```
CATEGORIES
expression;
stmt;
dataType;

NODES

program -> classDef global? create feature* runInvocation;

classDef -> name:string;

runInvocation -> procedure;

readStmt:stmt -> expression*;
printStmt:stmt -> expression* format:string;
assignStmt:stmt -> assignment;
ifStmt:stmt -> condition:expression ifStmts:stmt* elseStmts:stmt*;
fromStmt:stmt -> declarations:assignment* condition:expression stmts:stmt*;
procedureStmt:stmt -> procedure;
returnStmt:stmt -> returnInvoc;

assignment -> left:expression right:expression;

intLiteral:expression -> value:string;
realLiteral:expression -> value:string;
charLiteral:expression -> value:string;
variable:expression -> name:string;
procedureExpression:expression -> procedure;
arrayExpression:expression -> array:expression index:expression;
structExpression:expression -> struct:expression field:string;
minusExpression:expression -> expression;
notExpression:expression -> expression;
cast:expression -> dataType expression;
arithmeticExpression:expression -> left:expression operator:string right:expression;
comparisonExpression:expression -> left:expression operator:string right:expression;
logicExpression:expression -> left:expression operator:string right:expression;

procedure -> name:string expression*;

integerType:dataType -> ;
doubleType:dataType -> ;
characterType:dataType -> ;
structType:dataType -> name:string;
arrayType:dataType -> size:string dataType;
voidType:dataType -> ;
errorType:dataType -> ;

create -> idents:string* ;
feature -> name:string params:varDefinition* dataType? localBlock? doBlock;
returnInvoc -> expression?;
localBlock -> varDefinition*;
doBlock -> stmt*;

global -> globalTypes? varsTypes?;
globalTypes -> deftuple*;
varsTypes -> varDefinition*;

deftuple -> name:string field*;
field -> name:string type:dataType;

varDefinition -> name:string type:dataType;

// --------------------------------
ATTRIBUTE GRAMMAR Identification

variable -> definition:varDefinition ;
structType -> deftuple;
procedure -> invocation:feature;
field -> deftuple;
varDefinition -> scope:string;

// --------------------------------
ATTRIBUTE GRAMMAR TypeChecking
```

```
74
75  expression -> type:dataType;
76  expression -> lvalue:boolean;
77  stmt -> feature;
78  stmt -> returnable:boolean;
79  doBlock -> returnable:boolean;
80  feature -> returntype:dataType;
81  feature -> constructor:boolean;
82
83  // --------------------------------
84  ATTRIBUTE GRAMMAR MemoryAllocation
85
86  varDefinition -> [inh] address:int;
87  field -> offset:int;
88
89  // --------------------------------
90  CODE SPECIFICATION Mapl
91
92  run[program]
93
94  metadata[program]
95  metadata[global]
96  metadata[globalTypes]
97  metadata[varsTypes]
98  metadata[deftuple]
99  metadata[field]
100 metadata[varDefinition]
101
102 execute[runInvocation]
103 execute[assignment]
104 execute[feature]
105 execute[returnInvoc]
106 execute[doBlock]
107 execute[stmt]
108
109 value[expression]
110
111 address[expression]
112
113
```