



Silas Hoffmann, inf103088  
3. Fachsemester  
3. Verwaltungssemester

1. Januar 2019

Dokumentation

## **Programmierpraktikum**

im Wintersemester 2018/19

Dozent: Prof. Dr. Andreas Häuslein

Fachbereich Informatik  
Fachhochschule Wedel

# Inhaltsverzeichnis

|  |   |
|--|---|
| <b>I. Allgemeine Problemstellung</b>     | <b>4</b>                                    |
| 1. Spielregeln                           | 4   |
| 2. Spielablauf                           | 4   |
| 3. Anlegeregeln                          | 4   |
| 4. Spielende                             | 5   |
| 5. KI                                    | 5   |
| 6. Oberfläche                            | 5   |
| 7. Karten                                | 6   |
| 8. Log                                   | 7   |
| 9. Spielstand                            | 7   |
| <br>                                     |   |
| <b>II. Benutzerhandbuch</b>              | <b>9</b>                                    |
| 10. Ablaufbedingungen                    | 9   |
| 11. Programminstallation / Programmstart | 9   |
| 12. Bedienungsanleitung                  | 9   |
| 12.1. Hintergrundinformationen . . . . . | 9   |
| 12.2. Programmfunctionalitaet . . . . .  | 14  |
| <br>                                     |   |
| <b>III. Programmierhandbuch</b>          | <b>19</b>                                   |
| 13. Entwicklungskonfiguration            | 19  |
| 14. Problemanalyse und Realisation       | 19  |
| 14.1. Problemanalyse . . . . .           | 19  |
| 14.2. Realisationsanalyse . . . . .      | 20  |
| 15. Beschreibung grundlegender Klassen   | 22  |
| <br>                                     |   |
| <b>A. Im Anhang Eins</b>                 | <b>ii</b>                                   |
|  | Literaturverzeichnis<br>noch an-<br>fuehren |

## **Abbildungsverzeichnis**

|     |  |    |
|-----|--|----|
| 1.  | Erstes Selektieren . . . . .   | 10 |
| 2.  | Schwebender Domino ueber gueltiger Position . . . . .                | 11 |
| 3.  | Schwebender Domino ueber ungueltiger Position . . . . .              | 11 |
| 4.  | Menueoptionen . . . . .  | 12 |
| 5.  | Nachtraegliches Abspeichern . . . . .                                | 12 |
| 6.  | Filechooser . . . . .  | 13 |
| 7.  | Unterschiedliche Fehlermeldungen beim Einlesen einer Datei . . . . . | 14 |
| 8.  | Spielbeginn . . . . .  | 15 |
| 9.  | Initiales Selektieren - oben . . . . .                               | 15 |
| 10. | Initiales Selektieren - mittig . . . . .                             | 16 |
| 11. | Erstes Rotieren . . . . .  | 17 |
| 12. | Erste Ablage . . . . .   | 17 |

# Teil I.

# Allgemeine Problemstellung

Zu implementieren ist ein Dominospiel, bei dem vier Spieler jeweils ihre eigene Stadt gestalten. Ziel des Spiels ist es, möglichst viele Stadtteile mit Prestige zu gestalten. [1]

## 1. Spielregeln

Jeder Spieler besitzt ein eigenes 5\*5-Zellen großes Spielfeld und legt zu Beginn sein Stadtzentrum mittig ab. Ein Spielbeutel für alle Spieler enthält 48 Spielkarten in der Größe von zwei Zellen, die auf ihren zwei Hälften jeweils einen (evtl. auch den gleichen) Stadtteiltyp anzeigen. Die Stadtteiltypen unterscheiden sich durch Bild und Hintergrundfarbe voneinander. Jede Spielkarte besitzt eine definierte Wertigkeit. Auf manchen Stadtteilen sind zusätzlich ein bis drei Prestigesymbole abgebildet. Es werden vier Karten gezogen und im ersten Auswahlbereich angezeigt. Dabei wird die niederwertigste Karte zuoberst, die höchstwertigste zuunterst einsortiert. Der erste Spieler markiert die Karte im Auswahlbereich, die er gerne nehmen würde, die anderen Spieler treffen ihre Auswahl der Reihe nach ebenfalls und markieren die jeweils gewünschte Karte. Wurden alle Karten markiert, dann werden wieder vier Karten gezogen und ebenso sortiert im zweiten Auswahlbereich angezeigt.

## 2. Spielablauf

Derjenige, der die oberste Karte im ersten Auswahlbereich markiert hat, beginnt eine Runde, es folgen der Reihe nach die Spieler, die die jeweils darunterliegende Karte markiert haben. In einer Runde wird zunächst eine Karte aus dem zweiten Auswahlbereich markiert und dadurch für die kommende Runde gewählt. Je wertvoller also seine markierte Karte in dieser Runde ist, desto später ist der Spieler am Zug und desto weniger Auswahl hat er für die kommende Runde.

## 3. Anlegeregeln

Die erste Karte muss an das Stadtzentrum angrenzen. An das Stadtzentrum darf jeder Stadtteil angrenzen. Legt man eine Karte an eine andere Karte an, so muss mindestens eine Hälfte mit einer Seite an einen identischen Stadtteiltyp einer liegenden Karte angrenzen. Passt die abzulegende Karte weder an das Stadtzentrum noch an eine bereits ausliegende Karte, so wird sie verworfen. Alle Spielkarten müssen in das 5\*5-Feld passen, keine Hälfte darf hinausragen. Das Stadtzentrum muss aber nicht in der Mitte liegen, sondern kann im Spielverlauf verschoben werden, wodurch sich alle bereits gelegten Karten mit verschieben. Eine abgelegte Karte kann nicht verschoben werden.

## 4. Spielende

Wurden alle Spielkarten aus dem Beutel gezogen und von den Auswahlbereichen auf die Spielfelder platziert bzw. verworfen, werden die Punkte ermittelt.

- Jede Stadt besteht aus mehreren Stadtteilen. Ein Stadtteil setzt sich aus waagerecht und/oder senkrecht verbundenen Zellen desselben Stadtteiltyps zusammen. Das Stadtzentrum zählt zu keinem Stadtteil dazu.
- Die Punkte eines Stadtteils ergeben sich aus der Anzahl seiner Zellen multipliziert mit der Anzahl darin enthaltener Prestigesymbole.
- Innerhalb einer Stadt kann es mehrere voneinander getrennte Stadtteile desselben Typs geben. Jeder Stadtteil ist einzeln auszuwerten.
- Stadtteile ohne Prestigesymbole bringen keine Punkte.

Für die Auswertung wird für jeden Spieler die Summe der Punkte seiner Gebiete ermittelt. Gewonnen hat der Spieler mit den meisten Punkten. Bei einem Gleichstand gewinnt der Spieler mit dem größten einzelnen Gebiet. Besteht auch hier Gleichstand, so siegen beide Spieler gleichermaßen.

## 5. KI

Außer dem menschlichen Spieler, der im Spiel stets beginnt, existieren 3 computergesteuerte Spieler. Diese sollen einer sehr primitiven Logik folgen:

- bei der Auswahl wird die Karte markiert, mit der bei Auslage im eigenen Feld aktuell am meisten Punkte erzielt werden könnten
- dafür wird für jede freie Karte der Auswahlbank auf jeder freien Position des Spielfeldes und in jeder Rotation der Punktgewinn ermittelt
- bei Punktgleichheit mehrerer Positionen wird darauf geachtet, dass keine leeren Einzelzellen erzeugt werden bei der Ablage wird die so ermittelte Position genutzt
- die mögliche Verschiebung des Stadtzentrums wird nicht durchgeführt

Wer möchte, kann zusätzlich intelligentere KIs implementieren, die z.B. das Stadtzentrum verschieben, die Kartenwahl der kommenden Runde einbeziehen, verhindern, dass andere Spieler viele Punkte erhalten, oder Schlüsse aus den bereits abgelegten Karten ziehen.

## 6. Oberfläche

Existieren müssen folgende Elemente:

- ein Spielfeld für den menschlichen Spieler

- ein erster Auswahlbereich für die aktuelle Runde
- ein zweiter Auswahlbereich für die kommende Runde
- das Legen einer Karte auf das Spielfeld per Drag and Drop, gültige Ablegepositionen werden beim DragOver sichtbar markiert
- eine Möglichkeit, um die Karte zu verwerfen. Das kann z.B. ein Button sein, der zum Verwerfen der aktuellen Karte betätigt wird, oder auf den die aktuelle Karte gezogen wird. Verworfene Karten müssen nicht angezeigt werden.
- die Spielfelder der drei KI-Spieler, so dass die dort abgelegten Karten jederzeit erkennbar sind.

Mögliche Lösungen für das Legen einer Karte:

- Die aktuell zu legende Karte kann in einer Drehbox erscheinen, in der die Karte durch einfaches Anklicken gedreht werden kann. Hat sie die gewünschte Orientierung erreicht, so kann die Karte per Drag and Drop auf das eigene Spielfeld gelegt oder verworfen werden.
- Die aktuell gedragte Karte wird durch Tastendruck unter dem Mauscursor gedreht.

Die Reihenfolge der Spieler muss erkennbar sein. Es muss also zugeordnet werden können, welches der angezeigten Spielfelder zu welcher Kartenauswahl im Auswahlbereich gehört (z.B. durch gleiche Symbole oder farbliche Markierungen an beiden Stellen). Das Stadtzentrum eines Spielfeldes muss zusammen mit allen bereits gelegten Karten verschoben werden können, entweder per Drag and Drop oder beispielsweise durch Buttons am Spielfeldrand. Dabei dürfen keine Stadtteile aus dem Spielfeld geschoben werden. Die Auswertung eines Spiels muss für jeden Spieler die erreichten Punkte pro Stadtteiltyp darstellen. Die Bedienung des Spiels muss intuitiv möglich sein für jemanden, der die Spielregeln kennt. Die Größe des Fensters darf zu Spielbeginn höchstens 1600 \* 900 Pixel betragen.

## 7. Karten

Die für ein Spiel vorhandenen Karten sind in dieser Datei definiert. Die zu den Stadtteiltypen gehörigen Bilder findet man hier. Pro Zeile wird eine Karte mit ihren beiden Hälften und ihrem Wert festgelegt: <Art><Symbolanzahl>, <Art><Symbolanzahl>, <Wert> Art ist dabei der Anfangsbuchstabe eines Stadtteiltyps (Amusement, Industry, Office, Park, Shopping, Home), die Symbolanzahl eine Ziffer von 0 bis 3. Eine mögliche Zeile wäre also *H1,P0,24* für eine Karte mit einem Symbol auf einem Haus und ohne Symbol in einem Park und einem Wert von 24 Punkten.

## 8. Log

In einer Datei (gleichzeitig auch auf System.out) sind durchgeführte Aktionen zu protokollieren. Der zuerst angegebene Stadtteil einer Karte ist dabei immer der an der angegebenen Position, bei horizontaler Ausrichtung liegt der zweite Stadtteil rechts davon, bei vertikaler darunter. Beispiel:

BOT1 chose [H1, P0] at index 1 for next round  
 BOT1 put [A0, P2] horizontally to (1, 2)

HUMAN chose [P0, S0] at index 0 for next round  
 HUMAN dragged center to (2, 3)  
 HUMAN put [A0, A0] vertically to (0, 0)

BOT3 chose [O0,I2] at index 3 for next round  
 BOT3 did not use [O0, A1]

## 9. Spielstand

Der aktuelle Spielstand soll gespeichert und geladen werden können. Laden/Speichern soll nur möglich sein, wenn der menschliche Spieler am Zug ist. Eine Spielstandsdatei enthält die 4 Spielfelder der Spieler in ihrer Reihenfolge (das erste Feld gehört immer dem menschlichen Spieler 0), die zwei Auswahlbereiche und die im Beutel verbliebenen Karten mit folgenden Bereichen:

```
<Spielfeld>
<Spielfeld>
<Spielfeld>
<Spielfeld>
  <Bänke>
  <Beutel>
```

Die einzelnen Bereiche enthalten jeweils eine Einführungszeile (einen Kommentar) gefolgt von Inhaltsangaben:

- Ein Spielfeld enthält einen Kommentar, zu wem es gehört, und in Folge für jede Zelle eine Inhaltsangabe:
  - ‘-’ für eine nicht belegte Zelle,
  - ‘CC’ für das Stadtzentrum und
  - zwei Buchstaben für eine Hälftenbeschreibung.
- Die Zellen sind durch Leerzeichen separiert. Beispiel:  
 <Spielfeld>  
 - - - -

-- H1 P0 --

-- CC --

-----

-----

- Die Bänke enthalten Angaben für die aufliegenden Karten und von wem diese bereits gewählt wurde. Die erste Bank kann weniger als vier Karten enthalten, in entsprechender Anzahl enthält die zweite Bank dann bereits Markierungen (sonst '-' für fehlende Markierung). Die erste Bank wird als erste Markierung immer Spieler 0 (den menschlichen Spieler) aufweisen. Beispiel:

<Bänke>

0 H1P0,2 P0O1,3 I1P0

- P0P0,- A0A0,1 H0A0,- P1H0

- Der Beutel enthält alle im Beutel befindlichen Karten kommasepariert. Im folgenden Beispiel befinden sich nur noch 4 Karten im Beutel

<Beutel>

P0P0,P0P0,A1H0,I2P0

# Teil II.

# Benutzerhandbuch

## 10. Ablaufbedingungen

Ueberblick ueber die benoetigten Hardware und Softwarekomponenten die fuer die Ausfuehrung des kompilierten Programms benoetigt werden.

| Softwarekomponenten      |         |
|--------------------------|---------|
| Name                     | Version |
| Java Runtime Environment | 1.8     |

Fussnote mit Link zur Webseite einbinden

## 11. Programminstallation / Programmstart

Genauen Namen angeben

Font angeben

Muss noch gemacht gemacht werden

## 12. Bedienungsanleitung

### 12.1. Hintergrundinformationen

#### Spielinteraktion

**Selektierungsvorgang** Um einen gewuenschten Domino zu selektieren muss der Spieler auf einen der schwarzen Kaesten rechts neben dem angezeigten Domino per Mausklick auswaehlen (siehe Abbildung 10) und es erscheint die Zahl 1 in diesem Feld. Um dem Spieler deutlich zu machen von welcher Bank, oder ob er ueberhaupt in seinem aktuellen Zug einen Domino selektieren darf, kann er nur auf der Bank welche nicht verschwommen ist einen Domino auswaehlen. Um jederzeit ablesen zu koennen welcher Spieler gerade am Zug ist gibt es hierfuer ein Feld oberhalb der Bank fuer die naechste Runde.

**Justierung des Dominos** Nachdem der Spieler erfolgreich saemtliche Selektierungsschritte auf den beiden Baenken absolviert hat, kann er seinen zuvor ausgewahlten Domino in dem dafuer vorgesehenen Kasten drehen. Um den Domino um 90 Grad zu drehen muss der Spieler lediglich einen Mausklick auf dem Domino ausfuehren.

**Positionierung auf dem Spielfeld** Um den justierten Domino nun auf dem Feld zu platzieren zieht der Benutzer den Domino an die gewuenschte Stelle auf dem Spielfeld. Waehrend dem Ziehen faerben sich zugrunde liegenden Felder jeweils gruen, falls es moeglich sein sollte den Domino an dieser Stelle anzulegen (siehe Abbildung 11), beziehungsweise rot, falls dies nicht der Fall sein sollte (siehe Abbildung 3, fuer genauere Informationen siehe Abschnitt 12.2). Falls der Domino an der gewuenschten Stelle nicht

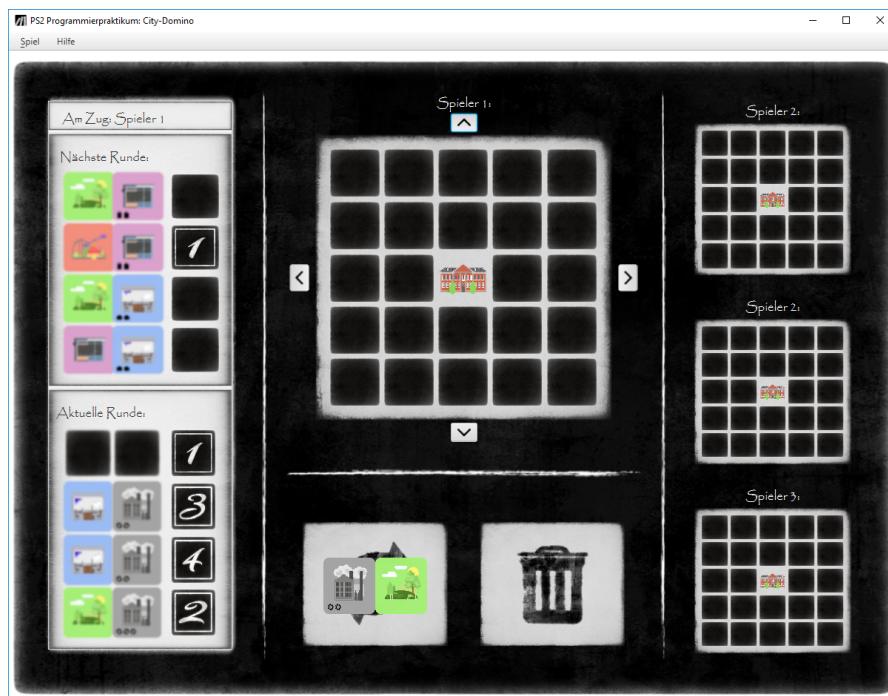


Abbildung 1: Erstes Selektieren auf der Bank fuer die naechste Runde

passen sollte und dennoch versucht wird ihn an dieser Stelle zu platzieren, passiert nichts denn der Domino befindet sich weiterhin in dem Kasten zum justieren der Ausrichtung und es kann ein neuer Versuch gestartet werden.

**Verwerfen des Dominos** Um den Domino zu verwerfen reicht es per Mausklick einmal auf das Muelleimer-Symbol rechts neben dem Domino zu klicken. Der Domino wird anschliessend aus dem Rotationsfeld entfernt.

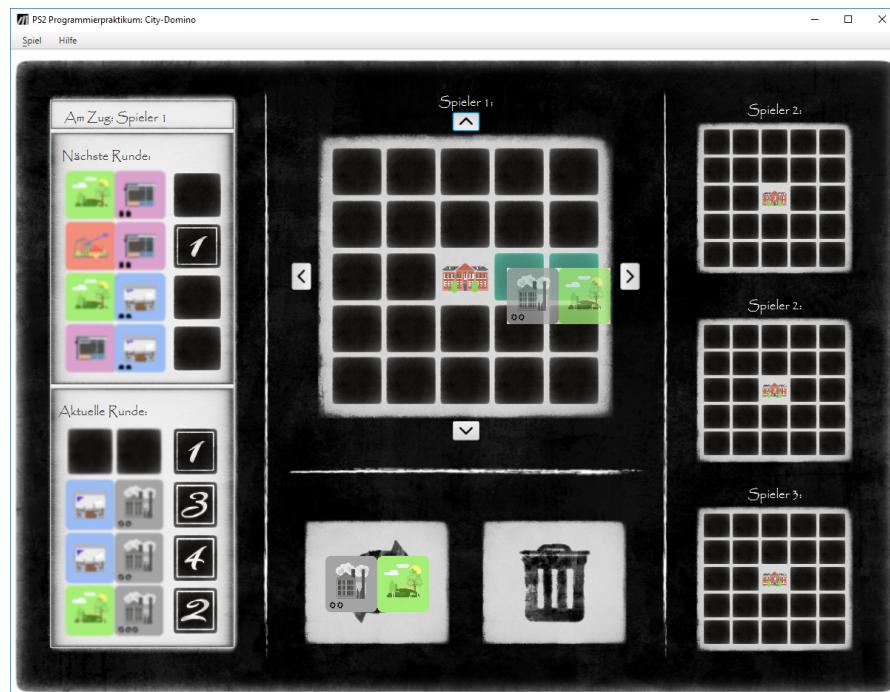


Abbildung 2: Schwebender Domino ueber gueltiger Position

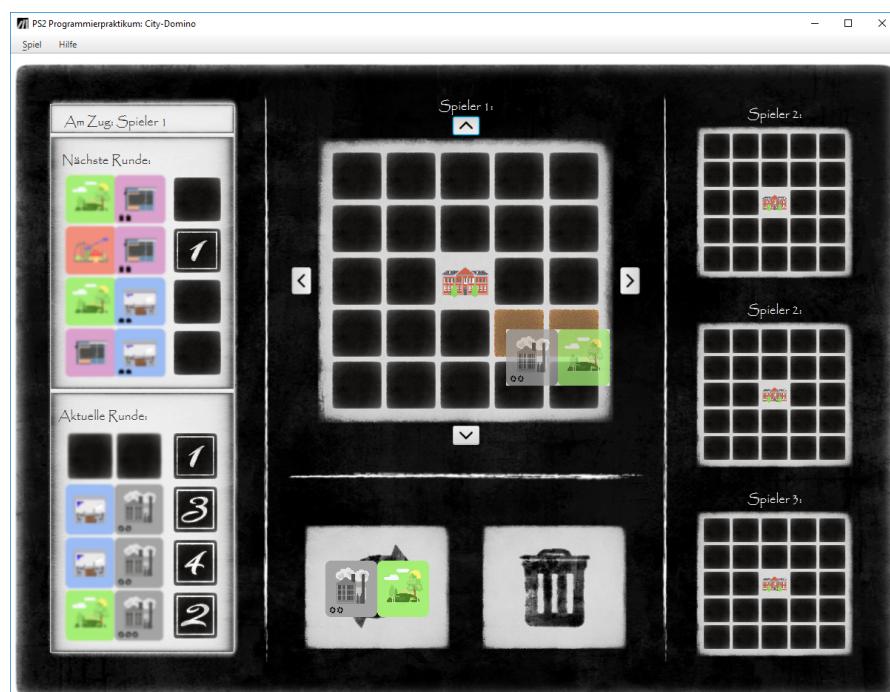


Abbildung 3: Schwebender Domino ueber ungueltiger Position



Abbildung 4: Menueoptionen

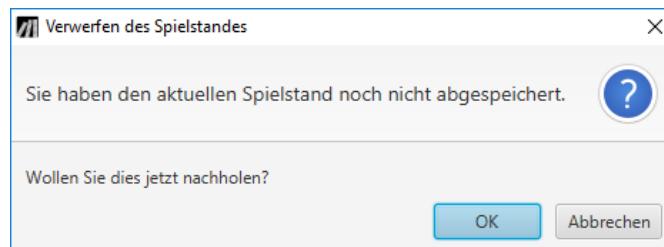


Abbildung 5: Nachtraegliches Abspeichern

## Menueinteraktion

**Starten bzw. Schliessen** Um ein neues Spiel zu starten wählt der Benutzer den Menuepunkt "Neues Spiel". Alternativ ist dies auch per Tastenkombination **strg + N** möglich. Um das geöffnete Fenster zu schliessen und das bestehende Spiel zu verwerfen wählt der Benutzer den Reiter **Beenden** (Tastenkombination **strg + E**). Falls der Benutzer das Spiel nicht vorher gespeichert hat erscheint hierbei ein weiteres Fenster welches den Benutzer darauf hinweist und ihm die Möglichkeit gibt dies nachzuholen (siehe folgenden Abschnitt). Möchte er fortfahren ohne den aktuellen Spielstand muss der Button mit der Aufschrift **Abbrechen** betätigt werden (siehe Abbildung 5).

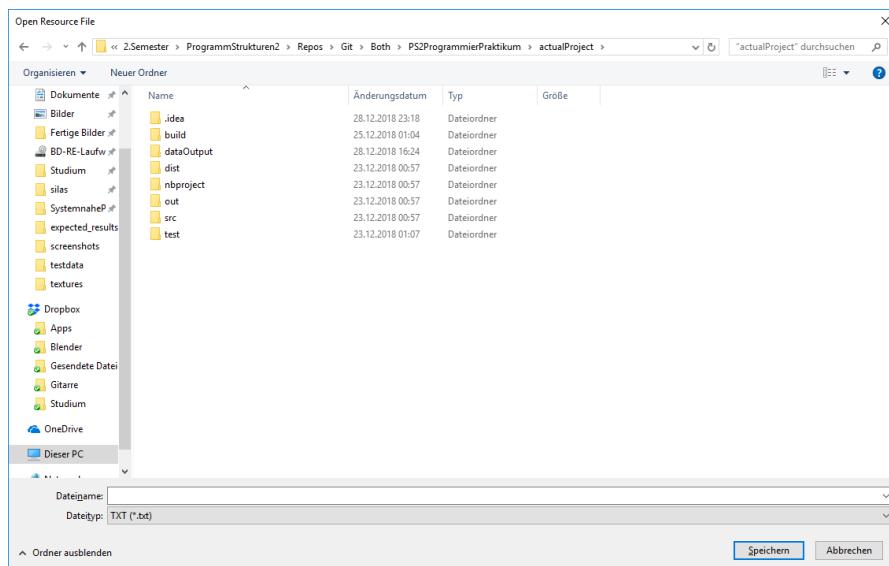


Abbildung 6: Filechooser zum Abspeichern eines Spielstandes

**Spielstand abspeichern** Um ein Spielstand zu speichern gibt es zwei Reiter mit folgenden Moeglichkeiten.

1. Speichern: **Strg + S**
2. Speichern unter: **Strg + Shift + S**

Speichern unter gibt dem Benutzer die Moeglichkeit, ausgehend vom Ablageverzeichnis den gewuenschten Speicherort anzugeben. Hierzu navigiert man mit dem gegebenen Filechooser an den gewuenschten Speicherort und gibt der abzuspeichernden Datei einen Namen, die benoetigt Dateiendung *.txt* ist bereits ausgewaehlt, sodass der Benutzer seine Auswahl lediglich auf dem Feld *Speichern* per Mausklick zu bestaetigen braucht (siehe Abbildung 6).

Der Reiter *Speichern* ermoeglicht es einen bereits gespeicherten Spielstand ohne oeffnen eines Filechoosers zu ueberschreiben. Falls der Benutzer diesen Reiter betaetigt ohne dass zuvor ein Spielstand des aktuellen Spiels abgespeichert wurde ist, oeffnet sich bei der Auswahl dieses Reiters dennoch ein Filechooser und es wird nach der Funktionsweise von *Speichern unter* vorgegangen.

**Oeffnen** Aehnlich wie beim Reiter *Speichern unter* wird hier ein Filechooser geoeffnet. Dieser wird jedoch dazu verwendet eine Datei auszuwaehlen um aus dieser einen Spielstand zu lesen. Falls die Datei nicht der geforderten Syntax entspricht , erscheint eine Fehlermeldung in Form eines Popup-Fensters mit dem einer groben Fehlermeldung wo der Fehler liegt(Siehe ...) . Falls der Benutzer vor dem Oeffnen eines neuen Spielstands den alten nicht gespeichert hat wird er aehnlich wie beim Beenden darauf hingewiesen und es wird per Filechooser eine Moeglichkeit bereitgestellt dies nachzuholen.

Screenshots  
der  
Fehlermel-  
dungen  
und  
Referenz  
auf Er-  
klaerung  
einfuegen

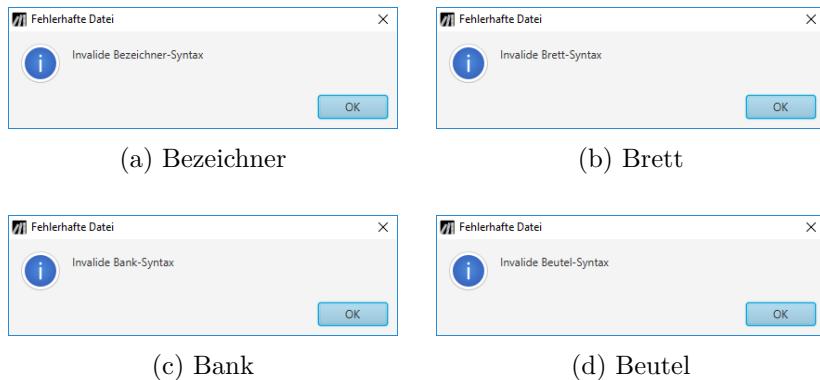


Abbildung 7: Unterschiedliche Fehlermeldungen beim Einlesen einer Datei

**Hilfestellung** Unter dem Reiter *Hilfe* ist die Aufgabenstellung im Pdf-Format zu finden. Beim Auswaehlen des Menuepunktes *Aufgabenstellung* oeffnet sich diese im, vom Benutzer standardmaessig genutzten, Pdf-Reader.

## 12.2. Programmfunctionalitaet

Generell gilt es zwischen einem standardmaessig ausgefuehrten Spiel und einem eingelesenen Spiel zu unterscheiden.

### Spiel ohne Einlesen einer Datei

**Spielbeginn** Ein Spielbeutel für alle Spieler enthält 48 Spielkarten in der Größe von zwei Zellen, die auf ihren zwei Hälften jeweils einen (evtl. auch den gleichen) Stadtteiltyp anzeigen. Die Stadtteiltypen unterscheiden sich durch Bild und Hintergrundfarbe voneinander. Jede Spielkarte besitzt eine definierte Wertigkeit. Auf manchen Stadtteilen sind zusätzlich ein bis drei Prestigesymbole abgebildet. Jeder Spieler besitzt ein eigenes 5\*5-Zellen großes Spielfeld und legt zu Beginn sein Stadtzentrum mittig ab [1]. (siehe Abbildung 8). Dies wird bereits vom Spiel uebernommen, sodass der erste Spielzug des Benutzers das initiales Selektieren vom ersten Auswahlbereich (hier mit *Aktuelle Runde* gekennzeichnet) darstellt. Um einen Domino selektieren zu koennen werden vier Karten gezogen und im ersten Auswahlbereich angezeigt. Dabei wird die niedwertigste Karte zuoberst, die höchswertigste zuunterst einsortiert. Der erste Spieler markiert die Karte im Auswahlbereich, die er gerne nehmen würde, die anderen Spieler treffen ihre Auswahl der Reihe nach ebenfalls und markieren die jeweils gewünschte Karte. Wurden alle Karten markiert, dann werden wieder vier Karten gezogen und ebenso sortiert im zweiten Auswahlbereich angezeigt. [1] (Siehe Abbildung 9)

**Spielablauf** Derjenige, der die oberste Karte im ersten Auswahlbereich markiert hat, beginnt eine Runde, es folgen der Reihe nach die Spieler, die die jeweils darunterliegende Karte markiert haben. In einer Runde wird zunächst eine Karte aus dem zweiten Auswahlbereich markiert und dadurch für die kommende Runde gewählt. Je wertvoller also

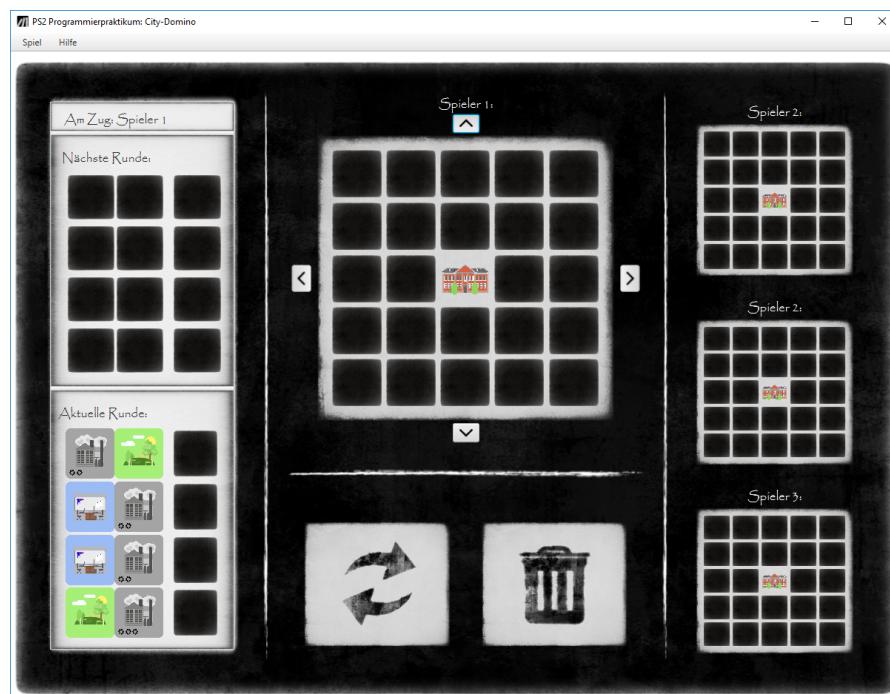


Abbildung 8: Spielbeginn nach Programmstart

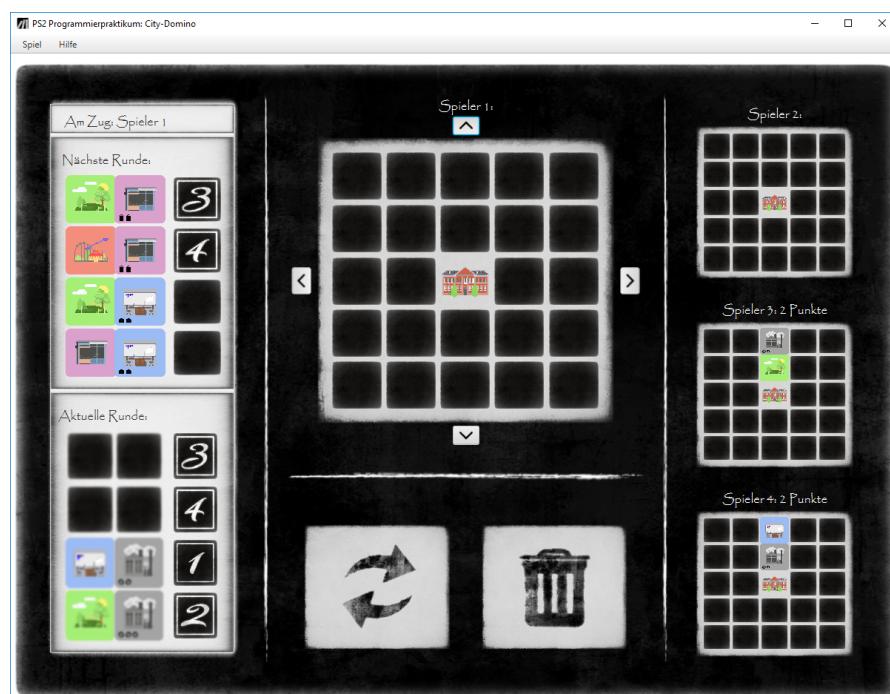


Abbildung 9: Initiales Selektieren (oben) nach Programmstart

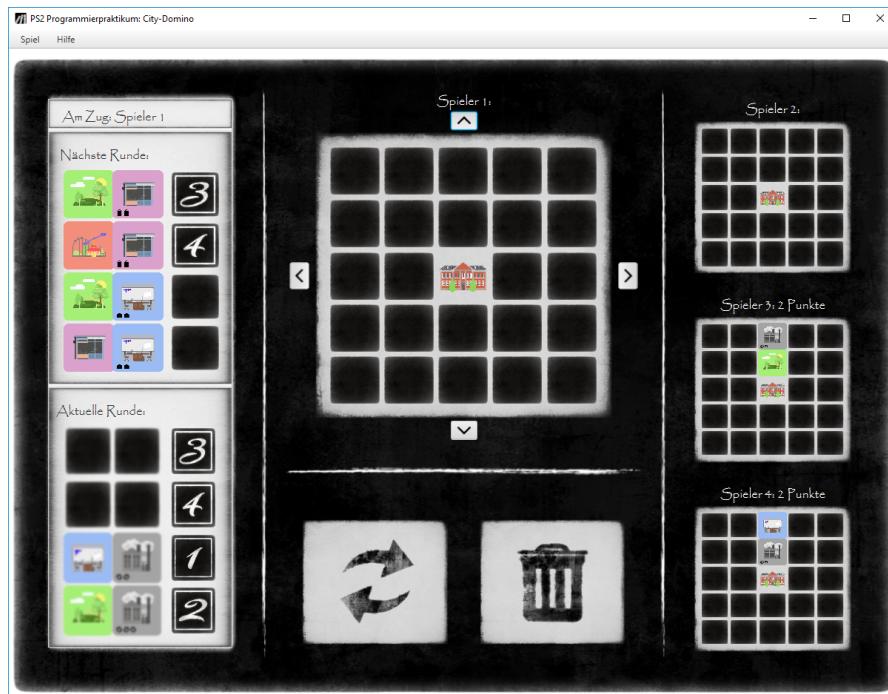


Abbildung 10: Initiales Selektieren (mittig) nach Programmstart

seine markierte Karte in dieser Runde ist, desto später ist der Spieler am Zug und desto weniger Auswahl hat er für die kommende Runde. [1] Zwischen dem initialen Selektieren und dem beschriebenen Spielablauf gibt es keine Pause. Wie man in Abbildung 10 sehen kann, ziehen die Gegner bereits ihre ersten Dominos auf ihr Feld wo der Benutzer noch gar nicht an der Reihe war. Nun kann der Benutzer einen Domino auf dem naechsten Auswahlstapel bzw. der naechsten Bank einen der uebrig gebliebenen Dominosteine auswaehlen. Nun wird der Domino welcher als erstes Selektiert wurde in den Kasten zum rotieren geladen (siehe Abbildung 11) und der Benutzer kann den Stein auf sein Brett legen (Abbildung 12). Nach dieser Aktion kann der Benutzer einen Domino auf der Bank fuer die naechste Runde auswaehlen. Danach muss er wieder "warten" bis er an der Reihe ist um einen ausgewaehlten Domino auf seinem Brett zu platzieren. Alternativ kann er seinen Domino aber auch verwerfen.

**Einlesen einer Datei** Nachdem der Benutzer eine Datei eingelesen hat ist dieser auch gleichzeitig am Zug. Je nachdem ob der Spielstand waehrend des initialen Selektierens oder waehrend einer standardmaessigen Runde abgespeichert wurde, muss der Benutzer entweder von der Bank fuer die aktuelle Runde oder von der Bank der naechsten Runde Selektieren. Generell gelten hier die gleichen Regeln zum Spielablauf wie beim Spielen ohne gespeicherten Spielstand, es kann nur der Schritt des initialen Selektierens uebersprungen werden.

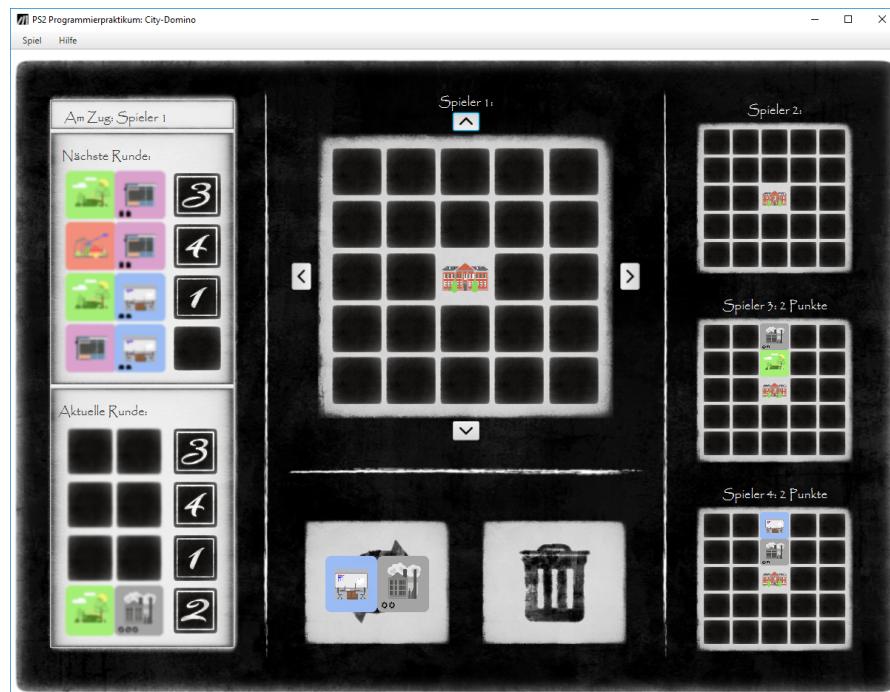


Abbildung 11: Erstes Rotieren

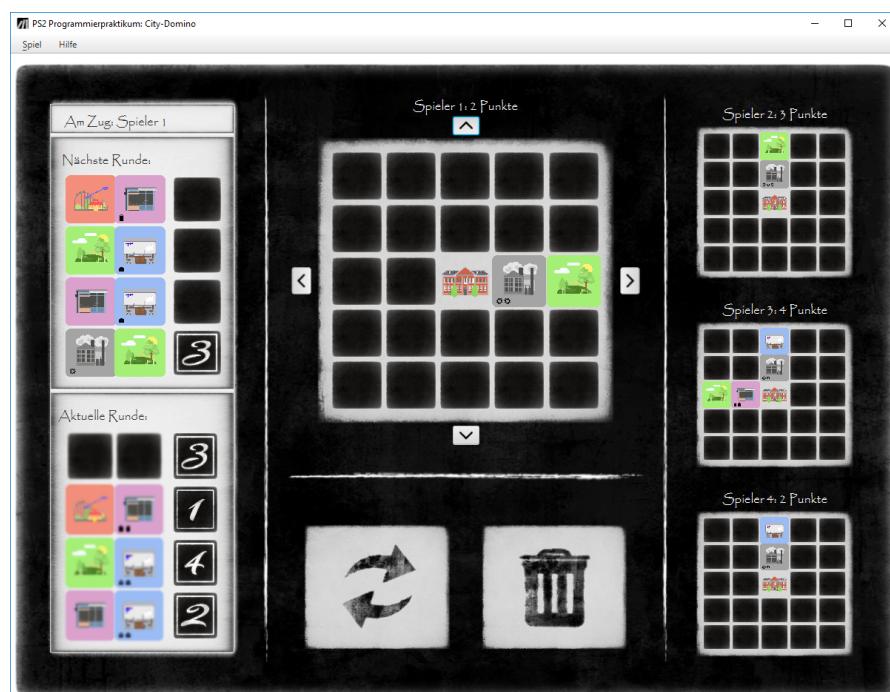


Abbildung 12: Erste Ablage auf dem Spielfeld

**Anlegeregeln** Die erste Karte muss an das Stadtzentrum angrenzen. An das Stadtzentrum darf jeder Stadtteil angrenzen. Legt man eine Karte an eine andere Karte an, so muss mindestens eine Hälfte mit einer Seite an einen identischen Stadtteiltyp einer liegenden Karte angrenzen. Passt die abzulegende Karte weder an das Stadtzentrum noch an eine bereits ausliegende Karte, so wird sie verworfen. Alle Spielkarten müssen in das 5\*5-Feld passen, keine Hälfte darf hinausragen. Das Stadtzentrum muss aber nicht in der Mitte liegen, sondern kann im Spielverlauf verschoben werden, wodurch sich alle bereits gelegten Karten mit verschieben. Eine abgelegte Karte kann nicht verschoben werden. [1]

**Spielende** Wurden alle Spielkarten aus dem Beutel gezogen und von den Auswahlbereichen auf die Spielfelder platziert bzw. verworfen, werden die Punkte ermittelt.

- Jede Stadt besteht aus mehreren Stadtteilen. Ein Stadtteil setzt sich aus waagerecht und/oder senkrecht verbundenen Zellen desselben Stadtteiltyps zusammen. Das Stadtzentrum zählt zu keinem Stadtteil dazu
- Die Punkte eines Stadtteils ergeben sich aus der Anzahl seiner Zellen multipliziert mit der Anzahl darin enthaltener Prestigesymbole.
- Innerhalb einer Stadt kann es mehrere voneinander getrennte Stadtteile desselben Typs geben. Jeder Stadtteil ist einzeln auszuwerten. Stadtteile ohne Prestigesymbole bringen keine Punkte.

Für die Auswertung wird für jeden Spieler die Summe der Punkte seiner Gebiete ermittelt. Gewonnen hat der Spieler mit den meisten Punkten. Bei einem Gleichstand gewinnt der Spieler mit dem größten einzelnen Gebiet. Besteht auch hier Gleichstand, so siegen beide Spieler gleichermaßen. [1]

# Teil III.

# Programmierhandbuch

## 13. Entwicklungskonfiguration

| Softwarekomponenten |                      |                 |
|---------------------|----------------------|-----------------|
| Art                 | Name                 | Version         |
| Betriebssystem      | Windows              | 10 Professional |
| Compiler            | Java development kit | 1.8             |

Sonstige  
Programme einfügen

## 14. Problemanalyse und Realisation

### 14.1. Problemanalyse

Hierbei habe ich mich einer Technik bedient in der man versucht sich in eine jeweilige Teilkomponente des Problems hineinzuversetzen und anzugeben fuer welchen Teilbereich diese Komponente verantwortlich ist. Anschliessend ist es etwas einfach sowie uebersichtlicher sich auf die Struktur festzulegen, da man sämtliche Nomen als Klassen ansehen kann (hier einmal orange dargestellt). Verben spiegeln die benoetigten Methoden wieder (hier gruen dargestellt).

#### 1. Benutzer

- als Benutzer möchte ich den aktuellen Spielstand in eine Datei mit Auswahl des Dateinamens speichern . Das Logging wird mit gespeichert .
- als Benutzer möchte ich eine Datei mit einem Spielstand auswählen und öffnen können.
- als Benutzer möchte ich ein Spiel initialisieren und neu starten.
- als Benutzer möchte ich ein Spiel beenden mit/ohne zu speichern .
- als Benutzer möchte ich das Laden oder Speichern loggen .

#### 2. Spieler

- als Spieler möchte ich einen Domino auswählen .
- als Spieler möchte ich einen Domino drehen .
- als Spieler möchte ich einen Domino setzen .
- als Spieler möchte ich das Stadtzentrum bewegen .

e) als **Spieler** möchte ich meine Aktionen **loggen**.

### 3. Spiel

- a) als **Spiel** möchte ich die **Spieldfelder** der Spieler **visualisieren**.
- b) als **Spiel** möchte ich die **Auswahlfelder** **visualisieren**.
- c) als **Spiel** möchte ich den aktuellen Spielstand der Spieler **anzeigen**.
- d) als **Spiel** möchte ich den **Gewinner** **anzeigen**
- e) als **Spiel** möchte ich den **Gewinner** **loggen**.

## 14.2. Realisationsanalyse

**Grundsätzlich benötigte Datenstrukturen** Um eine Partie spielen zu können werden erst einmal Spielsteine benötigt. Hierbei wurden diverse Klassen eingeführt die im Kapitel genauer beschrieben werden. Letztendlich bieten diese allerdings sämtliche benötigte Schnittstellen um einen Domino mit einer bestimmten Aufschrift an mit einer Position zu versehen.

Diese Dominos können auf Spielbrettern positioniert werden. Jeder Spieler besitzt hierbei eins, und die Gui ist in der Lage diese ordnungsgemäss darzustellen.

Um einen Domino wählen zu können ist es essentiell ein Konstrukt für eine Bank zu implementieren. Ich habe dabei eine Struktur gewählt in der sämtliche Spieler nicht anhand irgendeines Indices, sondern anhand ihrer Referenz gespeichert werden. Dies ermöglicht eine genaue Zuordnung, da es ja möglicherweise Spieler mit identischem Index geben könnte.

Kapitel mit Beschreibung der Dominos einbinden

**Benutzerschnittstelle** Der Punkt Benutzer entspricht in diesem Projekt sämtlichen Anfragen die ein Benutzer dem Programm stellen kann. Es bietet sich an eine Struktur zu wählen in der eine Klasse oder Schnittstelle als Anlaufstelle dient über die sämtliche Anfragen bearbeitet werden können. Hierbei ist nur abzuwägen ob man eventuell die Antwort des Programms gleich in diese Schnittstelle mit aufnimmt. Dies führt allerdings zu unübersichtlichem Code.

Anführungszeichen

**Spieler** Der Punkt Spieler beschreibt die wirklichen Spielteilnehmer. Er muss in der Lage sein selbstständig oder durch die Interaktion mit der Gui einen Zug vollziehen zu können. Hierzu gehört das auswählen, drehen und setzen eines Dominos. Hier gilt es abzuwägen ob dies durch eine gemeinsame Klasse geschehen soll, ich habe mich allerdings für eine Unterteilung entschieden da der menschliche Spieler auf die Eingabe des Benutzers, über die Benutzeroberfläche, arbeitet, während die Bots diese mehr oder weniger ignorieren und isoliert ihre Züge vollziehen.

Des Weiteren benötigt der Spieler ein Spielbrett auf dem er Dominos setzen kann. Man könnte das Spielbrett auch der Verwaltung (also der Spiel-Klasse) überlassen, dann wäre ein Spieler aber nicht mehr so unabhängig vom Spiel wie in diesem Fall. Es ist so möglich

eine minimale Schnittstelle dem Spiel gegenüber bereitzustellen, indem der Spieler selbst alle wichtigen Schritte ausführt um einen Zug zu machen.

Die Kapsellung des Spielerverhaltens ermöglicht es außerdem wartbaren Code zu erzeugen. Es ist einfacher Fehler zu beheben oder bestimmte Prozesse auszutauschen ohne das komplette Programm umstrukturieren zu müssen, aber am wichtigsten hierbei ist die Möglichkeit der Erweiterbarkeit. Durch die Kapselung ist es möglich sämtliche Schritte eines Spielers polymorph zu gestalten. Jede Spielerart reagiert also anders auf eine Anfrage und es nicht nötig den Aufruf hierfür zu verändern. All diese Möglichkeiten würden entfallen, wenn man das Spielverhalten der künstlichen Spieler in der Spiel-Klasse aufrufen würde.

Moegliche  
Erweiterung  
referenzieren...  
das intro  
fenster

**Spiel** Dieser Begriff beschreibt koordinierte Abarbeitung der Spielerzüge. Es wird der Benutzeroberfläche mitgeteilt was angezeigt werden soll. Außerdem werden sämtliche sämtliche Stapel von Dominos bereitgestellt die für ein vollwertiges Spiel genutzt werden sollen (Beutel, Baenke). Alle benötigten Dateioperationen werden hier eingeleitet aber nicht direkt in dieser Klasse bearbeitet. Durch das ganze Exceptionhandling wird es ziemlich unübersichtlich alles in das Spiel zu packen, da dieses in erster Linie für die übergeordnete Organisation des ganzen Programms gedacht ist.

**Log** Da man vermehrt, und vor allem an vielen verschiedenen Stellen im Code, eine neue Zeile in die Logdatei schreiben beziehungsweise auf der Konsole ausgeben möchte, bietet sich für die Implementierung des Loggers das *singleton Muster* an. Dieses Muster verwaltet eine einzige globale Instanz auf die immer wieder drauf zugegriffen wird. Das Muster wird eignet sich besonders gut für das Loggen von Daten, da man alles in die selbe Datei schreiben möchte und diese nicht jedesmal neu suchen muss. Im Logger kann man zum Beispiel einfach ein entsprechendes Feld anlegen.

**Speichern und Laden** Auch beim Speichern und Laden wird in diesem Entwurf ein *singleton Muster* verwendet. Da man beim Speichern jeweils den Dateipfad nach erstmaliger Eingabe nicht erneut eingeben möchte wenn dies nicht unbedingt erforderlich ist (siehe Abschnitt Speichern als gegenüber Speichern). Und auch das Speichern und Laden tritt immer wieder vermehrt und verteilt über den gesamten Code auf. Alternativ könnte man auch eine klassische Klasse verwenden, wegen den genannten Punkten empfiehlt sich aber gerade für diese beiden Anwendungsfälle dieses Muster.

Benutzerhandb.  
einbinden

**Große Klasseneuebersicht** Mit dieser Übersicht bin ich zu folgender groben Klassenstruktur gelangt:

- Interfaces: Waren zwar mehr oder weniger durch die Bonusaufgabe vorgegeben, aber gerade zum Testen des Programms bietet sich die Verwendung dieser Interfaces natürlich an.
  - GUI2Game
  - GUIConnector

Verweis  
auf ge-  
nauere  
Beschrei-  
bung des  
Musters  
einbinden

## **15 BESCHREIBUNG GRUNDLEGENDER KLASSEN**

---

- Klassen
  - Game
  - Player
  - Verschiedene Ausprägungen der abstrakten Spieler-Klasse
  - District
  - Bank
  - Entry
  - Board
  - Domino
  - Pos
  - Logger
- Aufzählungstypen
  - Tiles
  - SingleTile
  - DistrictType

## **15. Beschreibung grundlegender Klassen**

## **Literatur**

- [1] City Domino aufgabenstellung. <http://intern.fh-wedel.de/mitarbeiter/klk/programmierpraktikum-java/aufgabetermine/ss18-citydomino/>. Aufgerufen am: 29-12-2018.

**A. Im Anhang Eins**