```
/* You do not need to change anything up here. */
package lexer;

import frontend.Token;
import static frontend.Token.Type.*;

%%

%public
%final
%class Lexer
%function nextToken
%type Token
%unicode
%line
%column

%{
    /* These two methods are for the convenience of rules to create toke objects.
    * If you do not want to use them, delete them
    * otherwise add the code in
    */

    private Token token(Token.Type type) {
        return new Token(type, yyline, yycolumn, yytext());
    }

    /* Use this method for rules where you need to process yytext() to get the lexeme of the token.
     *
     * Useful for string literals; e.g., the quotes around the literal are part of yytext(),
     *       but they should not be part of the lexeme.
    */
    private Token token(Token.Type type, String text) {
        String substring = text.substring(1, text.length()-1);
        return new Token(type, yyline, yycolumn, substring);
    }
%}

/* This definition may come in handy. If you wish, you can add more definitions here. */
WhiteSpace = [ ] | \t | \f | \n | \r


%%
/* put in your rules here.    */

//keywords
```

```
47 "boolean" {return token(BOOLEAN);}
48 "break" {return token(BREAK);}
49 "else" {return token(ELSE);}
50 "false" {return token(FALSE);}
51 "if" {return token(IF);}
52 "import" {return token(IMPORT);}
53 "int" {return token(INT);}
54 "module" {return token(MODULE);}
55 "public" {return token(PUBLIC);}
56 "return" {return token(RETURN);}
57 "true" {return token(TRUE);}
58 "type" {return token(TYPE);}
59 "void" {return token(VOID);}
60 "while" {return token(WHILE);}
61
62
63 //punctuation
64 "," {return token(COMMA);}
65 "[" {return token(LBRACKET);}
66 "{" {return token(LCURLY);}
67 "(" {return token(LPAREN);}
68 "]" {return token(RBRACKET);}
69 "}" {return token(RCURLY);}
70 ")" {return token(RPAREN);}
71 ";" {return token(SEMICOLON);}
72
73 //operator
74 "/" {return token(DIV);}
75 "==" {return token(EQEQ);}
76 "=" {return token(EQL);}
77 ">=" {return token(GEQ);}
78 ">" {return token(GT);}
79 "<=" {return token(LEQ);}
80 "<" {return token(LT);}
81 "-" {return token(MINUS);}
82 "!=" {return token(NEQ);}
83 "+" {return token(PLUS);}
84 "*" {return token(TIMES);}
85
86 //identifier
87 [a-zA-Z_][a-zA-Z_0-9]* {return token(ID);}
88
89 //literals
90 [0-9]+ {return token(INT_LITERAL);}
91 \"(!(\"|\\n))*\" {return token(STRING_LITERAL, yytext());}
92
93 {WhiteSpace} {} //skip whitespace
94
95 /* You don't need to change anything below this line. */
96 .                           { throw new Error("unexpected character
```

```
 96   '" + yytext() + "'"); }
 97 <<EOF>>                     { return token(EOF); }
 98
```