How to Find Obscure Java UI Functions Using Jython

So in class today I noticed that something I didn't cover in detail in my lecture: say you need to use some feature in the Java UI framework that we haven't provided some example code that shows you how do it in Jython. The first thing to try is probably a google search, but you may not find an example out there before the builders of Jython generally assume that you are familiar enough with Java and the way Jython links up Java to look at the Java documentation to figure out what to do.

So let me explain how to do that. Say we want to make a window with a menu. You can do a search for java menu and maybe run across a tutorial like this. Or, if you're just looking for a specific function, you might take a look at the Java documentation (Javadoc) on Oracle's website. The nice thing about the Javadoc is that every function in the Java API is listed there. It's often what you need to usewhen you need to find some particularly obscure feature that the object can do. Here's the JMenu Javadoc.

Ok now let's look at two chunks of code. One is some Java code, and the other is the corresponding Jython code:

```
import javax.swing.JFrame;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;


// some boilerplate about java
classes

// and functions


JFrame frame = new
JFrame("Frame");

JMenuBar menuBar = new JMenuBar();

JMenu file = new JMenu("File");

menuBar.add(file);
```

```
import javax.swing as swing









frame = swing.JFrame("test1")

menuBar = swing.JMenuBar()

file = swing.JMenu("File")

menuBar.add(file)

open = swing.JMenuItem("Open...")
```

```
JMenuItem open = new
JMenuItem("Open...");

JMenuItem quit = new
JMenuItem("Quit");

file.add(open);

file.add(quit);

frame.setJMenuBar(menuBar);

frame.pack();

frame.show();
```

```
quit = swing.JMenuItem("Quit")

file.add(open)

file.add(quit)

frame.setJMenuBar(menuBar)

frame.pack()

frame.show()
```

As you can see, there's a simple 1-to-1 mapping for most Java code to Jython.

Now lets say I someday want to figure out how many items are in one of those menus. Looking at the Javadoc, I notice a function called `getItemCount()`. How do I use that to print out the number of items in my file menu? Well, one thing I could do is say something like:

```
print file.getItemCount()
```

But you can be fancier, because anytime a Java class has a method like getSomeCrazyBitOfData, python creates a property named someCrazyBitOfData. So you can also do this:

```
print file.itemCount
```

Note that the two above lines are actually exactly the same: that is file.itemCount is really calling getItemCount() behind the scences. The same trick works for setting stuff. So if I wanted to make the menu's popup menu visible I can do either of these:

```
file.setPopupMenuVisible(True)

file.popupMenuVisible = True
```

Responding to to UI events something you do a lot, but the code between Jython and Java looks perhaps more different than you would expect. Say we want to have our menu do something crazy on a double click. Can we even do that? Well, what we want to be on the lookout for is something called a listener in

Java. There should be a function like addSomeKindOfListener in the class you'd like to respond to the event. Looking at JMenu, we see something called addMenuListener. Will that do the trick? To figure it out I click on the MenuListener class. In there I see functions like menuSelected, menuCanceled - potentially interesting stuff to know about but not related to double clicking.

But, after some clicking around we notice that java.awt.Component, one of the parent classes of JMenu, actually has a method called addMouseListener. If I look at the Javadoc page for MouseListener, I see that *MouseListener* has a method called mouseClicked. That method takes a MouseEvent parameter that will tell me how many clicks this event has. To add a callback to my Menu in python, this is what I do:

```
def callbackForMouseClick(event):
    if(event.clickCount == 2):
        print "Double click!"
    else:
        print "Some other number of clicks"

file.mouseClicked = callbackForMouseClick
```

The fact that this works might be somewhat odd. But jython adds a property for every method of the listeners that an object can have added to it. So to see what events you can add callbacks for, look for addXXXListener methods, see what methods those Listeners define, and then you can set properties that correspond to those listener methods.

Hope that helps! For even more details on the linkages between python and Java, look at the Jython documentation.

Or you can contact me in whatever way you find most convenient.