2019 WWU ACM Hackathon

My Experience
Feb. 18, 2019

The session started with brainstorming; this didn't take long, as we had settled on a project idea beforehand. The general idea, I knew, was that the program needed to create a valid URL, use that URL to get JSON data from the Alpha Vantage API, and then relevant data needed to be extracted for whatever kind of output we wanted.

The first task I tackled was connecting to the API, given a URL that Timothy's (my partner) function would pass to me. Like most of the other tasks here, I'd never done something like this, but as usual some quick Googling helped me figure out the basics, and that part of the program was soon operational. It converted the string URL into the URL data type, opened a connection, scanned the page that opened and made it into one huge string, then closed the scanner and the connection.

Next, I needed to convert the string into JSON data that could be parsed in order to properly retrieve the relevant data within. I settled on org.json as the library with which to attempt this (a decision I would later regret) and began coding.

When we had the project mostly finished (or so we thought), we tried testing it in various different ways using the functions that we had created that gathered various stats and also attempted to make predictions. Unfortunately, we discovered that their values were well off target, and eventually we found out that the JSON data was totally unordered, so when we thought a function was accessing various data points throughout the timeline, it was actually just grabbing random data points that happened to be at those array indices. Apparently, org.json treats the JSONObject type as an unordered collection of other elements, not as an ordered one. Thus, when the object that stored the relevant data for most of our functions was converted into a JSONArray, the array simply preserved the jumbled nonsense order of the elements in the object. This would have been an important thing to research *before* choosing org.json, and if I encounter a similar scenario (multiple available frameworks or libraries for a given task), it would be prudent to get a rough idea of fairly important information like this.

With that aside, and midnight come and gone, it was time to find a solution. When briefly skimming a Stackexchange post about what libraries to use (before I settled on org.json because it appeared most popular) I noticed Gson. Naturally, I wasn't thinking clearly after 14 hours or so of coding, but it seemed well-documented, and I assumed that if Google or its employees made it, it probably wouldn't be too hard to implement it.

I was both right and wrong. It took less than 15 minutes to replace the old type and function names with their Gson counterparts, and after that I went home and went to bed. When I woke up, I was greeted by the snapchat notification icon and a helpful message from Timothy showing an error: _____ is not a JsonArray. The whole point of part of my functions had been to convert a particular JsonObject into an array of keys in order to grab different data points from throughout the Json file using those keys without too much hassle, and an error here would mean the entire project save the Foreign Exchange values would be derailed completely.

After some bleary-eyed Googling and browsing of StackExchange, I started to realize that unlike org.json, Gson didn't allow for directly converting JsonObjects to JsonArrays, so I had to find a different solution. Eventually, I realized that I could use the keyset() function on the relevant object and then convert the resulting ordered set into an array. Implementing that and one other minor change completed the project.

Looking Back: What I Learned

(Written July 2019)

- **Ensure** that a library you want to use in a program has all the functionality you need it to have (otherwise, as in this case, a lot of time will likely be wasted).
- **Build and test** the program often if possible; if I had run some tests before leaving for the night, I would have found the issue and been able to think about it at home instead of being blindsided in the morning.
- **Plan and design** before writing code, in detail if you can. One of my professors, a Microsoft alum, once said something to the effect of "if you fully plan out how to implement your design before you even sit down at the keyboard, you will usually have to spend very little time writing and debugging code." That advice definitely would have applied here, though I didn't hear it until later that quarter.