Sulfur Language

Eliminates whitespace and shortens your code length

Adam Omundsen

Sample Code

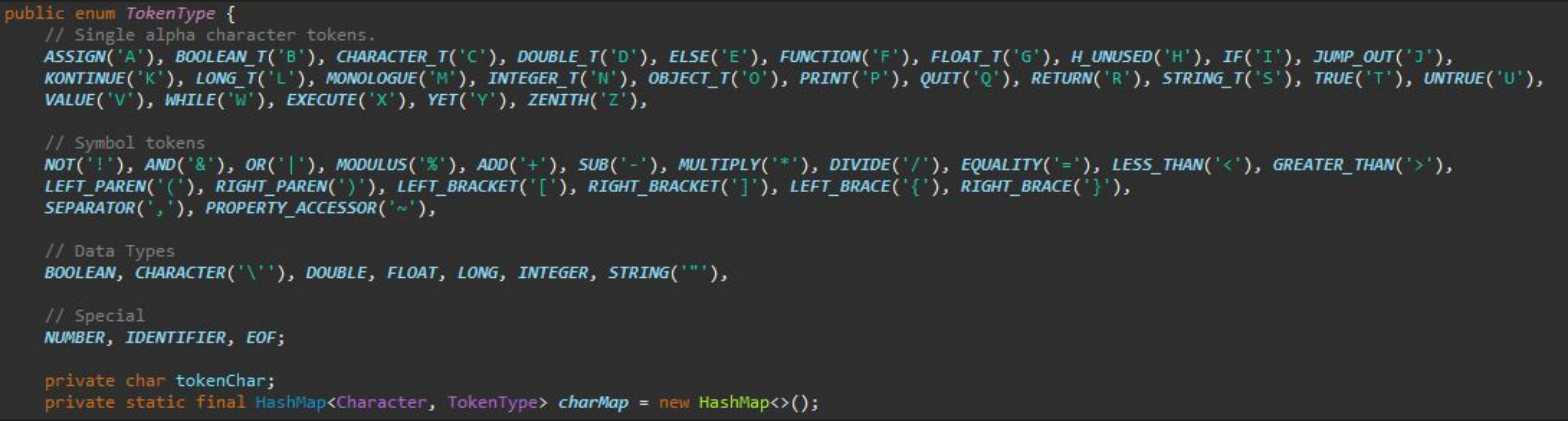
```
AnumNV2Wnum<100YAis_primeBVTAdivisorNV2Wdivisor<numYInum%divisor=2YAis_pr
imeBVUJZAdivisorNVdivisor+1ZIis_primeYP(num)P("\n")ZAnumNVnum+1Z
```

Same code with formatting —>

```
A num NV 2

W num<100 Y
A is_prime BV T
A divisor NV 2

W divisor<num Y

I num%divisor = 2 Y
A is_prime BV U
J
Z
A divisor NV divisor+1
Z

I is_prime Y
P(num)
P("\n")
Z
A num NV num+1
Z
```

Symbols used by the language

Each capital letter has its own meaning and usage. Symbols usage is relatively similar to Java. Variables must be lowercase and can contain underscores and numbers.



```java
public enum TokenType {
    // Single alpha character tokens.
    ASSIGN('A'), BOOLEAN_T('B'), CHARACTER_T('C'), DOUBLE_T('D'), ELSE('E'), FUNCTION('F'), FLOAT_T('G'), H_UNUSED('H'), IF('I'), JUMP_OUT('J'),
    KONTINUE('K'), LONG_T('L'), MONOLOGUE('M'), INTEGER_T('N'), OBJECT_T('O'), PRINT('P'), QUIT('Q'), RETURN('R'), STRING_T('S'), TRUE('T'), UNTRUE('U'),
    VALUE('V'), WHILE('W'), EXECUTE('X'), YET('Y'), ZENITH('Z'),

    // Symbol tokens
    NOT('!'), AND('&'), OR('|'), MODULUS('%'), ADD('+'), SUB('-'), MULTIPLY('*'), DIVIDE('/'), EQUALITY('='), LESS_THAN('<'), GREATER_THAN('>'),
    LEFT_PAREN('('), RIGHT_PAREN(')'), LEFT_BRACKET('['), RIGHT_BRACKET(']'), LEFT_BRACE('{'), RIGHT_BRACE('}'),
    SEPARATOR(','), PROPERTY_ACCESSOR('~'),

    // Data Types
    BOOLEAN, CHARACTER('\''), DOUBLE, FLOAT, LONG, INTEGER, STRING('"'),

    // Special
    NUMBER, IDENTIFIER, EOF;

    private char tokenChar;
    private static final HashMap<Character, TokenType> charMap = new HashMap<>();
```

Symbols used by the language (cont.)

Data Type Tokens:

B: Boolean
C: Character
D: Double
G: Float
L: Long
N: Integer
O: Object (WIP)
S: String
T: True / U: Untrue

Control Flow Tokens:

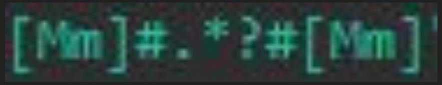
I: If
E: Else
W: While
J: Jump out (break)
K: Kontinue
F: Function
R: Return
Y: Yield ({)
Z Zenith (})
Q: Quit

Other Tokens:

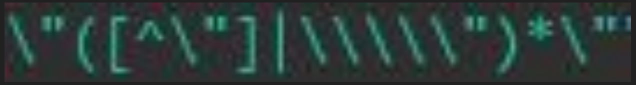
A Assign (=)
M Monologue (comment)
P Print
V Value (used with assignments)

Tools Used

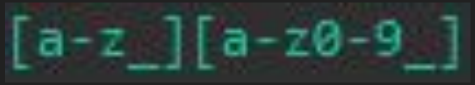
- Most of Sulfur’s tokens are only one character long and can be easily identified using a HashMap, therefore JFlex was not necessary and was not used, in part due to the extra complexity
- The Java regular expressions library was used to identify multi-character tokens like identifiers, strings, and numbers



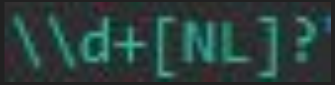
```
[Mm]#.*?#[Mm]
```



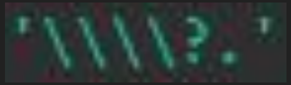
```
\"([^\"]|\\\\\\")*\"
```



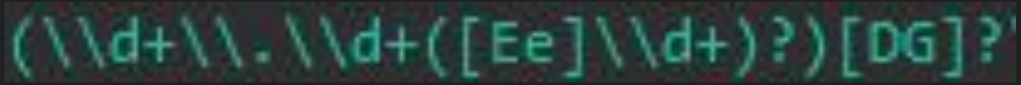
```
[a-z_][a-z0-9_]
```



```
\\d+[NL]?
```



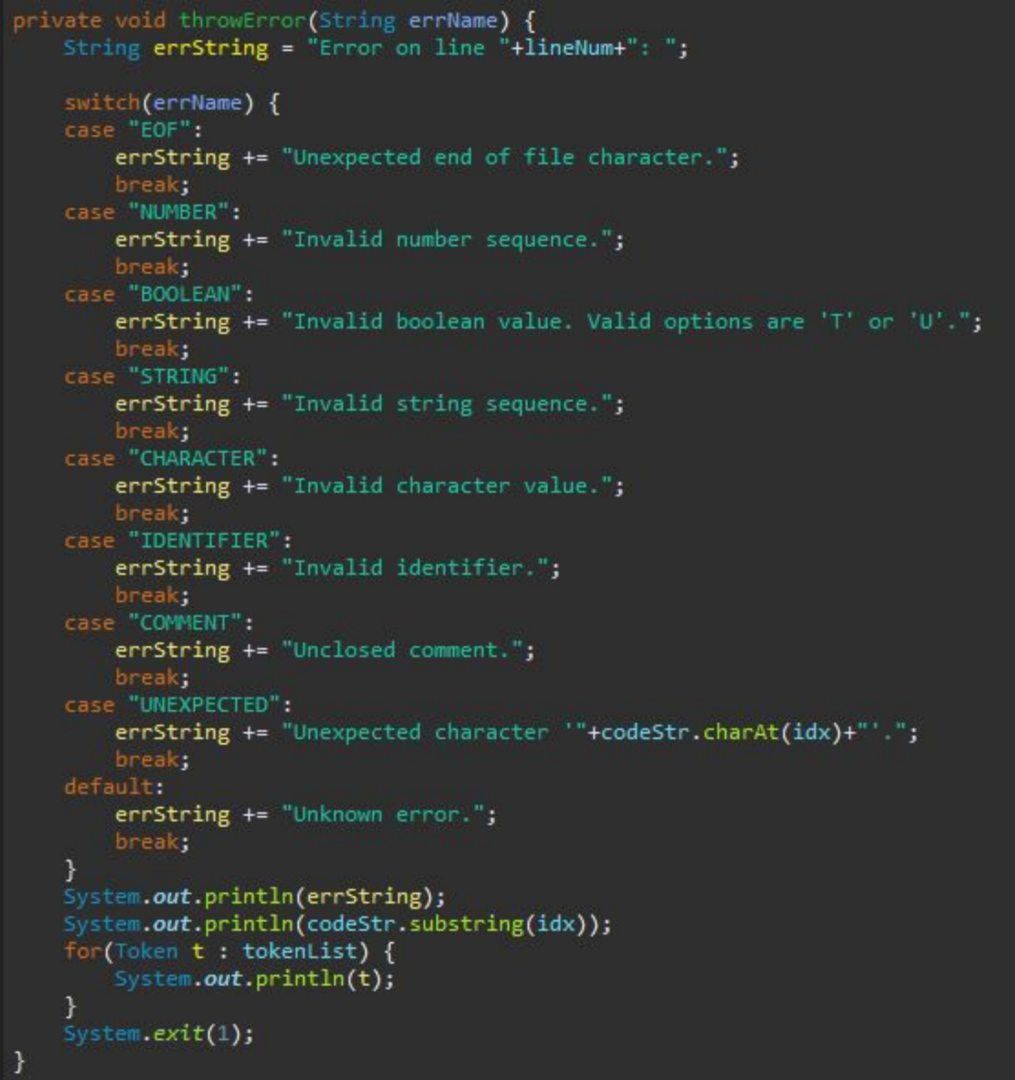
```
'\\\\?.'
```



```
(\\d+\\.\\d+([Ee]\\d+)?)[DG]?
```

Error Handling

Different parts of the lexing process can throw an error, which will be handled by a central error function



```java
private void throwError(String errName) {
    String errString = "Error on line "+lineNum+": ";

    switch(errName) {
    case "EOF":
        errString += "Unexpected end of file character.";
        break;
    case "NUMBER":
        errString += "Invalid number sequence.";
        break;
    case "BOOLEAN":
        errString += "Invalid boolean value. Valid options are 'T' or 'U'.";
        break;
    case "STRING":
        errString += "Invalid string sequence.";
        break;
    case "CHARACTER":
        errString += "Invalid character value.";
        break;
    case "IDENTIFIER":
        errString += "Invalid identifier.";
        break;
    case "COMMENT":
        errString += "Unclosed comment.";
        break;
    case "UNEXPECTED":
        errString += "Unexpected character '"+codeStr.charAt(idx)+"'.";
        break;
    default:
        errString += "Unknown error.";
        break;
    }
    System.out.println(errString);
    System.out.println(codeStr.substring(idx));
    for(Token t : tokenList) {
        System.out.println(t);
    }
    System.exit(1);
}
```

Factorial Example Program in Sulfur

```
M# Factorial function is recursive, takes an int as an argument and returns a long #M
A fac FLV (Nx) Y
  I x=0 | x=1 Y
    R 1
  ZEY
    R x*fac(x-1)
  Z
Z

P("Factorial of 7: ")
P(fac(7))
P('\n')
```

Questions?