

Design Decisions and Evaluation of Parser

Maryam Dabaghchian-u1078006

February 1, 2018

1 Grammar

$Exp \rightarrow LtLevel$
 $Exp \rightarrow Exp \text{ "&\&" } LtLevel$
 $LtLevel \rightarrow PlusLevel$
 $LtLevel \rightarrow LtLevel \text{ "<" } PlusLevel$
 $PlusLevel \rightarrow MultLevel$
 $PlusLevel \rightarrow PlusLevel \text{ "(" "+" "-" ")" } MultLevel$
 $MultLevel \rightarrow DotLevel$
 $MultLevel \rightarrow MultLevel \text{ "*" } DotLevel$
 $DotLevel \rightarrow PrimaryExp \text{ "(" "[" } Exp \text{ "]" | "." "length" | "." } Id \text{ "(" (} Exp \text{ ("," } Exp \text{)}^* \text{)}^*$
 $DotLevel \rightarrow \text{"!" } DotLevel$
 $PrimaryExp \rightarrow \text{"true" | "false" | "this" | "new" "int" "[" } Exp \text{ "]" | "new" } Id \text{ "(" ")" | "<"} \\ NUM > \text{ | "(" } Exp \text{ ")"}$

1.1 Eliminate Left Recursion

$Goal \rightarrow MainClass \text{ (} ClassDecl \text{)}^* < EOF >$
 $MainClass \rightarrow \text{"class" } Id \text{ "{" "public" "static" "void" "main" "(" "String" "[" "]" } Id \text{ ")" " {" } Stmt \text{ "}" "}"$
 $ClassDecl \rightarrow \text{"class" } Id \text{ ("extends" } Id \text{)? " {" (} VarDecl \text{)}^* \text{ (} MethodDecl \text{)}^* \text{ "}"}$
 $VarDecl \rightarrow Type \text{ } Id \text{ ";"}$
 $MethodDecl \rightarrow \text{"public" } Type \text{ } Id \text{ "(" (} Type \text{ } Id \text{ ("," } Type \text{ } Id \text{)}^* \text{)? ")" " {" (} VarDecl \text{)}^* \text{ (} Stmt \text{)}^* \text{ "return" } Exp \text{ ";" "}"$

 $Type \rightarrow \text{"int" "[" "]" | "boolean" | "int" | } Id$

 $Stmt \rightarrow \text{" {" (} Stmt \text{)}^* \text{ "}" |$
 $\text{"if" "(" } Exp \text{ ")" } Stmt \text{ "else" } Stmt \text{ |$
 $\text{"while" "(" } Exp \text{ ")" } Stmt \text{ |$
 $\text{"System.out.println" "(" } Exp \text{ ")" ";" |$
 $Id \text{ " = " } Exp \text{ ";" |$
 $Id \text{ "[" } Exp \text{ "]" " = " } Exp \text{ ";" |$

 $Exp \rightarrow LtLevel \text{ } Exp'$
 $Exp' \rightarrow \text{"&\&" } LtLevel \text{ } Exp'$
 $Exp' \rightarrow$

$LtLevel \rightarrow PlusLevel LtLevel'$
 $LtLevel' \rightarrow "<" PlusLevel LtLevel'$
 $LtLevel' \rightarrow$
 $PlusLevel \rightarrow MultLevel PlusLevel'$
 $PlusLevel' \rightarrow (" + "|" -) MultLevel PlusLevel'$
 $PlusLevel' \rightarrow$
 $MultLevel \rightarrow DotLevel MultLevel'$
 $MultLevel' \rightarrow "*" DotLevel MultLevel'$
 $MultLevel' \rightarrow$
 $DotLevel \rightarrow PrimaryExp ("[" Exp "]" | "." "length" | "." Id "(" (Exp ("," Exp)^*)? ")")^*$
 $DotLevel \rightarrow "!" DotLevel$
 $PrimaryExp \rightarrow "true" | "false" | "this" | "new" "int" "[" Exp "]" | "new" Id "(" ")" | <$
 $NUM > | "(" Exp ")"$

1.2 Left Factoring

$Goal \rightarrow "class" Id MainClass ("class" Id RegClass)^* < EOF >$
 $MainClass \rightarrow "{" "public" "static" "void" "main" "(" "String" "[" "]" Id ")" "{" Stmt "}" "}"$
 $RegClass \rightarrow ("extends" Id)? "{" (VarDecl)^* (MethodDecl)^* "}"$
 $VarDecl \rightarrow Type Id ";"$
 $MethodDecl \rightarrow "public" Type Id "(" (Type Id ("," Type Id)^*)? ")" "{" (VarDecl)^* (Stmt)^* "return" Exp ";" "}"$

 $Type \rightarrow "int" "[" "]" | "boolean" | "int" | Id$

 $Stmt \rightarrow "{" (Stmt)^* "}" |$
 $"if" "(" Exp ")" Stmt ElseStmt |$
 $"while" "(" Exp ")" Stmt |$
 $"System.out.println" "(" Exp ")" ";" |$
 $Id AssignStmt$
 $ElseStmt \rightarrow ("else" Stmt)?$
 $AssignStmt \rightarrow "=" Exp ";" |$
 $"[" Exp "]" "=" Exp ";"$

 $Exp \rightarrow LtLevel Exp'$
 $Exp' \rightarrow "&\&" LtLevel Exp'$
 $Exp' \rightarrow$
 $LtLevel \rightarrow PlusLevel LtLevel'$
 $LtLevel' \rightarrow "<" PlusLevel LtLevel'$
 $LtLevel' \rightarrow$
 $PlusLevel \rightarrow MultLevel PlusLevel'$
 $PlusLevel' \rightarrow (" + "|" -) MultLevel PlusLevel'$
 $PlusLevel' \rightarrow$
 $MultLevel \rightarrow DotLevel MultLevel'$
 $MultLevel' \rightarrow "*" DotLevel MultLevel'$
 $MultLevel' \rightarrow$
 $DotLevel \rightarrow PrimaryExp ("[" Exp "]" | "." "length" | "." Id "(" (Exp ("," Exp)^*)? ")")^*$

$DotLevel \rightarrow "!" \ DotLevel$
 $PrimaryExp \rightarrow "true" \mid "false" \mid "this" \mid "new" \ NewExp \mid < NUM > \mid "(" \ Exp \ ")"$
 $NewExp \rightarrow "int" \ "[" \ Exp \ "]" \mid Id \ "(" \ ")"$

2 Design Decisions