DD2480 - Report Assignment 4

Group 23
Lovisa Sjöberg
Oscar Arbman
Ziyuan Jia
Hugo Larsson Wilhelmsson
Erik Diep

March 7, 2025

The Gson library is a Java extended library which handles converting Json-objects into Java objects and vice versa.

OpenSource name: Gson
OpenSource URL: https://mvnrepository.com/artifact/com.google.code.gson/gson
OpenSource Github: https://github.com/google/gson
Project Group's repository URL: https://github.com/KTH-DD2480-group23/gson
Date: March 7, 2025

Contents

1 Onboarding Experience

The group decided not to move forward with the previous project since issues were not visible and they denied us access to this feature. Then we found another project, `gson`, which is a project which enables direct conversion between json-objects to Java-objects and vice versa. This repository had many open issues (192 issues) and even more closed ones (1007 issues). When browsing through these issues it was clear many issues that probably should have been closed, were still open, so features the group thought they could fix was already fixed through a connected PR.

Using the project within a Java-project was not too difficult as it supports Maven, so for Maven Java projects the integration was seamless with no problems. It also supports Gradle or downloading the .jar file.

2 Effort Spent

A full hour report is submitted in a separate document for improved readability of this report. It can be viewed on the `patch` branch here named "full_hour_report.pdf".

2.1 Lovisa

I spent a lot of time on the code, both configuring my setup and analyzing the code. Setting up the repository was painless using Maven, but there needed to be some workaround to be able to test the library in another Java test project. I needed to configure the test project to use the locally forked repository's snapshot instead of the Maven's Gson repository to be able to test my own adaptations to the Gson library locally. This setup the rest of the group also utilized when changing the logic of the code (i.e. not tests).

Then most of the time was spent on writing adaptations to the code and testing it through the setup. As I had to clean install every time I wanted to test the new version this took some time. Note that the time spent analyzing, writing code and running code is done interchangeable and not exactly one at a time. Finally I created the UML diagram and talked about the design patterns and architecture of our patch.

2.2 Hugo

I attended every meeting and we worked a lot together with the assignment during the meetings so I would assume I have spent around 6 hours in meetings during this assignment. We discussed nearly everything in the whole group, through the meetings, or through our communication channel, Discord, so I think I have spent less amount of time in discussions within parts of the group (maybe 4 hours).

Most of the time spent in this assignment was by reading documentation and configurating the setup. I had some problems setting up the project on my computer ans I was also responsible for finding and writing about the requirements so reading a lot of documentation was needed (I would say that this took around 11 hours).

Since I was responsible for finding the requirements, I did not write very much code, instead I was analyzing most of the time I interacted with the actual code. This included running the code. I would say I spended around 6 hours doing this.

2.3 Oscar

I spent a good amount of time trying to understand how gson library works, understanding Marcono1234 proof of concept code for this issue, that was writen in 2023. I also worked on improving his proof of concept to accuratly solve the issue, for achiving this Lovisas test enviorment was a great help. I have attended all but one of the meetings we have had, we all have discuss issue and studied the implementations.

2.4 Erik

I spend a significant amount of time trying to figure out the Gson library and researching the requested feature of the issue due to my lack of experience of Json. After having an ok grasp of the key functionallity of Gson (serialization and deserialization), I wrote some of the tests for our implementation of TypeAdapter. I have attended almost all of the meetings we have had, we all have discuss issue and studied the implementations. I have also contributed with improving the test file with some additional tests.

2.5 Ziyuan

I first tried to resolve the "Method fromJson throws JsonParseException instead of declared JsonSyntaxException" issue, after spending a good amount of time studying the Gson library and the exception structure, I've come to realize that the issue was caused by an incorrect use case and the conversion from JsonParseException to JsonSyntaxException I would've implemented will break the project's exception handling structure. So this issue was scraped.

Then I joined with others and worked on writing new test cases for the newly added adapter class, which actually found a bug in the new adapter. Then I collaborated with the others on fixing the bug and writing the report.

3 Overview of Issue(s) and Work Done

3.1 Method fromJson throws JsonParseException instead of declared JsonSyntaxException

URL: https://github.com/google/gson/issues/2816

To solve this issue, we want to modify the $fromJson$ method to catch the JsonParseException thrown by a custom adapter.read() and convert it into a JsonSyntaxEcception.

This solution later turned out to be unnecessary as we can actually define the custom adapter.read() to throw a JsonSyntaxEcception directly, which should be done by the user and is actually a better approach because JsonParseException is actually a parent class of JsonSyntaxEcception, and converting all JsonParseException into JsonSyntaxEcception could have unexpected consequences. Thus this issue was abandoned.

3.2 Parse nested json into flat model class objects

URL: https://github.com/google/gson/issues/2555

To solve the issue, we want to parse the json into the target model class without creating a nested class inside the target class.

Right now, we parse the json into the target model class but with a nested class in the target class. For example, the json can look like this:

```
{
    id : 1 ,
    name : "myname",
    address : {
    street : "my lane",
    city : "mycity" }
}
```

Instead, we want the json to look like this:

```
{
    id : 1 ,
    name : "myname",
    street : "my lane",
    city : "mycity"
}
```

4 Requirements for the New Feature of Requirements Affected by Functionality Being Refactored

4.1 Issue: Parse nested json into flat model class objects

In this section, we will discuss the functional requirements for implementing JSON flattening in Gson. Each requirement ensures that the solution meets the necessary serialization and deserialization.

To solve the issue of parsing nested JSON fields into flat model class objects without nested classes inside the target class, some requirements must be met. These requirements are:

Req 1: The implementation has to integrate with Gson's internal mechanism, ensuring it works seamlessly with Gson's APIs. Meaning the implementation should be backwards compatible and not ruin old implementations of the library.

Req 2: It requires transforming nested JSON objects into a flat structure concatenating nested keys using a separator ".".

Req 3: It must be able to reverse the flattening process by reconstructing the nested JSON object from the dot separated keys.

Req 4: If the current JSON objects have keys containing the separator ".", or after flattening it contains duplicate keys, the implementation must throw an IllegalArgumentException error.

5 Code Changes

5.1 Patch

View Patch

As Gson is a Java library, it can be imported within the users' own Java projects. Then that Gson-object can be used for serialization/deserialization of JSON- and Java-objects without reinstantiating it again. Customization for one's serialization and deserialization is possible where User guides on their Github explain how to do so. The issue concerned such a feature, wanting to adapt nestled referencing for Json-objects. The response to the issue suggested that this adaptation could be done in the user's own project but we wanted to create a patch which integrates this feature within Gson.

Branching off of the response to the issue from a prominent developer of the `gson`-library - it was suggested to customize a TypeAdapterFactory within the user's own Java project. Then instead of each user adding this adaptation to their own project, this customized FlatteningTypeAdapterFactory can be

instantiated and available when creating a Gson object with GsonBuilder registering the typeadapterfactory.

So the goal was to create an FlatteningTypeAdapterFactory class, which implements the abstract class TypeAdapterFactory class, with the proposed proof of concept code from the issue's response. The factories of the Gson handles how objects are created based on different types, and if the FlatteningTypeAdapterFactory is registered with GsonBuilder, then this adapter will serve as a translation layer to flatten the json object to the depth of 1, before calling the delegate adapter to convert the 1-depth json object to the corresponding model class for use.

6 Test Results

The log output from before any adaptations are in file `log.txt` and afer in `log_after.txt` and can be found on the branch `patch` here.

7 UML Class Diagram and its Description

7.1 Key Changes/Classes affected

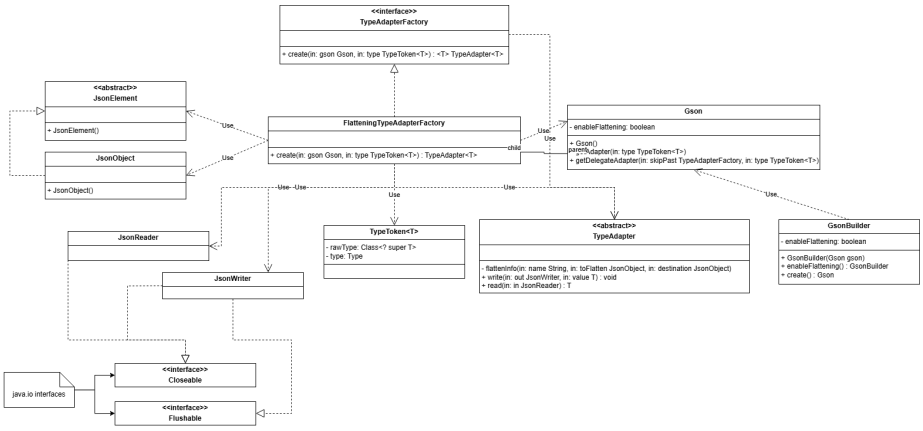

Figure 1: UML diagram connected to the added class FlatteningTypeAdapterFactory

In the UML class diagram certain variables and functions have been selected for visibility to enhance the readability. Since these classes contain on average more than 10 functions which are not really of importance to the patch, and over 20 field variables, we decided to keep the information to a minimum to get a good understanding of the code build for the patch. In the diagram the class `FlatteningTypeAdapterFactory` is added by us, which handles logic for flattening `json` objects. The dependencies, the used classes, are denoted with

dashed lines named “Use” as they are dependencies for the class’ logic but not for the class’ existence. Then further those dependencies are further expanded into their dependencies as many of the classes are derived from either an `gson` or Java defined interface or abstract class.

Furthermore the diagram shows the realization relationship between `FlatteningTypeAdapterFactory` and the interface `TypeAdapterFactory` it implements, which both use the abstract class `TypeAdapter` for reading and writing operations which handles the serialization and deserialization logic of the Java objects and `json` objects the factory handles.

Finally the dependency relationship between the factory and the Gson class. The factory uses a Gson-instance as parameter for its `create()` function, see Figure 1, and Gson adds an instance of the factory depending on flattening has been enabled - thus creating a parent-child relationship from the perspective of the Gson-class. The GsonBuilder enables flattening by creating an adapted instance of a Gson instance - which is why the relationship is “Usage”.

7.2 UML diagram relation to design patterns

The UML diagram, showcases a commonly used design pattern throughout the Gson project which is factory implementation. Since Gson have a standard way to implement factories to handle different type adapters to handle data in different ways. In their API docs they explain how to create a factory which writes all characters in lower case for example. Our situation is similar as we want the input data of a nested `json` to be flattened, which we then could do by implementing a TypeAdapterFactory which handled this logic. As the Gson object is instantiated with several default factories that handles common data, for example standard types strings, booleans etc., it would follow the convention by implementing this logic through a new factory.

8 Overall Experience

Our main take-aways from this project is that it is quite hard to work in an open source project. You need to read a lot of documentation, fight with setting it up on your computer and then understand all dependencies so you can contribute to the project. When finding issues to work on, you need to have some things in your mind. The first thing is to choose an issue that no one is working on. If someone is assigned to the issue, it is easy to understand that someone is working on it, but not everyone assign people to issues, but instead just start working on them.

It is also hard to know how long time it will take to solve an issue. The first issue we choosed didn’t sound very easy but it ended with that we solved it in 3 lines of code. The second issue we choosed, on the other hand, took a lot of time to solve but sounded pretty easy.

8.1 How did we grow as a team?

To evaluate how we have grown as a team, we use the Essence standard.

8.1.1 Principles Established

Checkpoints, principles and constraints were discussed and established amongst the team members in regards to the assignment's constraints and the course content's principles. We focus the work on what the graders will grade, for which the context of the assignment and project is built upon. We continued to use many of the tools we used in assignment one and two, such as Git (GitHub). We decided to work with a Java project. Before the first meeting, everyone were sent with homework to read through, understand and think about the assignment. On the first meeting, we discussed different issues and choosed one. Since we are happy with the way we have been working with issues, PRs and merge conflicts, we decided to continue like we did on the other assignments. Between the first and second meeting everyone looked into the issue we choosed in the first meeting. In the second meeting, we discusses the issue and solved it, leading to choosing another issue to work with. We also divided the issue in smaller parts and assigning them to different people. In the remaining meetings we worked with the issue and wrote the report. Each meeting were multiple hours meetings where we worked together.

8.1.2 Foundation Established

As the tools were established, the foundation was laid and because we worked on different parts of the issue, we could work quite easily in parallel. Problems encountered with the parts of the issue, were resolved both in creating new issues in the repo but also by communication via the agreed social platform (Discord) to avoid multiple members to work on the same bugs or missing features. After the previous assignments, everyone in the team was familiar to the other team members as well as we are quite similar of the way we are working, which has been improved after assignment one, two and three.

8.1.3 In Use

As the assignment are wrapping up and main functionalities are in process, the practices and tools have been used for real work. The issues and PRs are regularly reviewed, and are fixed by reviewers in case of merge conflicts or errors. Issues were set up continuously during the assignment to guide the work forward, but later issues have been more to address newly arisen errors or misinterpretations of the assignment, which showcases the group's way of working adapting towards the assignment's goals. The functionality of issues and commenting on reviews have also been a great way of understanding other people's code and pointing out misunderstandings.

8.1.4 In Place

We have worked on the parts of the issue quite separately so we do not have a perfect understanding of all parts in the code. On the other hand, we reviewed each others results in for example PRs which gave broad understanding of the whole project to all group members. To split the work like we did, led to that everybody is familiar with all the procedures and gives everyone something to do on their freetime away from the meetings. Everybody also has the right to handle the repository for the code and everybody is part of the meetings to review work that has been done or needs to be done.

8.1.5 Working well

In the end of assignment four, the team has gotten into the new practices, and mishaps and misunderstandings of using tools which occured in the previously assignments happens rarely. When problems occur we communicate via Discord and support each other so everyone can learn and contribute to the assignment.

8.1.6 Retired

Since this is the last assignment in the course, the group will after the presentation retire. The lessons learned in the way of working will be shared for future use.

8.1.7 Conclusion

Members of the group may sometimes, but rarely, need more practice. The need for more practice is way less than in the previous assignments. Therefore, we would say that we finally have reached the "Working Well" state. When problems occur we easily help each other out which results in that we are working well as a team. At the moment we think we just need to hold on to the things learned and developed in this team and to share our knowledge in future group projects at KTH or in the industry. Of course we will continue to learn more in future projects but right now we are happily with the results of our way of working we learned and used in this course.

8.2 How would we put our work in context with best software engineering practice?

Other than the Team and Way of Working alphas in the SEMAT kernel we could argue that the outcome of the patch and code changes agrees with the issue's proposed solution. From a stakeholder perspective, this solution integrated the directly into the Gson library which enhances the user experience as each individual user does not have to add the logic of flattening into their own projects. If the patch would to be accepted and integrated with the library, this added functionality would benefit the end users of Gson with more opportunities without having to add their adaptations.