Connecting Java and Matlab

The matlab desktop runs a java virtual machine. To use a java class, you must first import it. For example, you could type

```
import javax.swing.*;
```

You can then use any of these classes, but you must make some changes in syntax. The following is a list of those that come to mind:

- In matlab, you do not need to type "new"--objects are created as needed.
- In matlab, you use 'single quotes' where you use "double quotes" in java.
- In java you call a routine with no inputs by putting () after its name. This is unnecessary in matlab.

For example, try this:

```
import javax.swing.*
J = JFrame('Hi there')
L = JLabel('A Label');
P = J.getContentPane
P.add(L)
J.setSize(200,200);
J.setVisible(1)
```

You can also import your own classes. There are two things that you must get working in order to do this.

- You must tell matlab where to look for classes. To do this, you must make sure that the file "classpath.txt" contains the directories in which you class files reside. There is a default classpath.txt which does contain ".", so changing to the correct directory should work. For information on how to locate "classpath.txt" and create your own (you probably put it in your home directory or a directory called ~/matlab), look at the advice from MathWorks. Mathworks provides more help here
  In my current installation, then only way I can get matlab to read java classes is by starting matlab in the directory in which they reside

- Matlab does not necessarily accept code from the latest compilers. It might be necessary to add the flag "-target 1.3.1" to my call to javac. For example, you can compile my class VectorChooser.java by downloading it and typing:

  ```
  javac -target 1.3.1 VectorChooser.java
  ```

Let's now give VectorChooser a spin:

```
>> import VectorChooser
>> v = VectorChooser(4)

v =

VectorChooser[frame0,0,0,0x0,invalid,hidden,layout=java.awt.BorderLayout,resizabl

>> v.setVisible(1)
>> v.setSize(250,200)
>> v.getAll
```

```
ans =

     0
     0
     0
     0

>> v.setAll(10*rand(1,4))
>> v.getAll

ans =

     9
     2
     6
     4

>> v.setOne(1,0)
>> g.getOne(0)

and =

     9

>> v.getOne(0)
```

Try moving the sliders. You'll see that matlab can read them!

You can also have a java process running within matlab call matlab, but that is a story for another day.

Recompiling Classes

Unfortunately, if you recompile your java class, matlab will not know about it until you quit and restart matlab. However, Paul Milenkovic has found a cure for this. The rough idea is that you create a classloader for your class, and then access it through the classloader. After you recompile, you kill the classloader and then create a new instance of it, which then reads the recompiled class. Please see Paul Milenkovic's page for more information. Maybe I'll create an example using VectorChooser if there is confusion.