MiniJava Language Reference Manual

MiniJava is a subset of Java. The meaning of a MiniJava program is given by its meaning as a Java program. Overloading is not allowed in MiniJava. The MiniJava statement `System.out.println( ...);` can only print integers. The MiniJava expression `e.length` only applies to expressions of type `int[]`.

LEXICAL ISSUES

Identifiers: An *identifier* is a sequence of letters, digits, and underscores, starting with a letter. Uppercase letters are distinguished from lowercase. In this appendix the symbol *id* stands for an identifier.

Integer literals: A sequence of decimal digits is an *integer constant* that denotes the corresponding integer value. In this appendix the symbol *INTEGER_LITERAL* stands for an integer constant.

Binary operators: A *binary operator* is one of

```
&& < + - *
```

In this appendix the symbol *op* stands for a binary operator.

Comments: A comment may appear between any two tokens. There are two forms of comments: One starts with `/*`, ends with `*/`, and may be nested; another begins with `//` and goes to the end of the line.

GRAMMAR

In the MiniJava grammar, we use the notation N^*, where N is a nonterminal, to mean 0, 1, or more repetitions of N.

Program → *MainClass ClassDecl**

MainClass → **class** *id* { **public static void main** (**String** [ ] *id*)
{ *Statement* } }

ClassDecl → **class** *id* { *VarDecl* MethodDecl** }
→ **class** *id* **extends** *id* { *VarDecl* MethodDecl** }

VarDecl → *Type id* ;

MethodDecl → **public** *Type id* (*FormalList*)
{ *VarDecl* Statement** **return** *Exp* ; }

FormalList → *Type id FormalRest**
→

FormalRest → , *Type id*

Type → **int** [ ]
→ **boolean**
→ **int**
→ *id*

Statement → { *Statement** }
→ **if** (*Exp*) *Statement* **else** *Statement*
→ **while** (*Exp*) *Statement*
→ **System.out.println** (*Exp*) ;
→ *id* = *Exp* ;
→ *id* [*Exp*] = *Exp* ;

Exp → *Exp op Exp*
→ *Exp* [*Exp*]
→ *Exp* . **length**
→ *Exp* . *id* (*ExpList*)
→ *INTEGER_LITERAL*
→ **true**
→ **false**
→ *id*
→ **this**
→ **new int** [*Exp*]
→ **new** *id* ( )
→ ! *Exp*
→ (*Exp*)

ExpList → *Exp ExpRest**
→

ExpRest → , *Exp*