Lab 4 Frequently Asked Questions

Question 1:

The following chunk of text is copied from the in08.out file
```
################################################
Building Symbol Table
line 108: Redeclaration of identifier `main'
line 112: Redeclaration of identifier `Main'
line 115: Redeclaration of identifier `m'
line 116: Redeclaration of identifier `x'
Finished Symbol Table
################################################
```

In Lab4, is it a requirement to show the above message and line number when there is an attempt to redefine a name that has already been declared within the same scope?

Answer:

Yes it is a requirement.

There is a line number associated with every token (but NOT with non-terminals); the identifier name is a token in our language so the line number can be retrieved from the identifier token using method 'getLine()'. When an attempt to add a binding is unsuccessful (i.e., when false is returned), you must print the line number and message as shown above. In Lab5 you will also be printing error messages that include the line number whenever type errors are detected.

Question 2:

If class Main is redefined, then inserting the duplicate class into the symbol table will fail and return false. Since the current scope is GLOBAL, the method "int m()" will also fail when inserting ‘m()’ into the symbol table because the GLOBAL scope does not accept method declarations. So in order to get the same error message as the test cases, the CLASS scope would have to be entered even though the declaration of class Main failed. What is the best approach to this problem?

Answer:

Unless you enter the scope of class Main, even though it is a redeclaration, the fields and methods of the duplicate class will be inserted into the wrong scope entry. This could create INVALID error messages such as the one you mentioned about method "int m()" (the message is invalid because method "int m()" has not been declared in the global scope; it was declared in a class that was redefined and therefore should not fail).

The other possibility would be to immediately abort the program when there is a redeclaration of a class; this means that such conflicts would have to be corrected one at a time and any errors in the duplicate class would only be issued after all such duplicates have been corrected.

The usual approach is to try to give as many VALID error messages as possible and to avoid giving misleading error messages such as the one mentioned above. Therefore, it is better to enter the scope of the redeclared class, even though it has not been added to the global bindings.

Question 3:

What do I need to do about other constructs that create bindings such as local variable declarations inside blocks?

Answer:

In lab4, there is nothing to do for constructs such as local variable declarations because the bindings created by these declarations can only be referenced inside the block where it occurs and they cannot be mutually recursive (like methods can be). Therefore, they do not have to be inserted into the symbol table prior to type checking; they are inserted during phase 2 of the type checker, i.e., these constructs will be handled in lab5.