Assignment One -- Python

The assignment is to implement a simple domain specific language to control a Java program for Turtle Graphics. Parsing and translation and execution of the DSL should be implemented in Jython. The files that you need (assigment.jar) can be found as an attachment to this message.

DSL

The language to control the painting is an adoptation of the idea of turtle graphics. The language has 8 statements:

```
  pen_down - turn pen on (movement draws on canvas)

  pen_up - turn pen off (movement does not draw on canvas)

  move_forward - move one step forward

  move_backward - move one step backward

  move(STEPS, ANGLE) - move STEPS steps in ANGLE direction,
                       where STEPS and ANGLE are integer
                       literals (1, 2, ...) or loop
                       variables (see below)

  turn_cw(ANGLE) - turn clockwise, where ANGLE is an integer
                   literal or a loop variable (see below)

  turn_ccw(ANGLE) - turn counter clockwise, where ANGLE is
                    an integer literal or loop variable (see
                    below)

  put(XPOS, YPOS, ANGLE) - puts the pen on XPOS, YPOS
                           pointing in ANGLE direction

  for X=0 to 10 do - repeat STATEMENTS 10 times (STATEMENTS
  STATEMENTS end     may not contain another loop). X should
                     be visible to STATEMENTS (but not
                     outside the loop)
```

ANGLE is the number of degrees to turn clockwise from the current direction. STEPS is number of pixels.

All arguments passed to statements taking arguments must be integer literals, loop variables or a combination of the two separated by one of the following binary arithmetic operations: +, -, *

Ex:

```
  put(100, 100, 90) - place on 100x100 facing east

  move(10, 45) - move 10 steps in 45 degree angle of facing

  for X=0 to 4 do    - draws a square with the side 50 pixels
     move(50, X*90)
  end
```

The Task

The program consists of a Java application with a canvas and a textarea for turtle code. Your job is to write a Jython application that takes turtle code from the Java application, parses it with regular expressions and calls setPixel(x,y) in the Java application to draw the corresponding image.

The turtle remembers its current angle.

For example the following turtle code program draws a rectangle with the sides 50 and 100 pixels:

```
put(150, 150, 0)
move(50, 90)
move(100, 90)
move(50, 90)
move(100, 90)
```

The following turtle code program draws a tilted square with the side 50 pixels:

```
put(100, 100, 45)
for X=0 to 4 do
  move(50, 90)
end
```

For this task there is a preexisting bridge class called JythonTranslater.Jtrans, that accepts events from the Java program. You should subclass this class and override its methods.

Restrictions

You may only interact with the Java program by calling setPixel(x, y) to control the painting and getCode() to get the code entered in to the turtle code textarea. These methods are both defined in the DYPL Java class.

You may not change the Java code in any way.

All code to implement this assignment should be Python code.

Hints

Your Jython class(es) may be compiled to byte code with the command jythonc. If you use the -w flag with jythonc you can get it to put the resulting class files in your current directory (default is ./jpywork):

```
jythonc -w . MyJythonFile.py
```

When Jython classes are compiled with jythonc they are turned into regular Java classes and hence the module and class concepts get a bit cluttered. One could say that java packages are python modules. But also if you examine the class files produced by jythonc you'll see that the jython classes are inner classes in Java.

I previously said that you should name your class and file differently. This is not true. It is true for the general

case, but the DYPL program assumes that the file and class have the same name. So, in the general case you should name them differently, but not for this assignment.

In this example we have a file called MyJythonFile.py containing a class called MyClass and we have compiled with jythonc:

```
from MyJythonFile import MyClass
class MyOtherClass(MyClass):
    pass
```

If you keep all class files in the same directory you should have less classpath problems.

Don't forget to include jython.jar in your classpath

To get you started you get the pen_up and pen_down [regular expressions]:

```
"\s*pen_down\n"
"\s*pen_up\n"
```

A good idea might be to write a Python program that just parses a turtle code file in order to eliminate as many sources of error as possible for jour first attempts to write Python code. Jython is only required for the actual canvas drawing when the Java application must work together with the Python application.

Running the DYPL program

Unjar the assignment.jar file:

```
jar -xf assignment.jar
```

To run the DYPL program:

```
java -classpath /path/to/jython/home/jython.jar:. DYPL
```

To run the DYPL program with your JythonTranslater subclass:

```
java -classpath /path/to/jython/home/jython.jar:. DYPL
NameofSubclass
```

(Note that classpath separators in *nix are : but ; in DOS/windows. Thus, replace : by ; above if not running *nix)

Resources

The Jython Home Page: http://www.jython.org

The Python Home page: http://www.python.org

Regular Expressions: http://www.amk.ca/python/howto/regex

General knowledge: http://google.com