



9. NOVEMBER 2018


DOKUMENTATION

226B | RUGGIERO MARKUS

IFZ-724-004

NOÉ LÜTHOLD

DELFTERSTRASSE 24 | 5003 AARAU



Inhaltsverzeichnis

Konzept	2
Projektplan	2
Ausgangslage	2
Projektziel	2
Spielanleitung	2
Anforderungskatalog	3
Skizzen	3
UML	4
Projektablauf	5
Klassen Beziehungen	5
Versionen	5
Quellen	6

Konzept

Wir haben als Schulprojekt die Aufgabe bekommen für die Imaginäre Firma TOPOMEDICS ein Spiel mit Java zu Programmieren welches dazu dienen soll das Kinder verschiedener Altersstufen im Wartezimmer von Ärzten an Einem Bildschirm mit Tastatur beschäftigt werden können. Dabei soll das Kind gegen den Computer Spielen und auch eine Chance zum Gewinnen haben. Falls Jemand das Spiel nicht kennt soll es auch eine aufrufbare Spielanleitung haben. Das Spiel soll natürlich keine Urheberrechte verletzen und ausserdem geht es nicht darum ein perfektes Spiel zu entwickeln, sondern viel mehr um den Zeitvertreib.

Projektplan

Ausgangslage

Wir haben von der Firma TOPOMEDICS den Auftrag bekommen ein Spiel in Java zu Programmieren welche Kinder jeglicher Altersgruppen spielen können und hoffentlich auch mögen. Der sinn dahinter ist das man eine Beschäftigung in den Warteräumen der Ärzte auch für Kinder hat. Dieses Spiel soll an einem Computer mit einer Tastatur spielbar sein.

Projektziel

Ich habe mich dazu entschieden ein möglichst simples und Schnelles Spiel zu nehmen welches auch einfach zu spielen und lernen ist und welches in ziemlich jeder Altersgruppe beliebt ist.

Daher bin ich zum Schluss gekommen das Tic-Tac-Toe perfekt auf diese Kriterien übereinstimmt und somit das Perfekte Spiel ist.

Ich hatte zuerst den Wunsch das Spiel auf einem Grafischen Interface darzustellen was ich später aber doch nicht machen wollte das dies ein viel grösserer Aufwand wäre da ich nicht vertraut bin damit.

Mein Ziel ist es Schlussendlich das ich eine Konsolen Anwendung habe, welche ohne Probleme läuft, eine Aufrufbare Spiel- und Bedienungsanleitung hat und ein Computer, gegen den man Spielen und auch gewinnen kann.

Spielanleitung

Tic-Tac-Toe:

Auf einem 3x3 grossem Spielfeld müssen 2 Spieler (in diesem Fall Sie gegen den Computer) abwechselnd ihre Zeichen (entweder Kreuz oder Kreis) in ein leeres Feld setzen und dabei gewinnt der Spieler der zuerst 3 seiner Zeichen neben einander Platziert hat.

Dabei kommt es nicht darauf an ob diese Horizontal, Vertikal oder Diagonal nebeneinanderstehen.

Wenn alle 9 Felder Besetzt sind ohne das 3 der gleichen Zeichen nebeneinander Stehen ist das Spiel zu Ende und es ist Unentschieden.

Anforderungskatalog

Die Versionen des Programmes habe ich nach dem Format Major/Minor/Patch erstellt und somit Steht die Erste Zahl für grössere Veränderungen, die Zweite für kleinere und die Dritte für Bug Fixes etc.

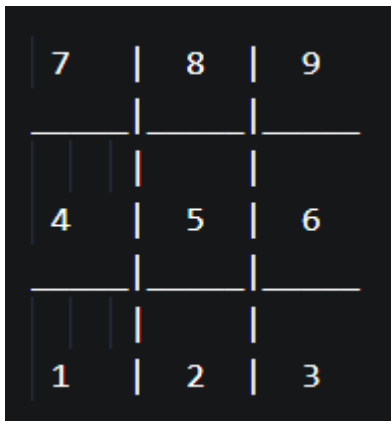
Funktionale Anforderungen	Version	Erfüllt
Abwechselnd Kreuz und Kreis	v0.0.0	Ja
Drei Zeichen führen zum Sieg	v0.0.1	Ja
1v1 im terminal	v0.1.0	Ja
Aufrufbare Spielanleitung	v0.1.1	Ja
Spielfähige "KI"	v1.0.0	Ja

Nicht-Funktionale Anforderungen	Version	Erfüllt
Keine Abstürze	v_._.+1	Ja
Keine Endlosschleifen	v_._.+1	Ja
Schnelle Reaktion nach Eingabe	v_._.+1	Ja
Ungültige Eingaben abfangen	v_._.+1	Ja
Hilfe Text in Deutsch	v_._.+1	Ja
Sinnvolle Kommentare im Code	v_._.+1	Ja

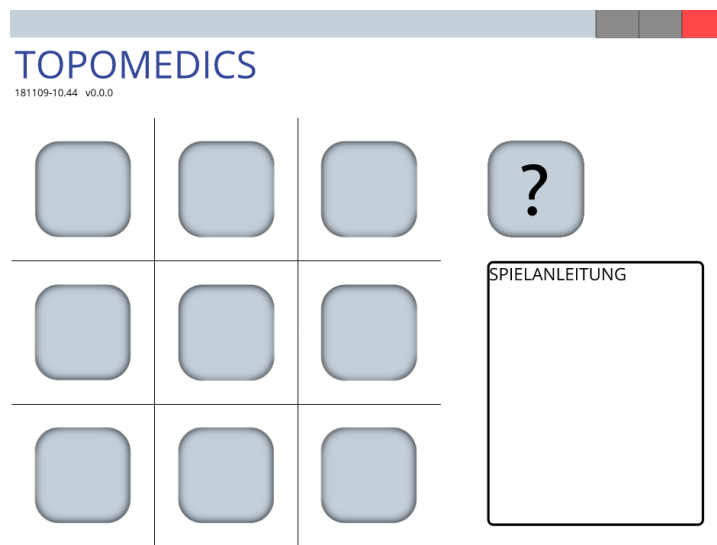
Skizzen

Erstmal als Konsolenanwendung und falls genügend Zeit dann noch eins Grafisches Interface:

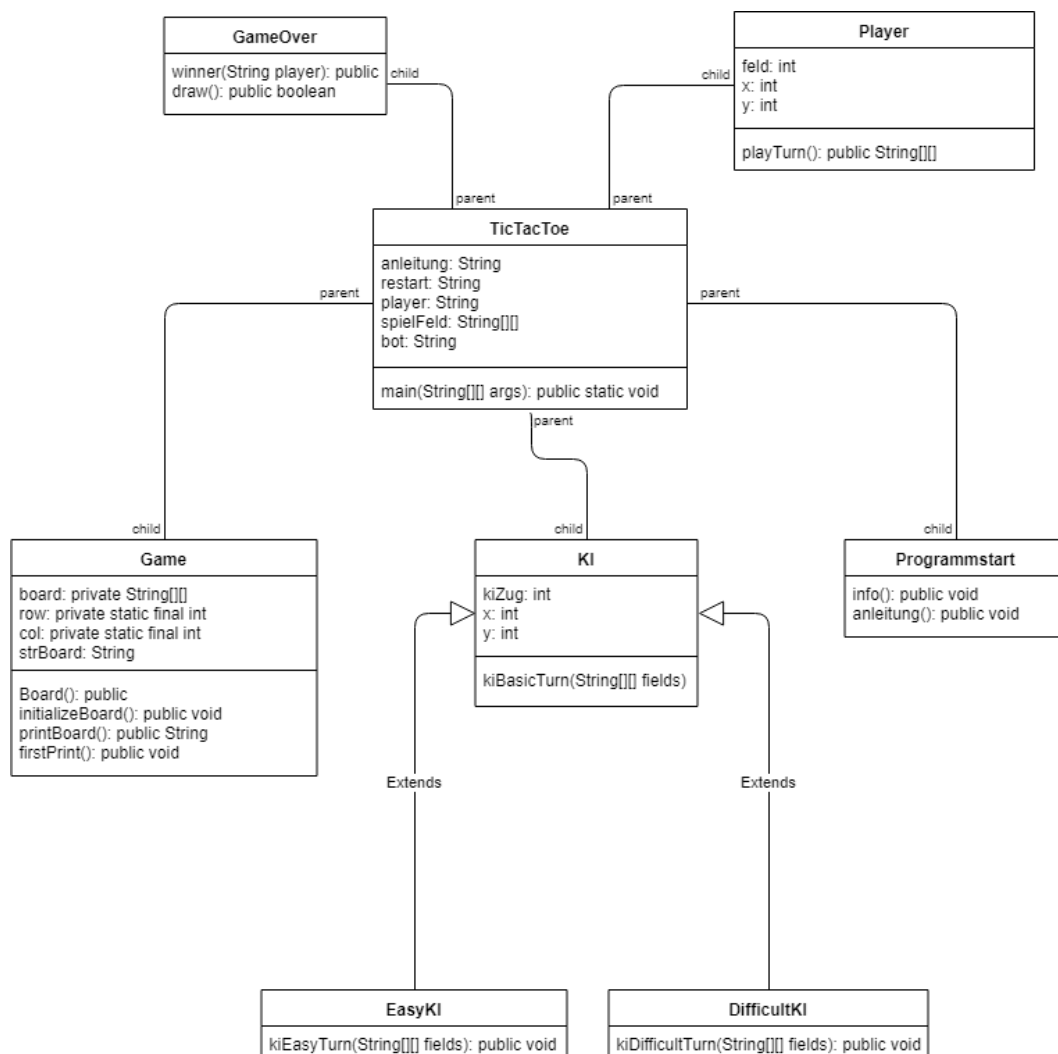
Dies ist eine Skizze für meine Konsolenabteilung, welche in einem Editor gemacht habe.



Und so sollte etwa das Grafische Interface aussehen falls ich genug Zeit habe



UML



Da ich leider nur noch das Bild habe und die veränderbare Datei verloren habe konnte ich die Methoden und variablen nicht mehr anpassen welche ich sowieso nicht genau schon im Voraus sagen kann.

Projektablauf

Zuerst habe ich alles in 2 Klassen gemacht, um zu sehen ob es überhaupt funktioniert so wie ich es gerne hätte, danach habe ich weitere Optimierungen vorgenommen und schlussendlich das Programm mithilfe von return values und Methoden Parametern in verschiedene Klassen so aufgeteilt wie ich es am liebsten hätte.

Ich habe mich ziemlich spät wieder daran erinnert, dass man Vererbungen benutzen muss und wusste nicht wo ich diese einfügen sollte, da ich keine Klasse habe, bei der es Sinn ergeben würde dass sie von einer andern erbt. Um dieses Problem zu beheben habe ich eine zweite Schwierigkeit für die KI erstellt, ich bin ziemlich stolz darauf auch wenn ich es nicht gerne sehe, dass eine Klasse mit einer Methode fast 300 Zeilen Code hat.

Klassen Beziehungen

Ich habe 7 verschiedene Klassen plus eine Klassen mit der Main Methode.

Wie diese Zusammenarbeiten kann man im UML oben gut sehen, dies ist nicht mehr das gleiche wie das aus dem Konzept vor einer Woche sondern hat jetzt alle aktuellen daten enthalten und stellt genau dar wie mein Programm jetzt aufgebaut ist.

Versionen

Version 0.1

In dieser Version habe ich erstmal das Spielfeld erstellt und den Benutzern erlaubt abwechselnd je ein Zeichen zu setzten, jedoch konnte man zweimal auf das gleiche Feld setzen, es passierte zwar nichts aber dann war wieder der andere dran.

Version 0.2

In dieser Version habe ich eingeführt das 3 gleiche Zeichen in einer Horizontalen, Vertikalen oder diagonalen Linie zum Sieg führen.

Version 0.3

In dieser Version habe ich eingeführt das man nicht 2 mal auf das gleiche Feld ein Zeichen setzten kann.

Version 0.4

In dieser Version habe ich erst bemerkt, dass es nicht im Unentschieden enden kann und musste dies Dringendst einfügen.

Version 0.5

Bis jetzt musste der Benutzer die Koordinaten des gewünschten Feldes eingeben und in dieser Version habe ich eingefügt, dass man es wie auf dem Numpad eingeben kann.

Version 1.0

In dieser Version habe ich die Erste KI erstellt, welche nur eine Zufällige Zahl zwischen 1 und 9 generierte und die wurde dann wie bei den benutzereingaben umgewandelt.

Version 1.1

In dieser Version habe ich das Board ein wenig angepasst und ein paar Bug fixes vorgenommen.

Version 1.2

In dieser Version habe ich die Klassen ein wenig weiter aufgeteilt, um das ganze Programm ein wenig übersichtlicher zu gestalten. Vor diesem Moment hatte ich etwa 4 und danach etwa 6

Version 1.3

In dieser Version habe ich einen weiteren Schwierigkeitsgrad für die KI entworfen welche natürlich noch ein paar Bugs hatten.

Version 1.4

Dies ist die finale Version und in dieser Version habe ich die Fehlerhaften Funktionen der Schweren KI abgeändert und ein paar Verbesserungen vorgenommen

Quellen

Die Basics für mein Programm habe ich von einem YouTube Video:

<https://www.youtube.com/watch?v=aSQgCtHBr8U>

Danach habe ich die meisten Probleme entweder mit ein paar Leuten von meiner Klasse besprochen und gelöst oder ich war auf dem Forum StackOverflow:

<https://stackoverflow.com/>