MiniJava Syntax

Below we give the concrete syntax of MiniJava. We assume the reader knows what a context-free grammar is, but this syntax is not given as a pure cfg — it uses some well-known shortcuts to make it easier to read. (This is called "extended BNF.") They work like this:

- Any syntactic item, or parenthesized group of syntactic items, can be followed by a `+`, `*`, or `?`. This means:
  - If followed by `+`, that item or group of items can be repeated as many times as you like, but at least once.
  - If followed by `*`, that item or group of items can be repeated as many times as you like, including zero.
  - If followed by `?`, that item or group of items can be either included (once) or omitted; i.e. it is optional.
- A list of syntactic items or parenthesized group can be separated by vertical bars (`|`), meaning that exactly one item or group from the list should be used at that point.

So, for example, the first rule says that a program consists of any number of class declarations, but at least one. Or, to take a more complicated case, a switch statement consists of the word "switch" followed by an expression in parentheses and then an open bracket; then any number (including zero) of cases, each of which has the word "case", an integer literal, a colon, and a non-empty list of statements; then a default case consisting of the word "default", a colon, and a non-empty list of statements; and, finally, a closing bracket.

```
   Program ::= (ClassDecl)+
 ClassDecl ::= "class" <IDENTIFIER> ("extends" <IDENTIFIER>)?
               "{" (VarDecl)* (MethodDecl)* "}"
   VarDecl ::= Type <IDENTIFIER> ";"
             | "static" Type <IDENTIFIER> ";"
MethodDecl ::= "public" Type <IDENTIFIER>
               "(" (Type <IDENTIFIER> ("," Type <IDENTIFIER>)*)?  ")"
               "{" (VarDecl)* (Statement)* "return" Expression ";" "}"
      Type ::= Type "[" "]"
             | "boolean"
             | "String"
             | "float"
             | "int"
             | <IDENTIFIER>
 Statement ::= "{" ( Statement )* "}"
             | "if" "(" Expression ")" Statement "else" Statement
             | "if" "(" Expression ")" Statement
             | "while" "(" Expression ")" Statement
             | "System.out.println" "(" Expression ")" ";"
             | <IDENTIFIER> "=" Expression ";"
             | "break" ";"
             | "continue" ";"
             | <IDENTIFIER> "[" Expression "]" "=" Expression ";"
             | "switch" "(" Expression ")" "{"
               ("case" <INTEGER_LITERAL> ":" (Statement)+)*
               "default" ":" ( Statement )+ "}"
Expression ::= Expression ( "&" | "|" | "<" | "+" | "-" | "*" | "/" ) Expression
             | Expression "[" Expression "]"
             | Expression "." "length"
             | Expression "." <IDENTIFIER> "(" (Expression ("," Expression)*)?  ")"
             | <INTEGER_LITERAL>
             | <FLOAT_LITERAL>
             | <STRING_LITERAL>
             | "null"
             | "true"
             | "false"
             | <IDENTIFIER>
             | "this"
             | "new" Type "[" Expression "]"
             | "new" <IDENTIFIER> "(" ")"
             | "!" Expression
             | "(" Expression ")"
```