**PFLICHTENHEFT**

**Version:** 2.0

**Datum:** 20.06.2020

**DOKUMENTVERSIONEN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versionsnr.** | **Datum** | **Autor** | **Änderungsgrund / Bemerkungen** |
| 1.0 | 20.06.2020 | Gruppe G | Ersterstellung |
| 2.0 | 20.06.2020 | Gruppe G | Finalisierung / Detaillierung Anforderungen + Zuweisung zu Meilensteinen |
|  |  |  |  |
|  |  |  |  |

**INHALT**

Inhalt

[1 Einleitung 4](#_Toc43556349)

[1.1 Allgemeines 4](#_Toc43556350)

[1.1.1 Ziel und Zweck dieses Dokuments 4](#_Toc43556351)

[1.1.2 Projektbezug 4](#_Toc43556352)

[1.1.3 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten 4](#_Toc43556353)

[1.2 Verteiler und Freigabe 4](#_Toc43556354)

[1.2.1 Verteiler für dieses Pflichtenheft 4](#_Toc43556355)

[2 Konzept und Rahmenbedingungen 5](#_Toc43556356)

[2.1 Übersicht der Meilensteine 5](#_Toc43556357)

[3 Anforderungsbeschreibung 6](#_Toc43556358)

[3.1 102.0 Refactoring anhand von Code-Smells 6](#_Toc43556359)

[3.1.1 Beschreibung 6](#_Toc43556360)

[3.1.2 Wechselwirkungen 6](#_Toc43556361)

[3.1.3 Risiken 6](#_Toc43556362)

[3.1.4 Vergleich mit bestehenden Lösungen 6](#_Toc43556363)

[3.2 102.1 Anforderung 6](#_Toc43556364)

[3.2.1 Beschreibung 6](#_Toc43556365)

[3.2.2 Refactoring-Steps 7](#_Toc43556366)

[3.3 102.2 Anforderung 7](#_Toc43556367)

[3.3.1 Beschreibung 7](#_Toc43556368)

[3.3.2 Refactoring-Steps 7](#_Toc43556369)

[3.4 102.3 Anforderung 7](#_Toc43556370)

[3.4.1 Beschreibung 8](#_Toc43556371)

[3.4.2 Refactoring-Steps 8](#_Toc43556372)

[3.5 102.4 Anforderung 8](#_Toc43556373)

[3.5.1 Beschreibung 8](#_Toc43556374)

[3.5.2 Refactoring-Steps 8](#_Toc43556375)

[3.6 102.5 Anforderung 8](#_Toc43556376)

[3.6.1 Beschreibung 9](#_Toc43556377)

[3.6.2 Refactoring-Steps 9](#_Toc43556378)

[3.7 102.6 Anforderung 9](#_Toc43556379)

[3.7.1 Beschreibung 9](#_Toc43556380)

[3.8 101.7 Projektplanung und Definition von Meilensteinen 9](#_Toc43556381)

[3.8.1 Beschreibung 9](#_Toc43556382)

[3.8.2 Wechselwirkungen 9](#_Toc43556383)

[3.8.3 Risiken 10](#_Toc43556384)

[3.8.4 Schätzung des Aufwands 10](#_Toc43556385)

[3.9 104.8 Werkzeugumgebung 10](#_Toc43556386)

[3.9.1 Beschreibung 10](#_Toc43556387)

[3.10 105.9 Anforderung 10](#_Toc43556388)

[3.10.1 Beschreibung 10](#_Toc43556389)

[3.10.2 Wechselwirkungen 10](#_Toc43556390)

[3.11 10. Anforderung 11](#_Toc43556391)

[3.11.1 Beschreibung 11](#_Toc43556392)

[3.12 11. Anforderung 11](#_Toc43556393)

[3.12.1 Beschreibung 11](#_Toc43556394)

[4 Genehmigung 12](#_Toc43556395)

# Einleitung

## Allgemeines

### Ziel und Zweck dieses Dokuments

Dieses Pflichtenheft beschreibt die Anforderungen an das vorliegende Refactoring-Projekt, die auf Grundlage des Lastenheftes spezifiziert werden. Es ist die 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, Addison-Wesley, 1999, 1. Auflage.

### Ablage, Gültigkeit und Bezüge zu anderen Dokumenten

Dieses Pflichtenheft beschreibt das Vorgehen zum lösen der Anforderungen des Auftraggebers des Lastenheftes. Die einzelnen Meilensteine werden in der GitHub-Historie ersichtlich.

## Verteiler und Freigabe

### Verteiler für dieses Pflichtenheft

|  |  |  |
| --- | --- | --- |
| Rolle | Name | E-Mail |
| Projektleiter | Mathis Kaspar-Müller | Kaspar-mueller.mathi@student.dh-karlsruhe.de |
| Requirements Engineer | Monika Bichlmaier | Bichlmaier.Monika@student.dh-karlsruhe.de |
| Programmer | Nico Sasse | Sasse.Nico@student.dh-karlsruhe.de |
| Programmer | Tim Schneider | Schneider.Tim2@student.dh-karlsruhe.de |
| Tester | André Göller | Goeller.Andre@student.dh-karlsruhe.de |

Die Rollen im Projekt werden entsprechend dem Wasserfallmodell verteilt.

# Konzept und Rahmenbedingungen

## Übersicht der Meilensteine

|  |  |
| --- | --- |
| 1. Meilenstein: Vorbereitungsphase | |
| Erstellung des Pflichtenhefts | Abschluss:12:00 |
| Abnahme Pflichtenheft von Auftraggeber | Abschluss:12:00 |
| Produktentwurf | Abschluss:12:00 |
| 1. Meilenstein: Implementierung (Refactoring) und Test | |
| Extract Method | Abschluss: 14:00 |
| Move Method | Abschluss: 14:00 |
| Replace Temp with Query | Abschluss: 14:00 |
| Replace Type Code with State/Strategy | Abschluss: 14:00 |
| Replace Switch with Polymorphism | Abschluss: 14:00 |
| Form Template Method | Abschluss: 14:00 |
| 1. Meilenstein: Test und Einführung | |
| Tests anpassen | Abschluss: 14:30 |
| Erstellen eines Benutzerhandbuchs | Abschluss: 14:30 |

# Anforderungsbeschreibung

Nachfolgend werden die Anforderungen des Lastenheftes und weitere entstehende Anforderungen erläutert.

## 102.0 Refactoring anhand von Code-Smells

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 2 | Nichttechnischer Titel | | Refactoring anhand von Code-Smells | | |
| Quellen | Anf. ID: 102 | | Verweise | Lastenheft | 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 der Präsentation von Martin Fowler können den nachfolgenden Anforderungen (3.3 bis 3.8) entnommen werden.

### Wechselwirkungen

Die von Martin Fowler in seinem Buch, bzw. in der Präsentation auf der JavaOne 2000

vorgestellten Refactorings gelten als Richtlinie für die notwendigen Code-Verbesserungen

durch den AN.

Mit Wissen über übliche Code-Smells muss der AN die bestehenden 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.

## 102.1 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 1 | Nichttechnischer Titel | | Extract Method | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | soll |

### Beschreibung

Die Wahrscheinlichkeit für Wiederverwendbarkeit steigt mit feinerer Granularität. Code in kleine Methoden aufteilen.

### Refactoring-Steps

* Create method named after intention of code
* Copy extracted code
* Look for local variables and parameters
  + Turn into parameter
  + Turn into return value
  + Declare within method
* Compile
* Replace code fragment with call to new method
* Compile and test

## 102.2 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 2 | Nichttechnischer Titel | | Move Method | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | soll |

### Beschreibung

Methoden sollten in der richtigen Klasse deklariert werden, da sonst die Kapselung durchbrochen wird oder die Koppelung zu eng wird.

A method is, or will be, using or used by more features of another class than the class it is defined on. Create a new method with a similar body in the class it uses most. Either turn the old method into a simple delegation, or remove it altogether.

### Refactoring-Steps

* Declare method in target class
* Copy and fit code
* Set up a reference from the source object to the target
* Turn the original method into a delegating method
  + amountOf(Rental each) {return each.charge()}
  + Check for overriding methods
* Compile and test
* Find all users of the method
  + Adjust them to call method on target
* Remove original method
* Compile and test

## 102.3 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 3 | Nichttechnischer Titel | | Replace Temp with Query | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | muss |

### Beschreibung

You are using a temporary variable to hold the result of an expression. Extract the expression into a method.Replace all references to the temp with the expression. The new method can then be used in other methods

### Refactoring-Steps

* Find temp with a single assignment
* Extract Right Hand Side of assignment
* Replace all references of temp with new method
* Remove declaration and assignment of temp
* Compile and test

## 102.4 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 4 | Nichttechnischer Titel | | Replace Type Code with State/Strategy | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | muss |

### Beschreibung

Neue Zustände können einfach ins System integriert werden,ohne bestehenden Code ändern zu müssen (OCP). Ermöglicht wird dies durch die Aufteilung in mehrere Klassen.

You have a type code which affects the behavior of a class but you cannot use subclassing. Replace the type code with a state object.

### Refactoring-Steps

* Create a new state class for the type code
* Add subclasses of the state object, one for each type code
* Create an abstract query in the superclass to return the type code. Override in subclasses to return correct type code
* Compile
* Create field in old class for the state object
* Change the type code query to delegate to the state object
* Change the type code setting methods to assign an instance of the subclass
* Compile and test

## 102.5 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 5 | Nichttechnischer Titel | | Replace Switch with Polymorphism | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | muss |

### Beschreibung

Vererbung und Polymorphismus verringern die Abhängigkeiten im System.

You have a conditional that chooses different behavior depending on the type of an object.

Move each leg of the conditional to an overriding method in a subclass.Make the original method abstract.

### Refactoring-Steps

* Move switch to superclass of inheritance structure
* Copy one leg of case statement into subclass
* Compile and test
* Repeat for all other legs
* Replace case statement with abstract method

## 102.6 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 6 | Nichttechnischer Titel | | Form Template Method | | |
| Quellen | Anf. ID:102 | | Verweise | Anf. 0 | Priorität | muss |

### Beschreibung

Identische Schritte von abgeleiteten Klassen in die Superklasse verschieben und lediglich unterschiedliche Schritte in den abgeleiteten Klassen implementieren.

You have two methods in subclasses that carry out similar steps in the same order, yet the steps are different. Give each step into methods with the same signature, so that the original methods become the same. Then you can pull them up.

* + 1. Refactorings-Steps
* Take two methods with similar overall structure but varying pieces
  + Use subclasses of current class, or create a strategyand move the methods to the strategy
* At each point of variation extract methods from each source with the the same signature but different body
* Declare signature of extracted method in superclass and place varying bodies in subclasses
* When all points of variation have been removed, move one source method to superclass and remove the other

## 101.7 Projektplanung und Definition von Meilensteinen

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 7 | Nichttechnischer Titel | | Projektplanung mit Meilensteinen und Definition von Artefakten | | |
| Quellen | Anf. ID: 101 | | Verweise | Lastenheft | 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.

## 104.8 Werkzeugumgebung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 8 | Nichttechnischer Titel | | Werkzeugumgebung | | |
| Quellen | Anf. ID: 104 | | Verweise | Lastenheft | Priorität | soll |

### Beschreibung

Moderne Softwareentwicklung verlangt die Nutzung geeigneter Werkzeugunterstützung, hierfür werden folgende Anwendungen eingesetzt:

1. Entwicklung (IDE): IntelliJ
2. Build-Management: Maven
3. Software-Versionsverwaltung: Github
4. Tests: JUnit

## 105.9 Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 9 | Nichttechnischer Titel | | Funktionale Erweiterungen | | |
| Quellen | Anf. ID: 105 | | Verweise | Lastenheft | 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 Movie mit speziellem Verhalten zu ergänzen ohne

die bestehende Logik verändern zu müssen.

Bei diesen Erweiterungen ist das Open Closed Principle (OCP) – offen für Erweiterungen,

geschlossen für Veränderungen – zu beachten.

### Wechselwirkungen

Diese Erweiterungen sind ebenfalls im Rahmen des Vortrags von Martin Fowler beschrieben

und müssen für die Spezifikation des Pflichtenhefts genutzt werden.

## 10. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 8 | Nichttechnischer Titel | | Change Name of Variables | | |
| Quellen | JavaOne-Martin Fowler | | Verweise |  | Priorität | kann |

### Beschreibung

Aussagekräftige Benennung der Variablen mit einheitlicher Formatierung.

## 11. Anforderung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr. / ID | 11 | Nichttechnischer Titel | | Change Accessors | | |
| Quellen | JavaOne-Martin Fowler | | Verweise |  | Priorität | kann |

### Beschreibung

Zugriffsmöglichkeiten auf Variablen auf das Minimum beschränken.

# Genehmigung

Die Genehmigung erfolgt...

|  |  |
| --- | --- |
| Datum: | 20.06.2020 |
| Unterschrift Auftraggeber: | Herr Auftraggeber |
| Unterschrift Projektleiter: | Herr Projektleiter |
| Weitere Unterschriften: | Herr Sonstiges |