Clifford Black

David Carlin

Christopher Houze

Program 6

**Below is an updated grammar for our language. The changes made to the language are shown in blue. For this assignment we had to make a few changes to our grammar. We needed to add { } around statement in our Statement → {Statement\*} because we were having a shift reduce conflict. We also had to add an “end if” label to our If statement because we had another shift reduce conflict. We also had to rework our Exp completely to allow for order of operations. We only had two difficulties with SableCC and both of them involved changing our grammar to remove the shift reduce conflicts. Once that was taken care of everything else went smoothly.**

Program → MainClass ClassDecl∗

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

ClassDecl → ClassDeclSpec ClassDeclDeff

ClassDeclSpec → class id

ClassDeclDeff → { VarDecl∗ MethodDecl\* }

→ **extends** id { VarDecl\* MethodDecl\* }

VarDecl → Type VarDeclType MultiDecl\*;

VarDeclType → id VarDeclTypeAssign

VarDeclTypeAssign → = Exp

→

MultiDecl → , id VarDeclTypeAssign

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

FormalList → Type id FormalRest\*

→

FormalRest → , Type id

Type → **int** IntType

→ **boolean**

→ id

IntType → []

→

Statement → { Statement\* }

→ **if** ( Exp ) { Statement } ElseIf**\*** end if

→ **do** { Statement } **while** (Exp);

→ **while** ( Exp ) Statement

→ **for**(InitializationStm; Exp; IncrementStm) Statement

→ **switch**(id){ CaseList }

→ System.out.println ( Exp? ) ;

→ id Assign ;

→ ( id Assign ) FormalVarExp\*;

Assign → = Exp

→ [ Exp ] = Exp

InitializationStm → Type id = Exp

→ id = Exp

→id [Exp] = Exp

IncrementStm → id = Exp

→id[Exp] = Exp

ElseIf → **else if** (Exp) Statement

FormalVarExp → ,(Type id Assign)

CaseList → **case** Exp : Statement CaseList

→ **default**: Statement

Exp → And Elist  
Elist → && And Elist  
 →   
And → Less AList  
AList → < Less AList  
 →   
Less → Term LList  
LList → + Term LList  
 → - Term LList  
 →   
Term → Not TList  
TList → \* Not TList  
 →   
Not → ! Not  
 → Factor DotArray\*  
DotArray → .Member  
 → [Exp]  
Member → length  
 → id (ExpList)  
ExpList → Exp ExpRest\*  
 →   
ExpRest → , Exp  
Factor → num  
 → true  
 → false  
 → id  
 → this  
 → New  
 → (Exp)  
New → int [ Exp ]  
 → id()

Submit the following deliverables on BlackBoard (include the names of your team members in the text submission area):

* Updated grammar for your language.
* SableCC input (a .grammar file)
* Sample programs which were used as test cases.
* Output, if any, for each sample program.
* Description of any changes which you have made to your language for this assignment.
* Description of difficulties encountered (such as conflicts from SableCC) and how you dealt with those issues.