

Final Report

Object Oriented Programming Project

Jari van den Broek, Johannes Hagspiel, Nathan Ordonez, Nick van Riet, Lukas Zimmerhackl, Ardy Zwanenburg | CSE1105 | 11-04-2019

Table of Contents

[General 3](#_Toc4678492)

[The Planning 3](#_Toc4678493)

[The Collaboration 3](#_Toc4678494)

[The Communication 3](#_Toc4678495)

[Using GitLab 3](#_Toc4678496)

[What did we learn as a group 3](#_Toc4678497)

[Design decisions 4](#_Toc4678498)

[Work division 4](#_Toc4678499)

[Technological Decisions 4](#_Toc4678500)

[The Server 4](#_Toc4678501)

[Client-Server Communication 5](#_Toc4678502)

[Point for improvement 5](#_Toc4678503)

[Software improvement 5](#_Toc4678504)

[Process Improvement 5](#_Toc4678505)

[Course improvement 6](#_Toc4678506)

[Individual Feedback 6](#_Toc4678507)

[Jari van den Broek 6](#_Toc4678508)

[Strong points 6](#_Toc4678509)

[Weak points 6](#_Toc4678510)

[Conflicts 6](#_Toc4678511)

[Johannes Hagspiel 6](#_Toc4678512)

[Strong points 6](#_Toc4678513)

[Weak points 6](#_Toc4678514)

[Conflicts 7](#_Toc4678515)

[Nathan Ordonez 7](#_Toc4678516)

[Strong points 7](#_Toc4678517)

[Weak points 7](#_Toc4678518)

[Conflicts 7](#_Toc4678519)

[Nick van Riet 7](#_Toc4678520)

[Strong points 7](#_Toc4678521)

[Weak points 7](#_Toc4678522)

[Conflicts 7](#_Toc4678523)

[Lukas Zimmerhackl 7](#_Toc4678524)

[Strong points 7](#_Toc4678525)

[Weak points 7](#_Toc4678526)

[Conflicts 8](#_Toc4678527)

[Ardy Zwanenburg 8](#_Toc4678528)

[Strong points 8](#_Toc4678529)

[Weak points 8](#_Toc4678530)

[Conflicts 8](#_Toc4678531)

[Sources 8](#_Toc4678532)

# General

This project was all about improving our knowledge of the Java programming language and seeing what it’s like to work with a team, in this case consisting of six team member, and a client. The goal was to build an application in which users would be able to track how much CO2 they have saved and also see how much their friends have saved.

## The Planning

The planning was done via the Scrum framework[[1]](#footnote-1). Scrum allowed us to set clear goals for each week and gave us a clear overview of what every member was responsible for and what still had to be done. After the weekly meetings on Monday, we assigned new tasks via the Scrumboard in our GitLab[[2]](#footnote-2). Every week we would make a Backlog and a Retrospective. The Backlog would contain the stories from the client and the tasks we assigned to our team members and the priority of the given task. The Retrospective would contain the problems we encountered during a certain week and how we would adjust to them to improve ourselves in the future.

## The Collaboration

Our collaboration went very smoothly. Everyone was committed to making every deadline and delivering a product that was solid. There were no major issues and everyone did their best, which resulted in a great working environment where everyone could speak their minds and voice their ideas, which greatly benefitted our creativity and thereby also the end-product.

## The Communication

The communication was mostly done during our weekly meetings on Monday and our group chat in WhatsApp. All our communication was in English so that everyone would be able to understand each other at any given point. If at any point we found it helpful to do so, we would also meet during other days of the week to discuss our work and help one another out with their issues. The communication went well and we always made sure everyone was involved in the conversation and got the chance to speak their minds.

## Using GitLab

GitLab was one of the most difficult parts about this project, but as we progressed throughout the project our understanding of its functions and the proper usage grew a lot. We did make some mistakes at the start of the project, such as committing directly to the masters, but with the help of our TA, we got better and better at using Git. At the later parts of our project, the biggest challenge was keeping our GitLab clean and our branches to a minimum.

## What did we learn as a group

The biggest takeaways from this project are the proper usage of Git, the better understanding of programming and how to use Scrum to manage your project.  
The usage of version control will be helpful in every project to come, and the same is true for the Scrum framework, which will help us in all our future projects, no matter what the subject is. The greater understanding of Java and, more in general, programming itself will benefit us greatly during our studies and will form a great foundation to build upon.

# Design decisions

## Work division

At the start of the project, we divided the project into several parts and assigned them to groups of people. The GUI was assigned to Ardy and Nick. They first worked on the login screen and after that created the second screen with the main application. Lukas assisted them whenever they needed him to and got rid of Checkstyle errors. The server side was assigned to Johannes, Nathan, and Jari. At first, they worked on making a server using telnet and combining that with a MySQL database. After a while, we switched to SPRING, which Johannes implemented with some help of Nathan and Jari. Nathan also set up a Raspberry Pi for continuous deployment. Making the backlog and retrospective for the Scrum aspects of the project was mostly done by Johannes, Nathan, and Jari. Making the Agenda was usually done by Nick. We all did our part in making sure everything was uploaded before every deadline. The Final Report and Presentation were mostly made by Jari.

## Technological Decisions

### The Server

In the development journey of our project, we implemented the server in two different ways: once based on telnet and then based on the SPRING framework[[3]](#footnote-3). Even though we achieved great results with our earlier version, our TA pointed out that we had to implement a REST API[[4]](#footnote-4), which was not possible while using telnet. Thus, we had to look for a second solution, where we landed on the SPRING framework. Making use of the tools provided by various SPRING projects such as SPRING Web MVC, we created a REST API. For the servlet container, we used Apache Tomcat and the underlying database was a MYSQL database. Additionally, we used hibernate for object-relational mapping[[5]](#footnote-5). The reason for these design decisions is that they were made following an in-depth tutorial that explained in detail how to best make use of the SPRING framework in the context of a REST API. Thus, what spoke for making use of these tools is that they were accompanied by a guide, that clearly laid out the strengths and the weaknesses of these tools as well as how to best implement them. Obviously, other tools were also considered, such as using a Postgres database. However, after researching the general consensus in the programming community as to the usefulness of all these tools and as well as after having taken into consideration the fact that already a working prototype was created, the decision was made to stick with these services. Looking back, the group unanimously would reaffirm this choice.

### Client-Server Communication

As stated previously, based on the TAs feedback, client-server communication had to be implemented via a REST API and via JSON[[6]](#footnote-6). The endpoints of the API were set up using the Web section of the SPRING framework and the parsing of POJO’s[[7]](#footnote-7) into JSON and back was achieved through the Jackson. In detail, the client would send, depending on the goal to be achieved, a PUT, GET, POST or DELETE request to a specific URL where the server then accepts this particular request, reads the body and then performs the appropriate action. Again, similarly to the server, this process was implemented by following a tutorial for the REST API and again, other alternatives were taken into consideration. However, after having seen that this particular way of implementing a REST API seems to be the standard, we decided to stick to it. Looking back at this decision, it is clear that this particular implementation provided us with the flexibility to add new and change already existing request endpoints without a lot of trouble. Conclusively, we no doubt would go down this path again.

# Point for improvement

## Software improvement

The biggest improvements to our application could be the implementation of our basic features. At this point the choices are all quite binary, you either did or you didn’t. For things such as lowering the temperature and taking the bike instead of the car, we could implement our features differently, like get the distance the user travelled by bike instead of by car and how many degrees did the user lower the heating. These improvements would give the users a more realistic number of how much CO2 they have actually saved. Another improvement could be the addition of information the user could access of their friends and the accomplishments of their friends.

## Process Improvement

Planning is something that could definitely be improved. We delayed our work until Sunday and then we started working on it, instead of doing it as soon as possible. On the week after our demo’s we would often not be very productive as there was no pressure of a deadline the next week. By setting clear goals and making sure everyone did their part every week we could improve our process a lot. We used the Scrumboard for this, but if we had put more emphasis on it, our process would have been better and our workloads would have been distributed more evenly over the weeks.

## Course improvement

The biggest improvements that should be made for next year would be the explanation of GitLab and giving us more good practice rules from the start. We made quite a lot of errors concerning Git that could have easily been avoided if that information had been more clear from the start. Another improvement would be more feedback from the TA on how many points we earned after a demo and if the team is doing well enough to get the points for subjects such as Scrum and the overall process.

# Individual Feedback

## Jari van den Broek

### Strong points

I learned a lot about programming in Java and programming in general. There were a lot of different ways to use Java which were really great to discover and see which works best for our purposes. I think my strongest point in this project was communication. Being active in all conversations and brainstorming with everyone and seeing what was still needed and doing that was great.

### Weak points

One of my weaker points was the difficulty of the things I coded. If I compare the level of my work to some of my peers, I think mine was a bit sub-par. Even though I learned quite a lot about the Java programming language, some of my teammates excelled at one subject while I stayed quite mediocre in all of them. Another weak point was my planning, a lot of the times I had to cram all of the work on the Sunday before the deadline instead of doing it over multiple days.

### Conflicts

There were no conflicts within the team, but that could also have been because there was a clear difference between how vocal everyone was. All of us improved our knowledge of Java, but not everyone improved their communicative skills during the project, which was a bit disappointing. Overall it was a good process and I was very content with my teammates.

## Johannes Hagspiel

### Strong points

Overall, I did learn a lot about Java programming, mainly about the SPRING framework. I think my stronger points were that I worked independently and on a self-motivated basis, which resulted in me creating on my own the server and the part of the client that is responsible for the communication with the server.

### Weak points

I think one of my weaker points was that I was not really able to explain how the server and the client-server communication worked. I feel particularly disappointed about this as I anticipated this and even told my teammates about it – I mentioned it as a weakness of mine in the README. Additionally, I did not really lead our team and push it during the periods when not much progress was being made, which was also partially caused by the point I mentioned earlier, my inability to teach other people about things that I have learned.

### Conflicts

As far as I remember, there were no major conflicts in the team. Even though on the surface, this looks good, I do think that this is also a sign of the fact that we did not engage too deeply into the areas that other people were responsible for. Overall, I think I had great teammates, I learned a lot about the SPRING framework and the importance as well as the difficulty of project management were highlighted again to me.

## Nathan Ordonez

### Strong points

### Weak points

### Conflicts

## Nick van Riet

### Strong points

### Weak points

### Conflicts

## Lukas Zimmerhackl

### Strong points

### Weak points

### Conflicts

## Ardy Zwanenburg

### Strong points

### Weak points

### Conflicts

## Sources

* https://www.scrum.org/resources/what-is-scrum - 28/03/2019
* https://git-scm.com/ - 28/03/2019
* https://en.wikipedia.org/wiki/Spring\_Framework - 28/03/2019

https://en.wikipedia.org/wiki/Representational\_state\_transfer - 28/03/2019

1. ‘Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.’ - https://www.scrum.org/resources/what-is-scrum [↑](#footnote-ref-1)
2. ‘Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.’ - https://git-scm.com/ [↑](#footnote-ref-2)
3. The SPRING Framework is an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application’ - https://en.wikipedia.org/wiki/Spring\_Framework [↑](#footnote-ref-3)
4. ‘Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services ‘ -https://en.wikipedia.org/wiki/Representational\_state\_transfer [↑](#footnote-ref-4)
5. Object-relational mapping is a programming technique that is used to convert data between incompatible type systems using object-oriented programming. [↑](#footnote-ref-5)
6. Java Script Object Notation. [↑](#footnote-ref-6)
7. Plain Old Java Object. [↑](#footnote-ref-7)