Language Processors Lab Week 1

The goal of this week's lab is to compile and execute the Straightline Programming Language interpreter.

1) Compiling the Straightline Interpreter

Start up a Unix shell window and get the environment ready by typing the command

```
> module add java soi
```

Move to the directory where you want to do your IN2009 work and copy the `straightline.tar` file from the module's CitySpace directory `Week1` or from the following URL: http://www.soi.city.ac.uk/~sbbc287/straightline.tar

Type the command `tar -xvf straightline.tar`. This creates the straightline directory and copies all related `.java` files. Change to the `straightline` directory and compile using the following:

```
> javac *.java
```

2) Running the Straightline interpreter

Run the program with > **`java Interpreter`**

The interpreter is implemented as an infinite loop that asks the user to enter a Straightline program and - after the user hits enter – prints out the result of interp. Recall the Straightline Programming Language Syntax:

Stm → Stm ; Stm | id := Exp | print(ExpList)
Exp → id | num | Exp Binop Exp | (Stm , Exp)
ExpList → Exp , ExpList | Exp
Binop → + | - | * | /

Try the following examples:

a) print 5 // This should give you a parse error – correct it
b) print(1,1+2,3*5) // This should be ok
c) print(1,1+2), print(3*5+5) // Another parse error. What's wrong?
d) x = 12; print(10 + x)
e) x := 12; print(10 * y + x) // A semantic analysis should've detected this
f) x := 5; print(3, (x := x * 10, x+2),500) // try more of these

The file `Interpreter.java` contains the entry point to the program: an infinite loop that reads input from the user, parses it and prints out the result. The parser has been implemented using Javacc (to be learned in the following weeks); it performs lexical analysis and creates an instance of the `Stm` class. The file `straightlineAST.java` contains all the classes that implement the programming language's abstract syntax trees and the implementations of `interp`. Inspect both files and understand the code that implements the interpreter.