**Refactoring 2 with M. Fowler**

**Version:** 0.1

**Datum:** 13.02.2021

DOKUMENTVERSIONEN

|  |  |  |  |
| --- | --- | --- | --- |
| Versionsnr. | Datum | Autor | Änderungsgrund / Bemerkungen |
| 0.1 | 13.02.2021 | Dr. Marc Schanne | Portfolio-Prüfung in SE II |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

INHALT

[DOKUMENTVERSIONEN 1](#_Toc64110474)

[INHALT 2](#_Toc64110475)

[1. Einleitung 3](#_Toc64110476)

[1.1 Allgemeines 3](#_Toc64110477)

[1.1.1 Ziel und Zweck dieses Dokuments 3](#_Toc64110478)

[1.1.2 Projektbezug 3](#_Toc64110479)

[1.1.3 Abkürzungen 3](#_Toc64110480)

[1.1.4 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten 3](#_Toc64110481)

[1.2 Verteiler und Freigabe 3](#_Toc64110482)

[1.2.1 Verteiler für dieses Lastenheft 3](#_Toc64110483)

[1.3 Reviewvermerke und Meeting-Protokolle 4](#_Toc64110484)

[2. Anforderungsbeschreibung 5](#_Toc64110485)

[2.1 1. Anforderung 5](#_Toc64110486)

[2.1.1 Beschreibung 5](#_Toc64110487)

[2.1.2 Wechselwirkungen 5](#_Toc64110488)

[2.1.3 Risiken 5](#_Toc64110489)

[2.1.4 Schätzung des Aufwands 5](#_Toc64110490)

[2.2 2. Anforderung 5](#_Toc64110491)

[2.2.1 Beschreibung 5](#_Toc64110492)

[2.2.2 Wechselwirkungen 6](#_Toc64110493)

[2.2.3 Risiken 6](#_Toc64110494)

[Mittels Definition geeigneter Tests muss für das Refactoring eine geeignete Basis geschaffen werden, damit der bestehende Code ohne Sorge um funktionale Veränderungen überarbeitet werden kann. 6](#_Toc64110495)

[2.2.4 Vergleich mit bestehenden Lösungen 6](#_Toc64110496)

[2.3 3. Anforderung 6](#_Toc64110497)

[2.3.1 Beschreibung 6](#_Toc64110498)

[2.3.2 Wechselwirkungen 6](#_Toc64110499)

[2.3.3 Vergleich mit bestehenden Lösungen 6](#_Toc64110500)

[2.4 4. Anforderung 7](#_Toc64110501)

[2.4.1 Beschreibung 7](#_Toc64110502)

[2.5 5. Anforderung 7](#_Toc64110503)

[2.5.1 Beschreibung 7](#_Toc64110504)

[2.5.2 Wechselwirkungen 7](#_Toc64110505)

[3. Genehmigung 8](#_Toc64110506)

[4. Anhang 9](#_Toc64110507)

# Einleitung

Dieses Lastenheft ist Teil der Portfolio-Prüfung des Moduls Software Engineering II an der DHBW Karlsruhe. Die Prüfungsleistung als Portfolio enthält zwei Workshops zum Nachweis der Lerninhalte aus den Semestern 3 und 4, sowie Grundlagen aus dem Modul Software Engineering I.

## Allgemeines

In diesem Dokument werden konkrete Anforderungen für die Umsetzung in einem Phasen- bzw. Dokument-getriebenen Softwareenwicklungsprozess (Wasserfall) vorgegeben.

### Ziel und Zweck dieses Dokuments

Ziel dieser Spezifikation ist es Anforderungen des Auftraggebers (AG) an das vorliegende Refactoring-Projekt festzuhalten, damit der Auftragnehmer (AN) auf dieser Grundlage eine Lösung in einem Pflichtenheft spezifizieren kann.

Das Pflichtenheft ist dann Basis aller weiteren vertraglichen Vereinbarungen, insbesondere für die Definition der Abgabe-Artefakte mit entsprechenden Meilensteinen.

### Projektbezug

Das Refactoring-Projekt entspricht dem Code-Beispiel aus dem Buch von Martin Fowler. Refactoring: Improving the Design of Existing Code (2nd edition), Addison-Wesley, 2019.

### Abkürzungen

TBC …

### Ablage, Gültigkeit und Bezüge zu anderen Dokumenten

Dieses Lastenheft ist die Basis für ein vom AN zu erstellenden Pflichtenhefts, das in klarer Form beschreibt, wie er die Anforderungen des Lastenhefts zu lösen gedenkt.

Ausgangspunkt für die Refactorings ist ein Code-Repository auf Github. Hier sind neben dem Code auch dieses Lastenhefts und Vorlagen für ein Pflichtenheft.

## Verteiler und Freigabe

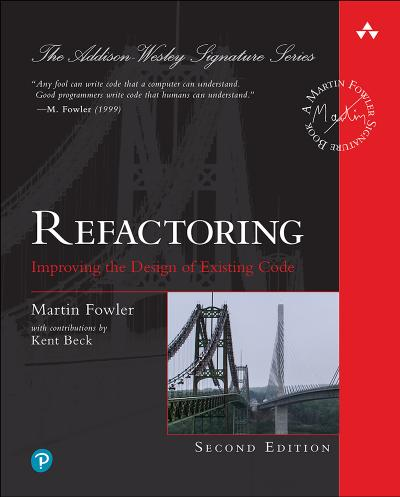
### Verteiler für dieses Lastenheft

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rolle | Name | Telefon | E-Mail | Bemerkungen |
| Projektleiter | Dr. Marc Schanne | 015 771 750 265 | marc@schanne.org |  |
| Kunde, AG | ??? |  |  |  |
| Projektteam, AN | ??? |  |  |  |
|  |  |  |  |  |

## Reviewvermerke und Meeting-Protokolle

Der im Github Repository hinterlegte Code enthält alle von Martin Fowler für das Refactoring-Beispiel seines Buches spezifizierte Klassen in Java, ab Version 8.   
Die verwendete Projektstruktur nutzt Apache Maven als Build-System.

# Anforderungsbeschreibung

Die Anforderungen dieses Refactoring-Projekts betreffen ein Theater-Projekt wie es Martin Fowler in seinem Buch Refactoring: Improving the Design of existing Code (2nd Edition) aus dem Jahr 2019 als Beispiel einführt.

Die Anforderungen sind aus Kundensicht und für das anstehende Refactoring muss der AN diese sinnvoll verfeinern und in einem Pflichtenheft die gesamte Entwicklung mit Meilensteinden planen.

## 1. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | 101 | **Nichttechnischer Titel** | | Projektplanung mit Meilensteinen und Definition von Artefakten | | |
| **Quelle** |  | | **Verweise** |  | **Priorität** | muss |

### Beschreibung

Vor Umsetzung des geforderten Refactorings (Anforderung 102) und notwendiger Erweiterungen (Anforderungen 105) ist die vollständige Planung des Projekts (Anforderungen 101, 103, 104) erforderlich.

### Wechselwirkungen

Artefakte des vorliegenden Refactoring-Projekts sind neben einer vollständigen und kleinschrittigen Projekt-Historie in einer Software-Versionsverwaltung (z.B. Github) zu archivieren. Dies umfasst insbesondere eine saubere Projektplanung mit Pflichtenheft und Meilensteinplan.

### Risiken

Die sinnvolle Dokumentation des Refactoring-Fortschritts durch Commit-Messages ist obligatorisch. Die dadurch entstehende Projekt-Historie muss dem AG eine Abnahme des Projekts in den definierten Meilensteinen ermöglichen.

### Schätzung des Aufwands

Für das gesamte Projekt steht dem AN nach Übergabe dieses Lastenhefts die Entwicklungszeit im Rahmen eines Workshoptages mit 4 Zeitstunden zur Verfügung. Die sinnvolle Strukturierung der notwendigen Arbeiten in 4 Meilensteinen (Zeitaufwand je 1 Stunde) ist somit angeraten.

## 2. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | 102 | **Nichttechnischer Titel** | | Refactoring anhand von Code-Smells | | |
| **Quelle** |  | | **Verweise** |  | **Priorität** | muss |

### Beschreibung

Funktionale Hauptanforderung des Refactoring-Projekts ist das „Cleanup“ des bestehenden Beispiel-Projekts. Die Verwendung moderner OO-Programmierrichtlinien und eine Software-entwicklung nach bekannten Best Practices muss das Ziel sein, damit der Code auch für zukünftige funktionale Erweiterungen und Anpassungen vorbereitet ist.

Refactorings analog Martin Fowler sind:

* Extract Method
* Move Method
* Replace Temp with Query
* Replace Type Code with State/Strategy
* Replace Switch with Polymorphism
* Form Template Method

### Wechselwirkungen

Die von Martin Fowler in seinem Buch vorgestellten Refactorings gelten als Richtlinie für die notwendigen Code-Verbesserungen durch den AN.

Mit Wissen über übliche Code-Smells muss der AN die bestehenen Unzulänglichkeiten des Beispielcodes beseitigen.

### Risiken

### Mittels Definition geeigneter Tests muss für das Refactoring eine geeignete Basis geschaffen werden, damit der bestehende Code ohne Sorge um funktionale Veränderungen überarbeitet werden kann.

### Vergleich mit bestehenden Lösungen

Die Nutzung bekannter Entwurfsmuster in der objektorientierten Programmierung sind obligatorisch.

## 3. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | 103 | **Nichttechnischer Titel** | | Softwareentwicklung nach Wasserfall | | |
| **Quelle** |  | | **Verweise** |  | **Priorität** | muss |

### Beschreibung

Die Softwareentwicklung mit einem Entwicklungsprozess nach Wasserfall-Modell ist obligatorisch und der AN muss notwendigen Dokumente für den Übergang der einzelnen Phasen im Entwicklungsprozess explizit nutzen.

### Wechselwirkungen

Genutze Dokumente muss der AN in einer Software-Versionsverwaltung archivieren.

### Vergleich mit bestehenden Lösungen

Auch wenn der klassische Prozess des Softwareentwicklung nach Wasserfall die sequentielle Entwicklung mit Weitergabe von Informationen innerhalb des (großen) Entwicklungsteams über Dokumente vorsieht ist für die Programmierung im kleinen Team auch die Nutzung von modernen Prozessmerkmalen denkbar.

Ansätze wie Pair Programming (zwei Entwickler vor einem Rechner, ein Driver, ein Navigator, im Wechsel), Test Driven Development (TDD) oder Continuous Integration (z.B. mit Jenkins) sind hier erlaubt.

## 4. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | 104 | **Nichttechnischer Titel** | | Werkzeugumgebung | | |
| **Quelle** |  | | **Verweise** |  | **Priorität** | soll |

### Beschreibung

Moderne Softwareentwicklung verlangt die Nutzung geeigneter Werkzeugunterstützung für Entwicklung (IDE), Build-Management, Software-Versionsverwaltung und Tests.

## 5. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nr. / ID** | 105 | **Nichttechnischer Titel** | | Funktionale Erweiterungen | | |
| **Quelle** |  | | **Verweise** |  | **Priorität** | kann |

### Beschreibung

Ausgangspunkt des Refactoring-Projekts ist eine schlecht strukturiertes Programm. Nach Refactoring (Anf. 102) ist es möglich die Funktionalität der Customer-Komponente zu erweitern und leicht neue Typen für die Komponente Play mit speziellem Verhalten zu ergänzen ohne die bestehende Logik verändern zu müssen. Als weitere Typen sind jetzt und evtl. zukünftig die folgenden Möglichkeiten im Gespräch: history, pastoral, pastoral­comical, historical­pastoral, tragicalhistorical, tragical­comical­historical­pastoral, scene individable, oder poem unlimited.

Bei diesen Erweiterungen ist das Open Closed Principle (OCP) – offen für Erweiterungen, geschlossen für Veränderungen – zu beachten.

Eine zusätzliche Erweiterung des Bestandssystems ist für den Druck der Aufstellungen geplant.

Hier soll neben der Ausgabe von Plain-Text auch HTML unterstützt werden.

### Wechselwirkungen

Die aktuelle Logik muss bei allen umgesetzten Erweiterungen unverändert erhalten bleiben.

# Genehmigung

Die Genehmigung erfolgt...

|  |  |
| --- | --- |
| Datum: |  |
| Unterschrift Auftraggeber: |  |
| Unterschrift Projektleiter: |  |
| Weitere Unterschriften: |  |

# Anhang