Dokumentation der Praktischen Arbeit zur Prüfung zum Mathematisch-technischen Softwareentwickler

1. April 2020

**Bjarne-Dag Herrmann**

Prüfungsnummer: 187

Programmiersprache: Java

Inhalt

[1. Aufgabenanalyse 4](#_Toc36714049)

[1.1 Analyse 4](#_Toc36714050)

[1.2 Eingabe 4](#_Toc36714051)

[1.3 Ausgabeformat 4](#_Toc36714052)

[1.4 Programmanforderung 4](#_Toc36714053)

[1.5 Sonderfälle 5](#_Toc36714054)

[1.6 Fehlerfälle 5](#_Toc36714055)

[2. Verfahrensbeschreibung 6](#_Toc36714056)

[2.1 Vorgehensweise 6](#_Toc36714057)

[2.2 Sonderfälle 6](#_Toc36714058)

[2.3 Fehlerfälle 6](#_Toc36714059)

[3. Systembeschreibung 6](#_Toc36714060)

[3.1 Programm 6](#_Toc36714061)

[3.2 Model-Classes 6](#_Toc36714062)

[3.3 View-Classes 6](#_Toc36714063)

[3.4 Controller-Classes 6](#_Toc36714064)

[4. Datenfluss 7](#_Toc36714065)

[5. Programmbeschreibung 7](#_Toc36714066)

[5.1 Pakete 7](#_Toc36714067)

[5.2 UML Diagramme 7](#_Toc36714068)

[5.3 Schnittstellen 7](#_Toc36714069)

[5.4 Präzisierung 7](#_Toc36714070)

[6. Testdokumentation 7](#_Toc36714071)

[7. Ausführliches Beispiel 8](#_Toc36714072)

[8. Zusammenfassung und Ausblick 8](#_Toc36714073)

[8.1 Zusammenfassung 8](#_Toc36714074)

[8.2 Ausblick 8](#_Toc36714075)

[9. Abweichungen und Ergänzungen zum Vorentwurf 8](#_Toc36714076)

[10. Benutzeranleitung 9](#_Toc36714077)

[10.1 Verzeichnisstruktur 9](#_Toc36714078)

[10.2 Vorbereitung des Systems 9](#_Toc36714079)

[10.3 Installation 9](#_Toc36714080)

[10.4 Kompilieren 9](#_Toc36714081)

[10.5 Programmaufruf 9](#_Toc36714082)

[11. Entwicklungsumgebung 9](#_Toc36714083)

[12. Verwendete Hilfsmittel 9](#_Toc36714084)

[13. Erklärung 9](#_Toc36714085)

[14. Anhang 9](#_Toc36714086)

[14.1 Aufgabenstellung 9](#_Toc36714087)

[14.2 In- und Output der Testdokumentation 9](#_Toc36714088)

# Aufgabenanalyse

## Analyse

Gegeben ist ein Puzzle, welches durch gegebene Felder definiert ist. Felder besitzen, da diese aneinander liegen, Nachbarn. Auf jedem Feld gehört eine dazugehörige Karte, welche dieselbe Anzahl an Kanten beziehungsweise Ecken besitzt. Zur Vereinfachung wird im Programm ein 12-teiliges Puzzle mit 5-eckige Feldern / Karten verwendet. Abbildung 1 zeigt ein mögliches Beispiel eines solchen Feldes:

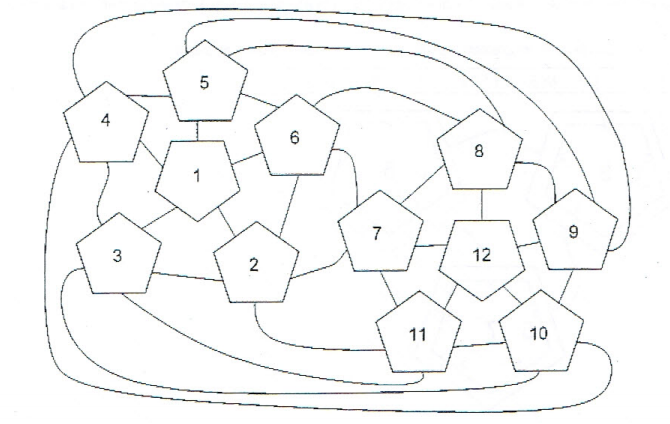


Abbildung 1: Beispiel eines 12-teiligen, 5-eckigen Puzzles

Die Karten besitzen an jeder Kante eine Ziffer – 0, 1 oder 2 – an denen jeweils eine passende Kante einer anderen Karte mit derselben Ziffer angelegt werden muss. Folglich besitzt jede Karte eine 5-stellige Ziffernfolge, die ausgehen von einer sogenannten Grundkante im Uhrzeigersinn die Kantenzugehörigkeit beschreibt. Somit gibt es – bezogen auf dieses Szenario – für eine Karte fünf Möglichkeiten, diese auf einen Platz zu legen.

Die Eingabe erfolgt über eine Datei, in der Zeilenweise die Kanteninformation pro Karte enthält. Diese wird mit Leerzeichen getrennt als Ziffer angegeben. Optional können Kommentare in der Datei enthalten sein, welche für das Programm irrelevant sind.

Ziel ist es, eine Anordnung aller gegebenen Karten zu finden, so dass alle Kanten jeder Karte an einer passenden Nachbarkante liegt (Ziffer 0 zu 0, 1 zu 1 und 2 zu 2).

Das Ergebnis wird in einer Ausgabedatei geschrieben, welche die ursprüngliche Eingabe sowie die Anordnungsinformation, also welches Feld welche Karte besitzt und an welchem Feld die Grundseite der draufliegenden Karte liegt, enthält.

## Eingabe

Die Eingabe erfolgt über eine Textdatei, in der 12 Zeilen mit den dazugehörigen Ziffernfolgen geschrieben stehen, welche die Kanten beschreiben. Kommentarzeilen werden mit einem führenden % gekennzeichnet. Abbildung 2 zeigt ein mögliches Eingabeformat:

%

% Beispiel der Aufgabenstellung IHK

%

0 1 2 1 2

2 2 1 2 0

0 0 1 2 0

1 0 1 1 2

1 2 0 0 1

0 2 2 2 1

1 2 2 1 1

2 1 0 0 2

0 2 1 0 0

2 1 1 0 1

1 1 0 1 1

2 0 1 0 1

Abbildung : Eingabebeispiel aus der Aufgabenstellung

## Ausgabeformat

Die Ausgabedatei beinhaltet zu Beginn den Inhalt der Eingabedatei. Im Anschluss erfolgt eine Ausgabezeile, welche die Lösung des Problems angibt (Lösung gefunden / keine Lösung gefunden). Wurde eine Anordnung der Karten gefunden, so folgt eine tabellarische Auflistung der Anordnungsinformation. Diese beinhaltet für jedes Feld die draufliegende Karte, sowie die Information, an welchem Feld die Grundseite dieser Karte liegt. Abbildung 3 zeigt ein Beispiel einer solchen Ausgabedatei:

---------------EINGABE----------------

%

% Beispiel der Aufgabenstellung IHK

%

Karte 0: [0, 1, 2, 1, 2]

Karte 1: [2, 2, 1, 2, 0]

Karte 2: [0, 0, 1, 2, 0]

Karte 3: [1, 0, 1, 1, 2]

Karte 4: [1, 2, 0, 0, 1]

Karte 5: [0, 2, 2, 2, 1]

Karte 6: [1, 2, 2, 1, 1]

Karte 7: [2, 1, 0, 0, 2]

Karte 8: [0, 2, 1, 0, 0]

Karte 9: [2, 1, 1, 0, 1]

Karte 10: [1, 1, 0, 1, 1]

Karte 11: [2, 0, 1, 0, 1]

--------------------------------------

\*\*\*\* Lösung gefunden \*\*\*\*

Eine mögliche Zuordnung der Karten auf die Puzzlefelder:

Feld | Karte | Grundkante an 5-Eck

===========================================================

0 | 1 | 2

1 | 6 | 1

2 | 2 | 2

3 | 3 | 5

4 | 11 | 4

5 | 8 | 5

6 | 5 | 8

7 | 9 | 6

8 | 10 | 5

9 | 12 | 9

10 | 4 | 3

11 | 7 | 8

Abbildung 3: Beispiel einer Ausgabedatei

* 1. Programmanforderung

Das Programm orientiert sich nach dem MVC-Entwurfsmuster.

* Nach welchem Prinzip arbeitet das Programm (MVC, Patterns)?
* Regierung auf Fehler
* Qualitätssicherung durch Testfälle

## Sonderfälle

* Welche Fälle sind nicht klar definiert?
* Welche Fälle müssen extra behandelt werden?
* Umgang mit Sonderfällen in Verfahrensbeschreibung

## Fehlerfälle

* Technische Fehler
  + Dateizugriff nicht möglich, Datei nicht vorhanden, Schreiben von Dateien nicht möglich
* Syntaktische Fehler
  + Format der Eingabedatei nicht valide
* Semantische Fehler
  + Daten in Eingabedatei inkonsistent
* Liste aller Fehlerfälle angeben

# Verfahrensbeschreibung

## Vorgehensweise

* Formale Beschreibung des Ablaufes
  + Eingabe 🡪 Verarbeitung 🡪 Ausgabe

## Sonderfälle

* Wie wird mit Sonderfällen umgegangen?
  + Fehler werfen / Spezielle Ausgabe?

## Fehlerfälle

* Wie sehen Fehler aus?
* Welche Meldung?
* Programmabbruch oder nur aktuelle Datei?

# Systembeschreibung

## Programm

* Programmstart
* Ruft Haupt-Controller auf
  + Verarbeitet und übergibt Aufrufargumente (Input/Output an Controller)

## Model-Classes

* Speichern von Daten

## View-Classes

* Ausgabe der Daten

## Controller-Classes

* Verarbeitungsschnittstelle zwischen Eingabe und Ausgabe
* Ruft weitere Verarbeitungsprozesse / Controller auf

# Datenfluss

* Sequenzdiagramm
  + Beschreibt Systemablauf zwischen Main-Class, Eingabe, Controller und Ausgabe

# 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