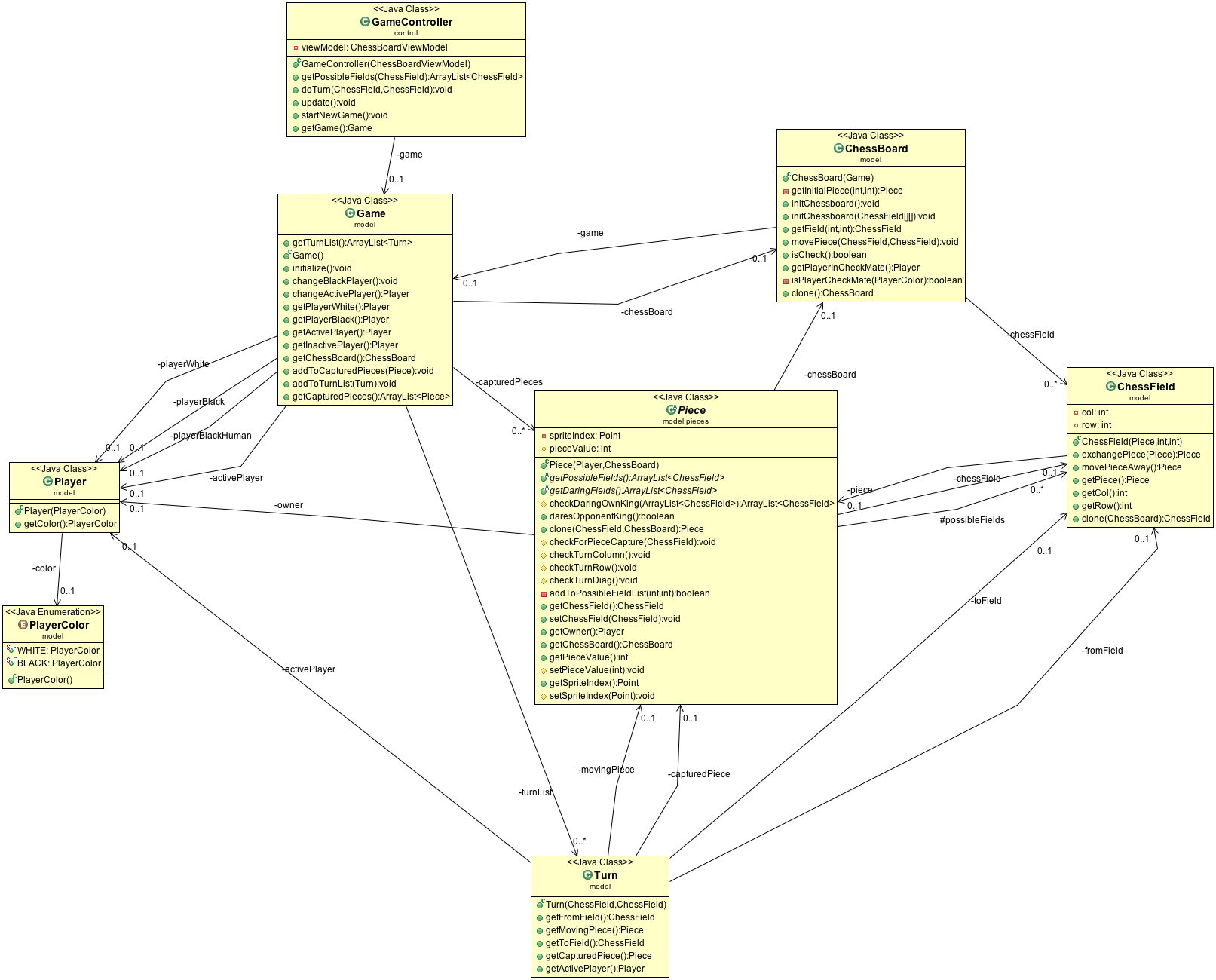
Schlussbericht

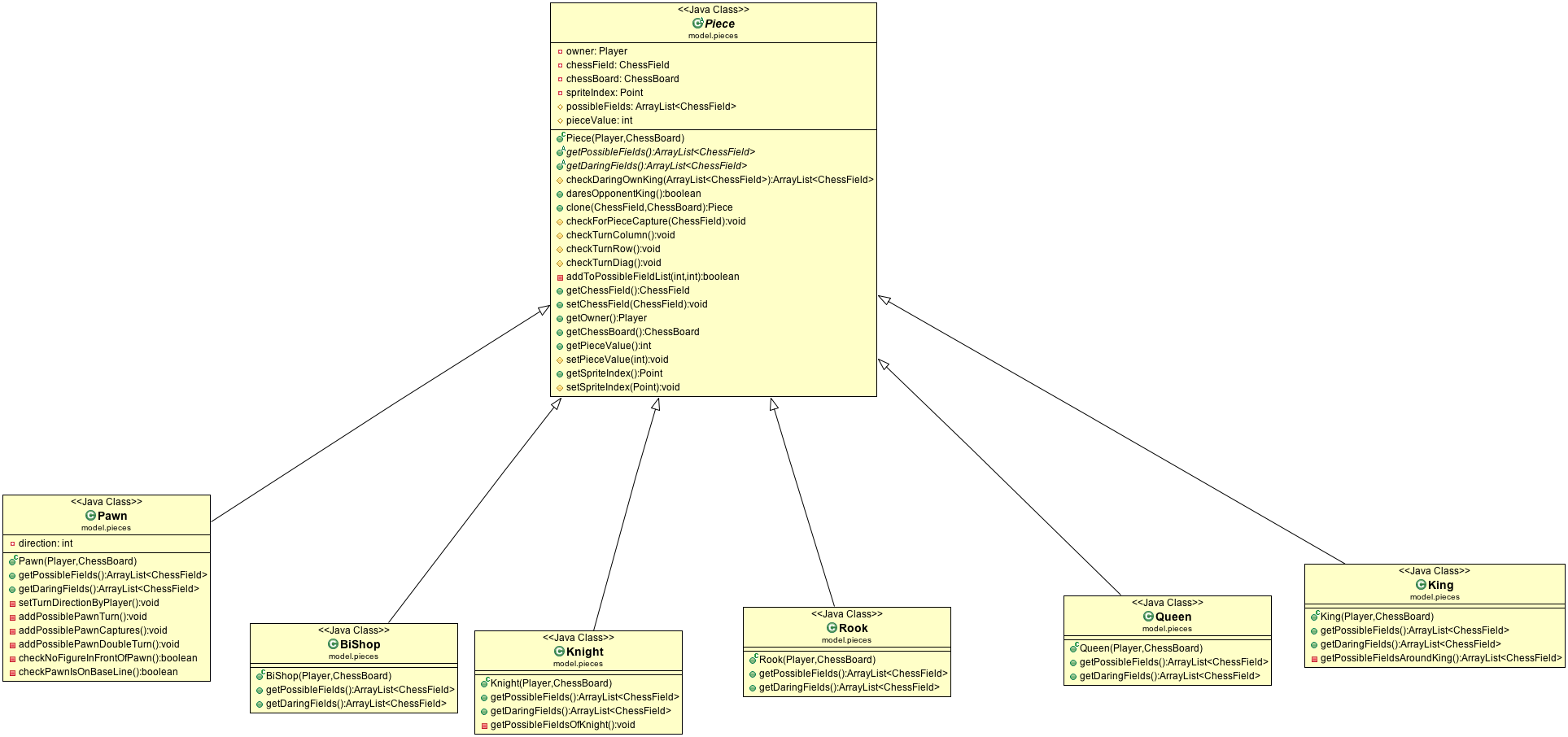
## Projektmanagement

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
|  |  |  |  |
|  |  |  |  |
| Bis und mit Meilenstein Schlusspräsentation | | |  |
| Mitarbeiter | Bezeichnung | Soll [h] | Ist [h] |
| NM | Projektmanagement erstellen | 1 | 1 |
| FB | Klassendiagramm | 1 | 1 |
| SS FB | Code | 15 | 17 |
| BH | Testbericht | 3 | 3 |
| NM | Bedienungsanleitung | 5 | 3 |
| NM | Zusammenfassung | 5 | 3 |
| BH | Abschlusspräsentation | 3 | 3 |
|  |  |  |  |
| Total |  | 33 | 31 |
|  |  |  |  |
|  |  |  |  |
| Gesamt Total |  | 93.5 | 91.5 |

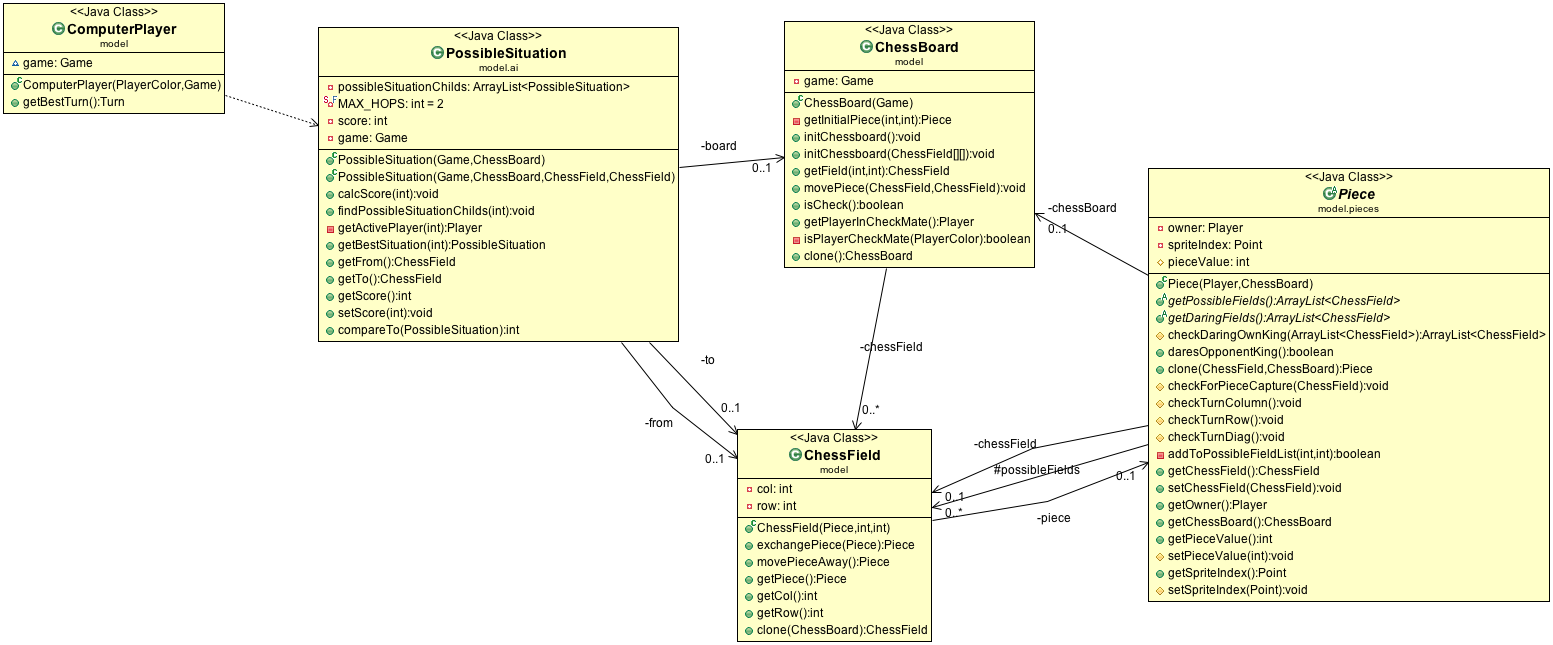
## Klassendiagramm



## Figuren (Pieces)



## Künstliche Intelligenz



## Code

Der Code befindet sich auf der CD:

* Source: Eclipse Projekt ch.zhaw.necarex. Enthält den gesamten Source-Code
* necaREx.jar : Ausführbare Dateien
* JavaDoc liegt unterhalb des Eclipse Projekts. Verweis auf oberster Ebene.

## Testbericht

## Bedienungsanleitung

# Installation

#### Installation des JRE

Um „necaREx“ auf Ihrem Computer zu installieren, benötigen Sie das **Java Runtime Environment (JRE)**.  
Die Installationsdateien können Sie unter folgendem Link herunterladen: <http://java.com/de/download/index.jsp>

Weitere Informationen zur Installation von JRE finden Sie unter folgendem Link:

<http://java.com/de/download/help/index_installing.xml>

#### Installation der „necaREx“ Anwendung

Kopieren Sie die auf der Installations-CD enthaltene Datei necaREx.jar in ein Verzeichnis Ihrer Wahl. Führen Sie Datei aus, um das Spiel zu starten.

# Spielregeln

Die Spielregeln von necaREx sind die des herkömmlichen Schachspiels.

# Beschreibung

„necaREx“ ist ein Schachspiel welches in Schachvereinen und Schachkursen als effiziente Lernsoftware eingesetzt werden kann. Durch das schlichte und intuitive Design ist „necaREx“ einfach und schnell zu erlernen. Dank der übersichtlichen Benutzeroberfläche kann der Spieler zu jeder Zeit wichtige Informationen dem Spiel entnehmen.

Das Schachspiel„necaREx“ ist weitgehend selbst erklärbar und funktioniert auf den herkömmlichen Regeln des Schachs, weshalb sich diese Bedienungsanleitung nur auf die „necaREx“ spezifischen Eigenschaften bezieht und deshalb kurz ausgefallen ist um in einfacher und intuitiver Weise die Bedienung zu erleichtern.

Beim Start des Programms wird das Schachfenster, welches zugleich auch das Hauptfenster ist, gestartet.

### Das Schachbrett

Im Zentrum des „necaREx“ Schachprogramms ist das Schachbrett mit den Figuren abgebildet. Hier werden die Figuren auf dem Spielbrett dargestellt. Ihre Figuren bewegen Sie, indem Sie auf das Start- und Zielfeld klicken. Nachdem auf das Startfeld einer eigenen Figur geklickt wurde, werden die Felder, die diese Figur erreichen kann, in Graustufen markiert und das Feld der ausgewählten Figur wird rot eingefärbt. In der Horizontalebene sind die Felder von links nach rechts mit A, B, C, D, E, F, G und H und in der Vertikalebene von unten nach oben mit 1, 2, 3, 4, 5, 6, 7 und 8 beschriftet, damit man Züge benennen kann. Die Spielfiguren des Spielers mit der Bezeichnung „Weiss“ sind gräulich gefärbt, die des Spielers mit der Bezeichnung „Schwarz“ sind bräunlich gefärbt. Dies aus dem Grund um das ganze Spiel auch für die Augen etwas angenehmer zu gestaltet.

### Einstellungen und Spielinformationen

Auf der rechten Seite des „necaREx“ Schachprogramms sind die Einstellungen und Spielinformationen abgebildet.   
  
Neues Spiel  
Zuoberst auf der rechten Seite befindet sich der Button „Neues Spiel“, womit jederzeit das Spiel neu gestartet werden kann, hierbei wird der aktuelle Spielstand nicht gespeichert und die Aktion kann nicht wieder rückgängig gemacht werden.  
  
Gegenspieler  
Weiter unten, unter dem Einstellungspunkt „Gegenspieler“ sieht man gegen welche Art von Gegenspieler man spielt. Als Gegenspieler kann man zwischen „Mensch“ oder „Computer“ wählen, der aktuelle Gegenspieler wird mit einem grauen Hintergrund hervorgehoben. Hierbei gilt es zu beachten, dass ein allfälliger Gegenspielerwechsel stets vor dem eigenen Zug getätigt werden muss, da sich sonst der Gegenspieler immer noch im alten Zustand befindet. Beispielsweise ist als Gegenspieler „Mensch“ ausgewählt, man möchte jedoch gegen den Computer spielen, nun muss man bevor man seine eigene Figur bewegt, als Gegenspieler „Computer“ wählen und dann anschliessend den Zug tätigen.

Spieler am Zug  
Auf mittlerer Höhe der rechten Seite befindet sich der Informationpunkt „Am Zug“, hier wird angezeigt welcher Spieler am Zug ist, somit sieht man jederzeit welcher Spieler am Zug ist und kann so Missverständnisse vermeiden. Nochmal zur Wiederholung, „Weiss“ heisst der Spieler, welcher die weissgrauen Figuren besitzt und dessen Figuren sich zu Spielbeginn auf den Zeilen 1 und 2 befinden, dementsprechend ist „Schwarz“ der Spieler, welcher die braunen Figuren besitzt und dessen Figuren sich zu Spielbeginn auf den Zeilen 7 und 8 befinden.  
  
Anzahl Züge  
Unter dem Informationspunkt „Anzahl Zuege“ sieht man wie viele Züge in dieser Partie getätigt worden sind. Diese Zahl dient rein zur Visualisierung und hat keinen Einfluss auf das Spiel.  
  
Gewonnen  
Unter dem letzten Informationspunkt, welcher sich auf der rechten Seite zuunterst befindet, wird der Gewinner der Partie angezeigt, „Schwarz“ bzw. „Weiss“, solange die Partie noch läuft und noch kein Gewinner fest steht wird ein „-“ angezeigt. Die Partie gewonnen hat der Spieler, der den König des Gegners schlägt.

## Zusammenfassung

### Funktionalität

Das Schachspiel „necaREx“ soll in Schachvereinen und Schachkursen als effiziente Lernsoftware eingesetzt werden. Dies ist durchaus denkbar, da „necaREx“, mit Ausnahme von ein Paar speziellen Funkionen, mit den herkömmlichen Schachregelen zu Spielen ist.  
  
Durch das schlichte und intuitive Design ist „necaREx“ einfach und schnell zu erlernen. Dies war einer der wichtigsten Aspekte, welchen wir erreichen wollten, denn damit das Spiel Spass macht und man etwas dabei lernen kann darf es nicht zu komplex aufgebaut sein. Wir denken diese Ziel erreicht zu haben.  
  
Dank unserer übersichtlichen Benutzeroberfläche kann der Spieler zu jeder Zeit wichtige Informationen dem Spiel entnehmen Dies ist vor allem für Übersicht und für den Lernerfolg entscheident.

### Fehler

Fehler als solche haben wir nicht gefunden, jedoch haben wir eine weile gebraucht, bis wir uns entscheiden konnten, wie wir das Problem mit der Umstellung zwischen „Mensch“ und „Computer“ lösen wollten, es gab verschiedene Ideen, so wollten wir zuerst bei einem Zustandswechsel des Gegenspielers gleich noch das Spiel neu starten, um Probleme zu vermeiden. Zum Schluss konnten wir eine Lösung finden, dass man den Gegenspieler „on the fly“ wechseln kann, wobei man darauf achten muss wann man den Gegenspieler wechselt.

### Funktionen

Es konnten glücklicherweise alle vorgesehenen Funktionen implementiert werden, jedoch konnte nicht alle möglichen Erweiterungen realisiert werden.

Erweiterungen / Ideen di nicht realisiert werden konnten

* Erreicht ein Bauer das obere Spielfeldende, so wählt der Spieler die eizusetzende Figur aus.
* Rochade
* Schlagen „en passant“
* Verschiedene Computer-Schwierigkeitsstufen
* Spiel zweier Menschen über das Netzwerk

Das diese Erweiterungen und Ideen nicht realisiert werden konnte ist zwar schade, jedoch nicht weiter schlimm, da sie nicht zu den zwingenden Anforderungen definiert worden waren.

### Rückblick auf SEPS 2012 Gruppe 2

Für jedes Mitglied der Gruppe war es eine sehr lehrreiche und interessante Erfahrung, welche wir in dieser Gruppenarbeit und Konstellation erleben durften. Schlussendlich konnte jedes Teammitglied in allen Teilbereichen, vom Projektablauf bis zur Programmierung etwas zur Gruppenarbeit beitragen.

Obwohl die einzelnen Gruppenmitglieder vor dieser Teamarbeit nicht viel miteinander zu tun hatten, hat die Arbeit im Team äusserst gut funktioniert. Man hat sich gegenseitig abgesprochen und respektiert und darauf geachtet, dass kein Mitglied zu kurz kommt oder zu viel machen musste.

Spezielle positive Erfahrungen gab es in der Gruppe durch den Informationsaustausch in der Programmierung und im Projektmanagement.

An negative Erfahrungen könne wir uns nicht erinnern.