CS 371 Compiler Design
Stage 2: Typechecking I
Aashish Karki
February 24, 2014

## Processing  Variables, Methods and Arrays in *Phase1.java* :

At *process(AVarMaindecl)* method, the compiler checks whether variable being declared is void or not. If it is not a void type, it calls *createClassVar* non static method in *Typechecker* class.

At *process(AMethodMaindecl)* method, the compiler calls non static method *createMethod* in *Typechecker* class. *process(AMethodMainDecl)* calls *process(PParamlist)* to process list of parameter types for the method being processed.

*process(PParamlist)* calls *process(AlistParamlist)* if the parameter list is not empty. *process(AlistParamlist)* creates a new ArrayList that stores processed parameters. Then it calls *process(PParam)* on each parameter. Each parameter is then processed by process(AParam) that checks whether the parameter being declared is void or not. If not, it calls *process(PType)* which then calls *process(AType)* to return a type for the parameter.

*process(AType)* checks whether the type being declared has emptydims to detect arrays. If it is does not have emptydims, it calls *getType* method in Typechecker class to return type listed in the typeMap. If the process detects emptydims, it runs a for loop to get the number of emptydim and calls typechecker.makeArrayType each time inside the for loop feeding the previous returnType to the makeArrayType method. This lets it handle it multidimensional arrays. *makeArrayType* method in Typechecker class tests whether the type fed is a void or not to prevent it from creating an array of voids.

## Handling conflicts in *Typechecker.java* :

*createClassVar(String, Type, Token)* checks whether classVarMap already contains a variable with same name before adding the new variable entry into the classVarMap.

*createMethod(String,Type,List<Type>,Token)* calls *checkNameAndParamSize(List <Type>,String)* to check whether methodList already contains a method with same name and parameter size. If it does, it calls *checkParam(List <Type>,String,int)* to check whether both methods contain the same type of parameters. If they do, it throws a TypecheckerException. If not, it creates a new Method object with the information provided and adds it to the methodList.

## Method.java and Var.java:

These classes store information about methods and variables being created. Method class has String _name, Type _type and List<Type> _paramTypes as non-statics. Var class contains String _name and Type _type as non-statics.

## ArrayType.java

ArrayType class extends Type class and is used to handle arrays. It has Type _typeOfArray as non-static variable that stores the type of array. Since we extend ArrayType from Type, we can create one directional arrays of types (int, String, and boolean) and further create arrays of type arrayType to create multidimensional arrays.