Lab Exercise 11

Software Design **Allen B. Downey**
Spring 2006

Due: never!!!

11.1 Popups

1. Download and read

   ```
   wget http://wb/sd/code/Popup.py
   ```

   Now run it. It should create three Popup windows, where each window contains a label and a Close button. Notice that each Popup is running `mainloop` in a separate thread, so you can interact with the three windows independently.

   When you close all the windows and the last thread completes, the program quits. The object of this lab is to transform `Popup` so that it can be invoked remotely. Start by making a copy of `Popup.py` called `RemotePopup.py`.

2. Modify `RemotePopup.py` to define a new class called `PopupServer` that inherits from `RemoteObject`, and then define a method named `popup` that takes a string as a parameter and creates a Popup object that displays the given string.

3. Write a `main` function that creates a `PopupServer` and then invokes `requestLoop` on it. Connect your PopupServer to the name server on `ece` using a unique name.

4. At this point you have a working Remote Object. When you create a `PopupServer`, it will register with the name server running on `ece` and then wait for incoming requests. When you hit Control-C, it should unregister cleanly.

   If your program crashes without removing the entry in the name server, you won't be able to restart the program. To remove a name from the name server, start the Name Server Controller by typing:

   ```
   /usr/bin/python-pyro/xnsc &
   ```

   A window should appear. Type `ece` into the entry labelled Name Server Location, then hit enter. Press the List button to see the names that are defined. If you want to remove a name, type it into the entry labelled Command arguments and they press `Remove`. You should see a confirmation message in the text field.

5. Create a `PopupClient` that can talk to your server. It should look up your remote object by name and then invoke `popup` on it, passing a message as an argument.

6. Start your server in one window and leave it running. Run your client in another window and send yourself a message.

   If all goes according to plan, the server should pop up a window to display the message.

7. Find another student in the class that has a working client-server pair. See if you can get your client to talk to your partner's server, and vice versa. If not, help each other get it debugged.

   Congratulations! You just wrote an instant message system.