COP5555 Spring 2012
Project 5

Assigned March 20
Due: April 3 at 3pm

Follow the instructions carefully. Test cases that fail due to failure to follow instructions will not be given credit and no regrades will be allowed.

1. Implement a visitor class called **CodeGenVisitor** in package `edu.ufl.cise.cop5555.sp12.codegen` to traverse your AST from Project 4 and generate code. More specifically, visiting a Program ast node should result in code being generated for the program, and the resulting class file returned from the visit method as an array of bytes.
2. See the attachment for more information about what our language is supposed to do. Note that not all of the language implementation is required; the rest will be done in Project 6.
3. You have been provided with a skeleton implementation that will generate code for programs containing only print statements with integer literal arguments. Your solution should add or complete the methods as necessary to satisfy the requirements of this project.
4. You have also been given a class Compiler that sets everything up to read a source file in our programming language and generate code. Note that there are two constants, WRITE_TO_FILE and EXECUTE. The first will cause a classfile to be written to the file system where you should be able to execute it with the usual java command; the second causes your generated classfile to be immediately loaded and executed. In order for the Compiler class to compiler you will need to make the t field in the SyntaxException class from Project 2 public.
5. You may want to add fields and/or methods to code from earlier projects.
6. Watching the lectures from 3/16 and 3/20 before starting the project is highly recommended.

Use **ASM 4.0** to generate your classfiles. This can be obtained from asm.ow2.org. Specifically, you want to download asm-4.0.jar. There have been API changes from previous versions of ASM, so make sure you get the right one.

If you use eclipse, get the latest version (2.4.0) of bytecode-outline. Do not use the eclipse update site they link to on the web site—it is not up to date. One way to install it is to download the .update file,

de.loskutov.BytecodeOutline.update_2.4.0.zip

and uncompress it. Then, in eclipse go to Help/Install new software/Add, and add the folder as a local repository.

If you do not use eclipse, check the command line tools that come with the asm distribution. You may need to download another jar file containing utils.

Use the bytecode-outline asmifier or equivalent command line tool to help you figure out what kind of code to generate. There will be several situation where you will probably want to just write a java program that has equivalent semantics to some feature in our language and then use the tool to see what its bytecode should be and what the asm calls to generate it look like. We are using the ClassWriter.COMPUTE_FRAMES argument to the ClassWriter constructor. This tells asm to compute frame information for us. The asmifier code does not do this—it calls the ClassWriter constructor with argument zero and explicitly inserts the frame information with mv.visitFrame calls. Since we are having asm do this for us, just ignore the visitFrame calls you see in the asmified code. (Doing it explicitly rather than letting asm do it is more efficient, and we might care about that if we were using asm to, say, transform classfiles at loadtime. But since we aren't, we will choose the easiest approach for the programmer. You can read more about this in the asm documentation if you are interested.)

The asm documentation has a concise explanation of the JVM if you need more information. And of course you may also look at the JVM specification itself available at http://docs.oracle.com/javase/specs/. (This is a more recent version than the one given in the lecture slides.)

Turn in a jar file called P5.jar containing ALL of the sources needed to scan, parse, perform context checking, and generate code, including those from previous projects and the handouts for this one. Do not change the signatures of these methods or the package declarations or you will break the test script.