CSC 3315, Assignment 1

A Grammar for the X- Programming Language BNF Grammar Description & Lexical Analyzer (Scanner)

Ziad El Ismaili

Al Akhawayn University in Ifrane

Saturday, March 19, 2022

Supervised by:

Pr. Hamid Harroud

Table of Contents

I. Java- Language grammar and code structure description:

In this project, the goal is to implement a programming language of our choice. In this first part, we are going to define the BNF of our program language that we called ***Java-***. Our language is a subset of rules extracted from the Java language syntax structure or grammar.

We choose following rules in BNF Grammar format to define our programming language grammar:

Types:

```
<type> ::= void | char | short | int | long | signed |
             unsigned | boolean | <floating-point type>

<floating-point type> ::= float | double

<array type> ::= <type> []
```

Operators:

```
<assignment-operator> ::= = | *= | /= | += | -= | <=
                          | >=  | <>

<unary-operator> ::= & | * | + | - | | | . | ; | : | < | >
```

Expressions:

```
<expression> ::= <identifier> | <constant> | <string> |
             ( <expression> ) | <assignment-expression> |
                   <expression> , <assignment-expression>
                         |{ <expression> }
```

```
<constant> ::= <integer-constant> | <character-constant>
                   | <floating-constant>


<string> ::=

<assignment-expression> ::= <identifier>
                             <assignment-operator>
                             <identifier>
```

Statements:

```
<statement> ::= <expression-statement> | <if-statement> |
             <jump-statement> | <while-statement> |

<expression-statement> ::= {<expression>}? ;


<if-statement> ::= if ( <expression> ) <statement>
      | if ( <expression> ) <statement> else <statement>

<jump-statement> ::=   continue ; | break ; |
                         return {<expression>}? ;


<while-statement> ::=  while ( <expression> ) <statement>


<for-statement> ::=   for ( {<expression>}?
      ;{<expression>}? ; {<expression>}? ) <statement>
```

II. Java- lexer specifications:

In this second part, we will use a tool called Jflex, in order to build the lexer or the scanner for our Java- language. The following are the regular expressions of our language, based on the previous part of the grammar:

ALPHA=[A-Za-z]

DIGIT=[0-9]

NONNEWLINE_WHITE_SPACE_CHAR=[\ \t\b\012]

NEWLINE=\r|\n|\r\n

WHITE_SPACE_CHAR=[\n\r\ \t\b\012]

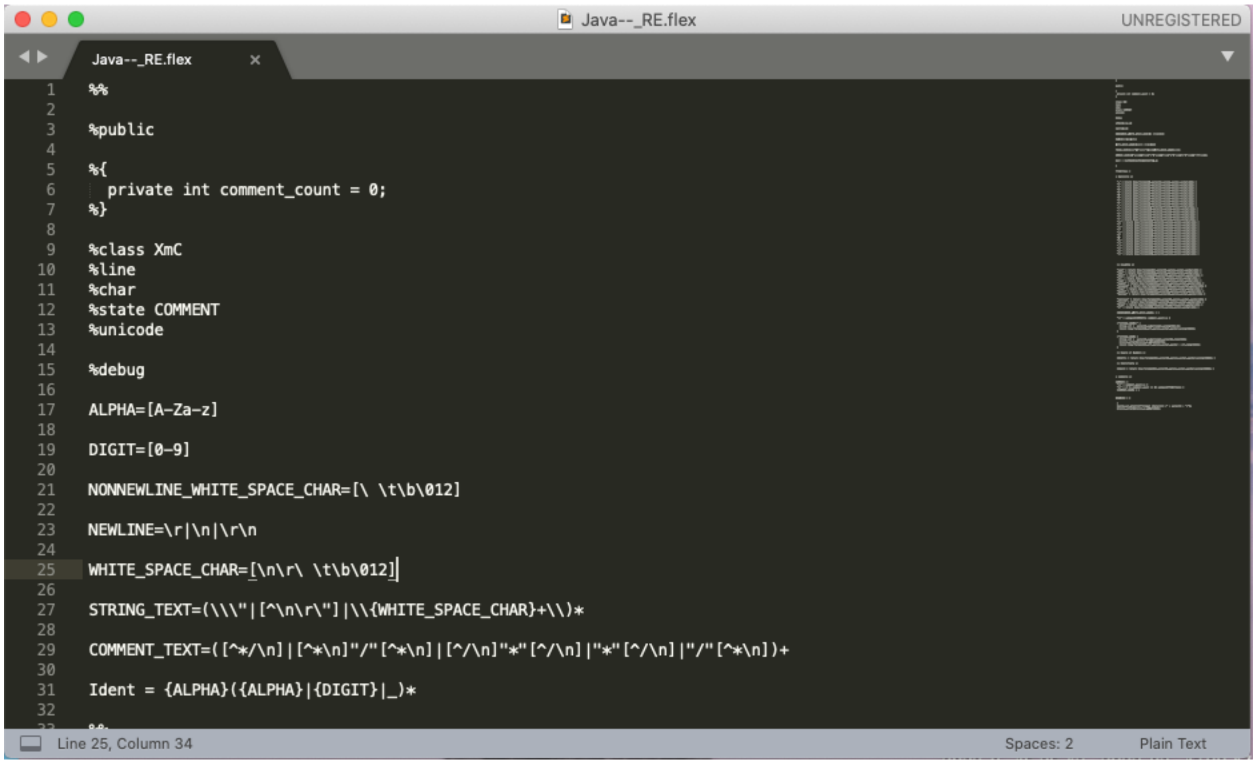
STRING_TEXT=(\\\"|[^\n\r\"]|\\{WHITE_SPACE_CHAR}+\\)*

COMMENT_TEXT=([^*/\n]|[^*\n]"/"[^*\n]|[^/\n]"*"[^/\n]|"*"[^/\n]|"/"[^*\n])+

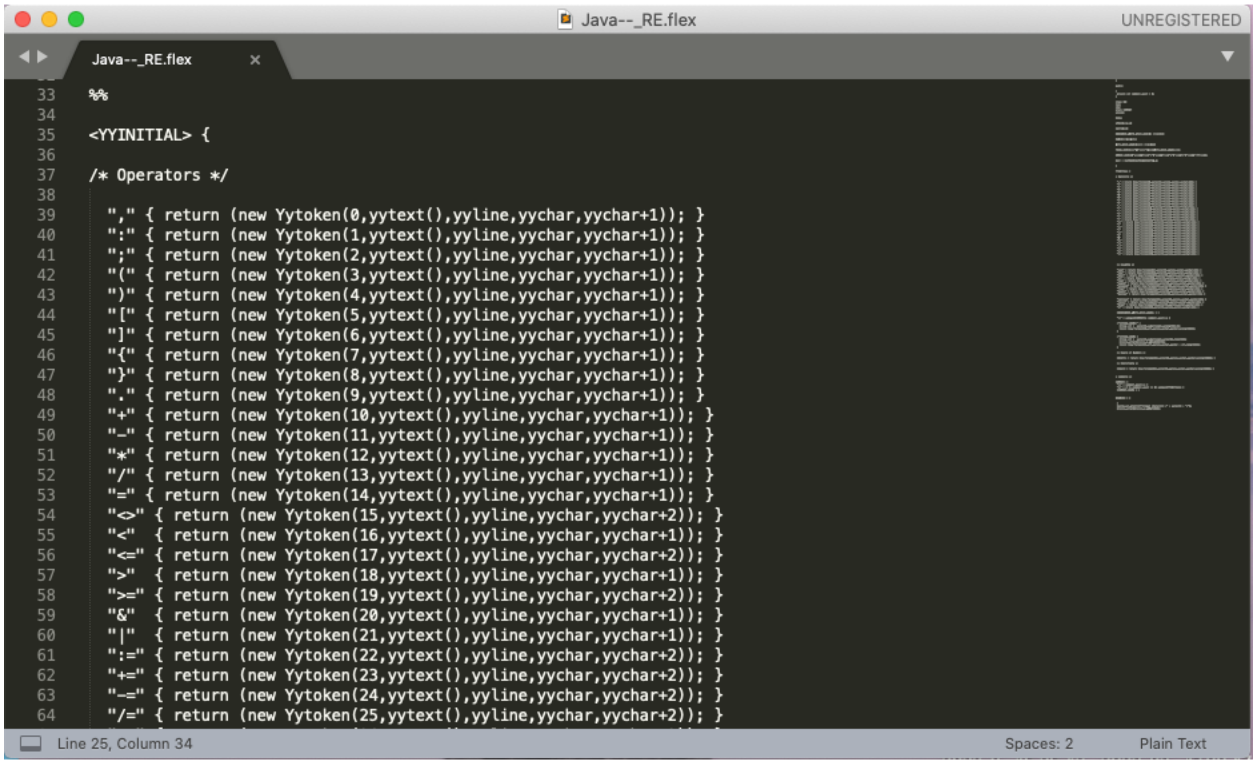
Ident = {ALPHA}({ALPHA}|{DIGIT}|_)*

Those regular expressions are going to be put in a .jlex file, along with other final keywords and operators, which will be the input of the JFlex application, that will generate a .java file, containing the necessary code to generate our lexical analyzer. The generated file will be named XmC.java. Below are some screenshots of the process.

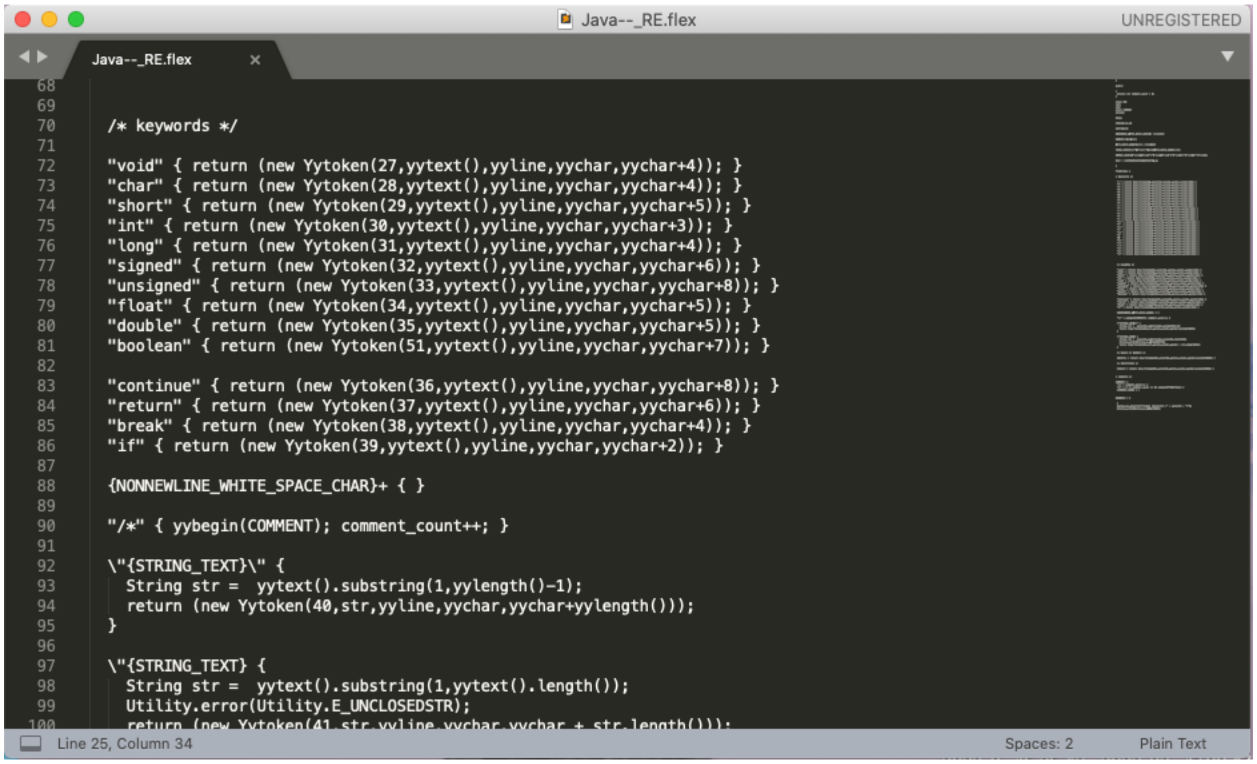
Step 1: *Creating and writing the Java--_RE.jlex file (View zip file for full code)*



```
%%

%public

%{
  private int comment_count = 0;
%}

%class XmC
%line
%char
%state COMMENT
%unicode

%debug

ALPHA=[A-Za-z]

DIGIT=[0-9]

NONNEWLINE_WHITE_SPACE_CHAR=[\ \t\b\012]

NEWLINE=\r|\n|\r\n

WHITE_SPACE_CHAR=[\n\r\ \t\b\012]

STRING_TEXT=(\\\"|[^\n\r\"]|\\{WHITE_SPACE_CHAR}+\\)*

COMMENT_TEXT=([^*/\n]|[^*\n]"/"[^*\n]|[^/\n]"*"[^/\n]|"*"[^/\n]|"/"[^*\n])+

Ident = {ALPHA}({ALPHA}|{DIGIT}|_)*
```

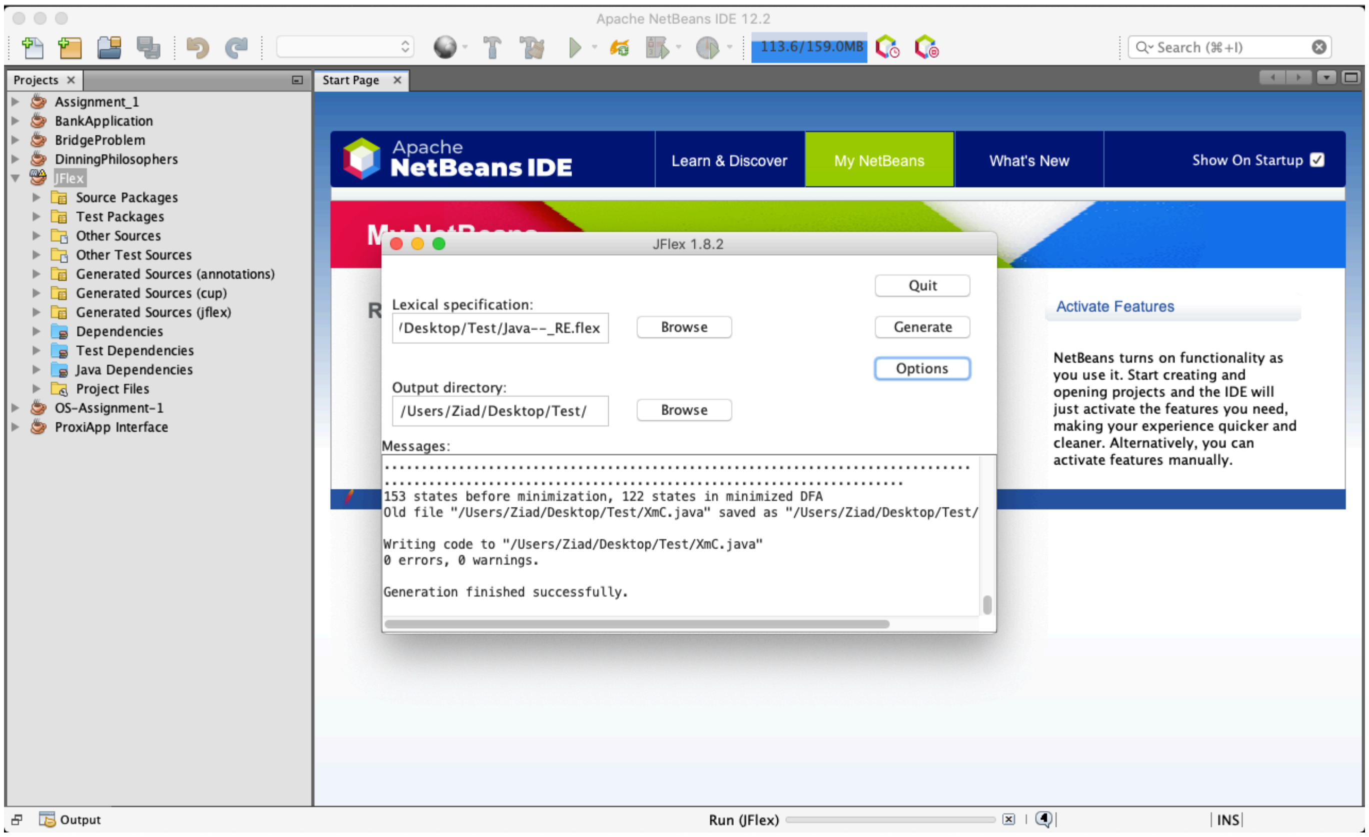


```
%%

<YYINITIAL> {

/* Operators */

  "," { return (new Yytoken(0,yytext(),yyline,yychar,yychar+1)); }
  ":" { return (new Yytoken(1,yytext(),yyline,yychar,yychar+1)); }
  ";" { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
  "(" { return (new Yytoken(3,yytext(),yyline,yychar,yychar+1)); }
  ")" { return (new Yytoken(4,yytext(),yyline,yychar,yychar+1)); }
  "[" { return (new Yytoken(5,yytext(),yyline,yychar,yychar+1)); }
  "]" { return (new Yytoken(6,yytext(),yyline,yychar,yychar+1)); }
  "{" { return (new Yytoken(7,yytext(),yyline,yychar,yychar+1)); }
  "}" { return (new Yytoken(8,yytext(),yyline,yychar,yychar+1)); }
  "." { return (new Yytoken(9,yytext(),yyline,yychar,yychar+1)); }
  "+" { return (new Yytoken(10,yytext(),yyline,yychar,yychar+1)); }
  "-" { return (new Yytoken(11,yytext(),yyline,yychar,yychar+1)); }
  "*" { return (new Yytoken(12,yytext(),yyline,yychar,yychar+1)); }
  "/" { return (new Yytoken(13,yytext(),yyline,yychar,yychar+1)); }
  "=" { return (new Yytoken(14,yytext(),yyline,yychar,yychar+1)); }
  "<>" { return (new Yytoken(15,yytext(),yyline,yychar,yychar+2)); }
  "<"  { return (new Yytoken(16,yytext(),yyline,yychar,yychar+1)); }
  "<=" { return (new Yytoken(17,yytext(),yyline,yychar,yychar+2)); }
  ">"  { return (new Yytoken(18,yytext(),yyline,yychar,yychar+1)); }
  ">=" { return (new Yytoken(19,yytext(),yyline,yychar,yychar+2)); }
  "&"  { return (new Yytoken(20,yytext(),yyline,yychar,yychar+1)); }
  "|"  { return (new Yytoken(21,yytext(),yyline,yychar,yychar+1)); }
  ":=" { return (new Yytoken(22,yytext(),yyline,yychar,yychar+2)); }
  "+=" { return (new Yytoken(23,yytext(),yyline,yychar,yychar+2)); }
  "-=" { return (new Yytoken(24,yytext(),yyline,yychar,yychar+2)); }
  "/=" { return (new Yytoken(25,yytext(),yyline,yychar,yychar+2)); }
```



```

  /* keywords */

  "void" { return (new Yytoken(27,yytext(),yyline,yychar,yychar+4)); }
  "char" { return (new Yytoken(28,yytext(),yyline,yychar,yychar+4)); }
  "short" { return (new Yytoken(29,yytext(),yyline,yychar,yychar+5)); }
  "int" { return (new Yytoken(30,yytext(),yyline,yychar,yychar+3)); }
  "long" { return (new Yytoken(31,yytext(),yyline,yychar,yychar+4)); }
  "signed" { return (new Yytoken(32,yytext(),yyline,yychar,yychar+6)); }
  "unsigned" { return (new Yytoken(33,yytext(),yyline,yychar,yychar+8)); }
  "float" { return (new Yytoken(34,yytext(),yyline,yychar,yychar+5)); }
  "double" { return (new Yytoken(35,yytext(),yyline,yychar,yychar+5)); }
  "boolean" { return (new Yytoken(51,yytext(),yyline,yychar,yychar+7)); }

  "continue" { return (new Yytoken(36,yytext(),yyline,yychar,yychar+8)); }
  "return" { return (new Yytoken(37,yytext(),yyline,yychar,yychar+6)); }
  "break" { return (new Yytoken(38,yytext(),yyline,yychar,yychar+4)); }
  "if" { return (new Yytoken(39,yytext(),yyline,yychar,yychar+2)); }

  {NONNEWLINE_WHITE_SPACE_CHAR}+ { }

  "/*" { yybegin(COMMENT); comment_count++; }

  \"{STRING_TEXT}\" {
    String str =  yytext().substring(1,yylength()-1);
    return (new Yytoken(40,str,yyline,yychar,yychar+yylength()));
  }

  \"{STRING_TEXT} {
    String str =  yytext().substring(1,yytext().length());
    Utility.error(Utility.E_UNCLOSEDSTR);
    return (new Yytoken(41,str,yyline,yychar,yychar + str.length()));
```

<u>Step 2:</u> *Generation of XmC.java using JFlex application.*



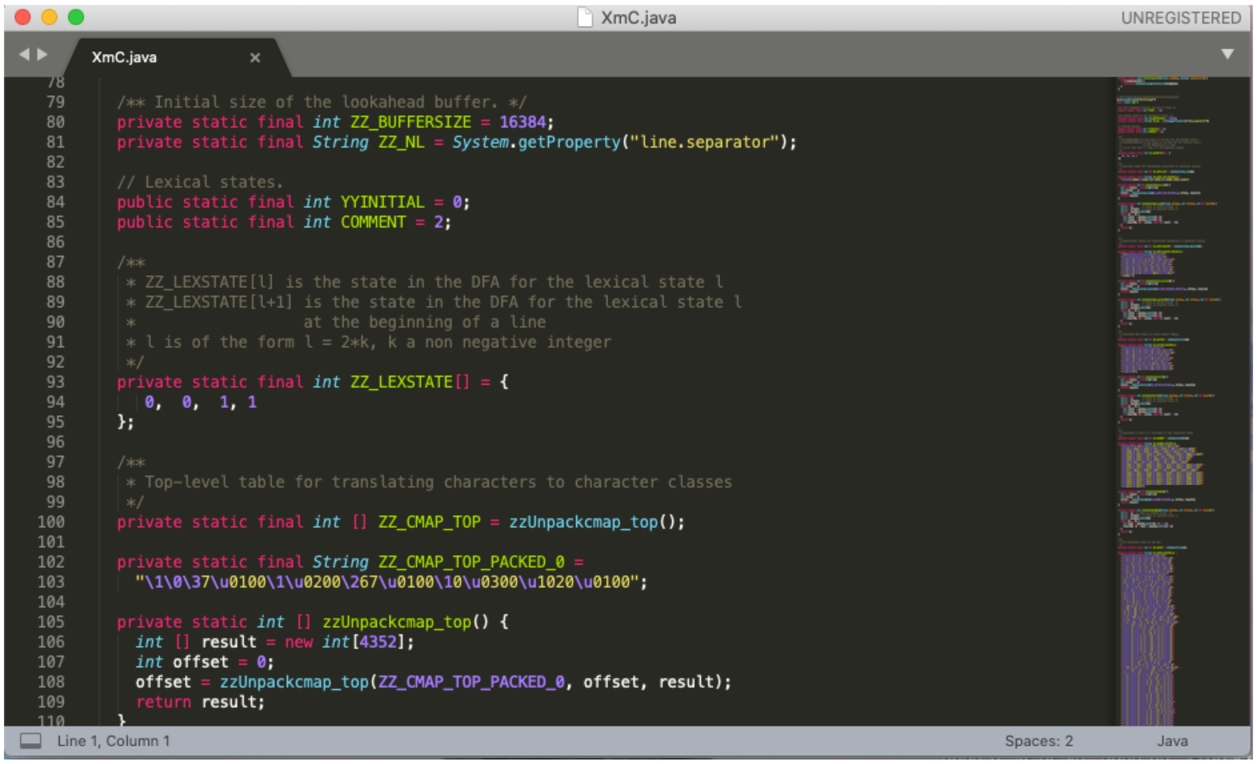
Lexical specification: /Desktop/Test/Java--_RE.flex

Output directory: /Users/Ziad/Desktop/Test/

Messages:

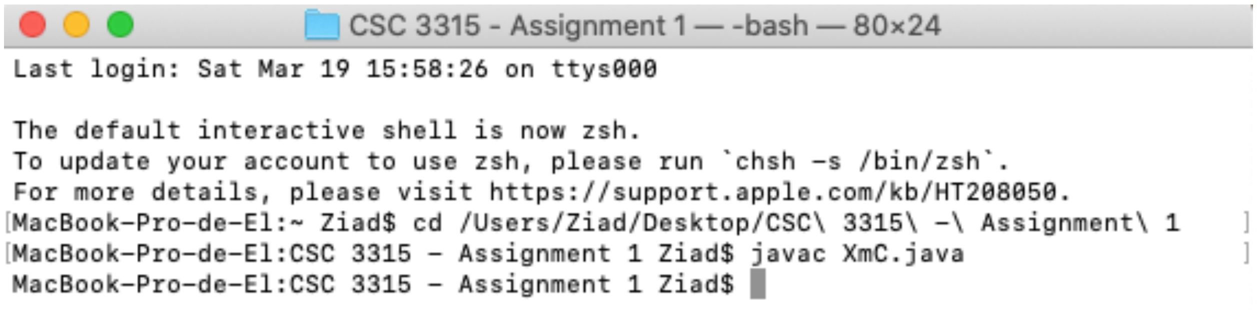
```
...............................................................................
......................................................................
153 states before minimization, 122 states in minimized DFA
Old file "/Users/Ziad/Desktop/Test/XmC.java" saved as "/Users/Ziad/Desktop/Test/

Writing code to "/Users/Ziad/Desktop/Test/XmC.java"
0 errors, 0 warnings.

Generation finished successfully.
```

<u>Step 3:</u> *XmC.java generated.*



```java
  /** Initial size of the lookahead buffer. */
  private static final int ZZ_BUFFERSIZE = 16384;
  private static final String ZZ_NL = System.getProperty("line.separator");

  // Lexical states.
  public static final int YYINITIAL = 0;
  public static final int COMMENT = 2;

  /**
   * ZZ_LEXSTATE[l] is the state in the DFA for the lexical state l
   * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
   *                  at the beginning of a line
   * l is of the form l = 2*k, k a non negative integer
   */
  private static final int ZZ_LEXSTATE[] = {
     0,  0,  1, 1
  };

  /**
   * Top-level table for translating characters to character classes
   */
  private static final int [] ZZ_CMAP_TOP = zzUnpackcmap_top();

  private static final String ZZ_CMAP_TOP_PACKED_0 =
    "\1\0\37\u0100\1\u0200\267\u0100\10\u0300\u1020\u0100";

  private static int [] zzUnpackcmap_top() {
    int [] result = new int[4352];
    int offset = 0;
    offset = zzUnpackcmap_top(ZZ_CMAP_TOP_PACKED_0, offset, result);
    return result;
  }
```

Step 4: Compiling *XmC.java using terminal.*



```
CSC 3315 - Assignment 1 — -bash — 80×24
Last login: Sat Mar 19 15:58:26 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[MacBook-Pro-de-El:~ Ziad$ cd /Users/Ziad/Desktop/CSC\ 3315\ -\ Assignment\ 1    ]
[MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ javac XmC.java                  ]
 MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ 
```

III. Test cases:

_ In this step, we are going to create a file we called input.txt, which will be passed as an argument to the XmC.java scanner, to test our lexical analyzer. The input.txt file will be overwritten 3 times for test purposes.

/* *Text* is the token.

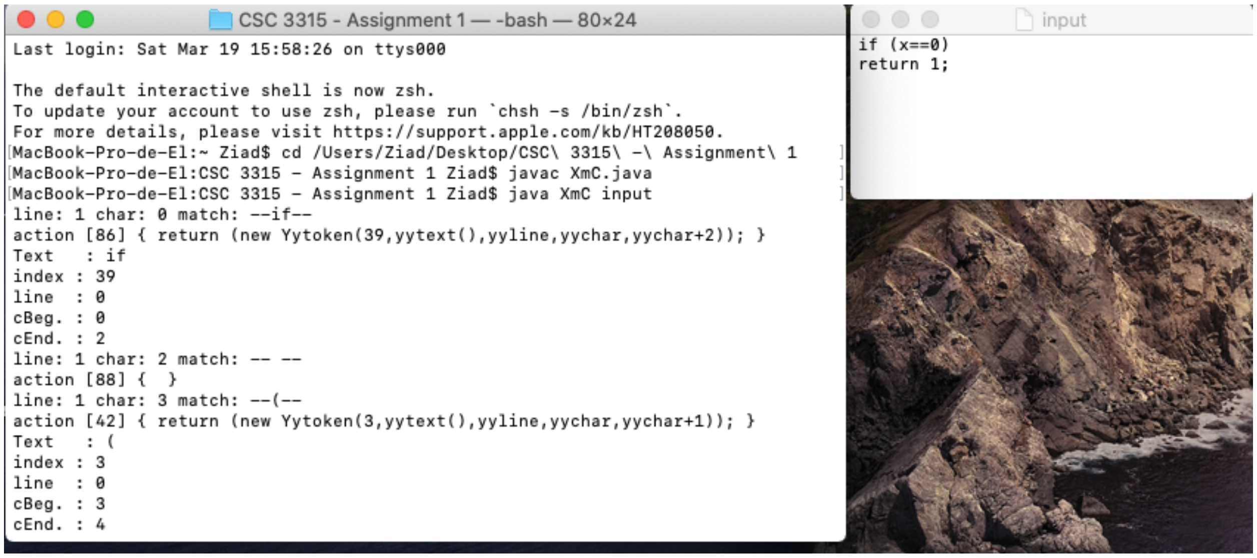
Index is the token number.

cBeg indicates the begging position of the token.

cEnd indicates the ending position of the token.

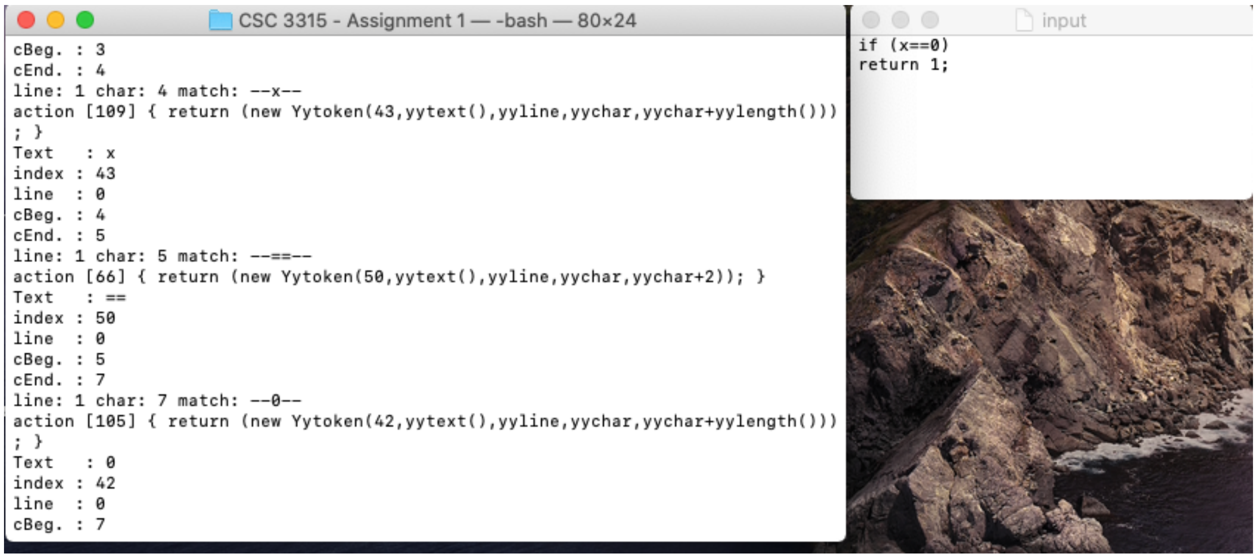
(cEnd – cBeg = length of the token) */

Case 1:



```
CSC 3315 - Assignment 1 — -bash — 80×24
Last login: Sat Mar 19 15:58:26 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[MacBook-Pro-de-El:~ Ziad$ cd /Users/Ziad/Desktop/CSC\ 3315\ -\ Assignment\ 1
[MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ javac XmC.java
[MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ java XmC input
line: 1 char: 0 match: --if--
action [86] { return (new Yytoken(39,yytext(),yyline,yychar,yychar+2)); }
Text   : if
index : 39
line   : 0
cBeg. : 0
cEnd. : 2
line: 1 char: 2 match: -- --
action [88] {  }
line: 1 char: 3 match: --(--
action [42] { return (new Yytoken(3,yytext(),yyline,yychar,yychar+1)); }
Text   : (
index : 3
line   : 0
cBeg. : 3
cEnd. : 4
```

```
input
if (x==0)
return 1;
```

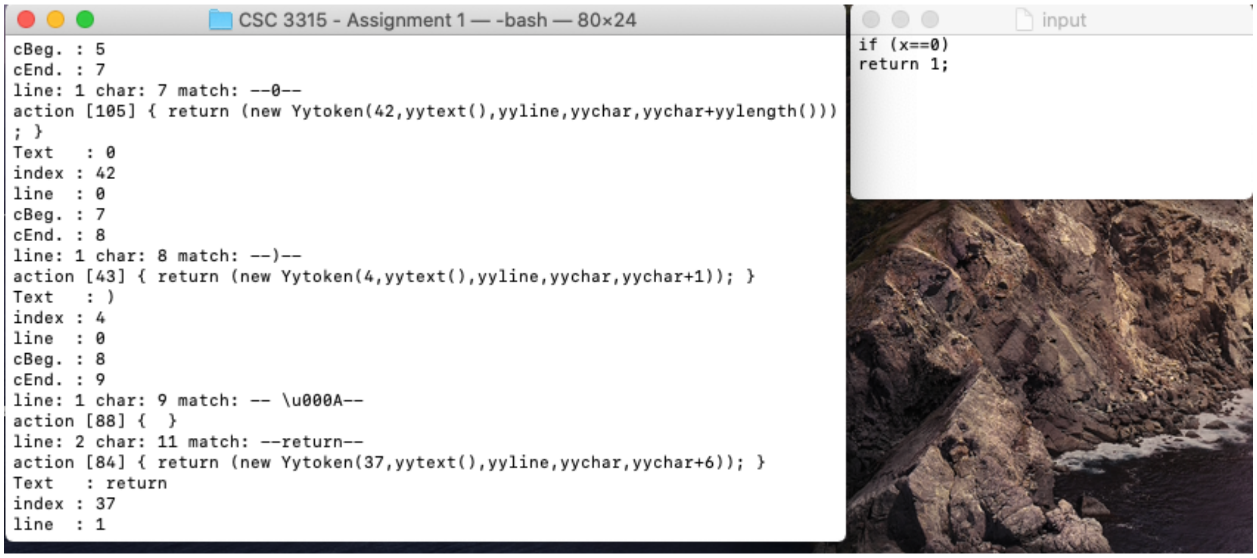


CSC 3315 - Assignment 1 — -bash — 80×24

```
cBeg. : 3
cEnd. : 4
line: 1 char: 4 match: --x--
action [109] { return (new Yytoken(43,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : x
index : 43
line  : 0
cBeg. : 4
cEnd. : 5
line: 1 char: 5 match: --==--
action [66] { return (new Yytoken(50,yytext(),yyline,yychar,yychar+2)); }
Text   : ==
index : 50
line  : 0
cBeg. : 5
cEnd. : 7
line: 1 char: 7 match: --0--
action [105] { return (new Yytoken(42,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : 0
index : 42
line  : 0
cBeg. : 7
```

input

```
if (x==0)
return 1;
```

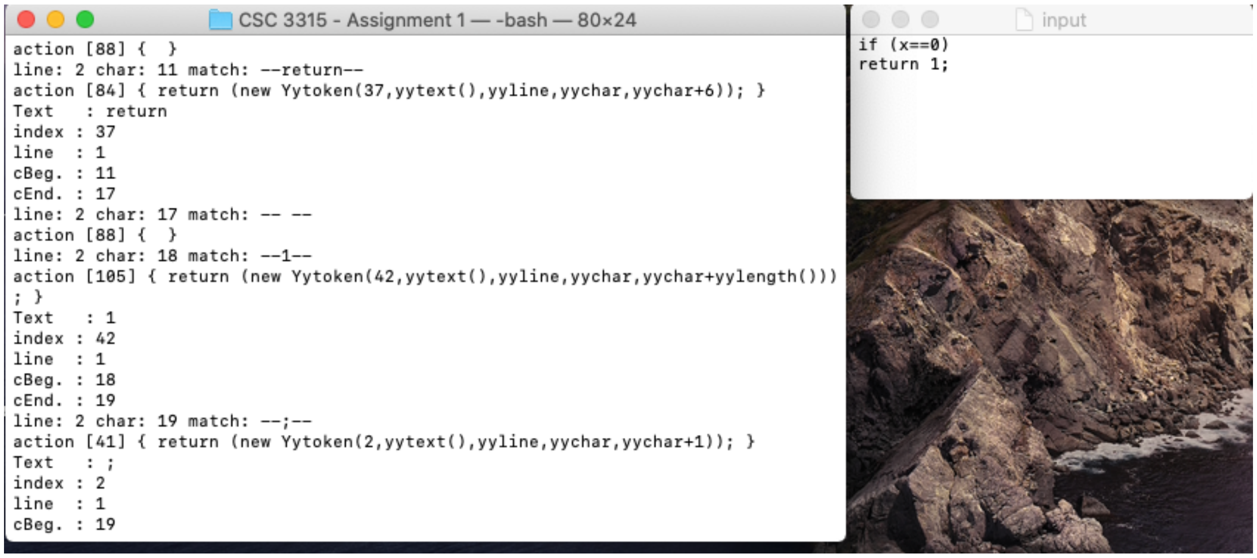


CSC 3315 - Assignment 1 — -bash — 80×24

```
cBeg. : 5
cEnd. : 7
line: 1 char: 7 match: --0--
action [105] { return (new Yytoken(42,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : 0
index : 42
line  : 0
cBeg. : 7
cEnd. : 8
line: 1 char: 8 match: --)--
action [43] { return (new Yytoken(4,yytext(),yyline,yychar,yychar+1)); }
Text   : )
index : 4
line  : 0
cBeg. : 8
cEnd. : 9
line: 1 char: 9 match: -- \u000A--
action [88] {  }
line: 2 char: 11 match: --return--
action [84] { return (new Yytoken(37,yytext(),yyline,yychar,yychar+6)); }
Text   : return
index : 37
line  : 1
```

input

```
if (x==0)
return 1;
```

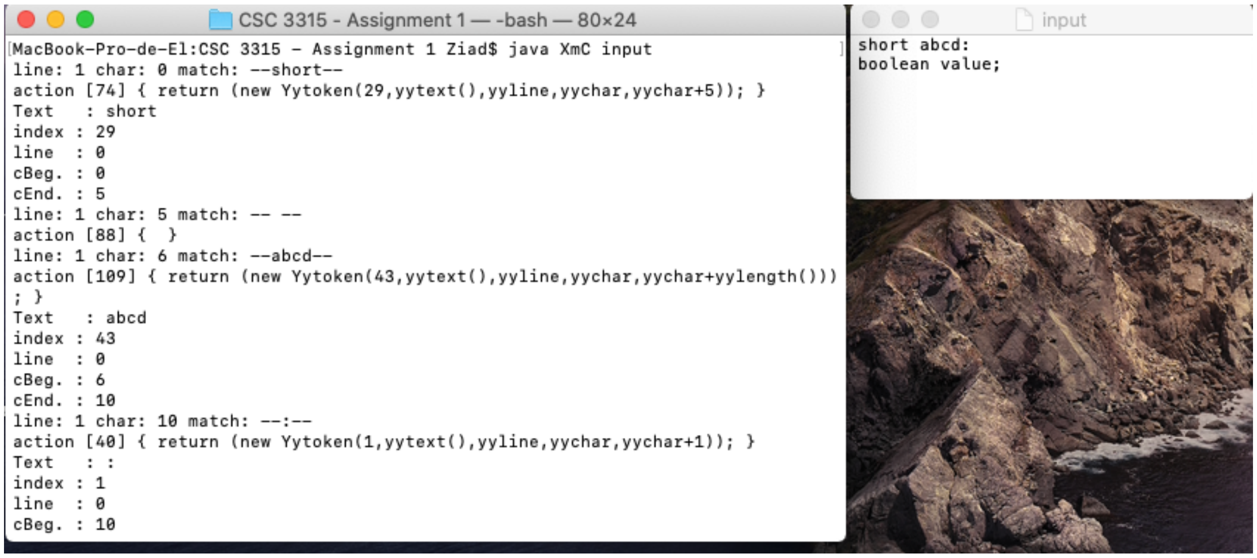

CSC 3315 - Assignment 1 — -bash — 80×24

```
action [88] {  }
line: 2 char: 11 match: --return--
action [84] { return (new Yytoken(37,yytext(),yyline,yychar,yychar+6)); }
Text   : return
index : 37
line  : 1
cBeg. : 11
cEnd. : 17
line: 2 char: 17 match: -- --
action [88] {  }
line: 2 char: 18 match: --1--
action [105] { return (new Yytoken(42,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : 1
index : 42
line  : 1
cBeg. : 18
cEnd. : 19
line: 2 char: 19 match: --;--
action [41] { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
Text   : ;
index : 2
line  : 1
cBeg. : 19
```

input

```
if (x==0)
return 1;
```

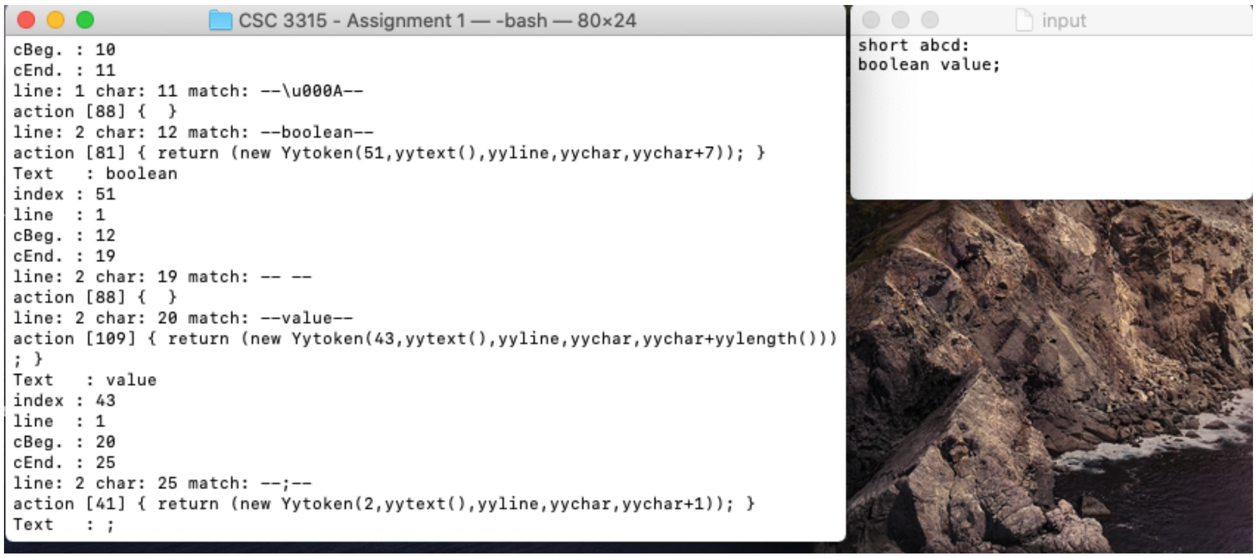
Case 2:


CSC 3315 - Assignment 1 — -bash — 80×24

```
[MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ java XmC input
line: 1 char: 0 match: --short--
action [74] { return (new Yytoken(29,yytext(),yyline,yychar,yychar+5)); }
Text   : short
index : 29
line  : 0
cBeg. : 0
cEnd. : 5
line: 1 char: 5 match: -- --
action [88] {  }
line: 1 char: 6 match: --abcd--
action [109] { return (new Yytoken(43,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : abcd
index : 43
line  : 0
cBeg. : 6
cEnd. : 10
line: 1 char: 10 match: --:--
action [40] { return (new Yytoken(1,yytext(),yyline,yychar,yychar+1)); }
Text   : :
index : 1
line  : 0
cBeg. : 10
```

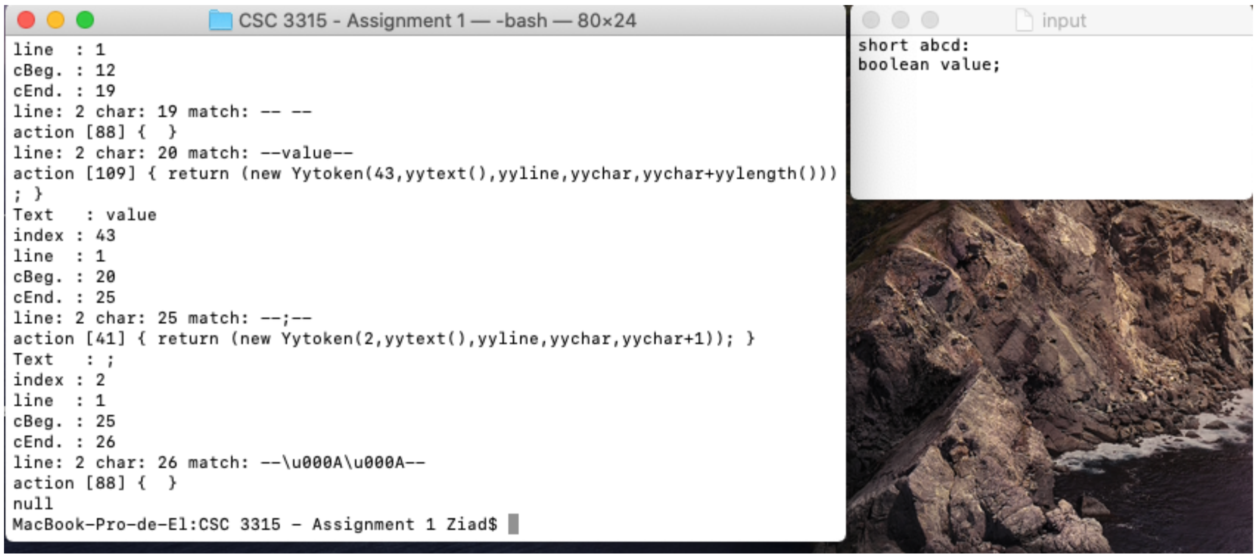
input

```
short abcd:
boolean value;
```



```
cBeg. : 10
cEnd. : 11
line: 1 char: 11 match: --\u000A--
action [88] {  }
line: 2 char: 12 match: --boolean--
action [81] { return (new Yytoken(51,yytext(),yyline,yychar,yychar+7)); }
Text   : boolean
index : 51
line  : 1
cBeg. : 12
cEnd. : 19
line: 2 char: 19 match: -- --
action [88] {  }
line: 2 char: 20 match: --value--
action [109] { return (new Yytoken(43,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : value
index : 43
line  : 1
cBeg. : 20
cEnd. : 25
line: 2 char: 25 match: --;--
action [41] { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
Text   : ;
```

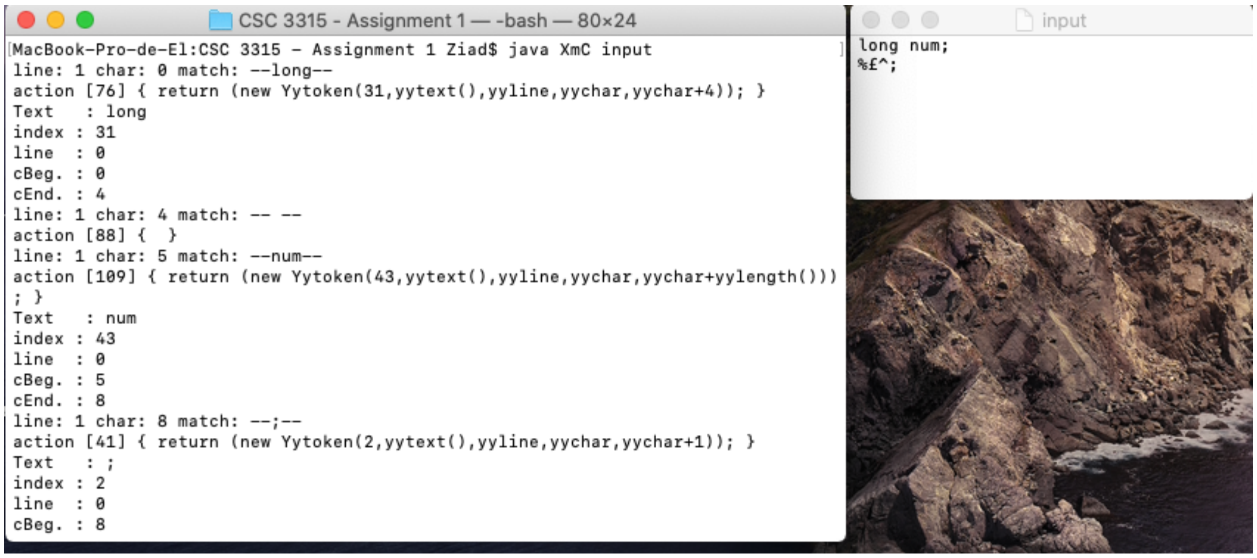
```
short abcd:
boolean value;
```



```
line  : 1
cBeg. : 12
cEnd. : 19
line: 2 char: 19 match: -- --
action [88] {  }
line: 2 char: 20 match: --value--
action [109] { return (new Yytoken(43,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : value
index : 43
line  : 1
cBeg. : 20
cEnd. : 25
line: 2 char: 25 match: --;--
action [41] { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
Text   : ;
index : 2
line  : 1
cBeg. : 25
cEnd. : 26
line: 2 char: 26 match: --\u000A\u000A--
action [88] {  }
null
MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$
```

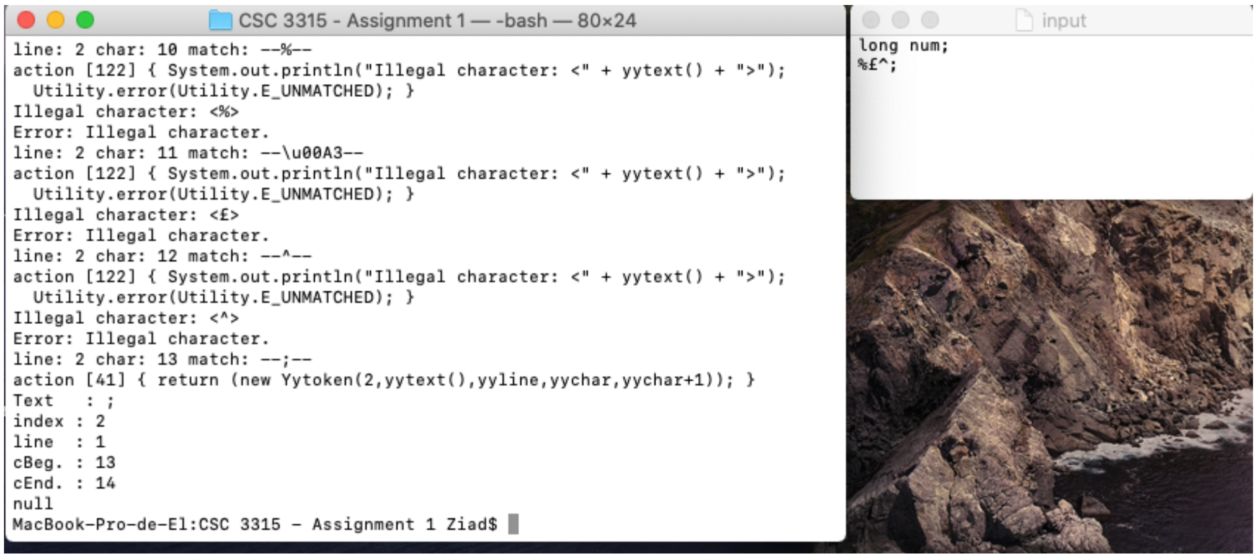
```
short abcd:
boolean value;
```

<u>Case 3:</u>



```
MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$ java XmC input
line: 1 char: 0 match: --long--
action [76] { return (new Yytoken(31,yytext(),yyline,yychar,yychar+4)); }
Text   : long
index : 31
line  : 0
cBeg. : 0
cEnd. : 4
line: 1 char: 4 match: -- --
action [88] {  }
line: 1 char: 5 match: --num--
action [109] { return (new Yytoken(43,yytext(),yyline,yychar,yychar+yylength()))
; }
Text   : num
index : 43
line  : 0
cBeg. : 5
cEnd. : 8
line: 1 char: 8 match: --;--
action [41] { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
Text   : ;
index : 2
line  : 0
cBeg. : 8
```

```
long num;
%£^;
```



```
line: 2 char: 10 match: --%--
action [122] { System.out.println("Illegal character: <" + yytext() + ">");
  Utility.error(Utility.E_UNMATCHED); }
Illegal character: <%>
Error: Illegal character.
line: 2 char: 11 match: --\u00A3--
action [122] { System.out.println("Illegal character: <" + yytext() + ">");
  Utility.error(Utility.E_UNMATCHED); }
Illegal character: <£>
Error: Illegal character.
line: 2 char: 12 match: --^--
action [122] { System.out.println("Illegal character: <" + yytext() + ">");
  Utility.error(Utility.E_UNMATCHED); }
Illegal character: <^>
Error: Illegal character.
line: 2 char: 13 match: --;--
action [41] { return (new Yytoken(2,yytext(),yyline,yychar,yychar+1)); }
Text   : ;
index : 2
line  : 1
cBeg. : 13
cEnd. : 14
null
MacBook-Pro-de-El:CSC 3315 - Assignment 1 Ziad$
```

```
long num;
%£^;
```

As you can see in case 3, since in our regular expressions we did not declare or define the characters %, £ and ^, the lexical analyzer therefore couldn't recognize them and gave and illegal character error message.

References:

Klein, G. (2022). JFlex - JFlex User's Manual. Retrieved 18 March 2022, from https://jflex.de/manual.html

Yytoken. (2022). Retrieved 17 March 2022, from https://ralleytn.github.io/SimpleJSON/de/ralleytn/simple/json/internal/Yytoken.html