Design

Projektmanagement

Benjamin Hohl	BH
Florian Bosshard	FB
Nadri Mamuti	NM
Sebastian Sprenger	SS

Bis und mit Meilenstein Projektskizze

Mitarbeiter	Bezeichnung	Soll [h]	Ist [h]
Gruppe	Projektskizze	8	8
Gruppe	Projektskizze Präsentation	3	4
NM	UML für Eclipse einrichten	1	2
SS	Github für Eclipse einrichten	2	2
FB & BH	Eclipse Projekt aufsetzen	1	1
SS	Java Framework Slik anschauen	2	1
Total		17	18

Bis und mit Meilenstein Elaboration

Mitarbeiter	Bezeichnung	Soll [h]	Ist [h]
NM	Projektmanagement erstellen	1	1
NM	Anwendungsfälle schreiben	4	3
SS	eine erste Architektur	2	2
BH	zusätzliche Spezifikationen	2	2
NM	Anwendugsfalldiagramm	1	1
FB	System-Sequenzdiagramm	2	2
FB	Systemoperationen	0.5	1
SS	Domänenmodell	4	3
FB	Glossar	1	1
SS	Software Codierung	4	4
Total		21.5	20

Bis und mit Meilenstein Design

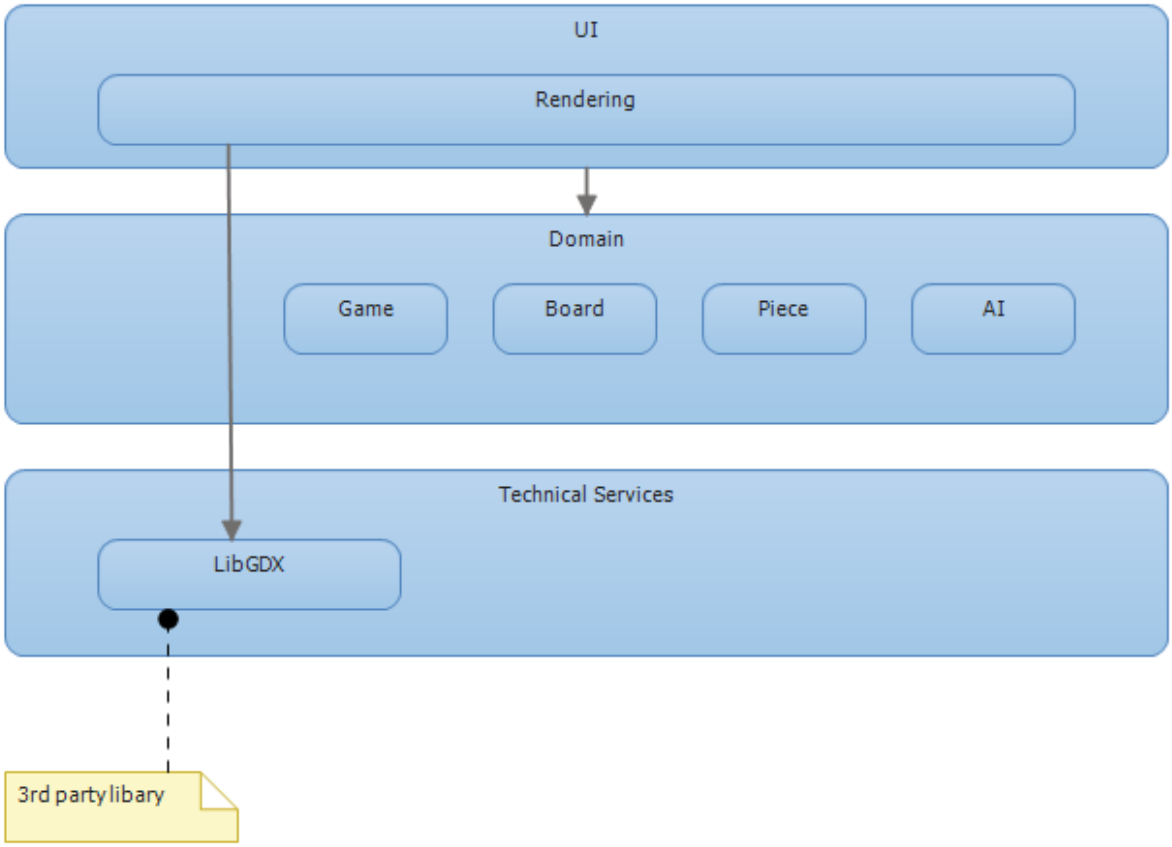
Mitarbeiter	Bezeichnung	Soll [h]	Ist [h]
NM	Projektmanagement erstellen	1	1
NM	definitive Architektur	2	1.5
FB	Design-Klassendiagramm	1	1.5
SS / FB	Klassenverantwortlichkeiten	2	2
FB	Zusammenarbeitsdiagramme	2	2
SS	GUI-Design	3	4
FB	Glossar	1	0.5
alle	Software Codierung	10	10
Total		22	22.5
Gesamt Total		60.5	60.5

Schätzung Meilenstein Schlusspräsentation

Mitarbeiter	Bezeichnung	Soll [h]	Ist [h]
NM	Projektmanagement erstellen	1	1
NM	Klassendiagramm	1	
SS	Code	15	
BH	Testbericht	3	
FB	Bedienungsanleitung	5	
SS	Zusammenfassung	5	
Total		30	1

Architektur

Wir haben uns entschieden necaREx mit Java zu entwickeln. Dies hat den Vorteil, dass wir mit sehr wenig Aufwand das Programm Plattformunabhängig entwickeln und anbieten können. Zusätzlich werden wir für das Frontend die Library „libgdx“ verwenden, diese bietet einfach zu verwendende Schnittstellen um 2d Anwendungen auf der Grafikkarte auszuführen. Ausserdem wird sie als zusätzlichen Abstraktionslayer im UI verwendet, anhand welchem wir, in einem nächsten Release, die Anwendung ebenfalls als iOS und Android App anbieten können.
Um eine möglichst flexible und erweiterbare Lösung aufzubauen haben wir uns für folgendes Layering entschieden:



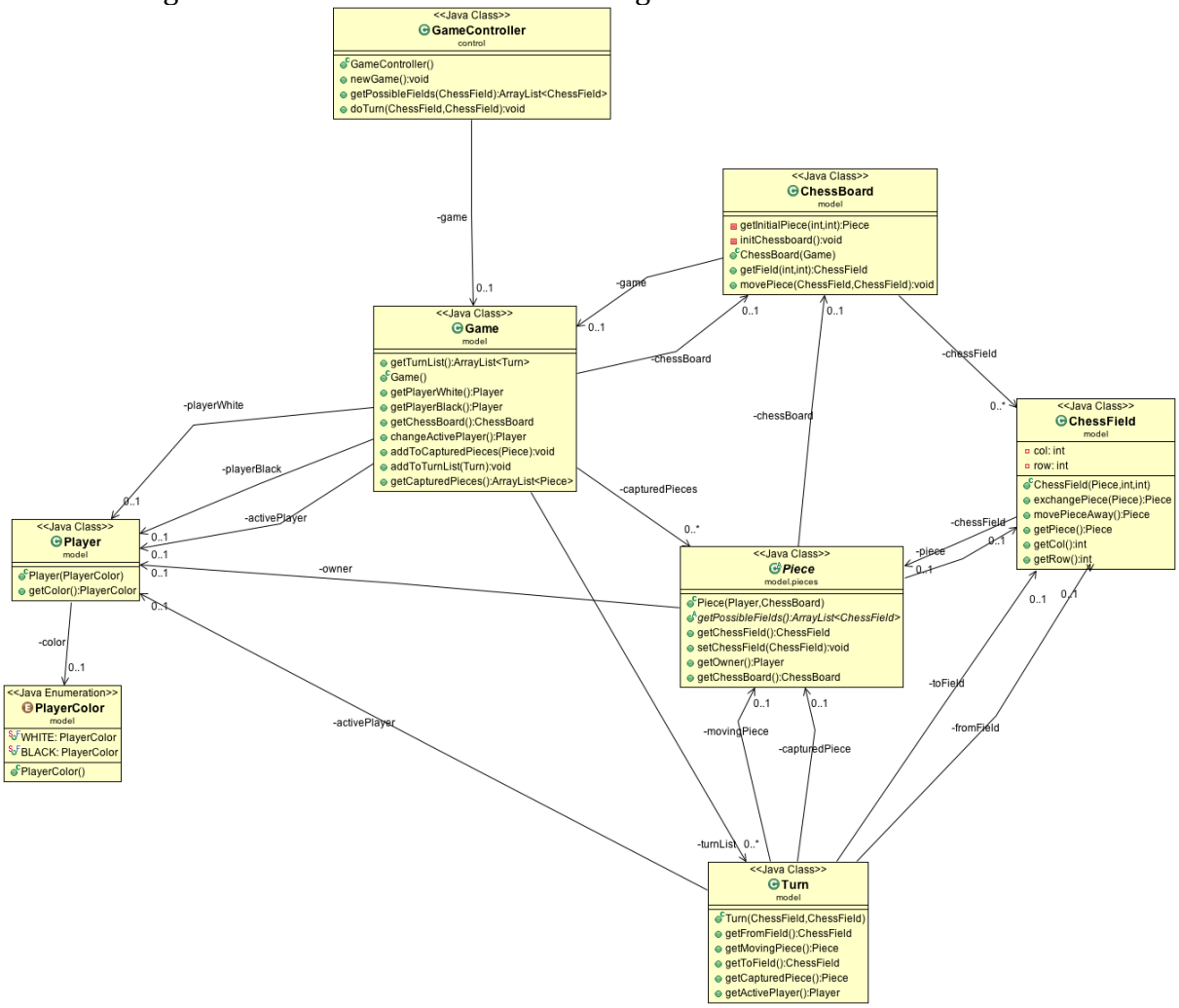
Erläuterungen zu den von uns entwickelten Packages:

- Rendering
  Hier wird das 2d Userinterface gerendert. Mit Hilfe der LibGDX Funktionalitäten werden hier die nötigen Informationen aus den Models gelesen und dargestellt.
- Game
  Das Game verwaltet die Spieler, Züge und getöteten Figuren.
- Board
  Hier geschieht die Verwaltung des Schachbretts. Es wird Logik zur Verfügung gestellt, um Figuren zu bewegen, Figuren zu töten und alle Figuren auf die Initialposition zu setzen.
- Piece
  Die einzelnen Figuren stellen Logik zur Verfügung um ihre möglichen Bewegungen zu berechnen.
- AI
  Logik für die künstliche Intelligenz.

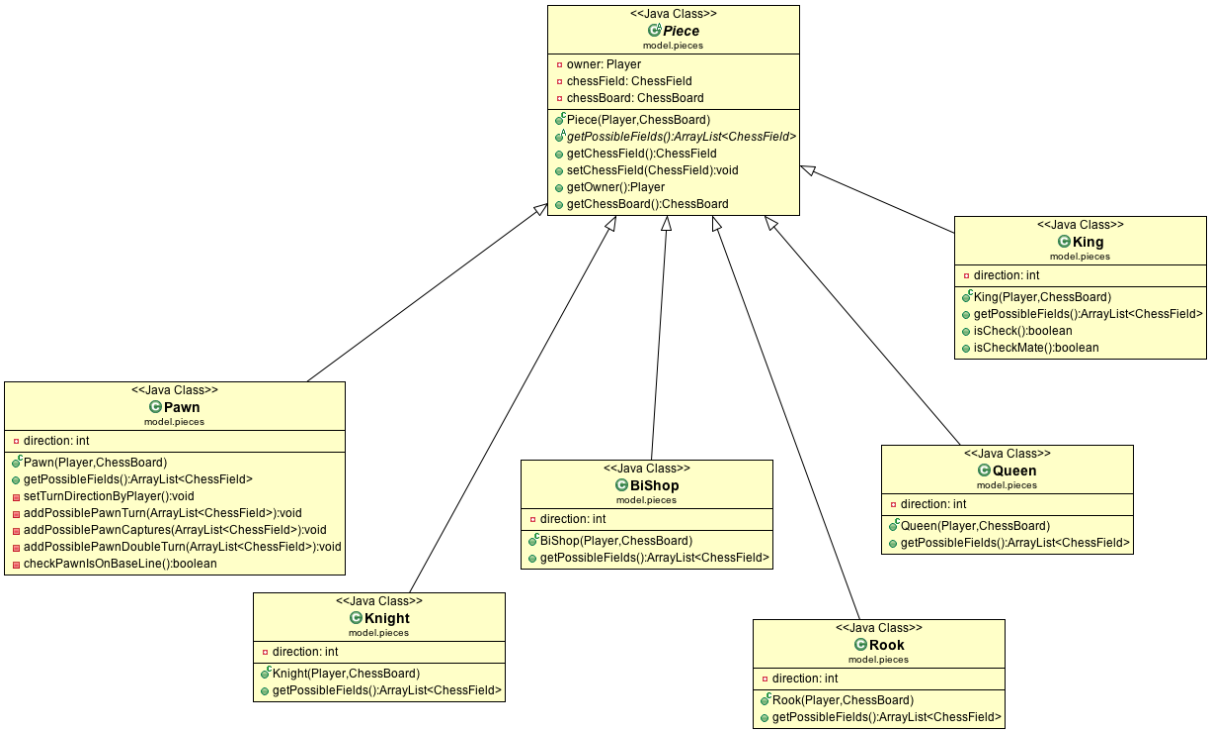
Design-Klassendiagramme

Diagramm Controller und Model

Das folgende Diagramm zeigt den GameController und das angehängte Model. Auf die einzelnen Figuren wurde aus Übersichtlichkeitsgründen bewusst verzichtet.

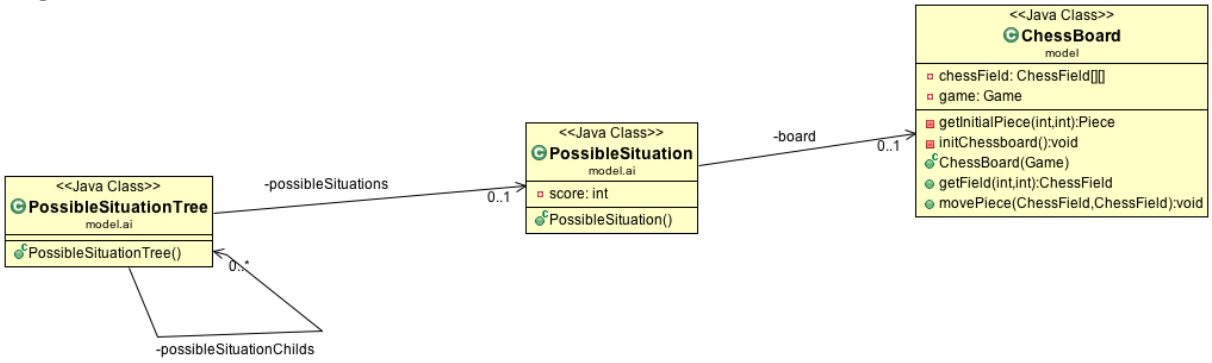


Schachfiguren



Artificial Intelligence

Die künstliche Intelligenz benötigt eine eigene Datenstruktur. Diese ist zur Zeit wie folgt angedacht.



Klassenverantwortlichkeiten

Im folgenden Abschnitt werden die Verantwortlichkeiten und Aufgaben der einzelnen Klassen kurz beschrieben.

Package client.UI

Klassenname	Verantwortlichkeiten / Aufgaben
NecarexDesktop	Einstiegsklasse für Desktop Anwendungen.
NecarexGame	LibGDX Main Klasse, übernimmt das Draw/Update.

Package client.UI.drawing

Klassenname	Verantwortlichkeiten / Aufgaben
BoardDrawer	Der BoardDrawer übernimmt alle Zeichnungsaufgaben bezogen auf das Schachbrett.
PieceDrawer	Der PieceDrawer übernimmt alle Zeichnungsaufgaben bezogen auf die Schachfiguren.

Package client.viewmodel

Klassenname	Verantwortlichkeiten / Aufgaben
ChessBoardViewModel	Model mit View-spezifischen Daten für das Schachbrett wie zum Beispiel das zuletzt vom Benutzer angeklickte Feld.

Package control

Klassenname	Verantwortlichkeiten / Aufgaben
GameController	Kontrolliert den Ablauf des Schachspiels

Package model

Klassenname	Verantwortlichkeiten / Aufgaben
ChessBoard	Das ChessBoard kontrolliert die Schachfelder und macht die Anfangsaufstellung zu Beginn des Spiels.
ChessField	Auf einem Schachfeld (ChessField), das auf einer Spalte und einer Zeile steht, kann eine Figur stehen.
ComputerPlayer	Stellt einen Computerspieler dar. Benutzt für die Zugberechnung das Package model.ai
Game	Ein Spiel (Game) besteht aus weissem und schwarzem Spieler, sowie einer Referenz des Schachbretts. Die Liste der vergangenen Züge und die bisher geschlagenen Figuren werden hier geführt.
Player	Ein Schachspieler (Player) besitzt eine Farbe.
PlayerColor	Enumeration für die beiden Figurenfarben im Schach: Weiss und Schwarz
Turn	Ein Zug (Turn) im Schach besteht aus einer Figur, die von einem Feld zu einem anderen zieht und dort allenfalls eine andere Figur schlägt. Die Attribute werden über den

	Konstruktor gesetzt, danach sind keine Änderungen mehr möglich, da ein gemachter Spielzug nicht geändert werden kann.

Package model.ai

Klassenname	Verantwortlichkeiten / Aufgaben
PossibleSituation	Stellt eine mögliche Schachbrettstellung dar. Diese Klasse nimmt die Bewertung der Schachsituation vor.
PossibleSituationTree	Stellt eine Baumstruktur für die möglichen Schachbrettstellungen (PossibleSituation) in der Zukunft dar und führt die Rekursion durch.

Package model.pieces

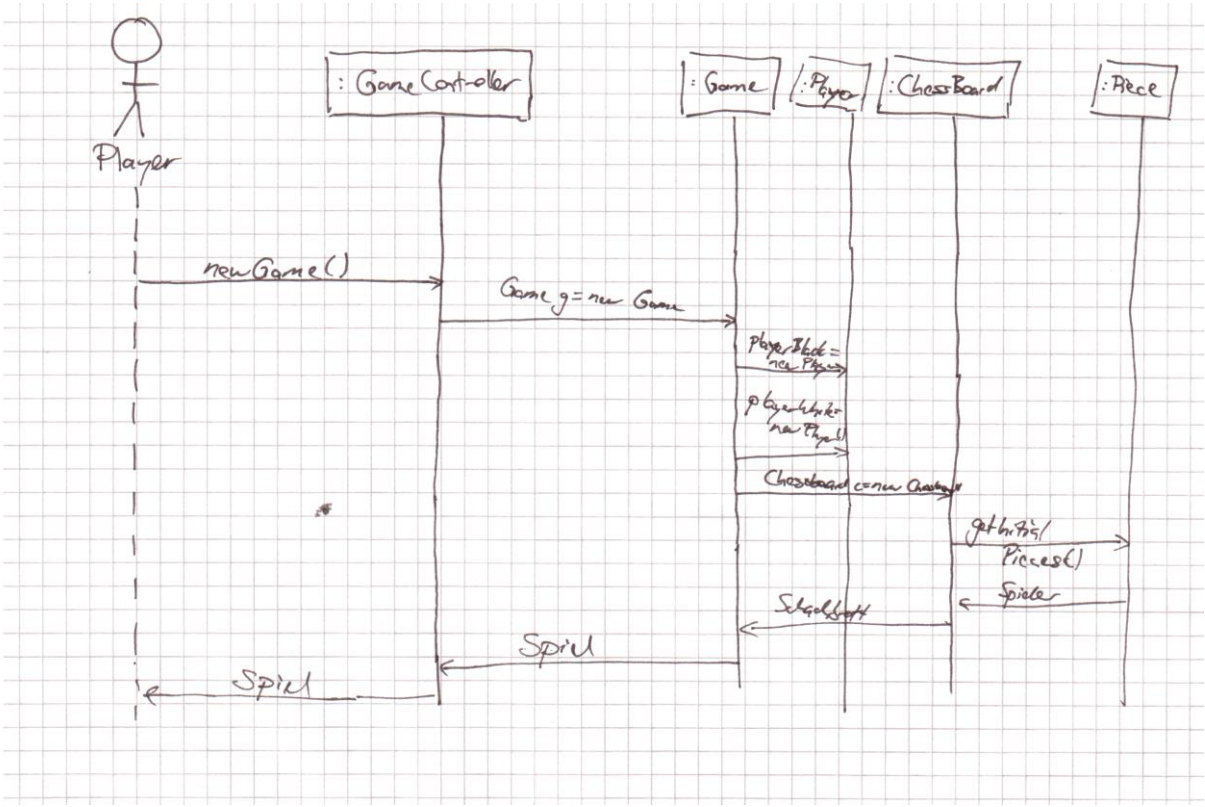
Klassenname	Verantwortlichkeiten / Aufgaben
Piece	Die Klasse Piece (abstrakt) beschreibt alles, was alle Schachfiguren gemeinsam haben. Alle Schachfiguren gehören einem Owner, stehen auf einem Schachfeld.Jede Schachfigur muss eine Methode implementieren, die die möglichen Felder zurück gibt.
Pawn	Spielfigur Bauer, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.
Knight	Spielfigur Springer, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.
Bishop	Spielfigur Läufer, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.
Rook	Spielfigur Turm, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.
Queen	Spielfigur Dame, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.
King	Spielfigur König, basiert auf Piece. Kennt die Gangart des und kann dessen mögliche Felder ausrechnen.

Test

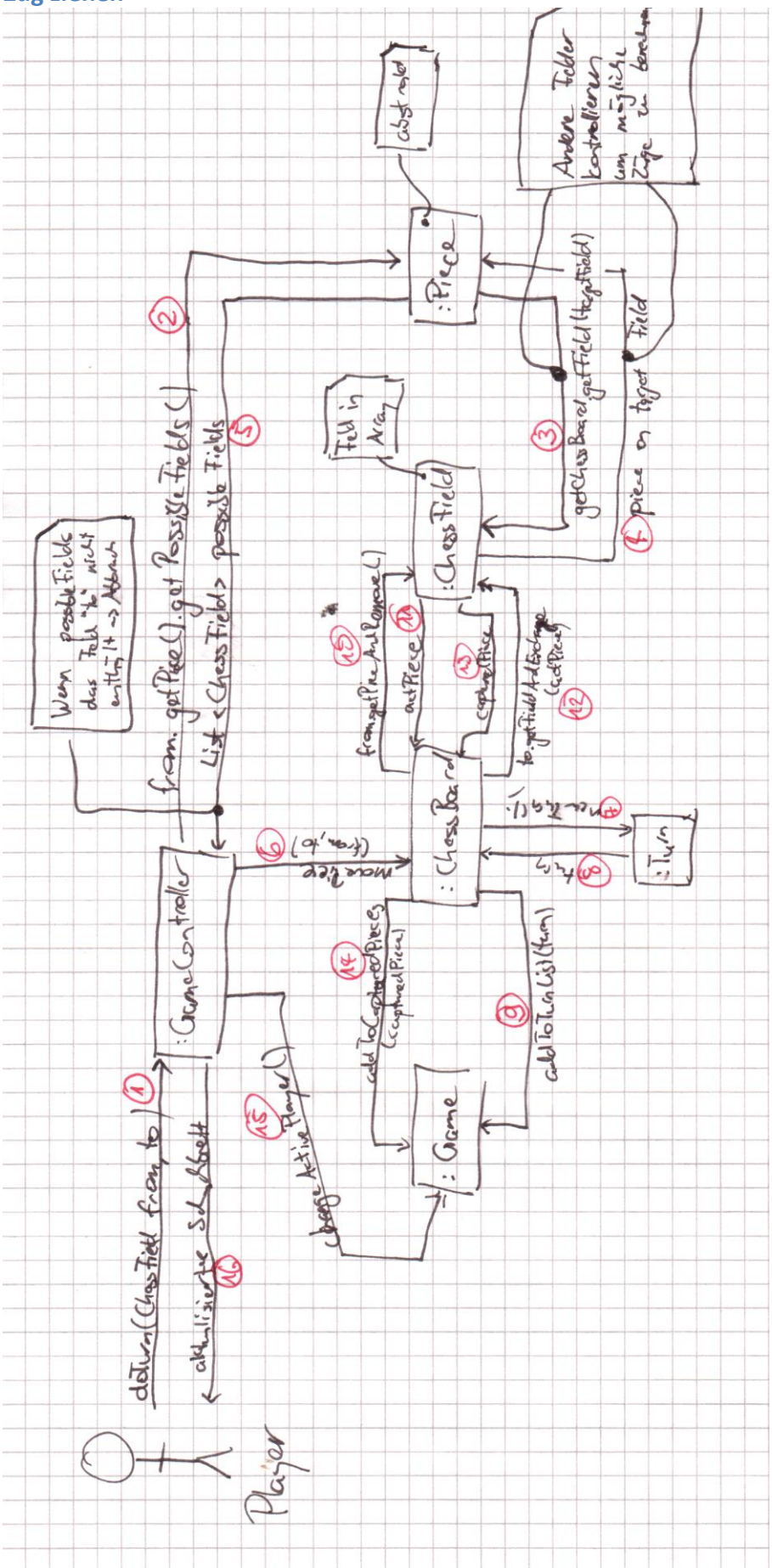
Die für den Test verwendeten Klassen werden hier nicht genauer spezifiziert.

Zusammenarbeitsdiagramme

Neues Spiel starten



Zug ziehen


Player
: GameController
: ChessBoard
: Game
: Piece
: ChessField
: Turn
Andere Felder kontrollieren um mögliche Züge zu berechnen
Wenn possible Fields das Feld "it" nicht enthält -> Abbruch
List <ChessField> possible Fields
Feld in Array

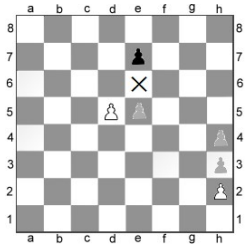
Glossar

Im folgenden Glossar sollen die Schachfiguren, mit ihren Zugsarten und die relevanten Schachregeln kurz erläutert werden. Für detailierte Informationen zum Schachspiel gibt es einiges an Fachliteratur.

Schachfiguren

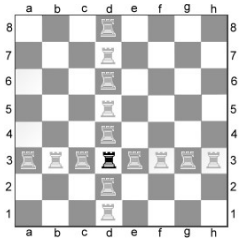
Bauer (1)

Die schwächste Figur im Schach und hat folgendes Zugverhalten:



- Ein Feld gerade vorwärts (in Richtung der gegnerischen Spielfeldseite).
- Schlagen einer Figur diagonal nach vorne (in beide Richtungen, wenn eine gegnerische Figur dort steht, die geschlagen werden kann und somit der Bauer nicht am Spielfeldrand steht)
- Steht der Bauer in der Grundstellung (Weiss A2 – H2; Schwarz A7 – H7) darf ein oder zwei Felder gerade vorwärts gemacht werden.
- **Schlagen en passant**: Ein Bauer, der ein Feld angreift, das von einem gegnerischen Bauern überschritten worden ist, der von seinem Ursprungsfeld aus in einem Zug um zwei Felder vorgerückt ist, darf diesen gegnerischen Bauern so schlagen, als ob letzterer nur um ein Feld vorgerückt wäre. Dieses Schlagen darf nur in dem Zug geschehen, der auf ein solches Vorrücken folgt.
- Erreicht ein Bauer den gegnerischen Spielfeldraum darf er vom Spieler in einer Figur seiner Wahl umgetauscht werden.

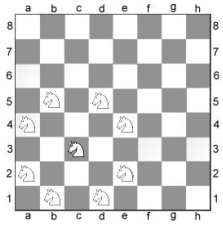
Turm (2)



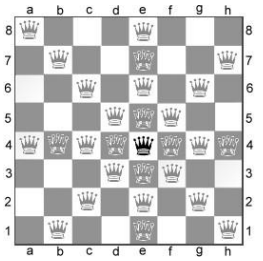
Figur, die sich auf den Linien und Reihen des Schachbretts bewegt.

Springer (3)

Einzige Schachfigur, die andere Figuren überspringen kann. Der Springer springt jeweils 2 Felder in eine Richtung (diagonal / horizontal) und danach ein Feld zur Seite, wobei die Seite wählbar ist. Dies ergibt einen L-förmigen Sprung.



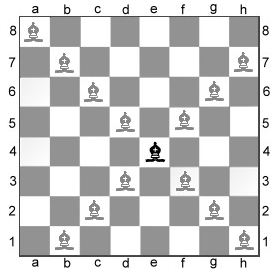
Dame (4)



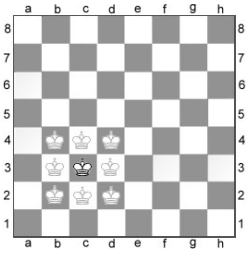
Die Schachfigur Dame darf auf ein beliebiges anderes Feld entlang der Linie, der Reihe oder einer der Diagonalen ziehen, auf welcher sie steht.

Läufer (5)

Schachfigur, die auf ein beliebiges Feld entlang der Diagonale ziehen darf, auf welcher er steht.



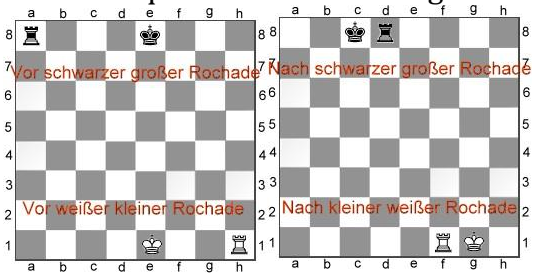
König (6)



Der König kann nur auf ein beliebiges angrenzendes Feld ziehen, das nicht von einer oder mehreren gegnerischen Figuren angegriffen wird.

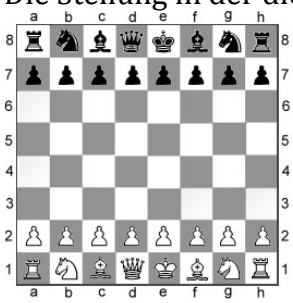
Wird der König in einem gegnerischen Zug angegriffen, so ist der Gegner dazu verpflichtet darauf mit dem Wort „**Schach!**“ hinzuweisen. Der Spieler muss dann die Schachsituation aufheben, beispielsweise durch einen Zug des Königs aus dem Schach.

Ein spezieller Zug des Königs ist die **Rochade**: Jeder Spieler kann einmal eine Rochade ausführen. Der König und der Turm werden bei der Rochade nach speziellem Muster verschoben (siehe Bild). Dies darf nicht gemacht werden, wenn Turm oder König bereits gezogen haben, eine Figur dazwischen steht oder eines der Felder, das der König betritt oder überquert von fremden Figuren bedroht ist.



Grundstellung / Anfangsstellung

Die Stellung in der die Figuren zu Beginn aufgestellt werden:



Spielablauf

Beim Spiel wird jeweils abgewechselt. Der Spieler mit den Figuren der Farbe weiss beginnt. Es besteht immer Zugpflicht.

Spielenden

Das Schachspiel hat verschiedene mögliche Spielenden:

Matt / Schachmatt

Der mattsetzende Spieler ist der Sieger des Spiels. Das Matt tritt ein, wenn der König bedroht ist und nicht mehr in Sicherheit gebracht werden kann, d.h. nicht mehr auf ein nicht bedrohtes Feld fahren oder die Bedrohung abwenden kann.

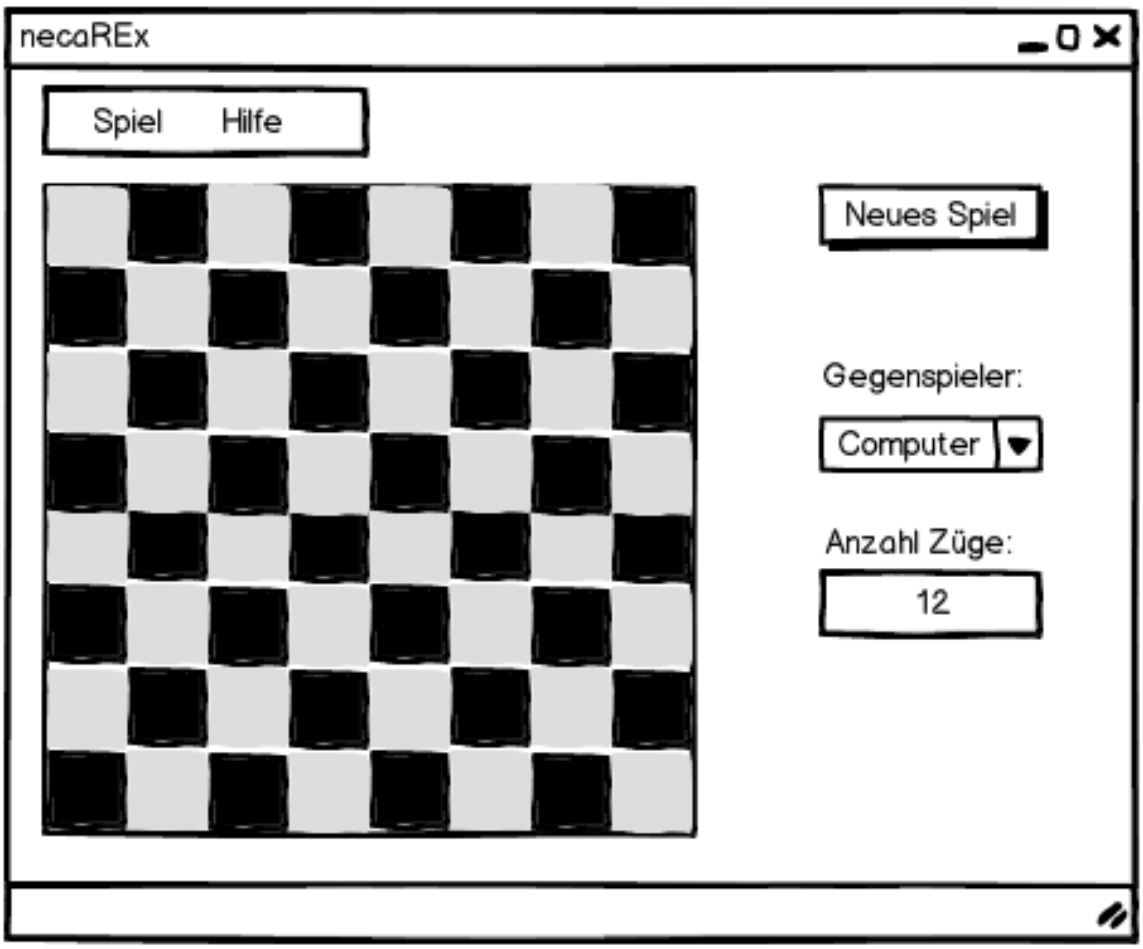
Remis

Remis bedeutet „Unentschieden“ und tritt in verschiedenen Fällen ein:

- Gegenseitige Einigung der beiden Spielenden
- **Patt** Der am Zug befindliche Spieler hat keine Möglichkeit einen Zug nach den Schachregeln zu machen, wobei sein König nicht im Schach steht, also nicht direkt bedroht wird.
- Nur noch 2 Könige sind auf dem Spielfeld. Kein Matt kann entstehen, dass die Könige sich gegenseitig nicht bedrohen können, da sie selbst nicht auf ein bedrohtes Feld (des gegnerischen König) fahren dürfen
- 3-mal die gleiche Stellung wird erreicht bei der gleichen Partei am Zug.
- 50 Züge ohne Bewegung eines Bauers und ohne schlagen einer Figur. Man geht davon aus, dass die Partei nicht vorwärts geht, da weder eine Figur geschlagen wurde, noch ein Bauer unterwegs ist zu einer Bauernumwandlung

GUI-Design

Nach dem das Spiel gestartet wird erscheint folgendes Fenster, welches die gesamte Benutzeroberfläche ausmacht.



Als Gegenspieler ist standardmässig der Computer aktiviert, durch anwählen des Dropdown Menu kann man zwischen Computer und Gegenspieler wählen. Die Anzahl gespielter Züge wird angezeigt. Mit dem Button Neues Spiel kann das Spielfeld jederzeit zurückgesetzt werden um ein neues Spiel zu spielen.