Dokumentation der Praktischen Arbeit zur Prüfung zum Mathematisch-technischen Softwareentwickler

1. April 2020

**Bjarne-Dag Herrmann**

Prüfungsnummer: 187

Programmiersprache: Java

Inhalt

[1. Aufgabenanalyse 4](#_Toc43571616)

[1.1 Beschreibung 4](#_Toc43571617)

[1.2 Analyse 4](#_Toc43571618)

[1.3 Eingabe 4](#_Toc43571619)

[1.4 Ausgabeformat 5](#_Toc43571620)

[1.5 Sonderfälle 5](#_Toc43571621)

[1.6 Fehlerfälle 5](#_Toc43571622)

[1.6.1 Technische Fehler 6](#_Toc43571623)

[1.6.2 Syntaktischer Fehler 6](#_Toc43571624)

[1.6.3 Semantische Fehler 6](#_Toc43571625)

[2. Verfahrensbeschreibung 7](#_Toc43571626)

[2.1 Logische Datenstruktur 7](#_Toc43571627)

[2.1.1 Program 7](#_Toc43571628)

[2.1.2 I\_FileReader 7](#_Toc43571629)

[2.1.3 I\_FileWriter 7](#_Toc43571630)

[2.1.4 TextFile 7](#_Toc43571631)

[2.1.5 TextFileReader 7](#_Toc43571632)

[2.1.6 TextFileWriter 8](#_Toc43571633)

[2.1.7 MainController 8](#_Toc43571634)

[2.1.8 I\_Mapper 8](#_Toc43571635)

[2.2 Vorgehensweise 8](#_Toc43571636)

[2.2.1 Einlesen von Eingabedateien 8](#_Toc43571637)

[2.2.2 Mappen in internes Datenmodell 8](#_Toc43571638)

[2.2.3 Berechnen der Lösung des Problems 8](#_Toc43571639)

[2.2.4 Mappen des Ergebnisses in externes Datenmodell 8](#_Toc43571640)

[2.2.5 Schreiben von Ausgabedateien 8](#_Toc43571641)

[2.3 Fehlerbehandlung 8](#_Toc43571642)

[2.3.1 Technische Fehler 9](#_Toc43571643)

[2.3.2 Syntaktische Fehler 9](#_Toc43571644)

[2.3.3 Semantische Fehler 9](#_Toc43571645)

[2.4 Behandlung von Sonderfällen 9](#_Toc43571646)

[3. Systembeschreibung 9](#_Toc43571647)

[3.1 Programm 9](#_Toc43571648)

[3.2 Model-Classes 10](#_Toc43571649)

[3.3 View-Classes 10](#_Toc43571650)

[3.4 Controller-Classes 10](#_Toc43571651)

[4. Datenfluss 10](#_Toc43571652)

[5. Programmbeschreibung 10](#_Toc43571653)

[5.1 Pakete 10](#_Toc43571654)

[5.2 UML Diagramme 10](#_Toc43571655)

[5.3 Schnittstellen 10](#_Toc43571656)

[5.4 Präzisierung 10](#_Toc43571657)

[6. Testdokumentation 11](#_Toc43571658)

[7. Ausführliches Beispiel 11](#_Toc43571659)

[8. Zusammenfassung und Ausblick 11](#_Toc43571660)

[8.1 Zusammenfassung 11](#_Toc43571661)

[8.2 Ausblick 11](#_Toc43571662)

[9. Abweichungen und Ergänzungen zum Vorentwurf 11](#_Toc43571663)

[10. Benutzeranleitung 12](#_Toc43571664)

[10.1 Verzeichnisstruktur 12](#_Toc43571665)

[10.2 Vorbereitung des Systems 12](#_Toc43571666)

[10.3 Installation 12](#_Toc43571667)

[10.4 Kompilieren 12](#_Toc43571668)

[10.5 Programmaufruf 12](#_Toc43571669)

[11. Entwicklungsumgebung 12](#_Toc43571670)

[12. Verwendete Hilfsmittel 12](#_Toc43571671)

[13. Erklärung 12](#_Toc43571672)

[14. Anhang 12](#_Toc43571673)

[14.1 Aufgabenstellung 12](#_Toc43571674)

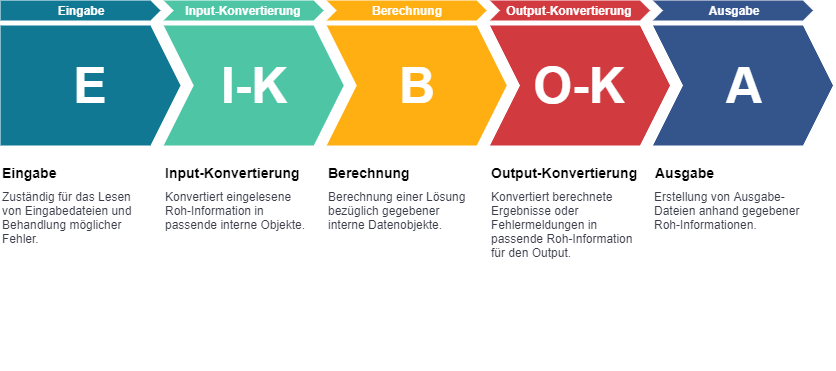
[14.2 In- und Output der Testdokumentation 13](#_Toc43571675)

# Beschreibung

* Wiedergabe der Situation
* Nur Text wiedergeben

# Analyse

Das gegebene Szenario lässt sich einteilen in folgende Teilbereiche:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zuständig für das Lesen von Eingabedateien und Behandlung möglicher Fehler. | Konvertiert eingelesene Roh-Information in passende interne Objekte. | Berechnung einer Lösung bezüglich gegebener interne Datenobjekte. | Konvertiert berechnete Ergebnisse oder Fehlermeldungen in passende Roh-Information für den Output. | Erstellung von Ausgabe-Dateien anhand gegebener Roh-Informationen. |

Ziel ist es, ein komplexes Problem in Teile zu splitten, sodass jeder Teilbereich einen Aufgabenbereich übernehmen kann.

## Eingabe

* Wie sieht die Eingabe aus?
  + Format, Struktur
* Welche Information ist enthalten?

## 

## Ausgabeformat

* Wie sieht die Ausgabe aus?
  + Format, Struktur, Datentyp
  + Positivfall / Fehlerfall
* Worauf muss geachtet werden?
* Welche Information ist enthalten?

## Konvertierung

Konvertierung blabla

## Berechnung

…….

## Sonderfälle

Als Sonderfall werden alle Szenarien beschrieben, welche nicht klar definiert worden sind oder durch spezielle Eigenschaften ein alternatives Vorgehen erfordern. Dazu zählen folgende Fälle:

* **S1: Triviale Lösung**

……

* **S2: Keine Eingabedateien vorhanden**

…….

* Welche Fälle sind nicht klar definiert?
* Welche Fälle müssen extra behandelt werden?

Wie damit verfahren wird, ist in Kapitel Fehlerbehandlung unter Verfahrensbeschreibung zu lesen.

## Fehlerfälle

Durch obige Analysen und Beschreibungen ergeben sich folgende Fehlerszenarien:

* Technische Fehler
* Syntaktische Fehler
* Semantische Fehler

Diese werden in folgenden Abschnitten genauer untersucht. Wie damit verfahren wird, ist in Kapitel Fehlerbehandlung unter Verfahrensbeschreibung zu lesen.

### Technische Fehler

Technische Fehler sind unerwartete Fehler, die meist vom Benutzer unabhängig sind. Dabei können folgende Fehler auftreten:

* **TF1: Der Inhalt einer Eingabedatei konnte nicht gelesen werden**

Beim Lesen einer Datei in angegebenen Eingabepfad ist ein Fehler aufgetreten. Ursache können fehlende Leserechte sein oder ein nicht mögliches Öffnen der Datei aufgrund eines Zugriffs von einem anderen Prozess sein.

* **TE2: Die Ausgabedatei konnte nicht generiert werden**

Beim Schreiben einer Datei in angegebenen Ausgabepfad ist ein Fehler aufgetreten. Ursache können fehlende Schreiberechte sein.

### Syntaktischer Fehler

Syntaktische Fehler treten auf, wenn ein nicht definiertes Format vorliegt. Dies kann beim Lesen einer Datei oder beim Programmstart durch Fehlerhafte Argumente vorkommen. Folgende Syntaxfehler können auftreten:

* **SYF1: Unpassende Anzahl an Argumenten**

Das Programm wurde mit einem oder mehr als zwei Argumenten aufgerufen.

* **SYF2: Invalide Argumente**

Angegebene Argumente sind keine validen Pfade.

* **SYF3: …….**

### 

### Semantische Fehler

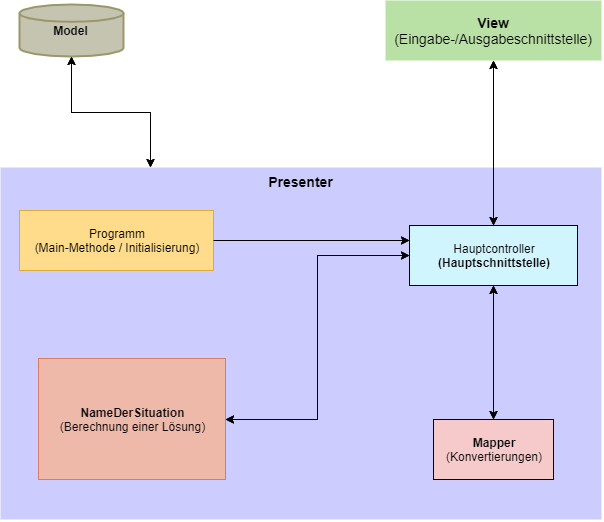
Semantische Fehler liegen vor, wenn die Syntax an sich korrekt ist – das heißt, technisch gesehen liegt kein Fehler vor und die beinhaltenden Daten könnten so gelesen / konvertiert werden – aber der Inhalt auf das Szenario bezogen Inkonsistenz ist. Folgende Semantikfehler können auftreten:

* **SEF1: …**

# Verfahrensbeschreibung

Die Verfahrensbeschreibung beschreibt zu Beginn den Aufbau und Ablauf sowie die logische Datenstruktur der Anwendung und folglich den Ablauf diverser Use-Case-Szenarien.

Für eine Logische Trennung der Daten und Aufgaben wird die Anwendung mittels eines Model-View-Presenter-Pattern entwickelt. So kann die Verarbeitung über eine – oder auch mehrere – Presenter-Klassen aufgeteilt werden. Verarbeitende Daten werden mittels Model-Klassen gesichert bzw. Verwaltet. Die View-Klassen behandeln die Eingabe und Ausgabe der Informationen für den Benutzer. Dies lässt sich im groben wie folgt darstellen:



## Logische Datenstruktur

### Program

Haupteinstiegspunkt der Anwendung, welche die Initialisierung wichtiger Parameter und übergebene Argumente verwaltet.

### I\_FileReader

Generisches Interface mit Methoden zum Lesen von Dateien eines Eingabepfades.

### I\_FileWriter

Generisches Interface mit Methoden zum Schreiben von Dateien eines Ausgabepfades.

### TextFile

Eine Klasse, die Objekte mit Informationen über eine Datei beinhaltet. Dazu gehören Name und Inhalt der Datei.

### TextFileReader

Klasse, welche generisches Interface IOFileReader sowie die Logik vom Lesen von Dateien durch die TextFile-Klasse implementiert.

### TextFileWriter

Klasse, welche generisches Interface IOFileWriter sowie die Logik vom Schreiben von Dateien durch die TextFile-Klasse implementiert.

### MainController

Haupt-Verarbeitungscontroller, der die oberste Verarbeitung verwaltet. Dazu zählt die Verarbeitung/Verwaltung von Lesen und Schreiben von Dateien sowie das Mapping von Datenstrukturen sowie die Übergabe an die Berechnungsinstanz für die Lösungsberechnung.

### I\_Mapper

Generisches Interface, welches für das Mapping von Datenstrukturen zuständig ist. Dazu gehören das Mapping von der internen Datenstruktur in die externe für das Schreiben von Objekten sowie umgekehrt für das Berechnen der Lösung eines Inhaltes einer Datei.

## Vorgehensweise

### Einlesen von Eingabedateien

Hierbei werden die Dateien mit einem fest definierten Pfad eingelesen. Ziel ist es, den Inhalt jeder einzelnen Datei getrennt in einer internen physikalischen Datenstruktur zu besitzen. In diesem Fall werden Dateien in Form von TextFile-Objekten gespeichert, sodass am Ende eine Liste von allen eingelesenen Dateien existiert. Hierbei können technische Fehler vorkommen – mit diesen wird wie oben beschrieben verfahren.

### Mappen in internes Datenmodell

Nachdem eine Liste mit allen Dateien in Form von TextFile-Objekten vorhanden ist, werden diese in ein internes Datenmodell gemappt. Das heißt, ….

### Berechnen der Lösung des Problems

…….

### Mappen des Ergebnisses in externes Datenmodell

…….

### Schreiben von Ausgabedateien

Sind alle Ergebnisse in Form von TextFile-Objekten vorhanden, so wird für jedes Objekt eine Ausgabedatei generiert. Hierbei können technische Fehler vorkommen – mit diesen wird wie oben beschrieben verfahren.

## Fehlerbehandlung

In Abschnitt XX Fehlerfälle wurden mögliche Fehlerszenarien aufgelistet. Im Folgenden ist beschrieben, wie mit einzelnen Verfahren umgegangen wird.

### Technische Fehler

Ein technischer Fehler kann keine gültige Ausgabedatei erzeugen. Dieser führt somit zu einer Konsolenausgabe oder einer alternativen Dateiausgabe mit dazugehöriger Information für den Anwender.

* **TF1: Der Inhalt einer Eingabedatei konnte nicht gelesen werden**

Die Datei wird übersprungen und es wird eine Ausgabedatei beinhaltend dieser Meldung erzeugt. Diese ist an der Bezeichnung “out\_TE1\_xxx“ innerhalb des Ausgabepfads erkennbar.

* **TE2: Die Ausgabedatei konnte nicht generiert werden**

Es wird keine Ausgabedatei erzeugt und der Inhalt der Datei wird mit relevanten Fehlerinformationen sowie dem Inhalt der Datei auf der Konsole ausgegeben.

### Syntaktische Fehler

Das Resultat ist eine passende Konsolenausgabe mit dazugehöriger Information. Wenn möglich, wird durch definierte Standardverfahren mit einer alternativen Prozedur weiter verfahren.

* **SYF1: Unpassende Anzahl an Argumenten**

Dies führt zu einer ungültigen Verarbeitung der Eingabe- und Ausgabepfade. Folglich wird mit einer Information in der Konsole mit den Standardwerten weiter verfahren.

* **SYF2: Invalide Argumente**

Dies führt zu einer ungültigen Verarbeitung der Eingabe- und Ausgabedateien. Folglich wird mit einer Information in der Konsole mit den Standardwerten weiter verfahren.

* **SYF3: …….**

### Semantische Fehler

Bei einem Semantikfehler wird……

* **SEF1: …**

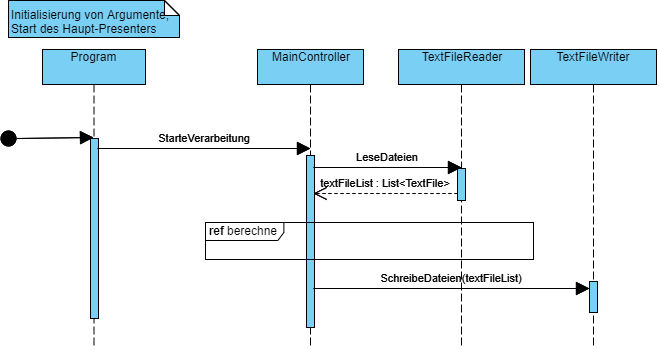
## Behandlung von Sonderfällen

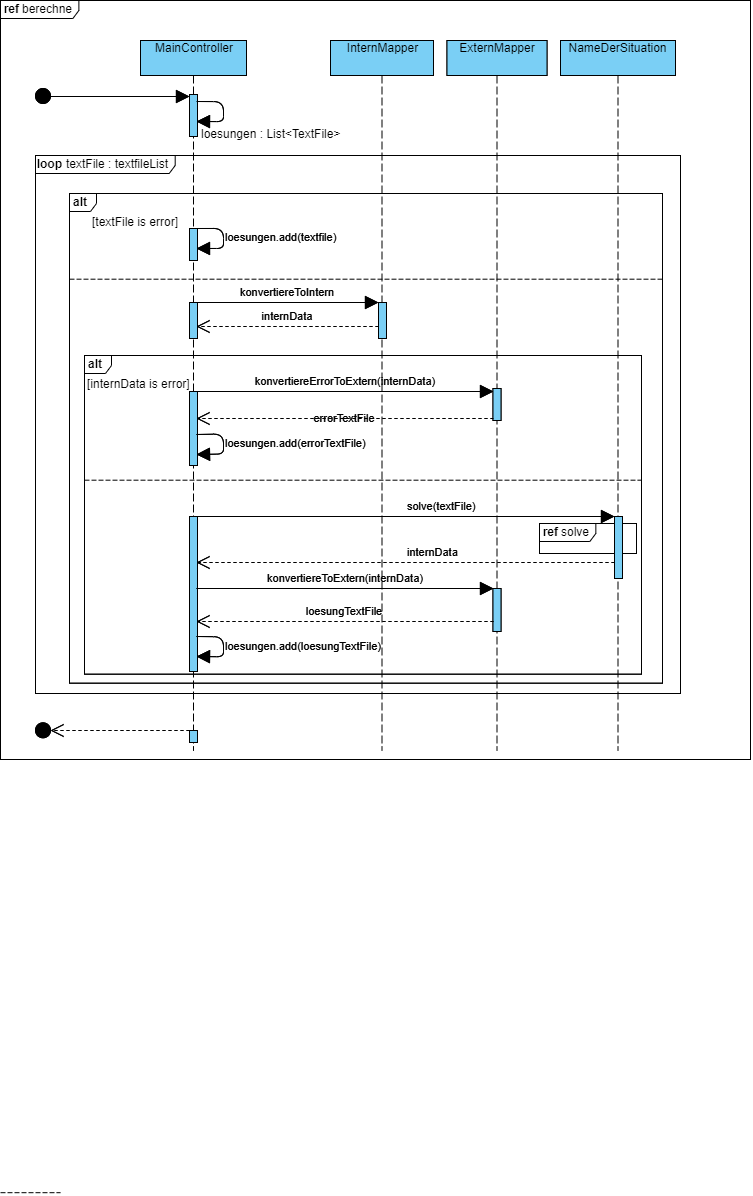
In Abschnitt XX Sonderfälle wurden Szenarien aufgelistet, mit denen gesondert verfahren wird. Im Folgenden ist beschrieben, wie mit jedem dieser Umgegangen wird.

# Programmkonzeption

## 4.1 Datenfluss

Folgende Abbildung beschreibt den high-level Datenfluss der Anwendung. Dargestellte Klassen sind als Singlelton-Instanzen bereits instanziiert und werden nacheinander für die Ausführung einer spezifischen Aufgabe aufgerufen.





# Programmbeschreibung

## Pakete

* Pakagediagramm

## UML Diagramme

* Diagramm abbilden inklusive Klassenbeschreibung

## Schnittstellen

* Attribute / Methoden der Klassen beschreiben

## Präzisierung

* Sequenzdiagramm 🡪 detaillierter Ablauf von Programm, Controller, Model, View
* Programmablaufpläne / Nassi-Schneiderman-Diagramme von wichtigen Algorithmen
* Textbeschreibung

# Testdokumentation

* Welche Tests für welche Fälle (Positivfälle, Sonderfälle, Fehlerfälle, Grenzfälle)?
* Minimalbeispiele / Maximalbeispiele

# Ausführliches Beispiel

* IHK-Beispiel zeigen

# Zusammenfassung und Ausblick

## Zusammenfassung

* Was kann erweitert werden?
* Was kann ausgetauscht werden?
* Stärken / Schwächen?

## Ausblick

* Mögliche Erweiterungen
  + Grafische Oberfläche
  + Weitere Eingabeformate
  + Anderer Algorithmus / Berechnung

# Abweichungen und Ergänzungen zum Vorentwurf

* Was wurde verändert / Ergänzt?
* Wieso wurde es verändert / Ergänzt?

# Benutzeranleitung

## Verzeichnisstruktur

## Vorbereitung des Systems

* Systemvoraussetzungen
  + Java
  + Powershell (Amdmin)

## Installation

## Kompilieren

## Programmaufruf

# Entwicklungsumgebung

# Verwendete Hilfsmittel

# Erklärung

# Anhang

## Aufgabenstellung

## In- und Output der Testdokumentation