**Entwurfsarbeit**

**„Qwixx“**

Grafische Nutzerschnittstellen, WiSe 2019/20

Unter der Leitung von Dipl.-Inf., Dipl.-Ing (FH) Michael Wilhelm

**Autor:**

Name: Oliver Lindemann

Matrikelnummer: 26264

Inhalt

[Problembeschreibung 3](#_Toc30598752)

[Projektziel 3](#_Toc30598753)

[Problemstellung 3](#_Toc30598754)

[Motivation 4](#_Toc30598755)

[Implementierung 4](#_Toc30598756)

[Aufbau 4](#_Toc30598757)

[Beschreibung der Klassen 5](#_Toc30598758)

[Die Würfel (Dice) 5](#_Toc30598759)

[Der Spieler (Player) 6](#_Toc30598760)

[Das Spielfeld (GameBoard) 7](#_Toc30598761)

[Das Spiel (Game) 8](#_Toc30598762)

[Das User Interface (UI) 9](#_Toc30598763)

[Zusammenfassung 13](#_Toc30598764)

[Erweiterung 13](#_Toc30598765)

# Problembeschreibung

## Projektziel

Das Ziel dieses Projektes ist es, das taktische Würfelspiel „Qwixx“ als lauffähiges Computerspiel zu realisieren.

## Problemstellung

Es soll das Gesellschaftsspiel „Qwixx“ als digitale Version entwickelt werden. Dieses Spiel muss mit mindestens zwei, aber maximal mit fünf Personen gespielt werden. Die Mitspieler werden in der digitalen Fassung durch Computergegner simuliert.  
Es gibt kein gemeinsam genutztes Spielbrett, sondern jeder Mitspieler erhält zu Beginn eines jeden Spiels sein eigenes Spielfeld bzw. seinen eigenen Spielblock. Dieser besteht aus vier unterschiedlich eingefärbten Reihen mit jeweils elf Zahlenfeldern von zwei bis zwölf (oder in umgekehrter Reihenfolge). Zusätzlich zu den Spielblöcken gibt es sechs Würfel, welche aus zwei weißen und jeweils einem roten, gelben, grünen und blauen Würfel bestehen.  
Ziel des Spiels ist es auf seinem eigenen Spielblock möglichst viele Felder in den vier (Farb-) Reihen anzukreuzen. Je mehr Kreuze ein Spieler in einer Farbreihe hat, desto mehr Punkte erhält er dafür. Gewonnen hat derjenige, der am Ende insgesamt die meisten Punkte hat.  
Auf dem Spielblock dürfen die Felder nur von links nach rechts angekreuzt werden. Dabei muss allerdings nicht zwingend beim ersten Feld begonnen werden. Zudem dürfen Felder auch übersprungen werden. Diese ausgelassenen Zahlenfelder dürfen nachträgliche allerdings nicht mehr angekreuzt werden.

Gespielt wird im Uhrzeigersinn. Der aktive Spieler würfelt mit allen sechs Würfeln. Nun werden zwei Aktionen hintereinander ausgeführt.

Aktion 1: Die Augen der beiden weißen Würfel werden zusammengezählt. Die Summe darf nun jeder Spieler (muss aber nicht) in einer beliebigen Farbreihe auf seinem Spielblock ankreuzen.

Aktion 2: Der aktive Spieler darf jetzt als einziger noch die Farbwürfel verwenden. Hierbei kombiniert er einen Farbwürfel seiner Wahl mit einem der beiden weißen Würfeln (ebenfalls frei wählbar). Von beiden ausgesuchten Würfeln addiert er nun die Augenzahl und trägt die Summe in die Farbreihe des gewählten Farbwürfels ein.

Falls der aktive Spieler keine der beiden Aktionen ausführen möchte oder kann, so muss dieser auf seinem Spielblock in der Spalte „Fehlwurf“ ein Kreuz machen.  
Nicht-aktive Spieler, die lediglich die erste Aktion ausführen durften, müssen allerdings kein Kreuz bei den Fehlwürfen machen, falls diese die erste Aktion nicht ausführen wollten oder konnten.

Nach Abschluss der beiden Aktionen ist der nächste Spieler an der Reihe. Dieser Würfelt nun erneut mit allen sechs Würfeln. Danach werden wieder beide oben beschriebene Aktionen ausgeführt.

Das Spiel endet, sobald ein Mitspieler alle vier Fehlwürfe angekreuzt hat oder eine zweite Farbreihe abgeschlossen hat.  
Eine Farbreihe kann abgeschlossen werden, indem die letzte Zahl in einer Reihe ankreuzt und insgesamt mindestens sechs Kreuze in dieser Reihe gemacht hat. Sobald also ein Mitspieler das letzte Feld einer Reihe ankreuzt und dieses Kreuz das mindestens sechste in dieser Farbreihe ist, so beendet er dieser Reihe und darf das Zusatzfeld neben der Reihe (durch ein Schloss gekennzeichnet) ebenfalls ankreuzen.  
Die abgeschlossene Reihe ist nun auch für alle anderen Spieler geschlossen und es dürfen keine weiteren Kreuze in dieser Reihe gemacht werden.

Nach Beenden der zweiten Farbreihe oder dem Ankreuzen aller Fehlwürfe errechnen alle Mitspieler ihre Punktzahl. Der Spieler mit der höchsten Punktzahl gewinnt.

## Motivation

Implementierung:

## Aufbau

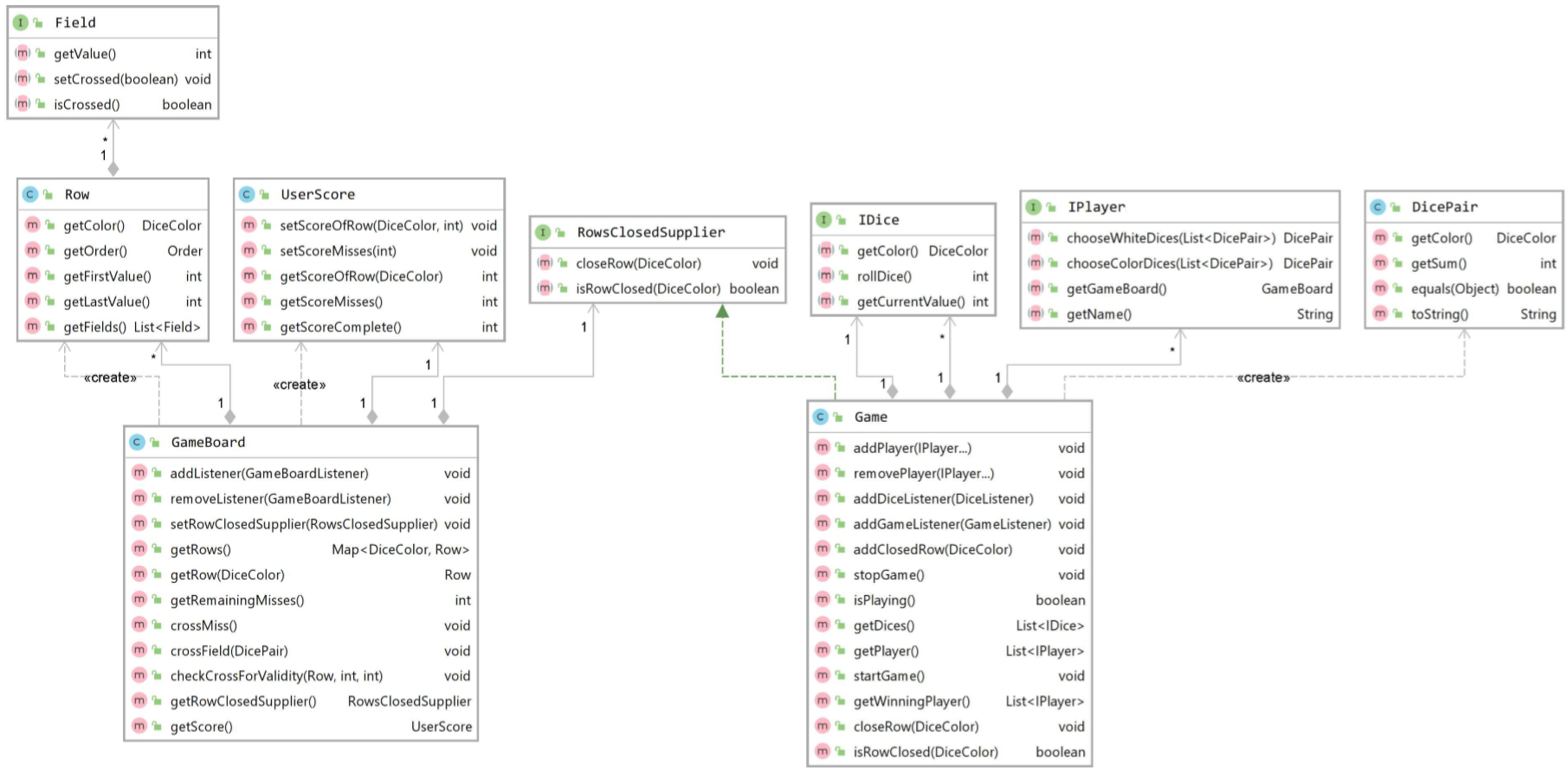
Ein Spiel (*Game*) besteht aus:

* 6 Würfeln (*IDice*) und die daraus kombinierten Würfelpaare (*DicePair*)
* Mehreren Mitspielern (*IPlayer*)

Ein Spieler hat genau ein Spielfeld (*GameBoard*), welches wiederum aus einem Score (*UserScore*) und verschiedenen Reihen (*Rows*) besteht. Jede Reihe hat zudem mehrere Felder (Fields).

Ein Spiel (*Game*) besteht aus 6 Würfeln (*IDice*) und mehreren Mitspielern (*Player*), welche jeweils ein Spielblock (*GameBoard*) mit verschieden farbigen Reihen (*Rows*) mit jeweils zwölf Feldern (*Fields*) haben.

Das Spiel (*Game*) implementiert zudem noch das Interface *RowClosedSupplier*, welches Methoden zur Abfrage von abgeschlossenen und zum Schließen von Reihen zur Verfügung stellt.  
Dieser *RowClosedSupplier* wird von jedem Spielfeld (*GameBoard*) benötigt, um eine Reihe abzuschließen.



## Beschreibung der Klassen

### Die Würfel (Dice)

Die Würfel haben in diesem Spiel repräsentieren nicht nur eine zufällige Augenzahl, sondern besitzen auch eine Farbe (DiceColor).

#### IDice

Das Interface IDice stellt drei Methoden zur Verfügung.

*getColor()* – liefert die Farbe dieses Würfels

*rollDice()* – würfelt diesen Würfel und liefert den geworfenen Wert

*getCurrentValue()* – Liefert den mit diesem Würfel zuletzt gewürfelten Wert

#### Dice

Diese Klasse implementiert das Interface IDice.

Über die Konstanten *MIN\_VALUE* und *MAX\_VALUE* stellt diese Klasse zudem den minimalen- und maximalen Wert zur Verfügung, der mit diesem Würfel geworfen werden kann.

#### DiceColor

Das Enum DiceColor beinhaltet verschiedene Farben, die ein Würfel haben kann. Dieses Enum beinhaltet noch zwei Methoden:

*getAsHex()* – liefert die Würfelfarbe in Hex-Darstellung  
*getDominantColor()* – liefert die dominante Farbe der gegebenen Farben

#### DicePair

Die Klasse DicePair stellt ein Paar von zwei Würfeln (IDice) dar. Ein Würfelpaar hat eine bestimmte Farbe (die dominante Farbe zweier Würfel) und eine Würfelsumme (Augenzahl zweier Würfel addiert).

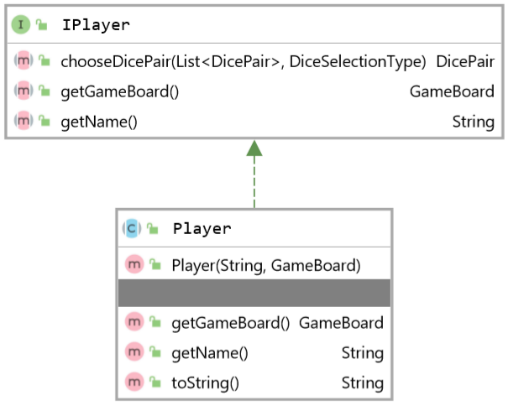
getColor() – liefert die Farbe dieses Würfelpaares  
getSum() – liefert die Summe dieses Würfelpaares

EMPTY – repräsentiert leeres Würfelpaar  
MISS – repräsentiert einen Fehlwurf

### Der Spieler (Player)

#### IPlayer

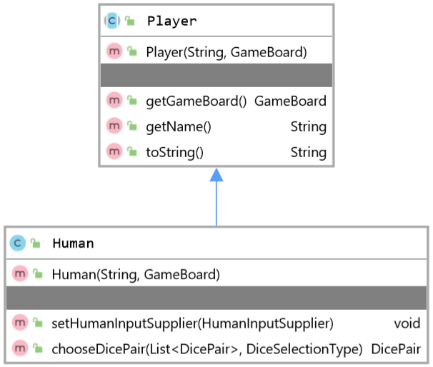
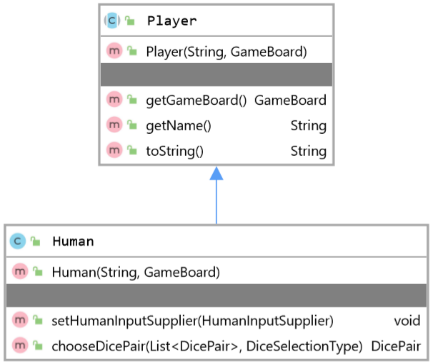
Ein Spieler hat einen Namen und ein Spielfeld. Zudem kann er über die Methode *chooseDicePair (List<DicePair>, …)* gefragt werden, welches Würfelpaar der Spieler aus der gegebenen Liste auswählen und auf seinem Spielfeld verwenden möchte.

*getName()* – liefert den Namen des Spielers  
*getGameBoard()* – liefert das Spielfeld des Spielers  
*chooseDicePair(…)* – fordert den Spieler auf, ein Würfelpaar aus der gegebenen Liste auszuwählen

#### Player

Die abstrakte Klasse Player implementiert die Methoden *getName()* und *getGameBoard()* des Interfaces IPlayer.  
Die Methode *chooseDicePair()* muss jedes Kind dieser Klasse noch selbst implementieren.

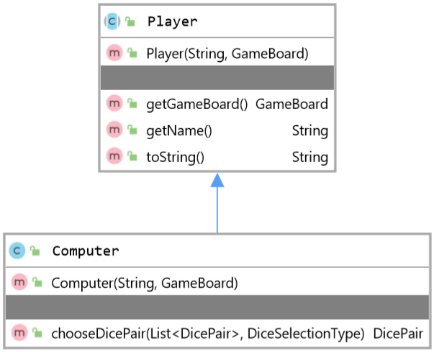
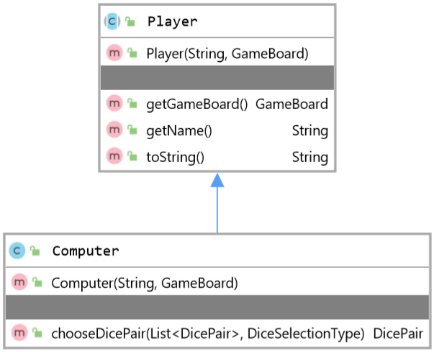
##### Human



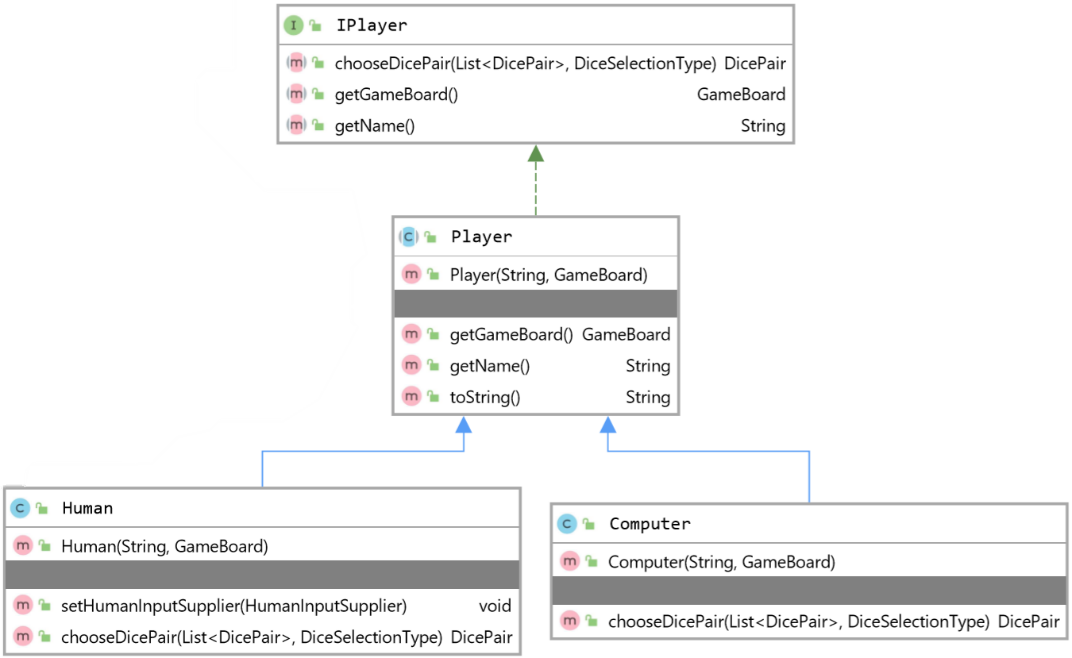
Die Klasse Human repräsentiert den menschlichen Spieler und erbt von der Klasse Player.

*setHumanInputSupplier(…)* – setzt für diesen menschlichen Spieler den *HumanInputSupplier*. Dieser entscheidet über die Auswahl des zu wählenden Würfelpaares.

##### Computer



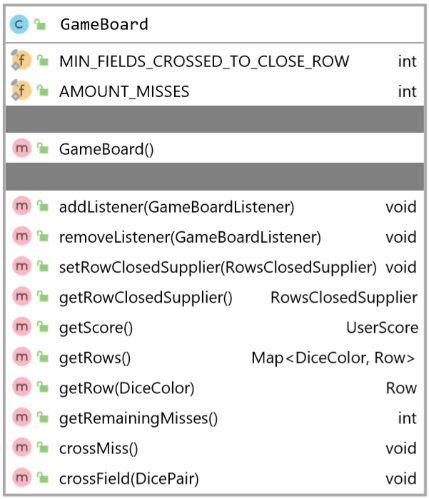
Die Klasse Computer repräsentiert einen Computerspieler und erbt von der Klasse Player. Diese Klasse stellt keine weiteren Methoden zur Verfügung.



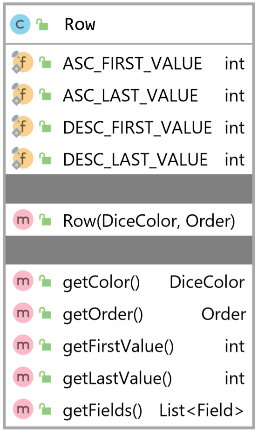
Die nebenstehende Grafik bildet bildet die vollständige Hierarchie der Player-Klassen ab.

### Das Spielfeld (GameBoard)

#### GameBoard

Das Spielfeld beinhaltet eine Liste von Reihen, den UserScore und die Anzahl der verbleibenden Fehlwürfe. Ein GameBoard benötigt zudem noch einen RowClosedSupplier, der über den entsprechenden Setter gesetzt werden kann.  
Um ein Feld auf diesem Spielfeld nun anzukreuzen, muss die Methode *crossField(DicePair)* aufgerufen werden. Diese prüft das gegebene Würfelpaar und kreuzt dieses in der entsprechenden Reihe an.  
Um einen Fehlwurf zu benutzen, muss die Methode *crossMiss()* verwendet werden.

Abseits der Methoden gibt es noch folgende Konstanten:  
*MIN\_FIELDS\_CROSSED\_TO\_CLOSE\_ROW* – Anzahl der Felder, die mindestens angekreuzt sein müssen, damit eine Reihe abgeschlossen werden kann  
*AMOUNT\_MISSES* – Anzahl der Fehlwürfe für dieses Spielfeld

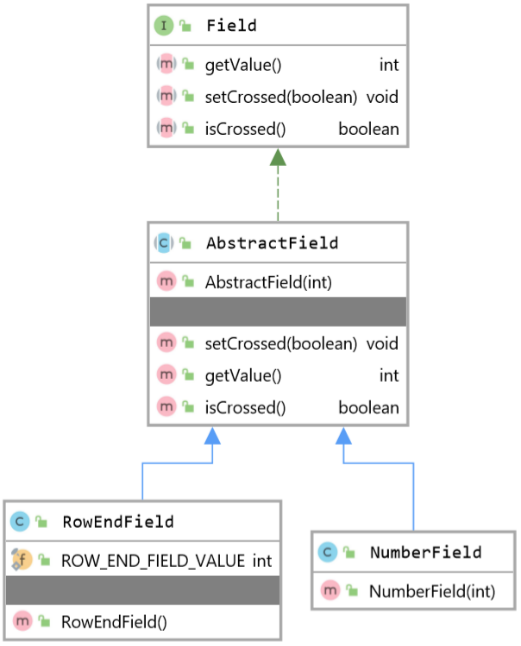


##### Row

Eine Reihe hat eine festgelegte Farbe und eine Zahlenreihenfolge (Order). Diese Reihenfolge bestimmt, ob die Werte der Felder dieser Reihe auf- oder absteigend sortiert sind.

Falls diese Reihe aufsteigend (ASC) sortiert ist, so besitzt das erste Feld den Wert der Konstante ASC\_FIRST\_VALUE und das letzte Feld den Wert von ASC\_LAST\_VALUE.   
Falls diese Reihe absteigend (DESC) sortiert ist, so werden die Werte der Konstanten DESC\_FIRST\_VALUE und DESC\_LAST\_VALUE verwendet.

###### Fields

Ein Feld hat einen festgelegten Wert und kann angekreuzt werden. Das Interface Field stellt folgende Methoden hierfür bereit:

*getValue()* – liefert den Wert des Feldes  
*setCrossed(boolean)* – der gegebene boolean gibt an, ob dieses Feld angekreuzt ist oder nicht  
*isCrossed()* – liefert boolean, ob dieses Feld angekreuzt ist

###### AbstractField

AbstractField implementiert alle Methoden des Interfaces Field.

###### NumberField

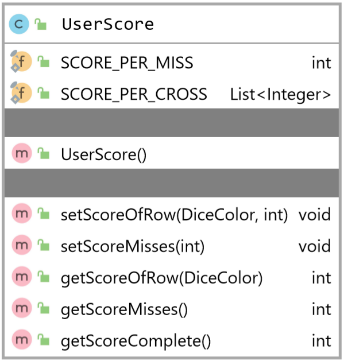
Die Klasse NumberField erbt von AbstractField und stellt ein normales Zahlenfeld dar.

##### RowEndField

Die Klasse RowEndField erbt von AbstractField und stellt das letzte Feld in einer Reihe dar. Es wird bei Abschluss einer Reihe angekreuzt ist somit ein Zusatzfeld. Es hat einen festen Wert, der in der Konstante ROW\_END\_FIELD\_VALUE festgelegt ist.

##### UserScore

Der UserScore repräsentiert den Punktestand eines Spielers (bzw. des Spielfeldes). Es werden die Punkte der einzelnen Reihen und der Fehlwürfe gespeichert.

*setScoreOfRow()* – Setzt für die gegebene Reihenfarbe die entsprechende Punktzahl. Die Punktzahl wird mittels der Liste *SCORE\_PER\_CROSS* ermittelt.  
*setScoreMisses()* – Setzt den Punktestand für die Anzahl der gegebenen Fehlwürfe (Anzahl der gegebenen Fehlwürfe \* *SCORE\_PER\_MISS*).  
*getCompleteScore()* – liefert die Gesamtpunktzahl

*SCORE\_PER\_MISS* – Punktzahl für einen Fehlwurf (Standard: -5 Punkte)

### Das Spiel (Game)

#### RowClosedSupplier

Das Interface RowClosedSupplier beinhaltet zwei Methoden.

*closeRow(DiceColor)* – die angegebene Reihe (Farbe) wird abgeschlossen  
*isRowClosed(DiceColor)* – gibt an, ob die gegebene Reihe abgeschlossen ist

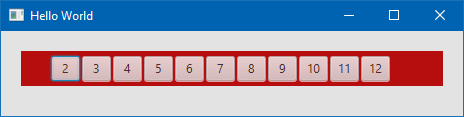
#### Game

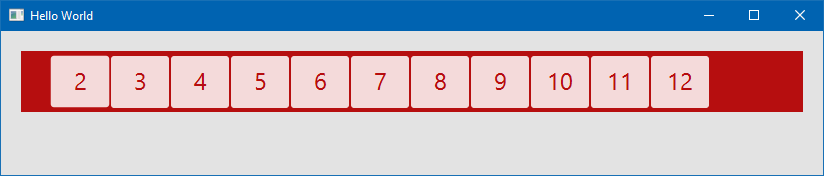
Diese Klasse repräsentiert ein Spiel. Hierfür stellt diese Klasse verschiedene Methoden zur Verfügung. Die wichtigsten werden im Folgenden erläutert:

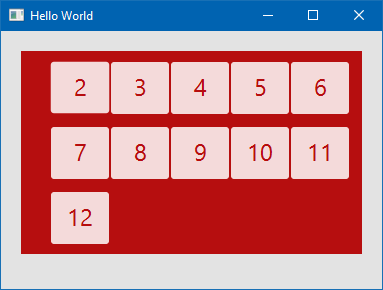
*addPlayer(…)* – Fügt die gegebenen Spieler diesem Spiel hinzu  
*startGame()* – Startet dieses Spiel in einem neuen Thread  
*stopGame()* – Stoppt dieses Spiel (und den dazugehörigen Spiel-Thread)  
*getWinningPlayer()* – Liefert eine Liste mit allen Spielern, die dieses Spiel gewonnen haben (mehrere bei Punktegleichstand)

Die Klasse Game implementiert zudem das Interface *RowClosedSupplier*.

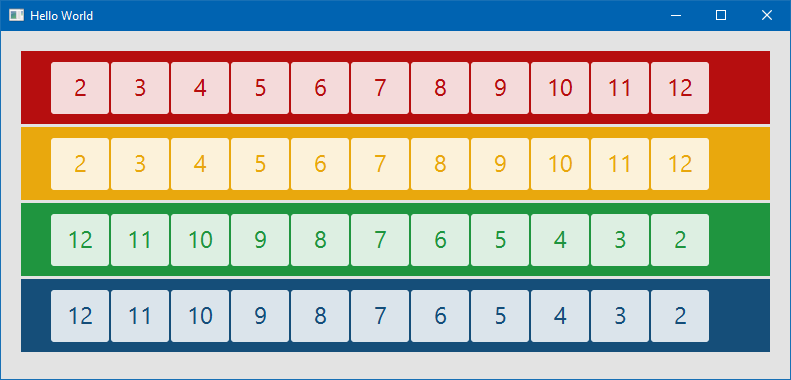
# Das User Interface (UI)

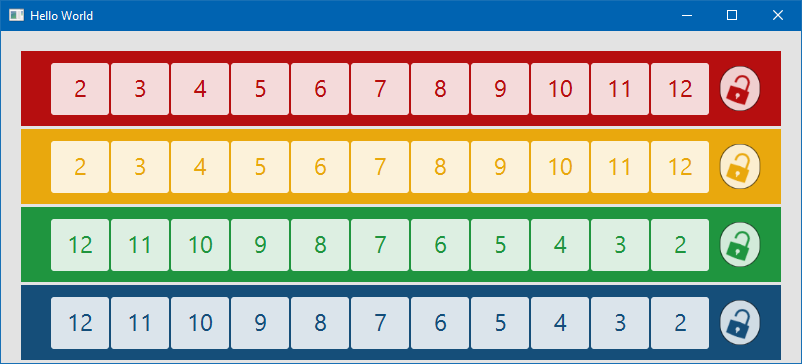


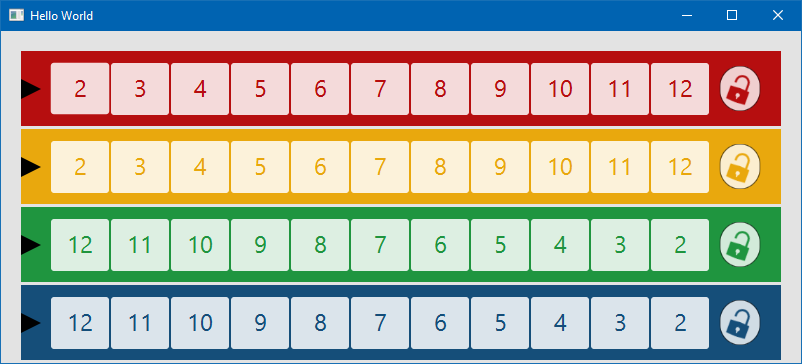


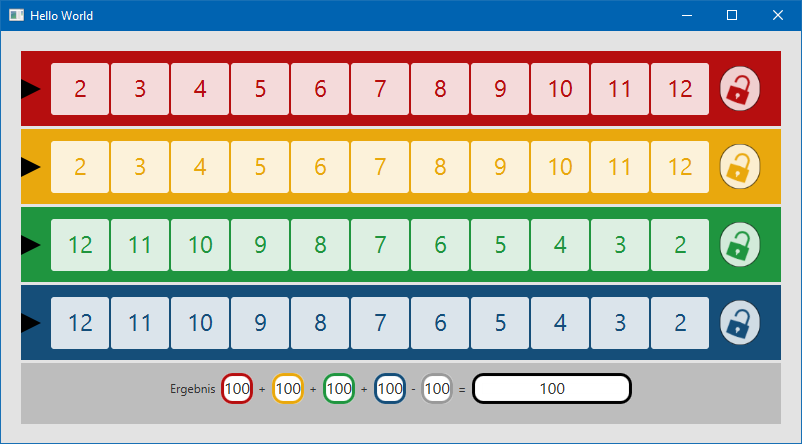


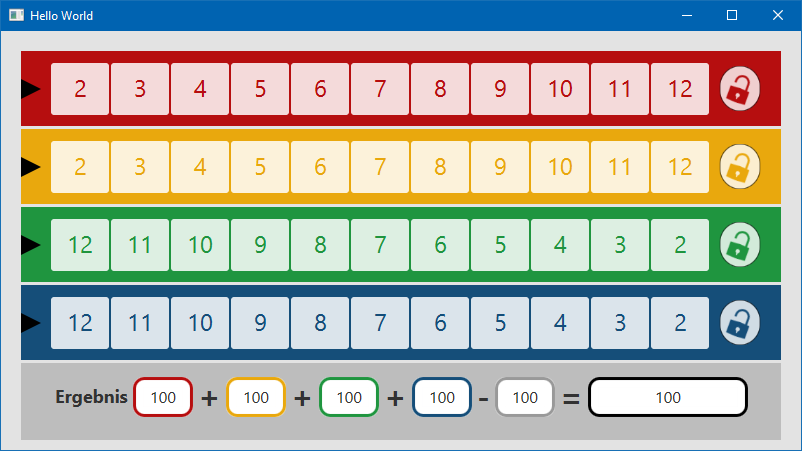


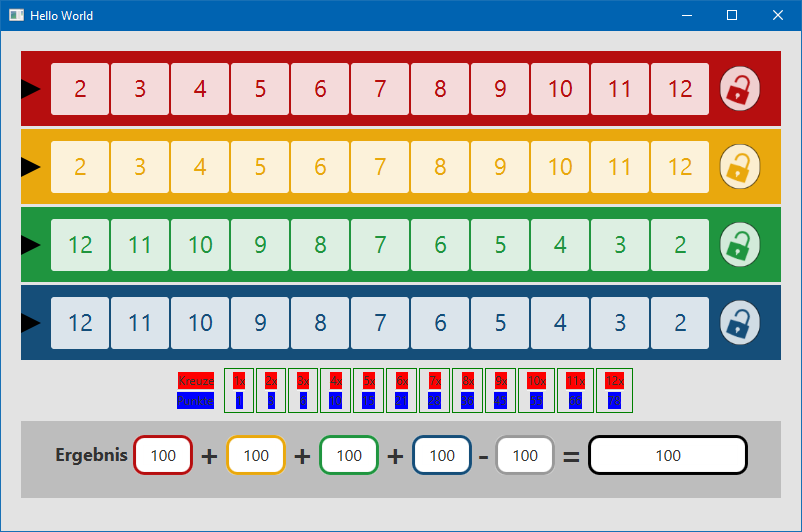




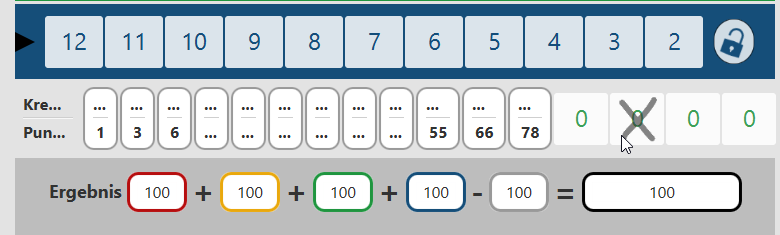


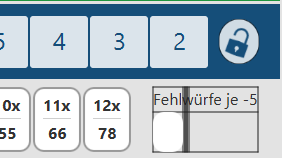
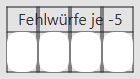
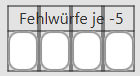
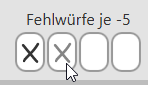


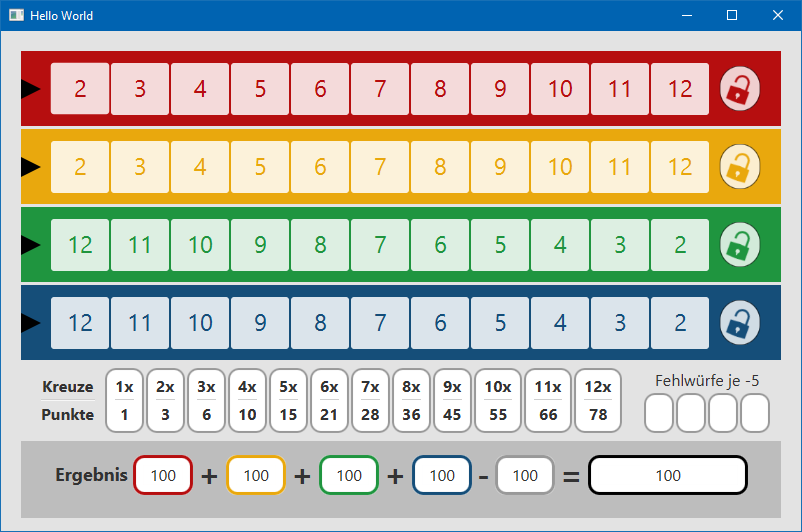










## Zusammenfassung

## Erweiterung

Statistiken einbauen mit Datenbankanbindung

