IN2009 Tutorial 2

In the Tutorial2 Resources folder in Moodle, you will find various files in sub-folders, as described below.

1. In *data* there are some small Calculon programs (*Ex0.calc*, *Ex1.calc*, etc). You will be building Abstract Syntax Trees (ASTs) for these programs. Copy these files into the *data* folder of your *Calculon* project.
2. In *src/compile/ast* there are some AST Java classes. *Program* is an updated version of an existing AST class, the others are new (as described in the Session 2 lecture). Copy them all into *src/compile/ast* within your *Calculon* project (the new version of *Program* should replace the old version).
3. The folder *src/handbuilt* contains a number of exercises in the form of java classes (*Ex0.java*, *Ex1.java*, etc). First, in IntelliJ create a new package called *handbuilt* within the *src* folder of your *Calculon* project; this creates the folder *src/handbuilt* (or just use your OS utilities to create the folder directly). Copy *Ex0.java*, *Ex1.java*, etc into the new folder. The aim is that each class will build an AST for the corresponding calc program; at end of each class are two lines of code which compile the AST and write the generated assembly code into a file.

Class *Ex0.java* has already been completed for you. Compare the Java code with the source code in *Ex0.calc*. Note that the comments at the top of *Ex0.calc* describe the expected *output* of the program. In this example, the variable declarations serve no actual purpose (the variables are never used) but the Java code in *Ex0.java* demonstrates how the corresponding part of the AST is constructed. Similarly, the use of a block in the body of *Ex0.calc* is essentially pointless (the statements within the block could simply be in-lined within the program body) but *Ex0.java* demonstrates how the AST for a block can be constructed. To test that *Ex0.java* works correctly, do the following:

1. Use IntelliJ to recompile the *Calculon* project (*Build* $\Rightarrow$ *Build Project*). **Note**: do this *every time* you change any of the Java source files, before further testing.
2. Within IntelliJ, open a Terminal window and enter the following command:

   ```
   java handbuilt.Ex0
   ```

   (Alternatively, just run *Ex0* from IntelliJ using the run button ▷). If all goes according to plan, this will create SSM assembly file *Ex0.ssma* in the root folder of your project. Open *Ex0.ssma* in the IntelliJ editor and check that the assembly code is as you would expect.
3. To test the generated code, assemble and execute:

   ```
   java Run Ex0.ssma
   ```

   Check that the output is correct.

Your job now is to complete as many of the remaining exercises as you can. In each case you may find that you need to write one or more new AST classes and/or extend the compile method in *ExpBinaryOp* to handle a new kind of operator.