Understanding Portfolio

Java Collections

I did some work with Java collections. I read up on all of the different types of collections and wrote up some code examples of an ArrayList, HashMap, and a TreeSet. These can be found at:

<https://github.com/Hope-Jordan/CIT360/tree/master/src/Collections>

there are three files there, one for each.

I also taught on the subject in a team meeting <https://plus.google.com/u/0/114178018165564065704/posts/EQgbcPxqigj>

I taught from 41:56 to 54:28.

I added an example of using Iterators for a LinkedList, HashSet, and HashMap. This can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/Collections/CollectionIteratorSandbox.java>

JSON

I have also been working with JSON. I have only got some examples using Bro. Barney's JSON library stringify method. But what I have so far can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/JSON/JSONSandbox.java>

MVC

This week I focused on Learning and understanding the MVC pattern. I think I have a good hold on what it is accomplishing and wrote an example of a portion of my app. There are other technologies that I want to integrate with it soon, but what I have so far can be found at:

<https://github.com/Hope-Jordan/CIT360/tree/master/src/MVC>

In this code, recipe.java has main and calls recipeControl’s method register. The control code then uses the view code to ask how many users the user wants to register (this will be taken out when the hibernate code and database are implemented, you can currently register multiple users to test the “database” in the model to see if the username selected already exists, since the set that is storing the usernames is empty at the start of when the app is run.) The control code then uses views getUsername and getPassword methods to create a new userBean. The control code then checks to HashSet in the model to see if that username has already been stored, and if it has prompts the user to try again, and if it the username is unique to what is stored in the HashSet it stores the username in the HashSet, confirms that the user registered successfully, and then uses the view code method displayUsernameAndPassword. register() will then calls the model’s method printSet to show what usernames are stored.

Some things I want to add, are handlers, a part of Application Controller Pattern, to have better control code, Hibernate, to use in my model to store and retrieve data from a database, and JSON to pass information between the model, control and view code.

Threads, Runnables, Executors

This week I have spent a portion of my time learning about Threads, Runnables, and Executors. I have read a lot of different things about parallel processing. Essentially there are three ways to create a thread. You can extend the thread class and create threads using that class. I have an example of this that can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/Threads/ThreadSandbox.java>

Threads are used to prevent an application from hanging, to be able to do multiple things in an app at the same time.

HttpUrlConnection

The other part of this week, I have been working on HttpUrlConnection. I know that this creates a connection to an http server. I played around with some of the different methods that HttpUrlConnection has, this can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/HTTPConnection/HTTPCommunicationSandbox.java>

To be completely honest, I don’t entirely understand what I am supposed to do with the connection to the server, or how to communicate with it after I have the connection. I have read a bit of the documentation, but a solid example of how it is used effectively would be nice to have.

Hibernate

I worked on hibernate the last couple of weeks. It was not very easy for me, mostly because I have never had any experience with MySQL databases before so I feel I had to teach myself a lot of that stuff. I’m not entirely sure of what all I need to do for hibernate, but here are a few of my examples from my project.

This is an example of a Hibernate Entity:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/JavaBeans/userBean.java>

I put my Hibernate code in my recipe model, to be used with MVC, I also used List to store the data retrieved by the queries, to be used in the application. This example can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/MVC/recipeModel.java>

My Hibernate config file is here:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/MVC/HibernateConfig.java>

Client and Server Socket Communication

I was also able to work on Client Server Socket communication this week. It felt like this topic was pretty strait forward and easy to understand compared to some of the other things that I have been working on recently. My server code can be found at:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/Sockets/Server.java>

In this code I use an example of a hashmap to store a message and a command associated with that message. I use JSON to serialize the hashmap and send it between the server and the client. I also use an Executor with a thread pool. I was wondering, the executor is pulling from Executors thread pool, is this what would make it so multiple clients would be able to connect with this server? Each connection would be running in its own thread? I guess I’m not entirely sure how that works.

The client code can be found here:

<https://github.com/Hope-Jordan/CIT360/blob/master/src/Sockets/Client.java>

Application Controller Pattern

I have been studying up on the Application Controller Pattern. I actually really liked learning about this topic. As far as I understood the topic, it is a pattern that helps modularize control code for an application. It makes it really easy to pick and choose what your application will and will not do at any given time. It prevents having to have all of your control login in large if else statements that decide when certain logic will need to be applied and when it will not. I used the application control pattern in my server code from my server client socket communication example.