Comparison of Manual and JFLEX Scanner

Implementation Approach

The manual scanner (ManualScanner.java) was built entirely by hand using a character-by-character DFA simulation. Each token type has its own scanXxx() method that manually checks characters using if/while logic. The JFlex scanner (Scanner.flex) achieves the same result by writing regex rules in a specification file. JFlex then **automatically generates** the DFA code (Yylex.java).

Token Output

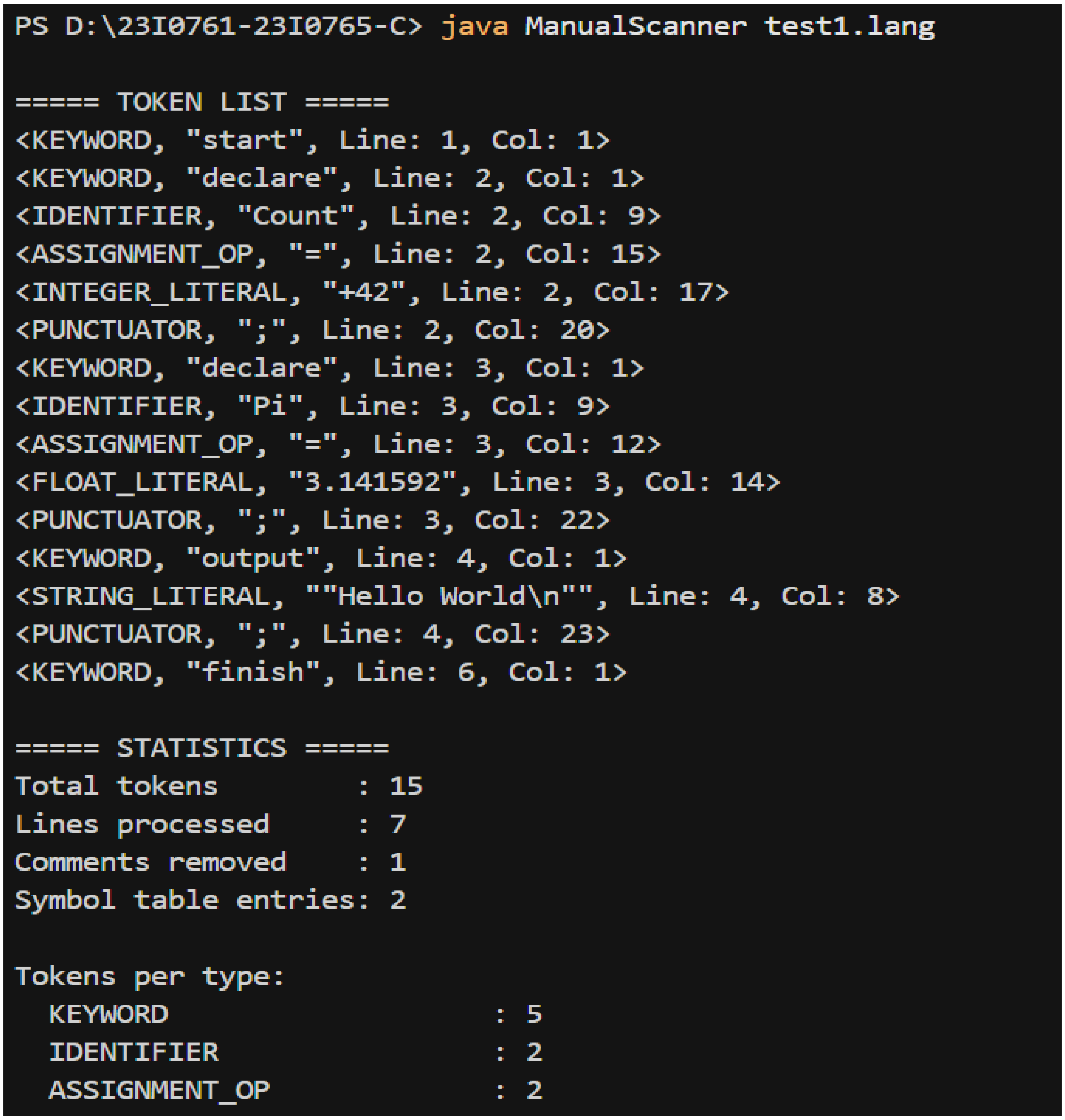
Both scanners produce **identical token output** on the same input files. Same format, same token types, same line/column numbers. This confirms that both implementations correctly follow the language specification.

Key Differences

Aspect	**Manual Scanner**	**JFlex Scanner**
Code written by you	~400 lines of Java	~100 lines of flex rules
DFA implementation	Hand-coded if/while checks	Auto-generated table lookup
Adding a new token	Write a new scanXxx() method	Add one regex rule
Error recovery	Custom restore() logic	Handled by catch-all [^] rule

Aspect	Manual Scanner	JFlex Scanner
Maintainability	Harder to modify	Easy to update rules
Learning value	Understand DFA internals deeply	Understand regex-to-DFA process

MANUAL SCANNER

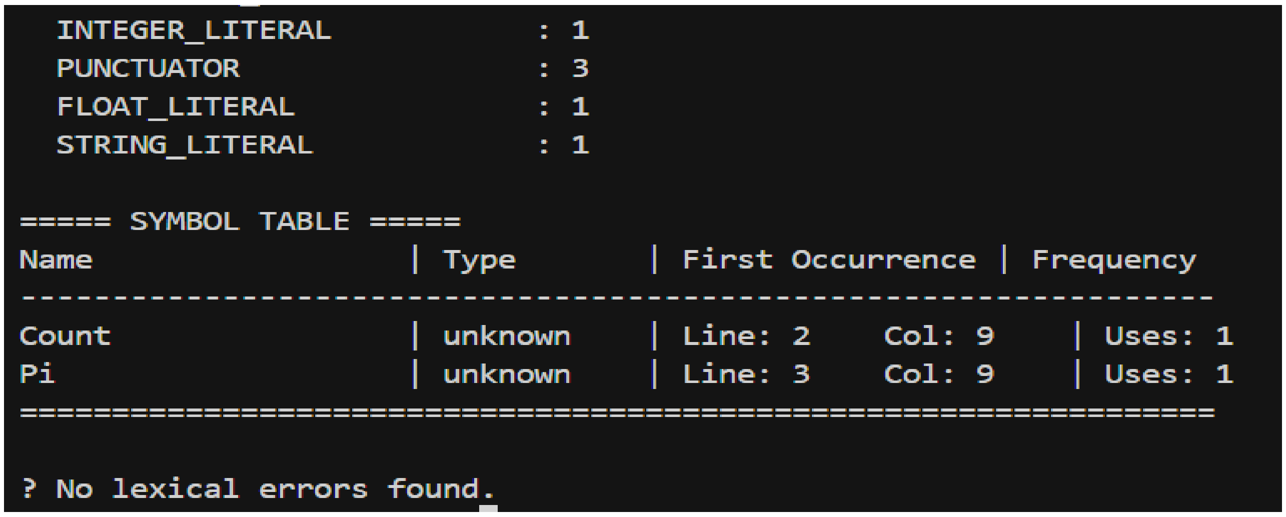


```
PS D:\23I0761-23I0765-C> java ManualScanner test1.lang

===== TOKEN LIST =====
<KEYWORD, "start", Line: 1, Col: 1>
<KEYWORD, "declare", Line: 2, Col: 1>
<IDENTIFIER, "Count", Line: 2, Col: 9>
<ASSIGNMENT_OP, "=", Line: 2, Col: 15>
<INTEGER_LITERAL, "+42", Line: 2, Col: 17>
<PUNCTUATOR, ";", Line: 2, Col: 20>
<KEYWORD, "declare", Line: 3, Col: 1>
<IDENTIFIER, "Pi", Line: 3, Col: 9>
<ASSIGNMENT_OP, "=", Line: 3, Col: 12>
<FLOAT_LITERAL, "3.141592", Line: 3, Col: 14>
<PUNCTUATOR, ";", Line: 3, Col: 22>
<KEYWORD, "output", Line: 4, Col: 1>
<STRING_LITERAL, ""Hello World\n"", Line: 4, Col: 8>
<PUNCTUATOR, ";", Line: 4, Col: 23>
<KEYWORD, "finish", Line: 6, Col: 1>

===== STATISTICS =====
Total tokens        : 15
Lines processed     : 7
Comments removed    : 1
Symbol table entries: 2

Tokens per type:
  KEYWORD                : 5
  IDENTIFIER             : 2
  ASSIGNMENT_OP          : 2
```

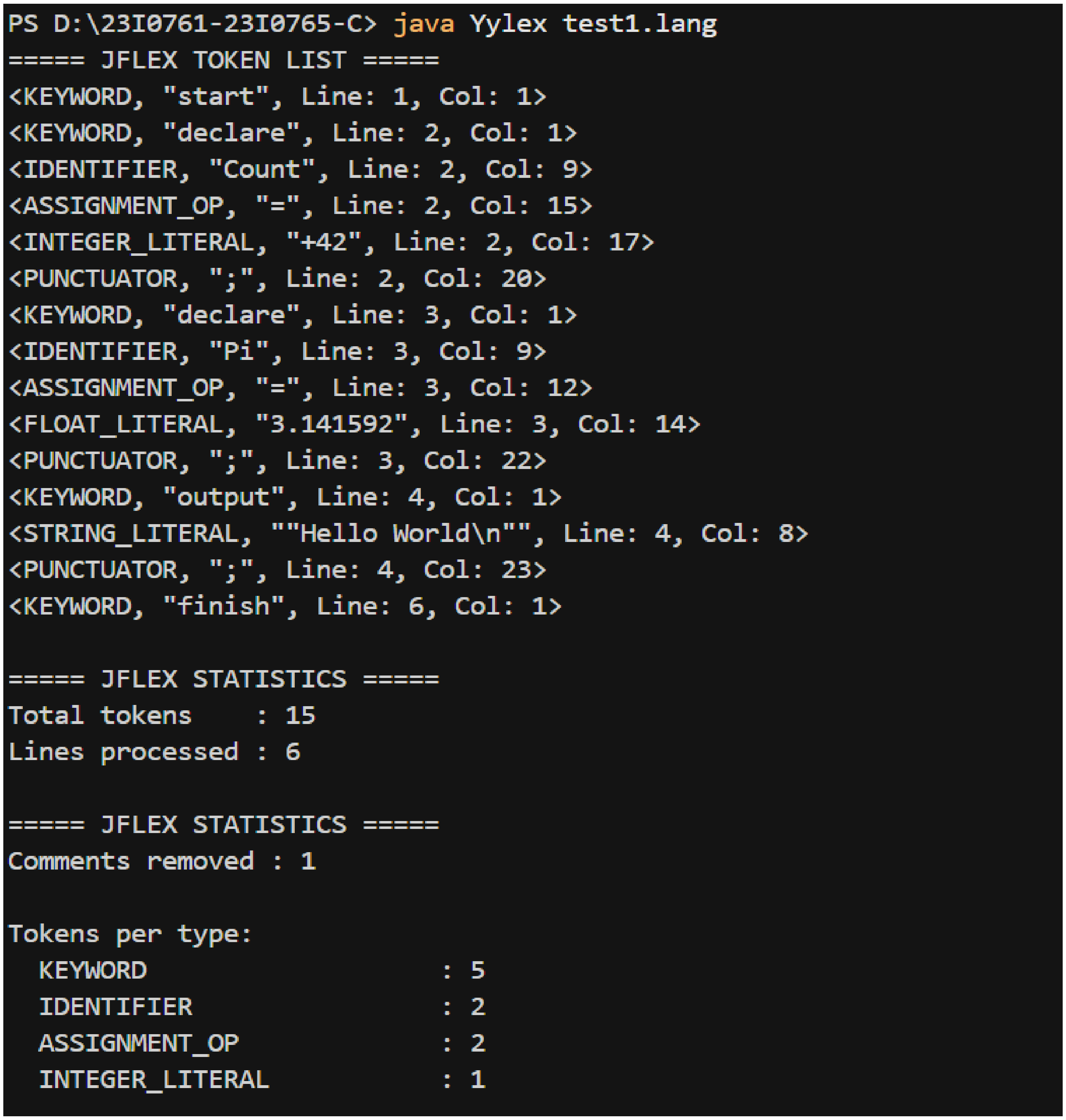


```
  INTEGER_LITERAL          : 1
  PUNCTUATOR               : 3
  FLOAT_LITERAL            : 1
  STRING_LITERAL           : 1

===== SYMBOL TABLE =====
Name                 | Type        | First Occurrence | Frequency
-----------------------------------------------------------------
Count                | unknown     | Line: 2    Col: 9     | Uses: 1
Pi                   | unknown     | Line: 3    Col: 9     | Uses: 1
=================================================================

? No lexical errors found.
```

JFLEX Scanner

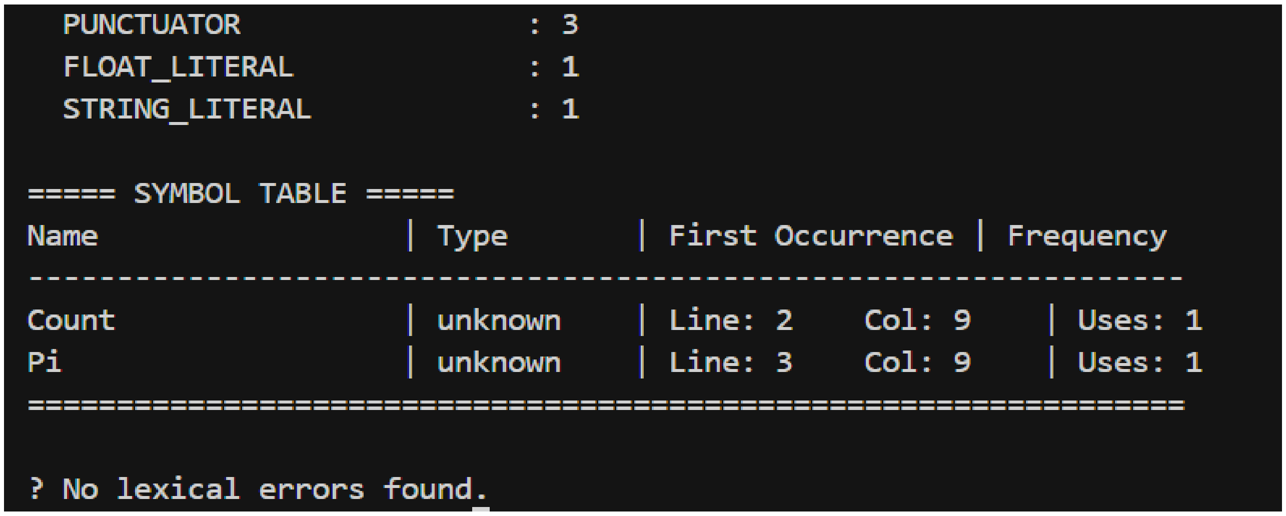


```
PS D:\23I0761-23I0765-C> java Yylex test1.lang
===== JFLEX TOKEN LIST =====
<KEYWORD, "start", Line: 1, Col: 1>
<KEYWORD, "declare", Line: 2, Col: 1>
<IDENTIFIER, "Count", Line: 2, Col: 9>
<ASSIGNMENT_OP, "=", Line: 2, Col: 15>
<INTEGER_LITERAL, "+42", Line: 2, Col: 17>
<PUNCTUATOR, ";", Line: 2, Col: 20>
<KEYWORD, "declare", Line: 3, Col: 1>
<IDENTIFIER, "Pi", Line: 3, Col: 9>
<ASSIGNMENT_OP, "=", Line: 3, Col: 12>
<FLOAT_LITERAL, "3.141592", Line: 3, Col: 14>
<PUNCTUATOR, ";", Line: 3, Col: 22>
<KEYWORD, "output", Line: 4, Col: 1>
<STRING_LITERAL, ""Hello World\n"", Line: 4, Col: 8>
<PUNCTUATOR, ";", Line: 4, Col: 23>
<KEYWORD, "finish", Line: 6, Col: 1>

===== JFLEX STATISTICS =====
Total tokens     : 15
Lines processed : 6

===== JFLEX STATISTICS =====
Comments removed : 1

Tokens per type:
  KEYWORD                : 5
  IDENTIFIER             : 2
  ASSIGNMENT_OP          : 2
  INTEGER_LITERAL        : 1
```


```
   PUNCTUATOR                : 3
   FLOAT_LITERAL             : 1
   STRING_LITERAL            : 1

===== SYMBOL TABLE =====
Name                 | Type       | First Occurrence | Frequency
-----------------------------------------------------------------
Count                | unknown    | Line: 2    Col: 9     | Uses: 1
Pi                   | unknown    | Line: 3    Col: 9     | Uses: 1
=================================================================

? No lexical errors found.
```