Hochschule Hamm-Lippstadt

Soziale Medien und Kommunikationsinformatik

Sommersemester 2017

Programmieren II

Dozent: Stefan Henkler

**„Vier Gewinnt“**

|  |  |  |  |
| --- | --- | --- | --- |
| Kevin-Karl Klages  Matr.-Nr.: 2150121 | Jens Michalke  Matr.-Nr.: 2152013 | Lucille Reeve  Matr.-Nr.: 2150229 | Hannah Kuhn Matr.-Nr.: 2150668 |

Lippstadt, 14.07.2017**Inhaltsverzeichnis**

1. Einführung 3

1.1 Zweck und Zielgruppe 3

1.2 Ziele des Projektes 3

1.3 Referenzen 3

2. Funktionale Anforderungen 4

2.1 UML-Diagramm 5

2.2 Muss-Funktionen 5

**2.2.1 GUI 5**

**2.2.2 Server 6**

**2.2.3 Client 6**

**2.2.4 Gewinnabfrage 6**

2.3 Benutzercharakteristika 7

2.4 Soll-Kriterien 8

2.5 Erweiterungen 8

3. Management Summary 8

3.1 Arbeitspakete 8

3.2 Probleme 9

3.3 Lösung 9

3.4 Lessons Learned 10

# 1. Einführung

Im nachfolgenden Kapitel werden die Ziele, die Zielgruppe und der Zweck des Projekts erläutert. Außerdem erfolgt eine Referenzangabe, der verwendeten Literaturquellen.

## Zweck und Zielgruppe

Zweck des „Vier Gewinnt“ Spiels ist die Unterhaltung. Das Spiel ist für Menschen jeder Altersklasse vorgesehen.

## 1.2 Ziele des Projektes

* Die Programmiersprache in der das Projekt implementiert werden soll ist Java
* Aufgabe ist es Mehrpersonenspiel umsetzen
* Mit Grafischer Oberfläche
* Verteiltes Spiel
* Spieler können an unterschiedlichen Rechnern gegeneinander spielen
* Computergegner nicht Teil der Aufgabe

## 1.3 Referenzen

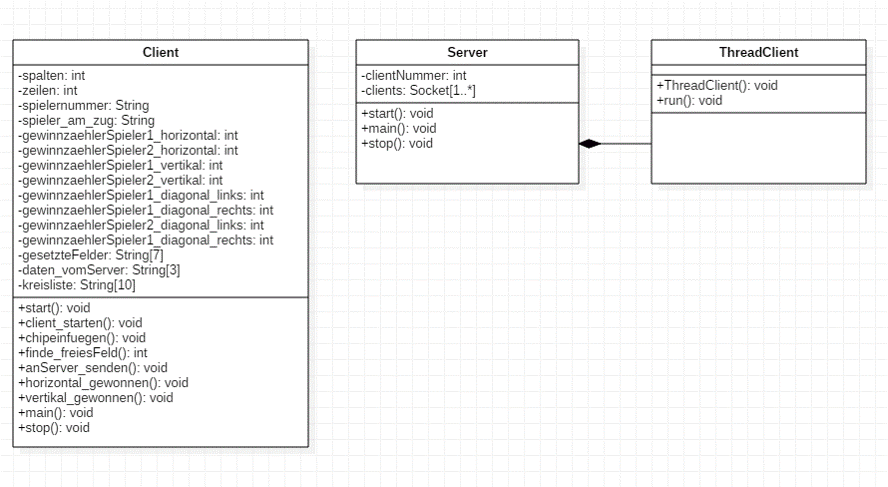
* Oracle Docs: https://docs.oracle.com/javase/7/docs/api/java/net/ServerSocket.html
* Java ist auch eine Insel: http://openbook.rheinwerk-verlag.de/javainsel/

# 2. Funktionale Anforderungen C:\Users\Hannah\Documents\Studium\4. Semester\2. Semester - Programmieren2\59fcf5e0-662c-4a76-8ccf-0bde09b7f5df.jpg

Zu Beginn haben wir erst ein Prototyp Paper angefertigt, um uns einen groben Überblick über den zu entwickelnden Code zu verschaffen. Der Fokus lag dort zum größten Teil auf der GUI mit dem Spielfeld und der Gewinnabfrage. Außerdem wurden zuerst die Spielregeln, abgeleitet vom „Originalen“ Brettspiel, zusammengetragen. Daraufhin haben wir uns für das klassische „Vier Gewinnt“ entschieden, bei dem das Spielfeld aus 7 Spalten und 6 Zeilen besteht. Jeder Spieler besitzt 21 Steine und setzt abwechselnd jeweils einen Stein in das Spielfeld. Dieser fällt immer bis zum untersten freien Platz durch. In dem Zusammenhang kam die Idee ein zweidimensionales Array zu verwenden. Die Spalten und Zeilen bilden die Indizes und verweisen auf den jeweiligen Spieler beziehungsweise auf ein leeres Feld. Gewonnen hat derjenige, der vier Steine einer Farbe entweder horizontal, vertikal oder diagonal angeordnet in einer Reihe hat. Tritt einer dieser Fälle nicht ein und das komplette Spielfeld ist belegt, liegt ein Unentschieden vor.

## 2.1 UML-Diagramm

Das nachfolgende UML Klassendiagramm zum „Vier Gewinnt“ Programm zeigt die Beziehungen der Klassen und deren Attribute und Operationen auf.



## 2.2 Muss-Funktionen

Zu den Muss-Funktionen des Programms zählen eine GUI für Eingaben der Spieler und Anzeigen des Spiels. Zusätzlich muss ein Server vorhanden sein, um das Spielen über ein Netzwerk zu ermöglichen. Ein Client muss in der Lage sein sich mit dem Server zu verbinden, mit ihm zu kommunizieren und ist unter anderem für die Gewinnabfrage zuständig.

**2.2.1 GUI**

Für die Benutzeroberfläche wurde auf die Benutzung der Java FX Bibliothek zurückgegriffen. Verwendet wird ein Grid Pane Layout zur Realisierung des Spielfeldes. Das Spielfeld besteht wie beim klassischen Vier Gewinnt Spiel aus sieben Spalten und sechs Zeilen. Das Hinzufügen der Kreise auf das Spielfeld wird durch eine for-Schleife realisiert, welche die einzelnen Kreise in das Grid Pane einfügt.

**2.2.2 Server**

**Pfhpwohfpwqohjfpo**

**2.2.3 Client**

Der Client beinhaltet das User Interface und die Logik des Spiels. Bei Start des Clients wird zuerst die Benutzeroberfläche generiert. Verbindet man sich mit einem Server, so wird dem Benutzer eine Spielernummer/ Zuschauernummer zugeordnet und ein neuer Thread gestartet, um Nachrichten vom Server abzufangen. In den Client sind neben den im folgenden Teil erklärten Gewinnabfragen noch weitere Methoden implementiert. Darunter zählen einen neuen Chip in das Spielfeld einzufügen, es an die richtige Stelle einzufügen und den Spielzug an den Server zu kommunizieren, damit auch der Mitspieler und die Zuschauer den aktuellsten Spielstand angezeigt bekommen.

**2.2.4 Gewinnabfrage**

* **Vertikale Gewinnabfrage**

Für die vertikale Gewinnabfrage wird der Spieler der zuletzt einen Stein gesetzt hat in einem zweidimensionalen Array abgespeichert. Verweisen die Indizes des Arrays, die in diesem Fall die Spalte und Zeile darstellen in der der Stein gesetzt wurde, auf Spieler 1 so wird die komplette Methode für den jeweiligen Spieler durchlaufen. Die Methode vergleicht dann jedes einzelne Feld in der festgelegten Spalten und überprüft ob vier Steine einer Farbe vertikal in einer Reihe sind und speichert die Anzahl der Steine einer Farbe in der Gewinnzählervariablen. Wenn die Gewinnzählervariable dann 4 beträgt, wird die Schleife beendet und ein Text ausgegeben.

* **Horizontale Gewinnabfrage**

Ähnlich wie bei der vertikalen Abfrage wird auch bei der horizontalen Abfrage der Spieler der als letztes einen Stein gesetzt hat in einem Array abgespeichert. Allerdings wird hier lediglich die Zeile in der der Stein gesetzt wurde überprüft und die Spalten bis zum letzten Feld abgegangen. Insgesamt läuft die Schleife der Methode mindestens vier Mal und maximal sieben Mal durch. Wenn ein Stein der anderen Farbe in der Reihe enthalten ist, dann wird die Gewinnzählervariable auf Null gesetzt und die gesetzte Zeile wird für den jeweiligen Spieler bis zur letzten Spalte zu Ende überprüft. Auch bei der horizontalen Abfrage wird ein Gewinntext ausgegeben, wenn die Variable vier beträgt.

* **Diagonale Gewinnabfrage**

Für die diagonale Gewinnabfrage existieren zwei verschiedene Funktionen. Eine überprüft die diagonale nach links oben und die andere nach rechts oben. Hierbei wird die Gewinnzählervariable auf 1 gesetzt, da wir davon ausgehen, dass ein Stein bereits gesetzt wurde. Beide Methoden besitzen zwei for-Schleifen, eine die die diagonale Reihe nach rechts unten testet und eine die nach links oben testet.

Dazu wird eine zweite Zählervariable i immer 1 aufaddiert und auf die aktuelle Zeile gerechnet bzw. von der aktuellen Zeile subtrahiert. So gelangt man dann zu jedem neuen Feld der diagonale nach links und kann dann überprüfen, ob Steine einer Farbe nebeneinander liegen.

## 2.3 Benutzercharakteristika

Grundsätzlich kann das Spiel von jedem verwendet werden, der einen Computer besitzt, welcher Java Applikationen ausführen kann. Die einzige Schwierigkeit könnte bei der Anmeldung an den Server liegen. Hierfür muss man wissen, welche IP Adresse dem Server zugeordnet ist, der verwendet werden soll.

## 2.4 Soll-Kriterien

Es ist nicht möglich ein Unentschieden zu erzielen, da die passende Gewinnabfrage nicht implementiert wurde.

**2.5 Erweiterungen**

Da es in der heutigen Zeit immer beliebter wird anderen beim Spielen zu zusehen, haben wir auch einen Zuschauermodus eingebaut. Die ersten beiden Spieler die sich beim Server einloggen sind aktive Spieler und alle weiteren haben zwar die Einsicht auf das Feld, können aber keine Steine setzten oder andere Aktionen ausführen.

**3. Management Summary**

In der Management Summary werden die umfassende Teamarbeit und deren Aspekte wie die Aufteilung der Arbeitspakete und Abläufe der Teammeetings, sowie die aufgetretenen Probleme und deren Lösungsansätze und die gelernten Lektionen aufgezeigt.

**3.2 Arbeitspakete**

Das Projekt ist in 5 Haupt-Arbeitspakete aufgeteilt.

* **Server programmieren**  
  Es musste ein Server erstellt werden, welcher das Spielen über ein Netzwerk ermöglichen soll. Clients können sich auf diesem Server anmelden und über ihn für das Spiel relevante Nutzdaten austauschen.
* **Benutzeroberfläche erstellen**  
  Um das Spiel Vier Gewinnt wirklich vernünftig spielen zu können benötigt es eine Benutzerschnittstelle, die auf entsprechende Benutzereingaben reagieren kann.
* **Server-Client Verbindung implementieren**

Der Client soll sich auf dem Server anmelden können und eine Rolle im Spiel zugeordnet bekommen (Spieler 1, Spieler 2, Zuschauer…)

* **Spiel-Logik programmieren (Gewinnabfragen, richtiges Einfügen von Spielsteinen)**

Damit das Spiel vollständig funktioniert muss ständig überprüft werden, ob bereits ein Spieler gewonnen hat. Zudem muss eine Benutzereingabe auch richtig verarbeitet werden können und ein Spielstein an die entsprechend richtige Stelle eingefügt werden.

* **Kommunikation des Spielstandes über den Server**

Nach erfolgreicher Implementierung einer Verbindung zwischen Server und Client soll diese so erweitert werden, dass eine Änderung im Spiel (in diesem Fall die Benutzereingabe mit setzen eines Spielsteins) an den Server übermittelt wird und dieser die Änderung allen angemeldeten Clients mitteilt.

**3.3 Probleme**

* Zu Beginn hatten wir beim Erstellen der GUI Probleme die weißen Kreise in das Layout einzufügen. Deshalb sind wir von der SWING Bibliothek auf Java FX umgestiegen.
* Die Umsetzung neuer Konzepte wie beispielsweise die RMI - Methode fiel zunächst schwer.
* Programm an manchen Stellen sehr umständlich und komplexer geschrieben als nötig.

**3.4 Lösungen**

* Zur Lösung des Kreisproblems wurde das in der Java FX – Bibliothek vorgefertigte Kreisobjekt verwendet.

**3.5 Lessons Learned**

Wir haben auf jeden Fall aus dem Projekt mitgenommen, dass wir uns die Arbeit erschwert haben und wir unser Programm mit anderen Konzepten leichter und auch schöner hätten umsetzen können. Unsere Teamarbeit war in jedem Fall gelungen, wir konnten immer eine positive und konstruktive Kommunikation führen und die regelmäßigen Teammeetings erfolgten effizient und effektiv. Die Aufteilung der Arbeitspakete verlief einstimmig und es konnte zu jedem Treffen etwas abgeliefert werden. Unser Team zeichnete sich durch Verlässlichkeit und offener und ehrlicher Kommunikation aus.