

Appendix A

Decaf Grammar

In this appendix, we give a description and grammar of the source language that we call “Decaf.” *Decaf* is a simple subset of the standard Java language. It does not include arrays, structs, unions, files, sets, switch statements, do statements, class definitions, methods, or many of the low level operators. The only data types permitted are int and float. A complete grammar for Decaf is shown below, and it is similar to the SableCC grammar used in the compiler in Appendix B.2. Here we use the convention that symbols beginning with upper-case letters are nonterminals, and all other symbols are terminals (i.e., lexical tokens). As in BNF, we use the vertical bar | to indicate alternate definitions for a nonterminal.

```
Program      →      class identifier { public static
                    void main (String[] identifier)
                    CompoundStmt }
Declaration  →      Type IdentList ;
Type         →      int
                    | float
IdentList    →      identifier , IdentList
                    | identifier
Stmt         →      AssignStmt
                    | ForStmt
                    | WhileStmt
                    | IfStmt
                    | CompoundStmt
                    | Declaration
                    | NullStmt
AssignStmt   →      AssignExpr ;
```

```

ForStmt      →   for ( OptAssignExpr; OptBoolExpr ;
                  OptAssignExpr ) Stmt
OptAssignExpr →   AssignExpr
                |   ε
OptBoolExpr  →   BoolExpr
                |   ε
WhileStmt    →   while ( BoolExpr ) Stmt
IfStmt       →   if ( BoolExpr ) Stmt ElsePart
ElsePart     →   else Stmt
                |   ε
CompoundStmt →   { StmtList }
StmtList    →   StmtList Stmt
                |   ε
NullStmt     →   ;
BoolExpr     →   Expr Compare Expr
Compare      →   == | < | > | <= | >= | !=
Expr         →   AssignExpr
                | Rvalue
AssignExpr   →   identifier = Expr
Rvalue       →   Rvalue + Term
                | Rvalue - Term
                | Term
Term         →   Term * Factor
                | Term / Factor
                | Factor
Factor       →   ( Expr )
                | - Factor
                | + Factor
                | identifier
                | number

```

This grammar is used in Appendix B as a starting point for the SableCC grammar for our Decaf compiler, with some modifications. It is not unusual for a compiler writer to make changes to the given grammar (which is descriptive of the source language) to obtain an equivalent grammar which is more amenable for parsing.

Decaf is clearly a very limited programming language, yet despite its limitations it can be used to program some useful applications. For example, a Decaf program to compute the cosine function is shown in Figure A.1.

```
class AClass
{

public static void main (String[] args)
{float cos, x, n, term, eps, alt;
// compute the cosine of x to within tolerance eps
// use an alternating series
  x = 3.14159;
  eps = 0.1;
  n = 1;
  cos = 1;
  term = 1;
  alt = -1;
  while (term>eps)
  {
    term = term * x * x / n / (n+1);
    cos = cos + alt * term;
    alt = -alt;
    n = n + 2;
  }
}
```

Figure A.1 A Decaf Program to Compute the Cosine Function