INDIAN INSTITUTE OF TECHNOLOGY, CSE DEPARTMENT

VEDANT SABOO CS19B074

CS3300 LAB 1: MacroJava to MiniJava Compiler

MacroJava specification

```
                Goal ::= (MacroDefinition)* MainClass ( TypeDeclaration )* <EOF>
           MainClass ::= class Identifier { public static void main ( String [] Identifier )
                         { System.out.println ( Expression ); } }
     TypeDeclaration ::= class Identifier { ( Type Identifier ;)* ( MethodDeclaration )* }
                       | class Identifier extends Identifier { ( Type Identifier;)*
                         ( MethodDeclaration )* }
  MethodDeclaration ::= public Type Identifier ( ( Type Identifier (, Type Identifier)*)? )
                         { ( Type Identifier ;)* ( Statement )* return Expression ; }
               Type ::= int [ ]
                       | boolean
                       | int
                       | Identifier
          Statement ::= { ( Statement )* }
                       | System.out.println ( Expression );
                       | Identifier = Expression ;
                       | Identifier [ Expression ] = Expression ;
                       | if ( Expression ) Statement
                       | if ( Expression ) Statement else Statement
                       | while ( Expression ) Statement
                       | Identifier ( (Expression (, Expression )*)?); /* Macro stmt call */
         Expression ::= PrimaryExpression && PrimaryExpression
                       | PrimaryExpression || PrimaryExpression
                       | PrimaryExpression != PrimaryExpression
                       | PrimaryExpression <= PrimaryExpression
                       | PrimaryExpression + PrimaryExpression
```

```
                    | PrimaryExpression - PrimaryExpression
                    | PrimaryExpression * PrimaryExpression
                    | PrimaryExpression / PrimaryExpression
                    | PrimaryExpression [ PrimaryExpression ]
                    | PrimaryExpression . length
                    | PrimaryExpression
                    | PrimaryExpression . Identifier ( (Expression (, Expression )*)? )
                    | Identifier ( (Expression (, Expression )*)? )/* Macro expr call */
PrimaryExpression ::= Integer
                    | true
                    | false
                    | Identifier
                    | this
                    | new int [ Expression ]
                    | new Identifier ( )
                    | ! Expression
                    | ( Expression )
MacroDefinition ::= MacroDefExpression
                    | MacroDefStatement
MacroDefStatement ::= #defineStmt Identifier (Identifier , Identifier, Identifier (, Identifier )*?
                      ) { ( Statement )* }/* More than 2 arguments */
                    | #defineStmt0 Identifier () { ( Statement )* }
                    | #defineStmt1 Identifier ( Identifier ) { ( Statement )* }
                    | #defineStmt2 Identifier (Identifier , Identifier ) { ( Statement )* }
MacroDefExpression ::= #defineExpr Identifier (Identifier , Identifier, Identifier (, Identifier )*?
                       ) ( Expression ) /* More than 2 arguments */
                    | #defineExpr0 Identifier () ( Expression )
                    | #defineExpr1 Identifier ( Identifier ) ( Expression )
                    | #defineExpr2 Identifier (Identifier , Identifier ) ( Expression )
    Identifier ::= <IDENTIFIER>
       Integer ::= <INTEGER_LITERAL>
```

MiniJava Specification

```
                    Goal ::= MainClass ( TypeDeclaration )* <EOF>
               MainClass ::= "class" Identifier "{" "public" "static" "void" "main" "(" "String"
                             "[" "]" Identifier ")" "{" PrintStatement "}" "}"
         TypeDeclaration ::= ClassDeclaration
                           | ClassExtendsDeclaration
        ClassDeclaration ::= "class" Identifier "{" ( VarDeclaration )* ( MethodDeclaration )*
                             "}"
ClassExtendsDeclaration ::= "class" Identifier "extends" Identifier "{" ( VarDeclaration )*
                             ( MethodDeclaration )* "}"
         VarDeclaration ::= Type Identifier ";"
      MethodDeclaration ::= "public" Type Identifier "(" ( FormalParameterList )? ")" "{"
                             ( VarDeclaration )* ( Statement )* "return" Expression ";" "}"
    FormalParameterList ::= FormalParameter ( FormalParameterRest )*
      FormalParameter ::= Type Identifier
    FormalParameterRest ::= "," FormalParameter
                   Type ::= ArrayType
                          | BooleanType
                          | IntegerType
                          | Identifier
              ArrayType ::= "int" "[" "]"
            BooleanType ::= "boolean"
            IntegerType ::= "int"
              Statement ::= Block
                          | AssignmentStatement
                          | ArrayAssignmentStatement
                          | IfStatement
                          | WhileStatement
                          | PrintStatement
                  Block ::= "{" ( Statement )* "}"
     AssignmentStatement ::= Identifier "=" Expression ";"
ArrayAssignmentStatement ::= Identifier "[" Expression "]" "=" Expression ";"
            IfStatement ::= IfthenElseStatement
                          | IfthenStatement
        IfthenStatement ::= "if" "(" Expression ")" Statement
    IfthenElseStatement ::= "if" "(" Expression ")" Statement "else" Statement
       WhileStatement ::= "while" "(" Expression ")" Statement
       PrintStatement ::= "System.out.println" "(" Expression ")" ";"
```

```
                Expression ::= OrExpression
                             | AndExpression
                             | CompareExpression
                             | neqExpression
                             | PlusExpression
                             | MinusExpression
                             | TimesExpression
                             | DivExpression
                             | ArrayLookup
                             | ArrayLength
                             | MessageSend
                             | PrimaryExpression
             AndExpression ::= PrimaryExpression "&&" PrimaryExpression
              OrExpression ::= PrimaryExpression "||" PrimaryExpression
         CompareExpression ::= PrimaryExpression "<=" PrimaryExpression
             neqExpression ::= PrimaryExpression "!=" PrimaryExpression
            PlusExpression ::= PrimaryExpression "+" PrimaryExpression
           MinusExpression ::= PrimaryExpression "-" PrimaryExpression
           TimesExpression ::= PrimaryExpression "*" PrimaryExpression
             DivExpression ::= PrimaryExpression "/" PrimaryExpression
              ArrayLookup ::= PrimaryExpression "[" PrimaryExpression "]"
             ArrayLength ::= PrimaryExpression "." "length"
             MessageSend ::= PrimaryExpression "." Identifier "(" ( ExpressionList )? ")"
          ExpressionList ::= Expression ( ExpressionRest )*
          ExpressionRest ::= "," Expression
       PrimaryExpression ::= IntegerLiteral
                           | TrueLiteral
                           | FalseLiteral
                           | Identifier
                           | ThisExpression
                           | ArrayAllocationExpression
                           | AllocationExpression
                           | NotExpression
                           | BracketExpression
          IntegerLiteral ::= <INTEGER_LITERAL>
             TrueLiteral ::= "true"
            FalseLiteral ::= "false"
              Identifier ::= <IDENTIFIER>
          ThisExpression ::= "this"
ArrayAllocationExpression ::= "new" "int" "[" Expression "]"
     AllocationExpression ::= "new" Identifier "(" ")"
           NotExpression ::= "!" Expression
```

BracketExpression ::= "(" Expression ")"

IdentifierList ::= Identifier (IdentifierRest)*

IdentifierRest ::= "," Identifier