Jonah Aney 10/11/25 CSD420 Module 11: JSON API

For this assignment I chose to research Gson API. Data exchanges need to happen among different applications and the JSON format makes it easier for both humans and machines to perform these operations. One of the most popular and accessible tools to use for this process in Gson. Gson is a library developed and maintained by Google. Gson is flexible and efficient as well as beginner friendly which is why I chose it as an API to research. Obviously we've worked with APIs before with python and I've heard it be discussed even outside of my classes but without having a lot of direct experience, I thought Gson was a good choice for me.

Gson is an open-source library designed to convert Java objects to JSON and back again. According to the repository, Gson supports "serialization and deserialization of Java objects, including preexisting objects without modification" (Google, n.d.). Methods such as toJson() for converting objects to Json strings and fromJson() for parsing JSON strings into Java objects help make JSON conversion very intuitive and straightforward. A nice thing about Gson especially for beginners is it requires little set up to start using when compared to other APIs. Developers only need to add the JAR file in the project making installation simple and easy, especially when using IDE like Eclipse or NetBeans.

The main feature of Gson library is the Gson class. This class handles serialization and deserialization. Developers are able to define how classes should be converted by the use of custom type adapters. This is especially helpful when working with legacy data. Gson also supports pretty printing which is somewhat of a self explanatory term. This means the JSON output is easy to read by being formatted with line breaks and indentation. Developers are also able to skip certain fields of data during serialization with an option called field exclusion. This can be done using @Expose or @SerializedName (TurtorialsPoint, n.d.). Gson can also work with nested objects and collections so object and arrays can be generated with minimal coding.
Gson has support for generic types and custom naming strategies. This feature is documented in Google's official user guide, which explains, the ability to, "serialize arbitrary Java objects including those with parameterized types" (Google, n.d.). Gson

instances can be configured using the GsonBuilder class, which has options like setting date formats, registering custom serializers. These features enables Gson to adapt to many data formats and programming styles yet the API remains simple to use.

When working with JSON data, Gson supports several processes. One of the main operations as mentioned before is serialization, converting a Java object into a JSON string, and then the reverse operation with deserialization. For large datasets Gson supports streaming processing using JaonReader and JsonWriter. Developers can also use the Tree Model API (JsonObject, JsonArray, and JsonElement) to manually navigate or construct JSON data when the structure is unknown at compile time. The tree model API provides a lot of flexibility and control over how data is read and written.

Gson was released by Google in 2008. It was an alternative to other JSON libraries like Jackson and org.json. It was designed to make JSON operations simple without needing to modify existing Java classes. Today, Gson has become popular and widley used, especially in development of Android and other Google owned projects. Due to is being open sourced, Gson receives much community support and continued updates through GitHub. (Google, n.d.).

To use Gson, developers can get it directly from the official GitHub repository or Maven Central. With in the GitHub project page are complied JAR files and the source code, as well as links to documentation like the user guide. The repository also provides pre-built ZIP files containing all necessary JAR files for manual installation (Google, n.d.). Maven users can add Gson as a dependency with one line of XML configuration, while those who prefer manual setup can download the gson, JAR file and add it to their project's classpath. In NetBeans, I added the JAR file to the libraries in the properties of the Java project, which I thought was pretty straighforward.

Gson remains one of the more accessible and efficient Java JSON APIs in today's tech landscape. It's great for all skills levels by having straight forward configuation, clear syntax, and a lot of flexibility. Combine that with its robust documentation as well as it's

support from Google and the open source community, Gson continues to play a major role in Java development.

References:

Google. (n.d.). Gson GitHub repository. GitHub. https://github.com/google/gson

GeeksforGeeks. (2022, December 15). *How to install Gson module in Java*. GeeksforGeeks. https://www.geeksforgeeks.org/java/how-to-install-gson-module-in-java/

Google. (n.d.). *Gson user guide*. GitHub. https://github.com/google/gson/blob/main/UserGuide.md

TutorialsPoint. (n.d.). *Gson – Quick guide*. TutorialsPoint. https://www.tutorialspoint.com/gson/gson_quick_guide.htm