Alex Smolders & Silvan Dubach

bbw  alex.smolders@lernende.bbw.ch & silvan.dubach@lernende.bbw.ch

Zork game

# Dokumentation 426/226A Zorkgame

## 226A:

Idee:

Wir implementieren ein Zork game, welches aus Rätseln besteht, nützliche Items wie ein Schlüssel und auch Schildern und Schwerter für Feinde. Man muss durch den verschiedenen Räumen. Rätsel und Items aufsammeln damit man zum Exit Raum gelangt und befreit ist. Auf dem Weg muss man 2 Monster besiegen, nämlich den Goblin und den Drachen. Hierfür wird ein eigenes Fight System gebaut.

Map:

Diagram

Description automatically generated

## 426:

### Tag organisieren/Abmachungen

Wir hatten die Abmachung, dass wir jeden Schultag ein kurzes Daily Scrum haben, heisst wir besprechen kurz wie weit wir sind, was noch implementiert werden muss und unsere Ziele für den heutigen Tag.

### JuniT Tests

Wir haben uns darauf geeinigt, dass jeder von uns seine eigene Klasse testet. Wir haben unsere eigenen Methoden, die wir selbst implementiert haben, getestet. Wir haben keine Rücksicht genommen auf Getter und Setter, weder noch die Konstruktoren, ausser wenn diese auch Logik beinhalten. Schauen wie sich die anderen Werten verändern. Für Jede classe gibt es einen Test File.

### Clean code

Zum clean code haben wir gemeinsam abgemacht, die Java naming Convention zu nehmen, heisst Methoden sind deklarativ, aber von der Länge des Namens nicht zu lang. Was auch noch die Abmachung ist, ist das wir in allen möglichen Fällen eine Stream Methode verwenden. Zudem haben alle Methoden einen Weg getestet zu werden. Dies bedeutet, dass wir schauten mussten, dass es entweder eine Rückgabe wert hatte, einen Scanner für Inputs mit einem Byte Array. Möglichst wenig Methoden die nur System.out.println() beinhalteten, da es wie nicht testbar ist. Auch noch zu jeder logischen Methode wie auch Klasse haben wir ein Javadoc erstellt und zusammengefasst was die Funktion macht

### GitFlow

Wir haben 2 Branches für den Default: main und dev

Falls es ein Projekt für den Betrieb wäre, wäre der Main branch für PROD Environment, und für den development und Features zu testen haben wir den dev Branch.

Für Jede Klasse, eine neue Implementation von einem Feature, haben wir einen separaten Branch erstellt und dort unsere Features fertig geschrieben. Die Syntax lautet: «feature/FeatureName». Hier haben wir entwickelt und anschliessend alles getestet damit es gut funktioniert. Danach auf dev gemerged und wenn es dort gut funktionierte auf main.

Wo wir dann Tests schrieben haben wir den gleichen Ansatz genommen wie beim Feature, einfach mit der Namensgebung «Test/TestClass»

Zudem haben wir regelmässig committed und gepuscht, nach jedem feature wo gut programmiert wurde.

### Reflexion

Aus unserer Sicht konnten wir gut mit den gegebenen Umgebungen gearbeitet. Mit Github kamen wir beide klar. Die vorgegebenen Aufgaben des Projektes konnten wir gut umsetzen und auch noch ein Fight feature einbauen konnten zwischen den Player und einem Enemy. Bei der Entwicklung kamen wir immer gut voran und haben immer die Aufträge des Tages gut erledigen können. Was wir beide verbessern würden, ist das wir früher uns mit dem Gameplay des Spieles hätten beschäftigen sollen. Wir haben nämliche ein wenig spät unsere Map im Spiel erstellt und auch zu diesem dann noch Weiteres geändert welches weitere implementation brauchte.

Insgesamt sind wir aber beide zufrieden mit unserem Endprodukt und konnten das geplante Einhalten und auch gute Test schreiben zu dem Entwickelten.