The miniJava Raw Grammar

The following context-free grammar specifies the miniJava language's syntax:

```
Program    -> {ClassDecl}

ClassDecl  -> "class" <ID> ["extends" <ID>] "{" {VarDecl} {MethodDecl} "}"

MethodDecl -> "public" ExtType <ID> "(" [Param {"," Param}] ")"
                "{" {VarDecl} {Stmt} "}"
           |  "public" "static" "void" "main" "(" "String" "[" "]" <ID> ")"
                "{" {VarDecl} {Stmt} "}"

Param      -> Type <ID>

VarDecl    -> Type <ID> ["=" InitExpr] ";"

ExtType    -> Type | "void"

Type       -> BasicType
           |  BasicType "[" "]"             // array type
           |  <ID>                          // object type

BasicType  -> "int" | "boolean"

Stmt  ->      "{" {Stmt} "}"                      // stmt block
           |  ExtId "(" [Args] ")" ";"            // call stmt
           |  Lvalue "=" InitExpr ";"             // assignment
           |  "if" "(" Expr ")" Stmt ["else" Stmt]
           |  "while" "(" Expr ")" Stmt
           |  "System" "." "out" "." "println"
                "(" [PrintArg] ")" ";"
           |  "return" [Expr] ";"

Args       -> Expr {"," Expr}

PrintArg   -> Expr | <STRLIT>

InitExpr   -> "new" BasicType "[" <INTLIT> "]"    // new array
           |  "new" <ID> "(" ")"                  // new object
           |  Expr

Expr       -> Expr BinOp Expr
           |  UnOp Expr
           |  "(" Expr ")"
           |  ExtId "(" [Args] ")"          // method call
           |  Lvalue
           |  Literal

Lvalue     -> ExtId "[" Expr "]"            // array element
           |  ExtId
```

```
ExtId       -> ["this" "."] <ID> {"." <ID>}  // object field or just ID

Literal     -> <INTLIT> | "true" | "false"

BinOp       -> "+"  | "-"  | "*" | "/"  | "&&" | "||"
            |  "==" | "!=" | "<" | "<=" | ">"  | ">="

UnOp        -> "-" | "!"
```

Footnotes

MiniJava is a strict subset of full Java. Any valid miniJava program can be compiled by Java compiler and executed. However, plenty of simplifications exist in miniJava. The following are some highlights.

- Variable declarations in a class scope must appear before all method declarations. Variable declarations in a method scope must appear before all statements.
- MiniJava has limited type support. It has only two basic types, integer and boolean; and arrays can only be defined over these two basic types.
- There is no support for array literals, nor object constructor with arguments. However, one can define static initial values for object fields.
- New array and new object expressions can only be used in declarations and assignment statements; they can not be embedded in other expressions.
- Strings can only appear as arguments to the print statement. A print statement can take only one expression or one string as argument; not a mix of the two.

This grammar is intended for people. It is not suitable for a compiler to use — it is ambiguous, has left-recursion, and has productions with common-prefix of unbounded length. Grammar transformations are needed before a parser can be built.

Operator Associativity and Precedence

The arithmetic and logical binary operators are all left-associative. The operators' precedence is defined by the following list (from high to low):

`new`, `()` | `[]`, ., method call | `-`, `!` | `*`, `/` | `+`, `-` | ==, `!=`, <, <=, >, >= | `&&` | `||`