**CS4031 — Compiler Construction — Assignment 01**

Scanner Comparison: Manual vs JFlex

**Group:** Ali Aamir Ashraf 22I-1199 & Dawood Cheema 22I-0875 | **Section:** G

---

**1. Overview**

Both scanners were tested on 5 test files covering all token types specified in Section 3 of the assignment. This document compares their outputs side-by-side, explains differences, and provides a performance comparison.

| Scanner | Approach | Language |
|---|---|---|
| ManualScanner.java | Hand-coded DFA with longest-match | Java |
| Scanner.flex (Yylex) | JFlex-generated scanner from regex rules | Java (generated) |

---

**2. Side-by-Side Output Comparison**

**Test 1 (test1.lang) — All Valid Tokens**

| Metric | JFlex | Manual | Match? |
|---|---|---|---|
| Identifiers | 17 | 17 | ✅ |
| Integer Literals | 5 | 5 | ⚠️ |
| Float Literals | 7 | 7 | ⚠️ |
| String Literals | 7 | 7 | ✅ |
| Char Literals | 8 | 8 | ✅ |
| Boolean Literals | 2 | 2 | ✅ |
| Comments Removed | 12 | 12 | ✅ |
| Errors | 0 | 0 | ✅ |

**Difference — Signed Literals:** JFlex produces INT_LITERAL "+12" and INT_LITERAL "-9999" (sign included in the token), while Manual Scanner produces OP_ARITHMETIC "+" + INT_LITERAL "12" (sign as a separate operator token). Same applies to signed floats like -0.123456 and +5.000001. This is because the Manual Scanner's signIsUnary() heuristic treats +/- at the start of a line as operators, whereas JFlex's regex [+-]?[0-9]+ greedily matches the sign as part of the literal.

**Impact:** Both approaches are valid. The signed values are semantically identical — +12 as one token or + then 12 as two tokens produce the same result in later compiler phases.

**Test 2 (test2.lang) — Mixed Tokens with Repeated Identifiers**

| Metric | JFlex | Manual | Match? |
|---|---|---|---|
| Identifiers | 9 | 9 | ✅ |
| Integer Literals | 5 | 5 | ⚠️ |
| Float Literals | 6 | 6 | ⚠️ |
| String Literals | 6 | 6 | ✅ |
| Char Literals | 8 | 8 | ✅ |
| Boolean Literals | 3 | 3 | ✅ |
| Symbol Table Entries | 4 | 4 | ✅ |
| Alpha frequency | 4 | 4 | ✅ |
| Errors | 0 | 0 | ✅ |

**Difference:** Same signed literal difference as Test 1. -20, +7, -0.500001 are single tokens in JFlex but split into operator + literal in Manual Scanner.

**Test 3 (test3.lang) — Strings & Chars with Escape Sequences**

| Metric | JFlex | Manual | Match? |
|---|---|---|---|
| String Literals | 6 | 6 | ✅ |
| Char Literals | 9 | 9 | ✅ |
| Identifiers | 1 (After) | 1 (After) | ✅ |
| Comments Removed | 5 | 5 | ✅ |
| Errors | 5 | 5 | ✅ |

**Error Comparison:**

| Line | JFlex Error | Manual Error | Match? |
|---|---|---|---|
| 22 | Malformed string literal | Malformed string literal | ✅ |
| 23 | Unterminated string literal | Unterminated string literal | ✅ |
| 24 | Malformed character literal | Malformed character literal | ✅ |
| 25 | Malformed character literal ('AB') | Malformed character literal ('AB') | ✅ |
| 26 | Unterminated character literal | Unterminated character literal | ✅ |

**No differences.** Both scanners correctly recover after each error and continue scanning. The post-error tokens (After, "ok", '\n') are identical.

**Test 4 (test4.lang) — Lexical Errors**

| Metric | JFlex | Manual | Match? |
|---|---|---|---|
| Valid Tokens | 7 | 9 | ⚠️ |
| Errors | 19 | 18 | ⚠️ |
| Comments Removed | 12 | 12 | ✅ |
| Unclosed Comment Detected | Yes | Yes | ✅ |

**Token Differences:**

| Line | Input | JFlex | Manual | Explanation |
|---|---|---|---|---|
| 7 | 9start | INT "9" only | INT "9" + KEYWORD "start" | JFlex treats start after 9 as an identifier error. Manual Scanner correctly separates them: 9 ends the integer DFA, then start matches as a keyword. |
| 22 | 1,000 | INT "1" + INT "000" | INT "1" + PUNCT "," + INT "000" | Manual Scanner correctly recognizes , as a punctuator (Section 3.9). JFlex treats it as an invalid character in this numeric context. |

**Error Differences:** JFlex reports start on line 7 as INVALID_IDENTIFIER and , on line 22 as INVALID_CHARACTER. Manual Scanner emits them as valid tokens instead. All other 17 errors are identical between both scanners.

**Test 5 (test5.lang) — Comments**

| Metric | JFlex | Manual | Match? |
|---|---|---|---|
| Identifiers | 5 | 5 | ✅ |
| Integer Literals | 1 | 1 | ✅ |
| String Literals | 2 | 2 | ✅ |
| Boolean Literals | 1 | 1 | ✅ |
| Comments Removed | 7 | 7 | ✅ |
| Errors | 0 | 0 | ✅ |

**No differences — perfect match.** Both scanners correctly handle ## and #*...*# comments, and both correctly preserve comment-like text inside string literals (e.g., "This is not a comment: ## hello").

---

### 3. Summary of All Differences

| Difference | Cause | Impact |
|---|---|---|
| Signed literals (+12, -9999) split into operator + number by Manual Scanner | signIsUnary() heuristic in Manual Scanner vs greedy regex in JFlex | No semantic impact — same result in parsing phase |
| 9start tokenized differently | Manual Scanner's DFA completes integer then starts new token; JFlex glues them | Both valid error recovery strategies |
| , in 1,000 treated as punctuator by Manual Scanner | , is a valid punctuator per Section 3.9 | Manual Scanner is technically more correct per spec |

**All other outputs are identical across all 5 tests**, including: identifiers, string/char literals with escape sequences, boolean literals, float literals, comment removal, symbol table entries and frequencies, and error detection/recovery.

---

### 4. Performance Comparison

| Metric | JFlex Scanner | Manual Scanner |
|---|---|---|

| | | |
|---|---|---|
| Implementation Time | ~2 hours (writing .flex rules) | ~8 hours (hand-coding DFA) |
| Code Size | ~150 lines (.flex) + generated code | ~450 lines (ManualScanner.java) |
| Maintainability | High — change regex, regenerate | Medium — must modify DFA logic manually |
| Flexibility | Limited to JFlex's capabilities | Full control over error recovery and edge cases |
| Compilation | Requires JFlex tool to generate .java | Direct Java compilation |
| Runtime Speed | Very fast (optimized table-driven DFA) | Fast (hand-coded conditionals and switches) |
| Error Recovery | Basic — skip to next valid token | Customizable — can skip to closing delimiter |
| Nested Comments | Requires manual state tracking | Naturally supported via depth counter |

**Key Observations:**

- **JFlex is faster to develop** — regex rules map directly from the assignment spec. The generated scanner is highly optimized with lookup tables.
- **Manual Scanner offers better control** — error recovery can be fine-tuned (e.g., scanning to closing quote on malformed literals rather than stopping immediately). Nested multi-line comment support was trivial to add with a depth counter.
- **Both produce functionally equivalent output** — the minor differences in signed literal handling and edge-case error reporting do not affect the correctness of tokenization for valid programs.

---

## 5. Conclusion

Both scanners successfully recognize all token types defined in Section 3 of the assignment specification. They produce identical output on valid inputs and handle error cases with appropriate recovery. The differences are limited to ambiguous edge cases (signed literals at line start, token boundary after digits) where both approaches are defensible. The Manual Scanner's DFA-based implementation validates the correctness of the JFlex specification, and vice versa.