INF 3110 Mandatory 1-- 2012

ROBOL Language

ROBOL is a language for controlling a very simple robot. A robot starts at a given point (x, y) on a grid with predefined boundary (x_bound * y_bound) in a given direction (x, -x, y, or -y), then moves on this grid (forwards, backwards, left, right) a number of steps given by an integer value until it stops. While moving on this grid, the robot may lower a pen so that it will draw lines for the subsequent moves until the pen is lifted.

Here is the grammar of the language:

```
<program> ::= <grid> <robot>

<grid> ::= <size>(x_bound * y_bound)
<size> ::= size

x_bound ::= <number>
y_bound ::= <number>

<robot> ::= <var-decl>* <stmt>*
<stmt> ::= <start>(<exp> , <exp>, <direction>)
         | <stop>
         | <move>(<exp>)
         | <pen> | <light>
         | <assignment> | <if-then-else> | <while>

<direction> ::= - ? x | - ? y

<var-decl> ::= var <identifier> = <exp>

<start> ::= start
<stop> ::= stop

<move> ::= <forward> | <backward> | <left> | <right>
<forward> ::= forward
<backward> ::= backward
<left> ::= left
<right> ::= right

<pen> ::= <pen-up> | <pen-down>
<pen-up> ::= up
<pen-down> ::= down

<assignment> ::= <identifier> = <exp>
<if-then-else> ::= if <boolean-exp> { <stmt>+ } |
                   if <boolean-exp> { <stmt>+ } else {<stmt>+ }
<while> ::= while <boolean-exp> { <stmt>+ }

<exp> ::=
    <identifier> | <number>| (<exp>) | <plus-exp> | <boolean-exp>

<boolean-exp> ::= <exp> > <exp> |
                  <exp> < <exp> |
                  <exp> = <exp>
```

```
<plus-exp> ::= <exp>  + <exp> |
               <exp>  - <exp> |
               <exp>  * <exp>
```

<number> is an integer.

Assignment

Make an interpreter for ROBOL in Java.

The classes representing nodes in the abstract syntax tree implements interfaces (or a single interface Robol) that have the method void interpret(). Given a program represented by an object of class Program and referenced by a reference p typed by the interface Robol, p.interpret() will interpret the program.

Requirements:

- Make the interpreter so that it is easy to extend.
- Feel free to display the result (in terms of lines) in whatever form you like, however, as a minimum the stop statement shall print the position (and direction) of the robot.
- Implement a pretty-printer for ROBOL. It is up to you to decide upon the layout, e.g. when to issue a new line, but use the concrete syntax given in the grammar (e.g. start and stop for <start> and <stop>, respectively.
- The interpreter shall check that the robot does not moves outside the grid.
- Write a short note (1 or 2 page) explaining how you designed and implemented the interpreter. And also specify how to configure and run your code in a certain IDE (e.g. Eclipse).

Test Programs

Test programs are provided as below. However, we do not use scanners and parsers to turn the test programs into a structure that is suitable for interpretation (e.g. abstract syntax tree). You are supposed to make Java code that makes the structure that represents each test program.

Demonstrate your implementations of ROBOL by means of the following test cases:

```
**************************************
Testing Code 1: Simple Example
size(64*64)
start(23,30, -x)
forward(15)
up
left(15)
right(2+3)
down
backward(10 + 27)
stop
**************************************
The result is (40,15,x)
```

```
**************************************
Testing Code 2: Example with variables
size(64*64)
var i = 5;
var j = 3;
start(23,6, -x)
forward(3*i)
up
right(15)
left(4)
down
backward(2*i + 3*j + 5)
stop
**************************************
The result is (28,21,x)


**************************************
Testing Code 3: Example with While loop
size(64*64)
var i = 5;
var j = 3;
start(23,6, -x)
forward(3*i)
up
right(15)
left(4)
down
backward(2*i + 3*j + 5)
while(j>0)
{
       right(j);
       j = j -1;
}

stop
**************************************
The result is (26,19,y)


**************************************
Testing Code 4: Example with if-then-else
size(64*64)
var i = 5;
var j = 3;
start(23,6, -x)
forward(3*i)
up
right(15)
left(4)
down
backward(2*i + 3*j + 5)
while(j>0)
{
       right(j);
       j = j -1;
}
if( i > 3)
{
       forward(14);
}
else
{
       backward(14);
}

stop
**************************************
The result is (26,33,y)
```

Delivery Requirement

Document the process that has been done.

- It is very important to document how you think and why you make the choices when designing the interpreter.
- Put snapshots of code in the document if you think that is necessary;
- Suggestions to improve the mandatory are also welcome;
- The document should be in PDF format.

Always zip all your files, in case of too big attachment. Name your zip file ‘INF3110 Oblig1 _<usernames>.Zip’. All the code and the document should be included in the delivery.

The email title should be ‘INF3110_2012_Oblig1_<usernames>’;

Solution of Mandatory 1 should be submitted to weiqingz@ifi.uio.no no later than 15.10.2012 23:59;