CITY UNIVERSITY LONDON

IN2009 Language Processors (2009/2010)

Programming Assignment 2 (Modifying the SPL Interpreter)

DUE: 8pm. Thursday, April 1st 2010.
HAND-IN: The sourcepl directory, zipped and tar-ed (sourcepl.tar.gz) with your new code.

Organisation: This coursework assessment may be completed individually or in pairs - there is no penalty for working in a pair. You will hand work in as individuals but the hand-in procedure will allow you to say who you collaborated with (and you must do so!).

Hand in: The deadline and the electronic hand-in procedure, and exactly what you should hand-in, are documented online. Obviously you should change file and directory permissions while you are working so that your work is not visible to others -- remember plagiarism carries severe penalties.

Instructions:

Download the file sourcepl.tar.gz from the CW2 CitySpace folder. Unzip and untar sourcepl.tar.gz. You will get the sourcepl directory with the following files and directories:

- sourcepl.jj : The JavaCC file that contains the grammar specification of SPL. It also contains Java code (semantic actions) that generates the abstract syntax tree for the parsed SPL program.
- Main.java: Driver class that calls the parser and then invokes two visitors that pretty-print and execute the SPL file passed as argument.
- test.pls: Text file that contains a sample SPL program.
- ast directory. Contains the Java classes that implement abstract syntax trees.
- visitor directory. Contains:
  - The visitor interface Visitor.java.
  - The PrinterVisitor.java visitor used to pretty-print SPL abstract syntax trees.
  - The Interpreter.java visitor used to execute SPL programs.
  - VisitorException.java.

The contents of the sourcepl directory correspond to the solution of lab6. Therefore, the instructions on how to compile and execute the program are the same as in lab6. You will be asked to extend the parser, pretty-printer and interpreter with a new expression and a new statement.

TASKS: Update the necessary files in order to implement the following:

1) The modulo operator. Add the modulus operator "%" to SPL. The modulus operator returns the remainder of two integers. For example, the following SPL code:

```
x := 12;  print (x % 2 );  print( x % 5) ;  print( x % 9 );  print( x % 3 );
```

should evaluate to: 0 2 3 0

The operator priority of the modulus operator should be the same as multiply, division and the comparison operators (hint: it should be part of the term() non-terminal). Therefore, the following SPL code fragment:

```
x := 12;  print (4 + x % 2 );  print( x % 5 * 2) ;  print( 9 * x % 9 );  print( x % 3 + 1);
```

Should produce the following output: 4 4 0 1

Note that same priority operators are parsed (by JavaCC) from left to right.

2) The RepeatUntil statement. Add the **RepeatUntil** statement defined by the following syntax:

RepeatUntilStm → "repeat" "{" *Statement*$^{+}$ } "until" "(" *Exp* ")"

Given abstract syntax RepeatUntilStm(List<Stm> body, Exp e), the interpreter executes the **RepeatUntil** statement by first executing the contents of its body and then evaluating test e. If the test returns true, execution ends. Otherwise, the interpreter re-executes body and re-evaluates e until the value of the test becomes true. The body of a RepeatUntil statement is executed at least once.

For example, the following SPL code:

```
x := 0; repeat { if (x%3 == 0) then { print (x); }  x ;= x + 1; } until (x == 16)
```

Should output: 0 3 6 9 12 15
Note that we are using the new modulus operator in order to test if a number is a multiple of 3. The PrinterVisitor should output the following:

```
x := 0;
repeat {
  if (x%3 == 0) then {
    print (x);
  }
  x ;= x + 1;
} until (x == 16)
```

What to do. In order to add a new expression or statement, you must:

- Modify sourcepl.jj in order to add new syntax and call the right ast class constructor.
- Create a new class inside the ast directory the implements the AST of the new expression or statement.
- Update the visitors' interface (Visitor.java)with a new visit method that takes as argument the new expression or statement AST class.
- Implement the new visit method in PrinterVisitor.java
- Implement the new visit method in Interpreter.java

These steps (while statement) are described in lab6.pdf.