Programmieren & Datenstrukturen

Dokumentation Projekt „Dots & Boxes“

R. Diehl, R. Gisler, M. Klaper, R. Meier

Versionen:

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Datum | Autor | Bemerkungen |
| 1.0.0 | 27.03.2015 | R. Diehl | Initiale Version |
| 1.1.0 | 27.04.2015 | Doz. Team | Überarbeitung aller Kapitel |
| 1.1.1 | 04.05.2015 | Doz. Team | Freigabe |
| 1.2 | 22.05.2015 | LMalushi | Klassendiagramm erstellt |

**Inhalt**

1. Einleitung 2

2. Anforderungen 3

2.1. Regeln „Dots & Boxes“ 3

2.2. Tipps zur Vorgehensweise 4

2.3. Environment 4

3. Systemspezifikation 5

3.1. Bausteinsichten 5

3.2. Datensicht 8

3.3. Netzwerkprotokoll 8

4. Erweiterungsmöglichkeiten 9

5. Lessons learned 10

5.1. Java 10

5.2. Projektdesign 10

5.3. Gruppendynamik 10

# Einleitung

Geschätzte Studierende

Im letzten Teil des Semesters bearbeiten Sie in einem kleinen Team ein Softwareprojekt. Mit diesem Projekt sind die folgenden Ziele verbunden:

* Sie wenden im Unterricht gelernte Konzepte der Sprache Java in einem grösseren Kontext an.
* Sie wiederholen wesentliche Elemente der Programmiersprache Java.
* Sie implementieren eine Softwarelösung im Team.
* Sie können Klassendiagramme lesen und interpretieren.
* Sie können Sequenzdiagramme lesen und interpretieren.
* Sie können Zustandsdiagramme lesen und interpretieren.
* Sie können Source-Code mit Hilfe von Klassendiagrammen dokumentieren.
* Sie können in einem grösseren Programm die Übersicht wahren.

Das gesamte Projekt ist als Lernprojekt zu verstehen, bei dem Sie Schritt für Schritt Kenntnisse erwerben und anwenden.

Die Dozierenden und Assistierende begleiten Sie während des Projekts und ermöglichen Ihnen, wesentliche Erfahrungen zu reflektieren. Zur Unterstützung dieses Prozesses erstellen Sie jede Woche für die Dozierenden einen kurzen Projekt-Statusrapport mit folgendem Inhalt:

* Welche Arbeiten wurden in der letzten Woche ausgeführt. Was hat gut geklappt, wo hatten oder haben Sie Probleme?
* Welche Tätigkeiten sind für die nächste Woche vorgesehen?
* Welche Knackpunkte (Herausforderungen oder Risiken) bestehen noch? Was gedenken Sie dagegen zu unternehmen?

Eine Vorlage für diesen Projekt-Statusrapport [2] finden Sie im ILIAS.

Viel Erfolg sowie spannende und wertvolle Projekterfahrungen wünschen Ihnen

Ihr Dozierendenteam

[1] Projektauftrag Dots & Boxes (dots\_and\_boxes.pdf)

[2] Projekt-Statusrapport (PRG2\_Projekt-Statusrapport.docx)

# Anforderungen

|  |  |
| --- | --- |
| **Name** | Spiel gegen Computer(AI) |
| **Beschreibung** | Ein lokaler Spieler kann gegen den Computer (AI) spielen. |

|  |  |
| --- | --- |
| **Name** | Speichern und laden des Spielstandes gegen Computer |
| **Beschreibung** | Spielstand muss als Datei speicherbar sein. Nach Neustart des Programms, muss der Spielstand wieder verfügbar sein. |

|  |  |
| --- | --- |
| **Name** | Spiel über das Netzwerk |
| **Beschreibung** | Über das Netzwerk kann man gegen einen menschlichen Gegenspieler antreten. Dazu wird die IP Adresse benötigt. |

|  |  |
| --- | --- |
| **Name** | Spielregeln |
| **Beschreibung** | Die Regeln für das Spiel Dots and Boxes müssen gemäss der Dokumentation von Ilse Fischer implementiert werden. |

|  |  |
| --- | --- |
| **Name** | Abgabetermin |
| **Beschreibung** | 28. Mai 2015 bis 12:00 Uhr |

|  |  |
| --- | --- |
| **Name** | Benutzeroberfläche |
| **Beschreibung** | Es wird ein GUI erstellt. In diesem sind die aktuellen Spieler, der aktuelle Spielstand sowie die Information für den Spieler, welcher den nächsten Zug setzen kann ersichtlich. |

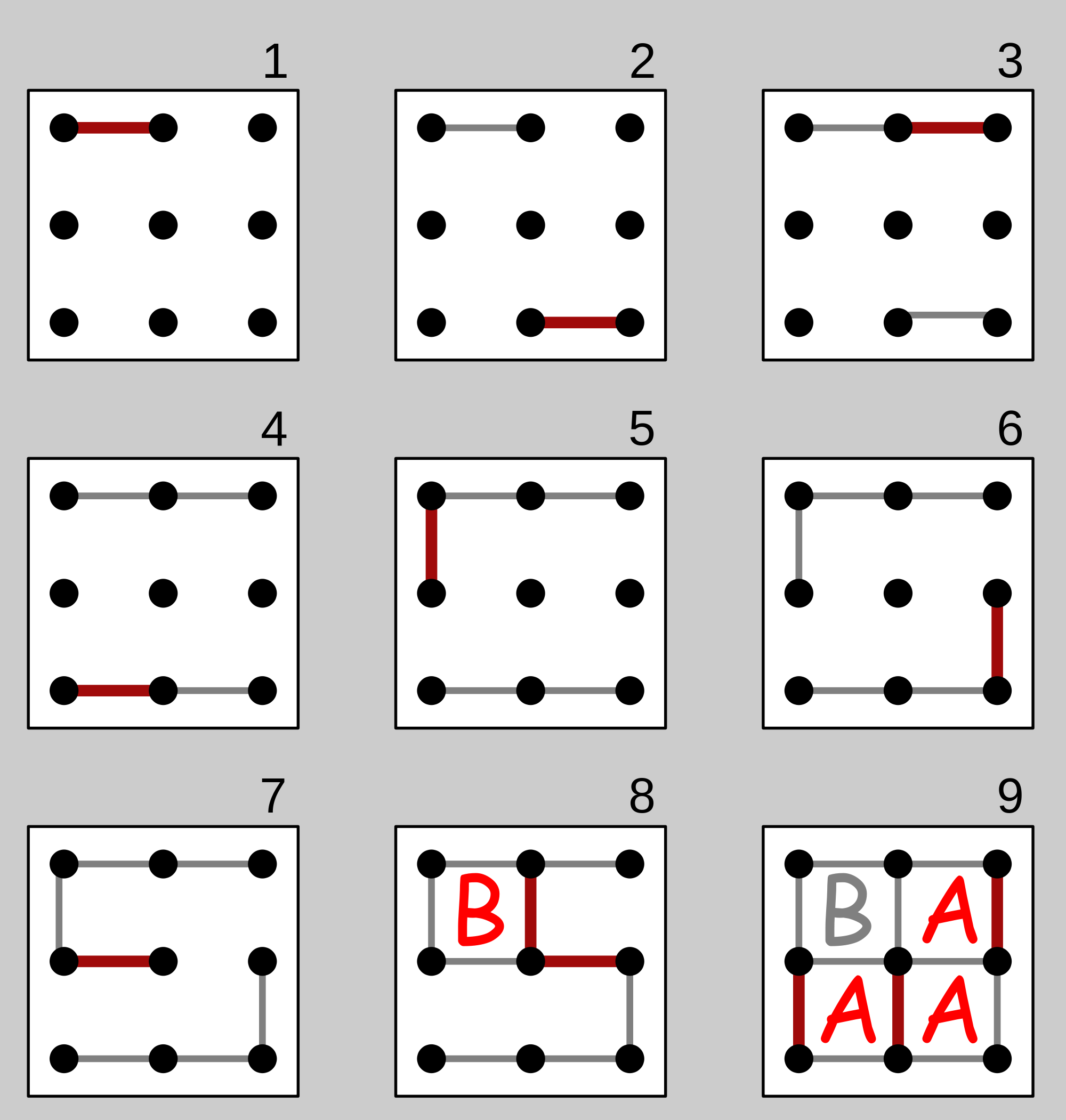
## Regeln „Dots & Boxes“

Zwei Spieler beginnen mit einer quadratischen (oder rechteckigen) Anordnung von Punkten.

Ein Zug besteht darin, zwei horizontal oder vertikal benachbarte Punkte durch einen Strich zu verbinden.

Hat ein Spieler, eine Spielerin bei ihrem Zug ein Einheitsquadrat vollendet, so kennzeichnet er/sie dieses Kästchen (z.B. füllt das Kästchen mit der Strichfarbe des Spielers) und macht einen weiteren Strich.

Das Spiel ist zu Ende, wenn alle Kästchen vollendet sind. Die Siegerin, der Sieger ist derjenige oder diejenige mit den meisten Kästchen.



## Tipps zur Vorgehensweise

Vorgehen:

* Identifizieren Sie die Objekte im Spiel
* Identifizieren Sie die Eigenschaften der Objekte – erstellen Sie daraus die Klassen
* Suchen Sie Beziehungen zwischen den Klassen

Beachten Sie dazu folgende Hinweise

* Entwerfen Sie in objektorientierter Denkweise
* Erstellen Sie kleine Klassen, dafür mehrere Klassen
* Beachten Sie Kopplung, Kohäsion und Information Hiding
* Setzen Sie Interfaces ein
* Benutzen Sie die Model-View-Controller Struktur (siehe Abschnitt 3.1.4)
* Denken Sie ans Testing und Debugging

## Environment

* Für das Spiel über Netzwerk muss Traffic über UDP erlaubt sein, insbesondere müssen UDP-Broadcasts erlaubt sein.
  + Weil UDP-Broadcast im HSLU-Netz nicht erlaubt sind, wird ein Wifi Access-Point im F‑Stock/Bunker zur Verfügung gestellt. Die Angaben dazu lauten:
    - SSID: PRG2\_WLAN
    - Passwort: JavaIsFun
* Java muss mit (mindestens) der Version 1.7 installiert sein.

# Systemspezifikation

## Bausteinsichten

### Systemübersicht



### Klassendiagramme

Für bessere Einsicht bitte zoomen.



### DotAndBoxes

Eigentliches Spiel: Erzeugt das Dialogfenster um die Parameter für ein neues Spiel abzufragen, lässt das entsprechende Spielfeld erzeugen, zeigt Spielfeld, Statusbar und Menu an, erzeugt den Computergegner,…

Hier ist auch die main-Methode untergebracht.

### CreateGameDialog

Erstellt ein Dialogfenster, das den Spieler nach dem Namen und der Spielfeldgrösse fragt. Diese Informationen werden dann verwendet, um das Spiel zu starten.

### StatusBar

Zeigt den Score der Spieler an und wer an der Reihe ist.

### Game

Diese Klasse enthält die Spiellogik. Sie enthält die Informationen über die Spieler, wer aktuell spielt und erzeugt die Liste der Boxen. Hier wird das GameEvent erzeugt, dass bei jedem PictureEvent ausgelöst wird.

### Player

Player ist der menschliche Spieler, aber auch die Oberklasse für Enemy. Sie speichert den Namen und den aktuellen Score.

### Enemy

Diese Unterklasse von Player wurde als Oberklasse für ComputerPlayer und ServerPlayer vorgesehen. Aus zeitlichen Gründen musste der ServerPlayer aber weggelassen werden. Enemy wurde als abstrakte Klasse realisiert, da sie das EnemyEvent auslöst, aber die Funktion play() nicht implementiert.

### ComputerPlayer

Für den Computergegner wurde die Klasse ComputerPlayer erstellt. Sie erzeugt einen Spieler mit dem Namen „Computer“, speichert den Score, implementiert die Funktion play() und erzeugt das EnemyEvent.

### Box

Die Box speichert die vier dazugehörigen Linien. Damit das Erstellen aus einer Liste funktioniert, wurden drei verschiedene Konstruktoren benötigt.

### Line

Line steht für die Linien zwischen den Boxen. Sie speichern nur, ob sie markiert wurden, oder nicht.

### Gameboard

GUI des Rasters: Stellt mit Hilfe der BoxPicture-, LinePicture- und DotPicture-Klassen das eigentliche Spielfeld dar. Für die Anordnung wird ein GridBagLayout verwendet.

### BoxPicture

GUI der Box-Klasse: Jede Box wird als Attribut an ein BoxPicture übergeben. Diese Klasse übernimmt dann die visuelle Darstellung der Box.

### LinePicture

GUI der Line-Klasse: Jede Line wird als Attribut an ein LinePicture übergeben. Diese Klasse übernimmt dann die visuelle Darstellung der Line. Dieses Objekt reagiert auf MouseClicked, MouseEntered und MouseExited (für das Hervorheben der Linie, wenn die Maus darüber fährt).

### DotPicture

GUI für einen Punkt. Wird nur für die Darstellung des Rasters verwendet.

## Datensicht

Aus zeitlichen Gründen musste auf eine Speicherung des Spielstandes verzichtet werden. Teilweise wären Konstruktoren vorhanden, um ein gespeichertes Spiel zu laden, aber nicht bei allen Klassen.

## Netzwerkprotokoll

Leider musste auch auf das Spiel über das Netzwerk aus zeitlichen Gründen verzichtet werden. In der Planung zu Beginn des Projekts wurde dieser Spielvariante zu wenig Beachtung geschenkt. Es wurden zwar Überlegungen angestellt, wie man dieses Spiel realisieren könnte, aber die Versuche mussten abgebrochen werden, da ein umfassendes Refactoring nötig wäre.

# Erweiterungsmöglichkeiten

Von den vorgeschlagenen Erweiterungen wurde die rechteckige Spielfeldform realisiert. Dadurch, dass bereits bei Beginn des Projekts alles auf einer rechteckigen Spielform aufbaute, konnte dieser Zusatz ohne mühsamen Aufwand umgesetzt werden. Auf weitere Ausschmückungen wurde verzichtet.

Mögliche Erweiterungen:

* Netzwerkspiel
* Speichermöglichkeit
* Individuelle Wahl der Farbe
* Multiplayer lokal
* Schwierigkeitsgrad des Computergegners
* Automatische Gegnersuche

# Lessons learned

## Java

Die ganze Arbeit mit dem GUI und den diversen Layouts (vor allem GridBagLayout) hat das allgemeine Verständnis stark verbessert. Was zusätzlich zum Unterricht noch erlernt werden musste, war die Verwendung von Constraints. Auch neue GUI-Komponenten (JDialog) mussten im Selbststudium erlernt werden.

## Projektdesign

Zu Beginn des Projekts wurden bereits Überlegungen getätigt, die zu tief in die Implementierung gingen. Wir wollten das Spiel in allen Facetten verstehen und verstrickten uns zuerst im Raster. Nach ein paar Tagen und ein paar ersten Implementationsversuchen, hatten wir das nötige Verständnis und auch die nötige Distanz um das Projekt ganzheitlicher zu sehen. Leider mussten wir gegen Ende des Projekts merken, dass zu Beginn zu wenige Überlegungen Richtung Netzwerk und Speicherung gingen. Als Folge davon mussten wir auf diese beiden Aspekte des Spiels verzichten.

## Gruppendynamik

In einer Gruppe gibt es immer Dinge, die besser klappen und Sachen die überhaupt nicht wollen. In unserer Gruppe hatten wir Glück, dass jeder seinen Teil (mehr oder weniger wie von den anderen erwartet) gemacht hat. Keiner war sich zu scheu, ein Blatt vor den Mund zu nehmen, wenn wir unterschiedlicher Ansichten waren. Trotzdem waren wir im Umgang miteinander stets freundlich und konstruktiv. Dieses Projekt hat uns gut gezeigt, wie man zuerst herausfinden muss, wer welche Stärken und Schwächen hat, und diese dann zielgerichtet ins Projekt einfliessen. Diese Erfahrung hat uns alle sehr bereichert.