**Parallel Processing**

**Summary:** Parallel Processing is something that exists all over the place; we simply don’t always know it. In its simplest form, parallel processing allows an application to perform different tasks on various threads simultaneously. This code creates a simple client-server socket, then allows the use of multiple clients to connect simultaneously without having to wait for the previous client to disconnect from the server. This is accomplished through creating a ThreadPool and executing a runnable via an executor every time a new client connects. This sample code also uses Socket I/O as well as Java Collections, QCJSON, and HTTP communication.

**Code from Internet:** I got the following code samples from Bro. Barney’s GitHub, where he shows the use of ThreadPools and Runnables in parallel processing. The first shows how a ThreadPool is used to execute a runnable: <https://github.com/yenrab/real_life_java/blob/master/SillyThreadPoolExample.java>

And the second shows exactly what a runnable is: <https://github.com/yenrab/real_life_java/blob/master/SimpleRunnableExample.java>

**My Sample Code:** This coding topic has two repositories: one for a client and another for a server. On the client side, found here: <https://github.com/Lundberg-Jonathan/Multithread-Client/blob/master/MultithreadClient/src/multithreadclient/MultithreadClient.java> the client simply connects to a server via a network socket. At that point, the server, found here: <https://github.com/Lundberg-Jonathan/Multithread-Server/tree/master/MultithreadServer/src/multithreadserver> uses 3 classes to first, fetch data on 25 Pokemon using HTTP then parses out various pieces of pokemon info (GetPokemonInfo.java). It then goes into a ‘listening’ state. It creates a ThreadPool and executes a runnable when a new client connects (MultithreadServer.java). The runnable (SendPokemonData.java) takes the existing socket and sends the pokemon data out to the client.

**Sharing Video:** <https://www.youtube.com/watch?v=6abvDUr2Nbc>

**Group Meeting Teaching:** Along with normal reasons for needing parallel processing, I shared why Android needs parallel processing (the AsyncTask) in order to run the main thread and the UI thread simultaneously, otherwise the app will crash: <https://youtu.be/VOB5NT3bSWA?t=6m35s> (ends at 7:00)

Also, I was able to share some code with my group mates and explain what is going on in the threads: <https://youtu.be/VOB5NT3bSWA?t=7m36s> (ends at 17:00)