Advanced Software Engineering Textbasierter Dungeon Crawler

Programmentwurf

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Sascha Vollmer

Abgabedatum:	30. Mai
Matrikelnummer, Kurs:	5700656, TINF19B5
Gutachter der Dualen Hochschule:	Maurice Müller

Eidesstattliche Erklärung

Wir Versichern hiermit, dass wir diese Programmentwurf mit dem Thema:

Advanced Software Engineering Textbasierter Dungeon Crawler

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Wir versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Oberkirch, den 30. Mai 2022

Vollmer, Sascha

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Übersicht über die Applikation . . . 1
- 1.2 Wie startet man die Applikation . . . 1
- 1.3 Technischer Überblick . . . 1

2 Clean Architecture **3**
- 2.1 Was ist Clean Architecture . . . 3
- 2.2 Analyse der Dependency Rule . . . 3
- 2.3 Analyse der Schichten . . . 4

3 SOLID **5**
- 3.1 Analyse SRP . . . 5
- 3.2 Analyse OCP . . . 5
- 3.3 Analyse [LSP/ISP/DIP] . . . 5

4 Weitere Prinzipien **6**
- 4.1 Analyse GRASP: Geringe Koppelung . . . 6
- 4.2 Analyse GRASP: Hohe Kohäsion . . . 6
- 4.3 DRY . . . 6

5 Unit Tests **7**
- 5.1 10 Unit Tests . . . 7
- 5.2 ATRIP: Automatic . . . 7
- 5.3 ATRIP: Thorough . . . 7
- 5.4 ATRIP: Professional . . . 7
- 5.5 Fakes und Mocks . . . 7

6 Domain Driven Design **8**
- 6.1 Ubiquitous Language . . . 8
- 6.2 Repositories . . . 8
- 6.3 Aggregates . . . 8
- 6.4 Entities . . . 8
- 6.5 Value Objects . . . 8

7 Refactoring **9**
- 7.1 Code Smells . . . 9
- 7.2 2 Refactorings . . . 9

8 Entwurfsmuster **10**
8.1 Entwurfsmuster: [Name] 10
8.2 Entwurfsmuster: [Name] 10

1 Einführung

1.1 Übersicht über die Applikation

Bei der Applikation handelt es sich um einen Textabsierten Dungeon Crawler, bzw. ein Framework mit dem ein solcher einfach umgesetzt werden kann. In der Applikation ist Test Text, sowie ein Test Dungeon enthalten durch das Nutzer sehen können was mit diesem Framework möglich ist. Diese Applikation hat als Ziel Dungeon Crawler ohne konkret Coden zu müssen anhand von speziel Formartierten Daten erstellen zu können.

1.2 Wie startet man die Applikation

write execution instructions

1.3 Technischer Überblick

Für die Implementierung wurden folgende Techniken eingesetzt.

1.3.1 Java

aus de zwei erlaubten sprachen ist Java dem Autor am vertauteren. Ist die haupt Sprache in der diese Applikation geschrieben ist.

1.3.2 Swing

Da für den Dungeon Crawler sehr viele Informationen auf einmal ersichtlich sein sollen und die Ausgabe des Textes in einer Kommandozeile hierdurch eingeschränkt ist, wurde sich entschieden die Textfelder in eine GUI einzubauen. Da Vorwissen in Swing vorhanden ist wurde sich für Swing entschieden.

1.3.3 Maven

für die Dependency Kontrolle wird Maven eingesetzt. Es werden nur 1 nicht Java interne Librarys verwendet,

1.3.4 org.json & JSON

Es wurde als Datenspeicherformat JSON gewählt da es von den grundlegenden eigenschaften her vergleichbar mit xml für dieses Projekt ist und eine präferenz für json vorhanden ist. Als Datenspeicher war eines der beiden notwendig um schnell und übersichtlich die informationen in die richtigen Klassen zu konvertieren. Für diese Konvertierung wurde zusätlich die org.json Library verwendet.

1.3.5 JUnit Testing

Als Testing Framework wurde sich für JUnit entschieden. Durch dieses werden die Unit-Tests für die Applikation durchgeführt.

2 Clean Architecture

2.1 Was ist Clean Architecture

add definition of clean architecture

2.2 Analyse der Dependency Rule

2.2.1 Positiv-Beispiel: Dependency Rule

2.2.2 Negativ-Beispiel: Dependency Rule

Die GUI Klasse welche für die SWING gui verantwortlich ist wird nicht sauber über Adapter Code abstrahiert. Die Dependency ist an sich falsch herum. Aktuell weiß die GUI Klasse nichts über die Main / BattleInterface Klassen. Grundlegend sollte dieses Wissen umgekehrt werden damit mein Domain Code

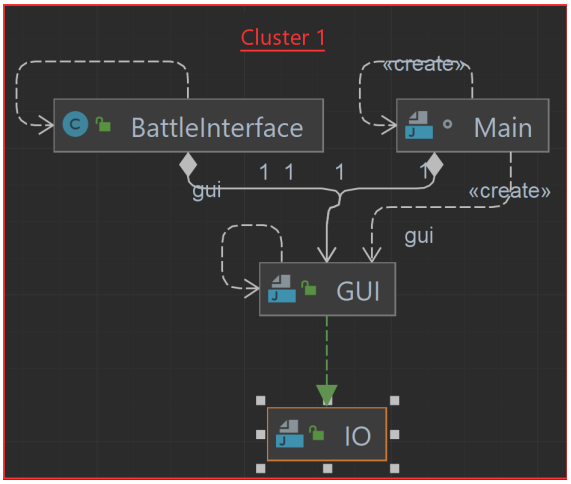

Abbildung 2.1: Alle Dependencys der GUI Klasse

2.3 Analyse der Schichten

2.3.1 Schicht: [NAME]

2.3.2 Schicht: [NAME]

3 SOLID

3.1 Analyse SRP

3.1.1 Positiv-Beispiel

3.1.2 Negativ-Beispiel

3.2 Analyse OCP

3.2.1 Positiv-Beispiel

3.2.2 Negativ-Beispiel

3.3 Analyse [LSP/ISP/DIP]

3.3.1 Positiv-Beispiel

3.3.2 Negativ-Beispiel

4 Weitere Prinzipien

4.1 Analyse GRASP: Geringe Koppelung

4.1.1 Positiv-Beispiel

4.1.2 Negativ-Beispiel

4.2 Analyse GRASP: Hohe Kohäsion

4.3 DRY

5 Unit Tests

5.1 10 Unit Tests

5.2 ATRIP: Automatic

5.3 ATRIP: Thorough

5.4 ATRIP: Professional

5.5 Fakes und Mocks

6 Domain Driven Design

6.1 Ubiquitous Language

6.2 Repositories

6.3 Aggregates

6.4 Entities

6.5 Value Objects

7 Refactoring

7.1 Code Smells

7.2 2 Refactorings

8 Entwurfsmuster

8.1 Entwurfsmuster: [Name]

8.2 Entwurfsmuster: [Name]