Chris Piech
CS 106A

Assn #7
June 1, 2018

Steam Tunnel Client Extension

Now that you have implemented the server part of SteamTunnel, as an extension you implement part (or all) of the client program.

At its core, the client program is a program with interactors and a canvas, much like the ones you have seen before. The main new concept is how to generate a request and send it to your server. As an example, in the starter code for **`SteamTunnelClient.java`**, which is the file where you should implement the client program, we create a ping request and turn the response into a **`GLabel`** that we display in the center of the canvas. The method you use to do this is

```
String SimpleClient.makeRequest(String host, Request request)
```

For instance, in your client program you could use it as follows:

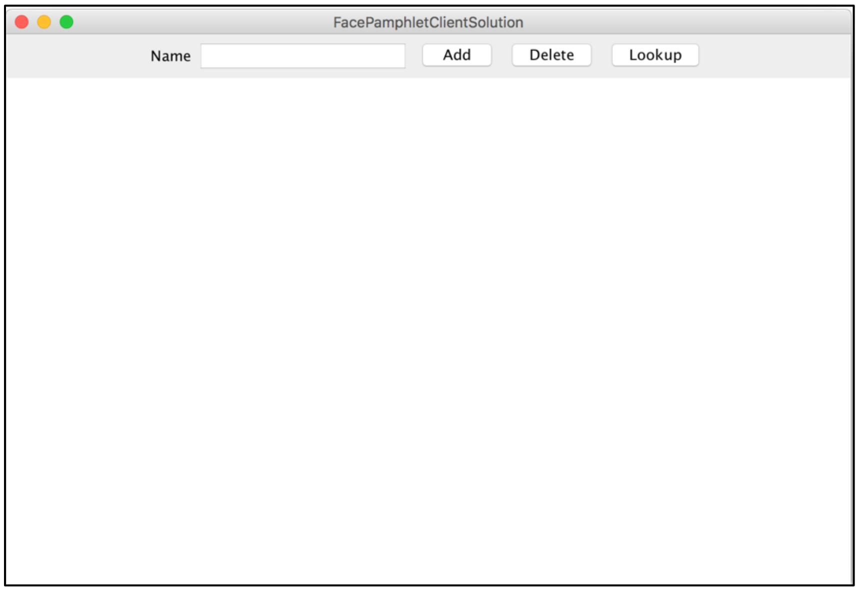
```
Request myRequest = ... // make a new Request and add any params

String response = SimpleClient.makeRequest(HOST, myRequest);
```

If an error occurs while sending the request or receiving the response, an *IOException* will be thrown. **This includes when the server sends back a response that begins with the text "Error:"** (as you did when implementing your server). To access the error message, you can say:

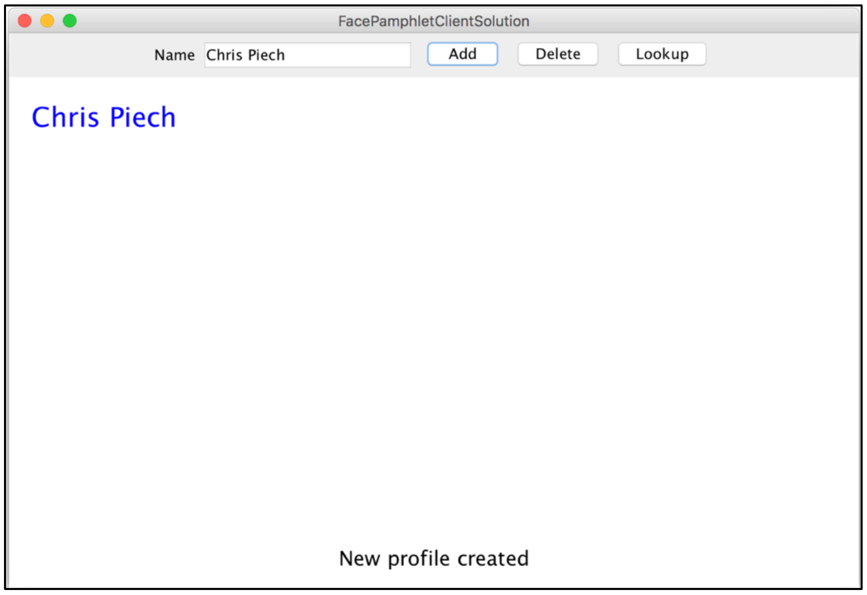
```
try {
  Request myRequest = ...
  String response = SimpleClient.makeRequest(HOST, myRequest);
  // if we get here, the request was successful - continue on...
} catch (IOException e) {
  // if we get here, there was an error
  String errorMessage = e.getMessage();
  // Do something with errorMessage
}
```

On launch, the client application you are required to implement should look like this:



Along the **`NORTH`** border of the application, is a “Name” label, a text field, and three buttons: **`Add`**, **`Delete`**, and **`Lookup`**. The text field should be **`TEXT_FIELD_SIZE`** characters wide, and should *not* respond to the ENTER key. The functionality of each button is described below.

Add
When the user types something into the text field and clicks “Add”, your client program should attempt to add a new profile to the SteamTunnel social network; you will need to communicate with the SteamTunnel server to do this. If the profile is successfully added, you should display the new profile name on the canvas, as well as a message indicating that the profile was successfully created:



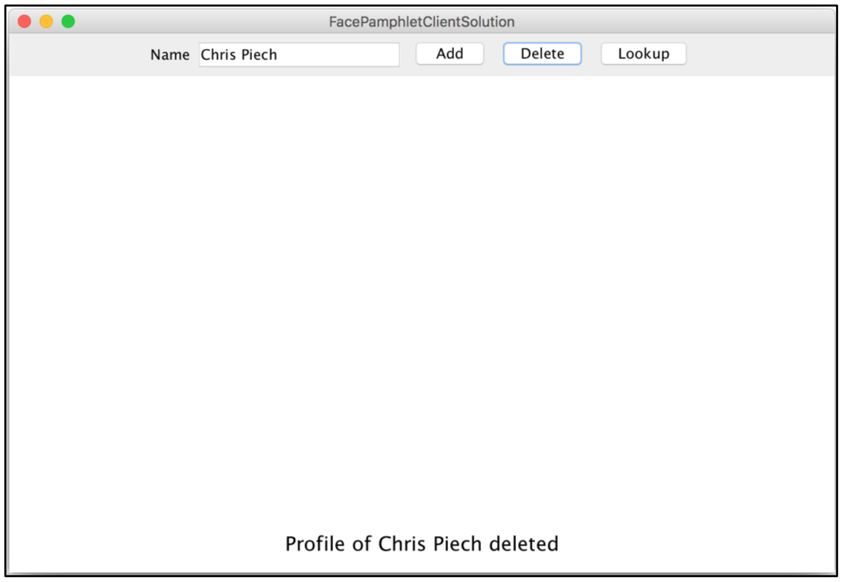
Chris Piech

New profile created

The name label should be colored blue, with font **`PROFILE_NAME_FONT`**, be positioned **`LEFT_MARGIN`** from the left side of the screen, and the *top* of the label should be **`TOP_MARGIN`** from the top of the screen. You should also add a success message saying "New profile created" at the bottom of the screen. This label should be colored black, with the font **`MESSAGE_FONT`**, centered horizontally on the screen, and the vertical *baseline* of the label should be **`BOTTOM_MESSAGE_MARGIN`** from the bottom of the screen.

In the event that an error occurs, you should simply display the error message received at the bottom of the screen. You should also remove any profile information that was currently displayed.

Delete
When the user types something into the text field and clicks "Delete", your client program should attempt to delete the profile for the entered name from the SteamTunnel social network; you will need to communicate with the SteamTunnel server to do this. Note that it does not matter what profile (if any) is currently displayed on the screen – deleting a profile involves only the text in the text field. If the profile is successfully deleted, you should remove any profile information that was currently displayed on the canvas, and display a message indicating that the profile was successfully deleted:

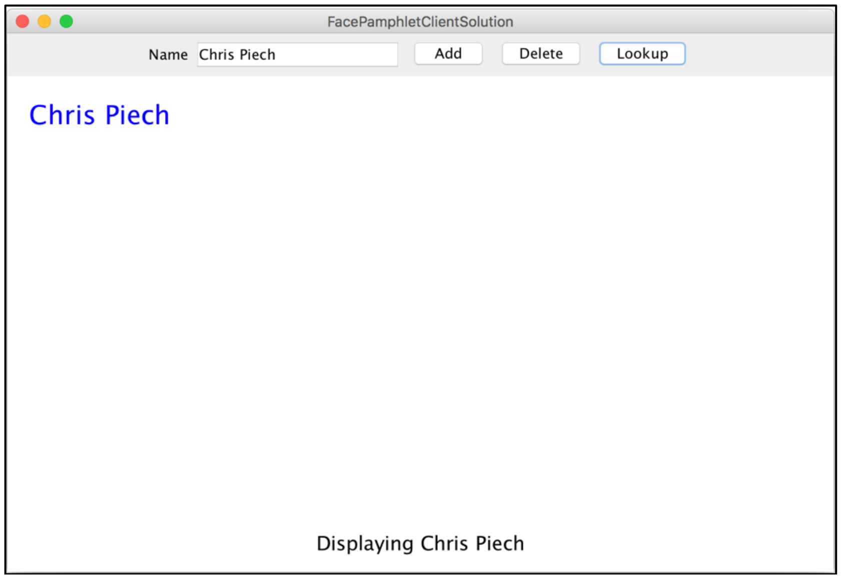


Profile of Chris Piech deleted

In the event that an error occurs, you should simply display the error message received at the bottom of the screen. You should also remove any profile information that was currently displayed.

Lookup
When the user types something into the text field and clicks "Lookup", your client program should attempt to display the profile corresponding to that name, if any, in the SteamTunnel social network; you will need to communicate with the SteamTunnel server to do this. If a profile exists for the given name, you should display the profile name on the canvas, as

well as a message that says "Displaying [NAME]" (replacing [NAME] with the entered name:



Displaying Chris Piech

The labels should be positioned the same as in the "Add" case.

If no profile exists for the entered name, you should clear the canvas of any existing profile information and display a message "A profile with the name [NAME] does not exist" (replacing [NAME] with the entered name).

In the event that an error occurs, you should simply display the error message received at the bottom of the screen. You should also remove any profile information that was currently displayed.

Other Notes:

- You are not required to implement any additional client functionality beyond what is specified above, though you are welcome to do so as an extension.
- None of the interactors above should respond if the text field contains no text (i.e. if the text field's text is the empty string).
- Now that you have implemented the server, as the client think about what types of requests you will need to send to get the information you need.
- Feel free to test your client using the provided server demo
- The cool thing about implementing the client and server is that, even if you close the client program, as long as the server is still running the social network data will still be there. In other words, you can quit your client, restart it, and view a profile you created earlier. Pretty neat!

Further Client Extensions

As mentioned earlier, you may add additional features, either that are included in our demo, or that you invent on your own. Here are some additional ideas for ways to extend your SteamTunnel program (some of which may require modifications in both the client and server). **Important:** please implement extensions in separate file(s). To create a new class in your Eclipse project, right-click on "default package" and select New -> Class. Enter the name you'd like, confirm, and it will be added to your project.

- *Implementing support for images.* You can add support in your client program for images, such as profile images (like in our client demo) or other image features. In your client, you have access to the same **`String`**/**`GImage`** conversion methods, but these methods are on the **`SimpleClient`** instead of the **`SimpleServer`**. In other words, the provided methods are

```
/* Converts a GImage to its string representation
String SimpleClient.imageToString(GImage image)

/* Converts a string representation of an image to a GImage
GImage SimpleClient.stringToImage(String str)
```

- *Keep track of additional information for each profile.* The current profile only keeps track of a name, image, status and a list of friends. In real social networks, there is much more information about users that is kept track of in profiles (e.g., age, gender, where they may have gone to school, etc.) Use your imagination. The more challenging issue will be how you appropriately display this additional information graphically in the client.
- *Click on friends to see their profile.* First of all, in your friends list you could keep a picture of each friend beside the name of the friend. Then, you could implement mouse listeners so that if the user clicks on one of the friend's photos (or name) you go straight to that friend's profile.
- *Support for groups.* Many social networking applications allow for keeping track of "groups" (or "communities") that profiles can belong to. In many ways, being a member of a group is similar to having that group as a "friend"—a "group" has a list of members (similar to a list of friends for a profile) and each profile can be a member of many groups (much in the same way that a profile can have many friends). Adding support for groups would help make your social network more realistic and may not actually require too much work if you can leverage some of the conceptual similarities with respect to "groups" being like "friends".
- *Finding friends of friends.* Another interesting aspect of social networks is not only keeping track of how many people you have as friends, but also how quickly that number grows as you consider all the friends of your friends, and their friends, and so on. Displaying these sorts of properties of the social network is a neat feature that shows just how few degrees of separation there are between people. Along these same lines, it would be interesting to find and display "friendship chains" that show the shortest sequence of friendship relations that create a chain from one profile to another. For

example, if X is a friend of Y, and Y is a friend of Z, then a friendship chain exist that goes: X → Y → Z. Finding longer chains can be a fun and challenging problem.

- *Adjust the profile display as the application window is resized.* You got some practice with this already with the NameSurfer application and it would be an interesting extension to apply some of those same ideas here. The more challenging issue is how you would decide to change font sizes and the size of the image as the display size grew or shrank.
- *Go nuts!* There's really no shortage of ways that you could extend your SteamTunnel application. In fact, whole companies have been started based on creating a social network application with some cool new features. And if you do end up starting the next multi-billion dollar company based on social networking, just remember where it all started... CS106A!