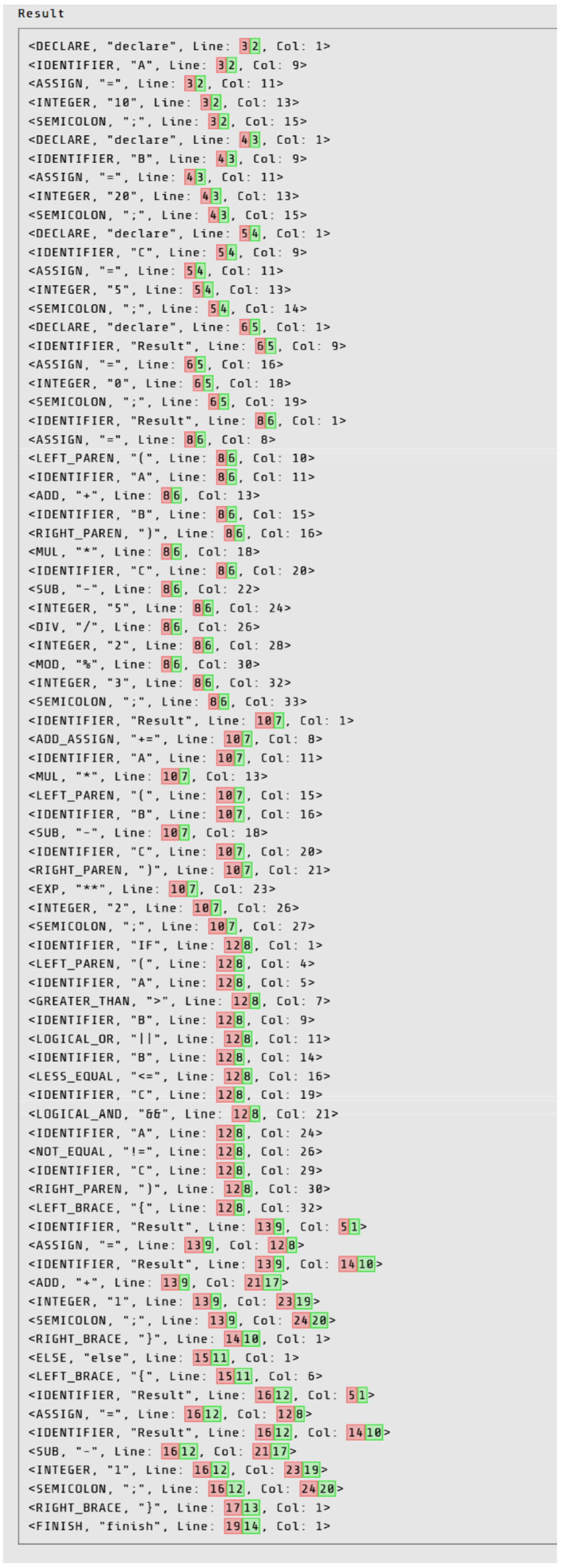
This comparative analysis establishes that both scanner implementations correctly recognize tokens according to the lexical specification of the Bili language. The outputs are functionally equivalent, with differences limited exclusively to positional metadata (line and column numbers) resulting from divergent preprocessing strategies.

Appendix



Result

```
<DECLARE, "declare", Line: 9 8, Col: 1>
<IDENTIFIER, "FLAG", Line: 9 8, Col: 9>
<ASSIGN, "=", Line: 9 8, Col: 14>
<BOOLEAN, "true", Line: 9 8, Col: 16>
<SEMICOLON, ";", Line: 9 8, Col: 20>
<DECLARE, "declare", Line: 10 9, Col: 1>
<IDENTIFIER, "DONE", Line: 10 9, Col: 9>
<ASSIGN, "=", Line: 10 9, Col: 14>
<BOOLEAN, "false", Line: 10 9, Col: 16>
<SEMICOLON, ";", Line: 10 9, Col: 21>
<DECLARE, "declare", Line: 12 10, Col: 1>
<IDENTIFIER, "STR", Line: 12 10, Col: 9>
<ASSIGN, "=", Line: 12 10, Col: 13>
<STRING, ""Hello World"", Line: 12 10, Col: 15>
<SEMICOLON, ";", Line: 12 10, Col: 28>
<DECLARE, "declare", Line: 13 11, Col: 1>
<IDENTIFIER, "ESC", Line: 13 11, Col: 9>
<ASSIGN, "=", Line: 13 11, Col: 13>
<STRING, ""Line\nTab\tQuote\"Backslash\\"", Line: 13 11, Col: 15>
<SEMICOLON, ";", Line: 13 11, Col: 46>
<DECLARE, "declare", Line: 14 12, Col: 1>
<IDENTIFIER, "CH1", Line: 14 12, Col: 9>
<ASSIGN, "=", Line: 14 12, Col: 13>
<CHARACTER, "'a'", Line: 14 12, Col: 15>
<SEMICOLON, ";", Line: 14 12, Col: 18>
<DECLARE, "declare", Line: 15 13, Col: 1>
<IDENTIFIER, "CH2", Line: 15 13, Col: 9>
<ASSIGN, "=", Line: 15 13, Col: 13>
<CHARACTER, "'\n'", Line: 15 13, Col: 15>
<SEMICOLON, ";", Line: 15 13, Col: 19>
<DECLARE, "declare", Line: 16 14, Col: 1>
<IDENTIFIER, "CH3", Line: 16 14, Col: 9>
<ASSIGN, "=", Line: 16 14, Col: 13>
<CHARACTER, "'\\'", Line: 16 14, Col: 15>
<SEMICOLON, ";", Line: 16 14, Col: 19>
<IDENTIFIER, "X", Line: 18 15, Col: 1>
<INC, "++", Line: 18 15, Col: 2>
<SEMICOLON, ";", Line: 18 15, Col: 4>
<IDENTIFIER, "Y", Line: 19 16, Col: 1>
<DEC, "--", Line: 19 16, Col: 2>
<SEMICOLON, ";", Line: 19 16, Col: 4>
<IDENTIFIER, "X", Line: 20 17, Col: 1>
<ADD_ASSIGN, "+=", Line: 20 17, Col: 3>
<INTEGER, "5", Line: 20 17, Col: 6>
<SEMICOLON, ";", Line: 20 17, Col: 7>
<IDENTIFIER, "Y", Line: 21 18, Col: 1>
<SUB_ASSIGN, "-=", Line: 21 18, Col: 3>
<INTEGER, "3", Line: 21 18, Col: 6>
<SEMICOLON, ";", Line: 21 18, Col: 7>
<IDENTIFIER, "X", Line: 22 19, Col: 1>
<MUL_ASSIGN, "*=", Line: 22 19, Col: 3>
<INTEGER, "2", Line: 22 19, Col: 6>
<SEMICOLON, ";", Line: 22 19, Col: 7>
<IDENTIFIER, "Y", Line: 23 20, Col: 1>
<DIV_ASSIGN, "/=", Line: 23 20, Col: 3>
<INTEGER, "4", Line: 23 20, Col: 6>
<SEMICOLON, ";", Line: 23 20, Col: 7>
<IDENTIFIER, "IF", Line: 25 21, Col: 1>
<LEFT_PAREN, "(", Line: 25 21, Col: 3>
<IDENTIFIER, "X", Line: 25 21, Col: 4>
<GREATER_EQUAL, ">=", Line: 25 21, Col: 6>
<IDENTIFIER, "Y", Line: 25 21, Col: 9>
<LOGICAL_AND, "&&", Line: 25 21, Col: 11>
<IDENTIFIER, "FLAG", Line: 25 21, Col: 14>
<LOGICAL_OR, "||", Line: 25 21, Col: 19>
<LOGICAL_NOT, "!", Line: 25 21, Col: 22>
<IDENTIFIER, "DONE", Line: 25 21, Col: 23>
<RIGHT_PAREN, ")", Line: 25 21, Col: 27>
<LEFT_BRACE, "{", Line: 25 21, Col: 29>
<OUTPUT, "output", Line: 26 22, Col: 5 1>
<LEFT_PAREN, "(", Line: 26 22, Col: 11 7>
<IDENTIFIER, "X", Line: 26 22, Col: 12 8>
<RIGHT_PAREN, ")", Line: 26 22, Col: 13 9>
<SEMICOLON, ";", Line: 26 22, Col: 14 10>
<RIGHT_BRACE, "}", Line: 27 23, Col: 1>
<ELSE, "else", Line: 28 24, Col: 1>
<LEFT_BRACE, "{", Line: 28 24, Col: 6>
<INPUT, "input", Line: 29 25, Col: 5 1>
<LEFT_PAREN, "(", Line: 29 25, Col: 10 6>
<IDENTIFIER, "Y", Line: 29 25, Col: 11 7>
<RIGHT_PAREN, ")", Line: 29 25, Col: 12 8>
<SEMICOLON, ";", Line: 29 25, Col: 13 9>
<RIGHT_BRACE, "}", Line: 30 26, Col: 1>
<LOOP, "loop", Line: 32 27, Col: 1>
<LEFT_PAREN, "(", Line: 32 27, Col: 6>
<IDENTIFIER, "X", Line: 32 27, Col: 7>
<LESS_THAN, "<", Line: 32 27, Col: 9>
<INTEGER, "100", Line: 32 27, Col: 11>
<RIGHT_PAREN, ")", Line: 32 27, Col: 14>
<LEFT_BRACE, "{", Line: 32 27, Col: 16>
<IDENTIFIER, "X", Line: 33 28, Col: 5 1>
<ASSIGN, "=", Line: 33 28, Col: 7 3>
<IDENTIFIER, "X", Line: 33 28, Col: 9 5>
<ADD, "+", Line: 33 28, Col: 11 7>
<INTEGER, "1", Line: 33 28, Col: 13 9>
<SEMICOLON, ";", Line: 33 28, Col: 14 10>
<CONTINUE, "continue", Line: 34 29, Col: 5 1>
<SEMICOLON, ";", Line: 34 29, Col: 13 9>
<BREAK, "break", Line: 35 30, Col: 5 1>
<SEMICOLON, ";", Line: 35 30, Col: 10 6>
<RIGHT_BRACE, "}", Line: 36 31, Col: 1>
<FUNCTION, "function", Line: 38 32, Col: 1>
<IDENTIFIER, "MyFunc", Line: 38 32, Col: 10>
<LEFT_PAREN, "(", Line: 38 32, Col: 16>
<RIGHT_PAREN, ")", Line: 38 32, Col: 17>
<LEFT_BRACE, "{", Line: 38 32, Col: 19>
<RETURN, "return", Line: 39 33, Col: 5 1>
<IDENTIFIER, "X", Line: 39 33, Col: 12 8>
<EXP, "**", Line: 39 33, Col: 14 10>
<INTEGER, "2", Line: 39 33, Col: 17 13>
<SEMICOLON, ";", Line: 39 33, Col: 18 14>
<RIGHT_BRACE, "}", Line: 40 34, Col: 1>
<FINISH, "finish", Line: 42 35, Col: 1>
```



Result

```
<DECLARE, "declare", Line: 3 2, Col: 1>
<IDENTIFIER, "A", Line: 3 2, Col: 9>
<ASSIGN, "=", Line: 3 2, Col: 11>
<INTEGER, "10", Line: 3 2, Col: 13>
<SEMICOLON, ";", Line: 3 2, Col: 15>
<DECLARE, "declare", Line: 4 3, Col: 1>
<IDENTIFIER, "B", Line: 4 3, Col: 9>
<ASSIGN, "=", Line: 4 3, Col: 11>
<INTEGER, "20", Line: 4 3, Col: 13>
<SEMICOLON, ";", Line: 4 3, Col: 15>
<DECLARE, "declare", Line: 5 4, Col: 1>
<IDENTIFIER, "C", Line: 5 4, Col: 9>
<ASSIGN, "=", Line: 5 4, Col: 11>
<INTEGER, "5", Line: 5 4, Col: 13>
<SEMICOLON, ";", Line: 5 4, Col: 14>
<DECLARE, "declare", Line: 6 5, Col: 1>
<IDENTIFIER, "Result", Line: 6 5, Col: 9>
<ASSIGN, "=", Line: 6 5, Col: 16>
<INTEGER, "0", Line: 6 5, Col: 18>
<SEMICOLON, ";", Line: 6 5, Col: 19>
<IDENTIFIER, "Result", Line: 8 6, Col: 1>
<ASSIGN, "=", Line: 8 6, Col: 8>
<LEFT_PAREN, "(", Line: 8 6, Col: 10>
<IDENTIFIER, "A", Line: 8 6, Col: 11>
<ADD, "+", Line: 8 6, Col: 13>
<IDENTIFIER, "B", Line: 8 6, Col: 15>
<RIGHT_PAREN, ")", Line: 8 6, Col: 16>
<MUL, "*", Line: 8 6, Col: 18>
<IDENTIFIER, "C", Line: 8 6, Col: 20>
<SUB, "-", Line: 8 6, Col: 22>
<INTEGER, "5", Line: 8 6, Col: 24>
<DIV, "/", Line: 8 6, Col: 26>
<INTEGER, "2", Line: 8 6, Col: 28>
<MOD, "%", Line: 8 6, Col: 30>
<INTEGER, "3", Line: 8 6, Col: 32>
<SEMICOLON, ";", Line: 8 6, Col: 33>
<IDENTIFIER, "Result", Line: 10 7, Col: 1>
<ADD_ASSIGN, "+=", Line: 10 7, Col: 8>
<IDENTIFIER, "A", Line: 10 7, Col: 11>
<MUL, "*", Line: 10 7, Col: 13>
<LEFT_PAREN, "(", Line: 10 7, Col: 15>
<IDENTIFIER, "B", Line: 10 7, Col: 16>
<SUB, "-", Line: 10 7, Col: 18>
<IDENTIFIER, "C", Line: 10 7, Col: 20>
<RIGHT_PAREN, ")", Line: 10 7, Col: 21>
<EXP, "**", Line: 10 7, Col: 23>
<INTEGER, "2", Line: 10 7, Col: 26>
<SEMICOLON, ";", Line: 10 7, Col: 27>
<IDENTIFIER, "IF", Line: 12 8, Col: 1>
<LEFT_PAREN, "(", Line: 12 8, Col: 4>
<IDENTIFIER, "A", Line: 12 8, Col: 5>
<GREATER_THAN, ">", Line: 12 8, Col: 7>
<IDENTIFIER, "B", Line: 12 8, Col: 9>
<LOGICAL_OR, "||", Line: 12 8, Col: 11>
<IDENTIFIER, "B", Line: 12 8, Col: 14>
<LESS_EQUAL, "<=", Line: 12 8, Col: 16>
<IDENTIFIER, "C", Line: 12 8, Col: 19>
<LOGICAL_AND, "&&", Line: 12 8, Col: 21>
<IDENTIFIER, "A", Line: 12 8, Col: 24>
<NOT_EQUAL, "!=", Line: 12 8, Col: 26>
<IDENTIFIER, "C", Line: 12 8, Col: 29>
<RIGHT_PAREN, ")", Line: 12 8, Col: 30>
<LEFT_BRACE, "{", Line: 12 8, Col: 32>
<IDENTIFIER, "Result", Line: 13 9, Col: 5 1>
<ASSIGN, "=", Line: 13 9, Col: 12 8>
<IDENTIFIER, "Result", Line: 13 9, Col: 14 10>
<ADD, "+", Line: 13 9, Col: 21 17>
<INTEGER, "1", Line: 13 9, Col: 23 19>
<SEMICOLON, ";", Line: 13 9, Col: 24 20>
<RIGHT_BRACE, "}", Line: 14 10, Col: 1>
<ELSE, "else", Line: 15 11, Col: 1>
<LEFT_BRACE, "{", Line: 15 11, Col: 6>
<IDENTIFIER, "Result", Line: 16 12, Col: 5 1>
<ASSIGN, "=", Line: 16 12, Col: 12 8>
<IDENTIFIER, "Result", Line: 16 12, Col: 14 10>
<SUB, "-", Line: 16 12, Col: 21 17>
<INTEGER, "1", Line: 16 12, Col: 23 19>
<SEMICOLON, ";", Line: 16 12, Col: 24 20>
<RIGHT_BRACE, "}", Line: 17 13, Col: 1>
<FINISH, "finish", Line: 19 14, Col: 1>
```

```
<DECLARE, "declare", Line: 32, Col: 1>
<IDENTIFIER, "S1", Line: 32, Col: 9>
<ASSIGN, "=", Line: 32, Col: 12>
<STRING, ""Hello\nWorld"", Line: 32, Col: 14>
<SEMICOLON, ";", Line: 32, Col: 28>
<DECLARE, "declare", Line: 43, Col: 1>
<IDENTIFIER, "S2", Line: 43, Col: 9>
<ASSIGN, "=", Line: 43, Col: 12>
<STRING, ""Tab\tSeparated"", Line: 43, Col: 14>
<SEMICOLON, ";", Line: 43, Col: 30>
<DECLARE, "declare", Line: 54, Col: 1>
<IDENTIFIER, "S5", Line: 54, Col: 9>
<ASSIGN, "=", Line: 54, Col: 12>
<STRING, ""Mix\n\t\\\""", Line: 54, Col: 14>
<SEMICOLON, ";", Line: 54, Col: 27>
<DECLARE, "declare", Line: 75, Col: 1>
<IDENTIFIER, "C3", Line: 75, Col: 9>
<ASSIGN, "=", Line: 75, Col: 12>
<CHARACTER, "'\\'", Line: 75, Col: 14>
<SEMICOLON, ";", Line: 75, Col: 18>
<DECLARE, "declare", Line: 86, Col: 1>
<IDENTIFIER, "C4", Line: 86, Col: 9>
<ASSIGN, "=", Line: 86, Col: 12>
<CHARACTER, "'\''", Line: 86, Col: 14>
<SEMICOLON, ";", Line: 86, Col: 18>
<FINISH, "finish", Line: 107, Col: 1>
```

```
<DECLARE, "declare", Line: 32, Col: 1>
[ERROR] Type: ERROR | Line: 32, Col: 9 | Lexeme: 'lowercase' | Reason: Invalid identifier (must start with Uppercase or be a keyword)
<ASSIGN, "=", Line: 32, Col: 19>
<INTEGER, "10", Line: 32, Col: 21>
<SEMICOLON, ";", Line: 32, Col: 23>
<DECLARE, "declare", Line: 43, Col: 1>
[ERROR] Type: ERROR | Line: 43, Col: 9 | Lexeme: 'TooLongIdentifierNameThatExceedsThirtyOneCharacters' | Reason: Identifier exceeds max
imummax length (31)
<ASSIGN, "=", Line: 43, Col: 61>
<INTEGER, "5", Line: 43, Col: 63>
<SEMICOLON, ";", Line: 43, Col: 64>
<DECLARE, "declare", Line: 64, Col: 1>
<IDENTIFIER, "WrongFloat", Line: 64, Col: 9>
<ASSIGN, "=", Line: 64, Col: 20>
[ERROR] Type: ERROR | Line: 64, Col: 22 | Lexeme: '12.3.4' | Reason: Malformed numericfloat:  literal (multiple decimal points)
<SEMICOLON, ";", Line: 64, Col: 28>
<DECLARE, "declare", Line: 85, Col: 1>
<IDENTIFIER, "BadString", Line: 85, Col: 9>
<ASSIGN, "=", Line: 85, Col: 19>
[ERROR] Type: ERROR | Line: 107, Col: 1 | Lexeme: '#* Unclosed multi-line comment
' | Reason: Unclosed multi-line comment  (missing *#)
```

```
<DECLARE, "declare", Line: 43, Col: 1>
<IDENTIFIER, "X", Line: 43, Col: 9>
<ASSIGN, "=", Line: 43, Col: 11>
<INTEGER, "10", Line: 43, Col: 13>
<SEMICOLON, ";", Line: 43, Col: 15>
<FINISH, "finish", Line: 107, Col: 13>
```