COP5555 Assignment 4

Assigned 4/4/2015
due Monday 4/13 at 10am.

Incrementally implement functionality for type checking and code generation.

I have given you skeleton versions of the TypeCheckVisitor and CodeGenVisitor classes along with an implementation of the SymbolTable. The SymbolTable class implements a LeBlanc-Cook symbol table as discussed in class and described in the textbook.

In general the TypeCheckVisitor does the following:

1. Enters and leaves scopes (calling appropriate SymbolTable methods at the beginning and end of blocks)
2. For each declaration, add the (name, Declaration) mapping to the symbol table, ensuring that the name has not already been defined in the current scope.
3. For each expression, infer its type and store in the (new) expressionType variable. If the expression has multiple operands, ensure that the types are correct. It is convenient to return the type of an expression from the expressions visit method.
4. When a variable is used (as in IdentExpression) its type is obtained by looking it up in the SymbolTable. If it isn't defined in the current scope, this is a type checking error.
5. Ensure that any expressions appearing in statements have an appropriate type for the constext.
6. Use JVM types to describe types. This will be convenient during code generation.
7. When a type checking error occurs, the throw a new TypeCheckException with a useful error message.

In generatl, the CodeGenVisitor does the following:

1. Generates a classfile that implements the Codelet interface. The Codelet interface contains one parameterless void method, execute.
2. The visitProgram method should set up the class and create the init method. It should set up the execute method and then visit its children to add variables and generate code.
3. When an expression is visited, it should generate code to leave the value of the expression on top of the stack.
4. When a declaration is visited, it should add the variable as a field to the classfile. (Later we will add some local variables, but we won't do that in this assignment)
5. See comments in the Assignment4Test class for more info.

You should be able to add your AST generating Parser and Scanner and run the tests in the Assignment4Tests suite. One test, printIntLiteral should pass initially.

I suggest you proceed through the tests, adding functionality as necessary for each test to succeed. Make sure to read the comments in the test file—there are many hints on how to do this. Think about what the AST of the program under test looks like—this will tell you which visit methods you need to worry about.

Remaining assignments will add test cases for additional features of our language.

TURN IN

jar file called cop5555hw4.jar with source code for ALL classes needed to run the test file.