# Lab Report

Abgabe <02>

<Jan> <Mühlbacher>

<Sebastian> <Toporsch>

<Tobias> <Weigl>

Repository: <<https://github.com/weTobias/2020-Abgabe02-Muehlbacher-Toporsch-Weigl>>

## Vorgehen

Übung 2 (25.06.2020):

Als erstes wurde besprochen welche Aufgaben jeder übernehmen soll. Wir einigten uns darauf, dass Tobias Weigl diesmal das Repository bereitstellt und die Tests schreibt, als auch die README erstellt. Jan Mühlbacher kümmert sich um die Queue und findet mit Sebastian Toporsch die Fehler. Sebastian Toporsch schreibt anschließend den Lab Report und hilft Tobias Weigl bei der Maven Site. Danach formatiert er den Lab Report entsprechend. Danach wurde besprochen wie jeder seine Aufgaben realisieren will und danach startete jeder mit seiner Arbeit.

#### Detailliertes Vorgehen Jan Mühlbacher:

Am Anfang habe ich eine neue Klasse namens “GenericQueue” erstellt. Diese befüllte ich anschließend mit allen nötigen Funktionen und schrieb die JavaDoc dazu.

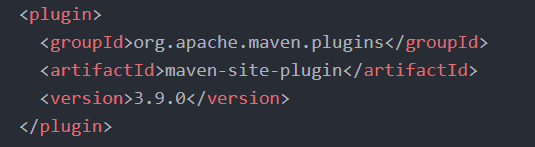
Danach besprach ich mit Sebastian Toporsch die möglichen Fehler der StringQueue-Klasse. Die 3 Fehler, de wir fanden waren:

* Der Konstruktor setzte die Klassenvariable “maxSize” nicht zum übergebenen Wert
* In der “poll”-Funktion musste der Vergleichsoperator von “==” zu “!=” gesetzt werden
* In der “remove”-Funktion musste die Zeile “element = “”” gelöscht werden

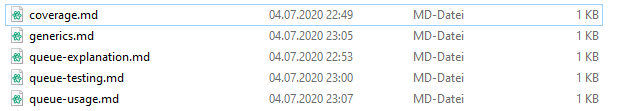
Danach pushte ich diese Änderungen.

Mit dieser groben Vorlage von StringQueue befüllte ich die GenericQueue-Klasse doch optimierte die Funktionen etwas. Diese Funktionen übernahm ich dann auch für die StringQueue-Klasse.

#### Detailliertes Vorgehen Sebastian Toporsch:

Als erstes arbeitet ich an der Maven Site. Damit die Erstellung der Site möglich war musste ich zuerst ein spezifisches Code-Snippet hinzufügen.

Danach erstellte ich zuerst die site.xml und erstellte die Inhalte, die in der Aufgabenstellungen verlangt waren.

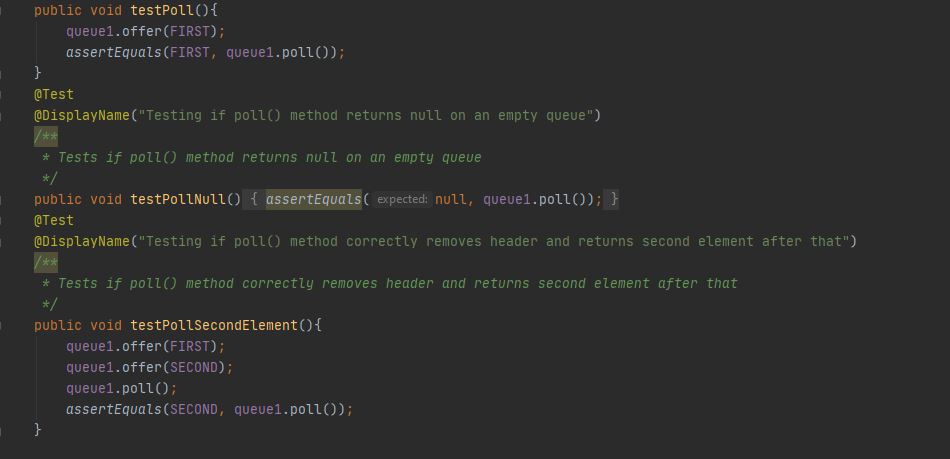


Ich versah dann die Dateien mit den benötigten Inhalten und benutze den Befehl „mvn site:site“ zur Erstellung der finalen Maven Site.

Sobald Jan Mühlbacher und ich die Fehler gefunden hatte, die in der StringQueue waren und er GenericQueue fertiggestellt hatte fing ich an den Lab Report zu formatieren. Dabei habe ich alle Anforderungen an den Lab Report überprüft und falls nötig verbessert.

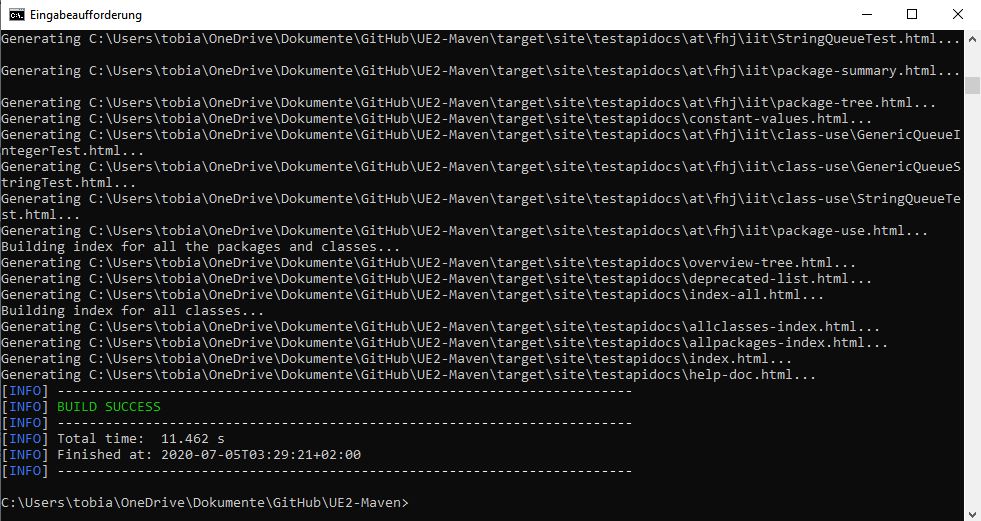
#### Detailliertes Vorgehen Tobias Weigl:

Als erste Aufgabe begann ich mit der Erstellung des GitHub Repositorys. Nachdem es erstellt war, habe ich meine Teamkollegen hinzugefügt und die Readme mitsamt Taskliste erstellt. Diese Tasks wurden dann abgearbeitet und als erledigt markiert, sobald die Aufgabe vollständig erfüllt war. Während meine Teamkollegen die Queue Implementation programmiert haben, habe ich parallel die Testklassen basierend auf der Dokumentation erstellt.



Als alles ausprogrammiert war, habe ich alle Tests durchlaufen lassen und 100% Coverage erreicht ohne Fehler.

Anschließend habe ich die pom.xml mit den nötigen Infos und Plugins befüllt, damit unser Build vollständig und fehlerlos erstellt werden kann.



Die Taskliste wurde während der Entwicklung immer wieder aktualisiert und neue Tasks wurden hinzugefügt um den Workflow zu steuern.

Sämtliche Änderungen wurden regelmäßig auf das GitHub Repository gepushed und mit erklärenden commit messages versehen.

## Links / Literatur

* Adaptieren Sie das Template für Ihre Bedürfnisse.  
  Beachten Sie dabei folgende Angaben.
* Entfernen Sie diese Checkliste
* Verwenden Sie folgendes Schema zum Speichern Ihres Lab Reports  
  <JAHR>-Abgabe<Nr>-<Nachname1>-<Nachname2>.pdf  
  **zB: 19-Abgabe01-Ulm-Hofer.pdf**
* Verwenden Sie dieselbe Namenskonvention für Ihre Abgaben bzw. das Repository  
  zB: https://<url-git-server>/19-Abgabe01-Ulm-Hofer/
* Min. 1 A4 für das jeweilige Kapitel in der Lehrveranstaltung, Dokumentieren Sie in einem sinnvollen Maß sodass Sie und StudienkollegInnen Ihre Dokumentation nachvollziehen können!