**JSON Serialization**

This was my assigned topic this week in our group presentations. I greatly enjoyed understanding this topic, as I had recently been exposed to JSON this past Christmas while participating in the Advent of Code challenges. I started with the link provided in the class topics at the following link:

<http://www.w3schools.com/json/default.asp>

It explained that JSON is just a format for storing and exchanging data. It was very simple to understand and has some very simple rules to follow. I then found a few YouTube videos that helped me understand the process of serialization to and from JSON.

<https://www.youtube.com/watch?v=vBkuLu-ycEc>

<https://www.youtube.com/watch?v=WWa0cg_xMC8>

I was then able to research more on how to serialize to and from JSON by researching examples on StackOverflow.com. Many of the examples provided used the Google GSON library which was very easy to understand and use. In my coding example that I have attached below only utilizes the toJson and fromJson methods from the GSON class type. This application creates a class called Vehicle and instantiates two instances of Vehicle named vehicle1 and vehicle2. It then changes the data to JSON and saves it to a file. With that completed, I wanted to see how easy it was to pull it in from a file, so the rest of the application reads the data from the file, converts it back to a Vehicle object and displays it’s data.

<https://github.com/hodges-olan/CIT360-Portfolio/tree/master/CIT360-Portfolio/src/JSONSerialization>

You can see me teaching the group about JSON as well as Java serialization in the following YouTube clip.

<http://www.youtube.com/watch?v=zEmwbfhQ3bQ>

**Hibernate**

After reading all of the links provided in the class topics section, I was feeling a little overwhelmed on this one topic. I understand that it provides a way for Java developers to interact with multiple types of databases. I also understand that it allows you to put Hibernate between your application and the database, providing you with the XML configuration file to remap objects to database entries instead of having to recode your entire application due to some simple database changes. The link that I found most helpful was the Tutorials Point website. They started from the beginning with an understanding of the architecture of Hibernate, down to examples on how to utilize it.

<http://www.tutorialspoint.com/hibernate/index.htm>

I decided to use their example to start my knowledge and understanding of how to use Hibernate. It was a simple employee database that would give you their entry within the database including their first and last name as well as their salary. I used my database server at home to perform this test against, which is running MySQL. Here is a link to their example.

<http://www.tutorialspoint.com/hibernate/hibernate_examples.htm>

And here is a link to my code and configuration files where I was able to finally get it to run successfully.

Main Class (Hibernate.java) - <https://github.com/hodges-olan/CIT360-Portfolio/blob/master/CIT360-Portfolio/src/Hibernate/Hibernate.java>

Employee Class - <https://github.com/hodges-olan/CIT360-Portfolio/blob/master/CIT360-Portfolio/src/Hibernate/Employee.java>

Employee Mapping XML File - <https://github.com/hodges-olan/CIT360-Portfolio/blob/master/CIT360-Portfolio/src/Hibernate/Employee.hbm.xml>

Hibernate XML Configuration File - <https://github.com/hodges-olan/CIT360-Portfolio/blob/master/CIT360-Portfolio/src/hibernate.cfg.xml>

In their code, they establish a connection to the database and add three employees to that database, returning the integers for the unique identifiers from the database (the primary key). They list the changes in the listEmployees() method call. They then update employee #1, and delete employee #2. Finally they list the data again to show the changes.

**Java Collections**

It was interesting to learn that certain classes that I had already been using all inherit the Collections class in the end. Using the links provided to us from the class, I was able to learn that this inheritance provides for a general set of tools inherent to all Collections type classes, but that after that each of them implements their own flavor of tools to work with that particular class. Some of these collection types are Lists, Maps, and Sets.

<http://docs.oracle.com/javase/tutorial/collections/index.html>

<http://www.tutorialspoint.com/java/java_collections.htm>

This past Christmas, a friend of mine introduced me to this Advent called Advent of Code. Each day you are challenged with different problems that Santa has encountered and you need to solve. On Day 7 was my first run in with using a collections type, even though I didn’t know it was inheriting from the Collections class. I used ArrayLists, Maps, and HashMaps to solve a problem with some bitwise logic gates. They wanted to know what the value would be at the end of a logical circuit after providing certain inputs. I used a HashMap with a matching ArrayList to accomplish this. The HashMap provided the circuit layout while the ArrayList handled the values for each wire. You can see this example and run it at the following Github link. Don’t forget to also use the provided text file for the input.

Java File - <https://github.com/hodges-olan/AdventOfCode2015/blob/master/src/Day7/Day7.java>

Text File - <https://github.com/hodges-olan/AdventOfCode2015/blob/master/day7.txt>