Elaboration

## Projektmanagement

#### Inception

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

#### 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 |

## Andwendugsfälle

### Fully dressed Anwendugsfall

#### „Zug ausführen“

**Hauptakteur:**

Spieler / Computer

**Stakeholders und ihre Interessen:**

* necaREx-Benuter: Möchte eine intuitive Benutzeroberfläche. Möchte seinen Zug nach den geltenden Schachregeln tätigen können. Benutzer welche mit den Schachregeln nicht sehr vertraut sind möchten, dass unzulässige Züge als solche angezeigt werden.
* Kunde: Möchte, dass seine Schüler mit necaREx ihre Schachfähigkeiten verbessern können. Möchte, dass die Schüler auf einfache Art Züge tätigen können.

**Vorbedingungen:**

Der Benutzer muss eine Figur ausgewählt haben und diese verschieben.

**Nachbedingungen:**

Der Benutzer hat den Zug getätigt und Figur verschoben.

**Haupterfolgsszenario:**

|  |  |
| --- | --- |
| Akteur:   1. Er wählt eine Figur aus   4. Er verschiebt die Figur | System:   1. Prüft welche Züge zulässig sind. 2. Stösst Regel-Engine an   5. Abfrage: Wird gegnerische Figur   geschlagen? 6. Allenfalls gegnerische Figur   entfernen. |

Dieses Szenario kann so oft wiederholt werden bis ein König geschlagen wurde und das Spiel beendet wird.

**Alternativer Ablauf:**

|  |  |
| --- | --- |
| Akteur:  6.b Akteur darf Zug nicht ausführen | System:  6.a Zug verstösst gegen Regeln |

**Häufigkeit des Auftretens:**

Dieser Anwendungsfall tritt auf solang das Spiel läuft.

### Casual Anwendungsfälle

#### „Regeln durchsetzten“

**Haupterfolgsszenario:**

Wählt der Benutzer eine Figur aus, wird die Regel-Engine angestossen. Diese überprüft welche Züge mit der gewählten Figur erlaubt sind. Zieht der Benutzer eine Figur auf ein unerlaubtes Feld, so wird die Figur auf das vorherige Feld zurückgesetzt.

**Alternatives Szenario:**

Wird einer der zwei Könige geschlagen, so wird das Spiel beendet.

#### „Figur wählen“

**Haupterfolgsszenario:**

Der Benutzer wählt eine Figur aus, mit der er den nächsten Zug tätigen möchte. Das System prüft ob die ausgewählte Figur dem Spieler gehört, welcher die Figur ausgewählt hat. Das Feld auf dem die ausgewählte Figur steht wird eingefärbt. Die möglichen Zielfelder werden ebenfalls eingefärbt.

**Alternatives Szenario:**

Falls die ausgewählte Figur nicht dem Spieler gehört, welcher die Figur ausgewählt hat wird die Figur wieder abgewählt.

#### „Neues Spiel starten“

**Haupterfolgsszenario:**

Sobald der Benutzer ein neues Spiel startet wird das Schachfeld initialisiert und die Figuren werden in Anfangsstellung bereitgestellt.

**Alternatives Szenario:**

Falls das Spiel nicht neu gestartet werden kann, wird vom System eine Fehlermeldung erzeugt.

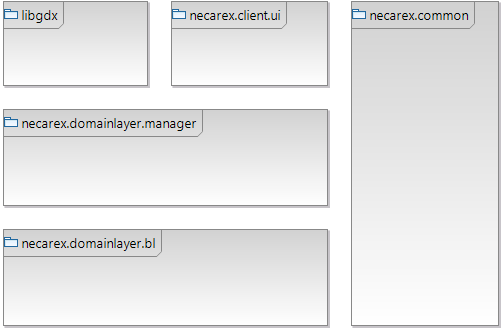
### Brief Anwendungsfälle

#### „Gegenspieler selektieren“

Der Benutzer wählt seinen Gegenspieler, entweder menschlicher Gegenspieler oder Computerspieler aus.

## Eine erste 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 folgende Packagestruktur entschieden:

Erläuterungen zu den von uns entwickelten Packages:

* necarex.common

Hier wird abgelegt, was in mehreren Packages verwendet wird. Hauptsächlich die Data Transfer Objets (nachfolgend DTO genannt), also die reinen Datenhalter.

* necarex.client.ui

Dieses Package ist für das UI-Rendering verantwortlich. Hier werden die Funktionalitäten von libgdx verwendet. Sämtliche Aktionen die entgegen genommen werden, werden an die Manager weiter gegeben.

* necarex.domainlayer.manager

Die Manager halten den Zustand des Programms, also die DTOs und führen Mutationen auf ihnen aus.

* necarex.domain.bl

BL ist die Abkürzung für „business logic“. Hier findet also die spielbezogene Logik statt, z.B. die Prüfung ob ein Spielzug gültig ist. Die Logikklassen sind zustandslos, die Manager, welche die Logik verwendet, übergeben jeweils die aktuellen Spielinformationen.

Zugriff von Package zu Package kann, mit Ausnahme vom Common und libgdx, nur von oben nach unten erfolgen. Weder das Common noch libgdx hat Zugriff auf andere Packages.

## Zusätzliche Spezifikationen

#### Functionality:

* Anhand von Statistiken kann der Spieler den Figurenverlust ablesen.

#### Usability:

* Die Gestaltung der Masken wird auf Ergonomie optimiert.
* Durch das schlichte und intuitive Design soll „necaREx“ einfach und schnell zu erlernen sein.

#### Reliability:

* Die Regel-Engine wird beim Zug des Spielers aktiv, um die Durchsetzung der Regeln zu garantieren.

#### Performance:

* Die Reaktionszeit des Programms, nicht des Computerspielers, muss unter zwei Sekunden liegen.

#### Supportability:

* Die Software wird modular aufgebaut, um Erweiterungen möglichst einfach ins System zu integrieren.
* „necaREx“ ist vorerst für den deutschen Sprachraum bestimmt.

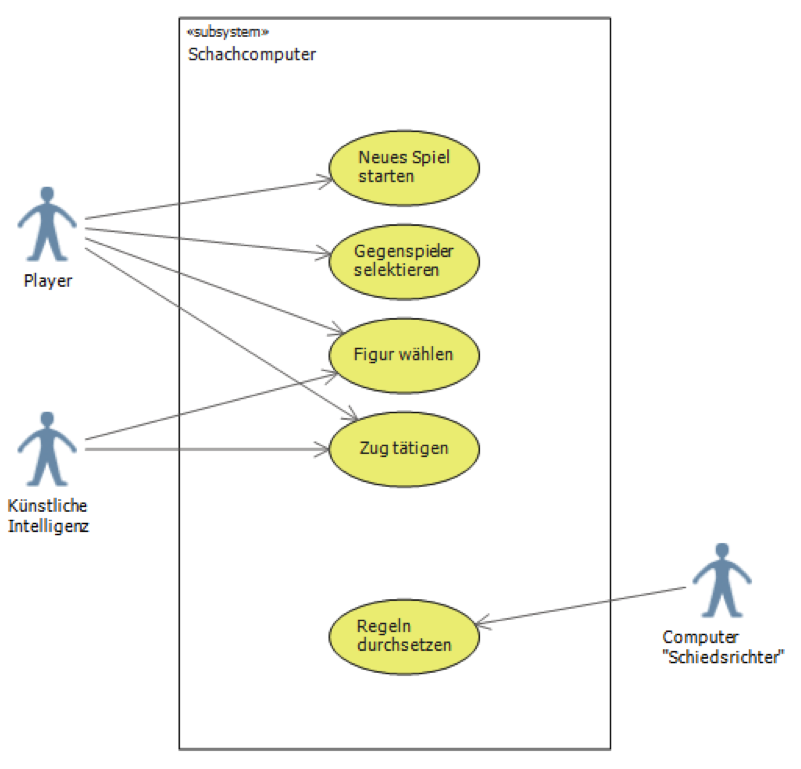
#### Implementation:

* „necaREx“ soll auf allen Standard-Java-Virtual-Maschinen mit Version 1.7 auf allen gängigen Betriebssystemen laufen.

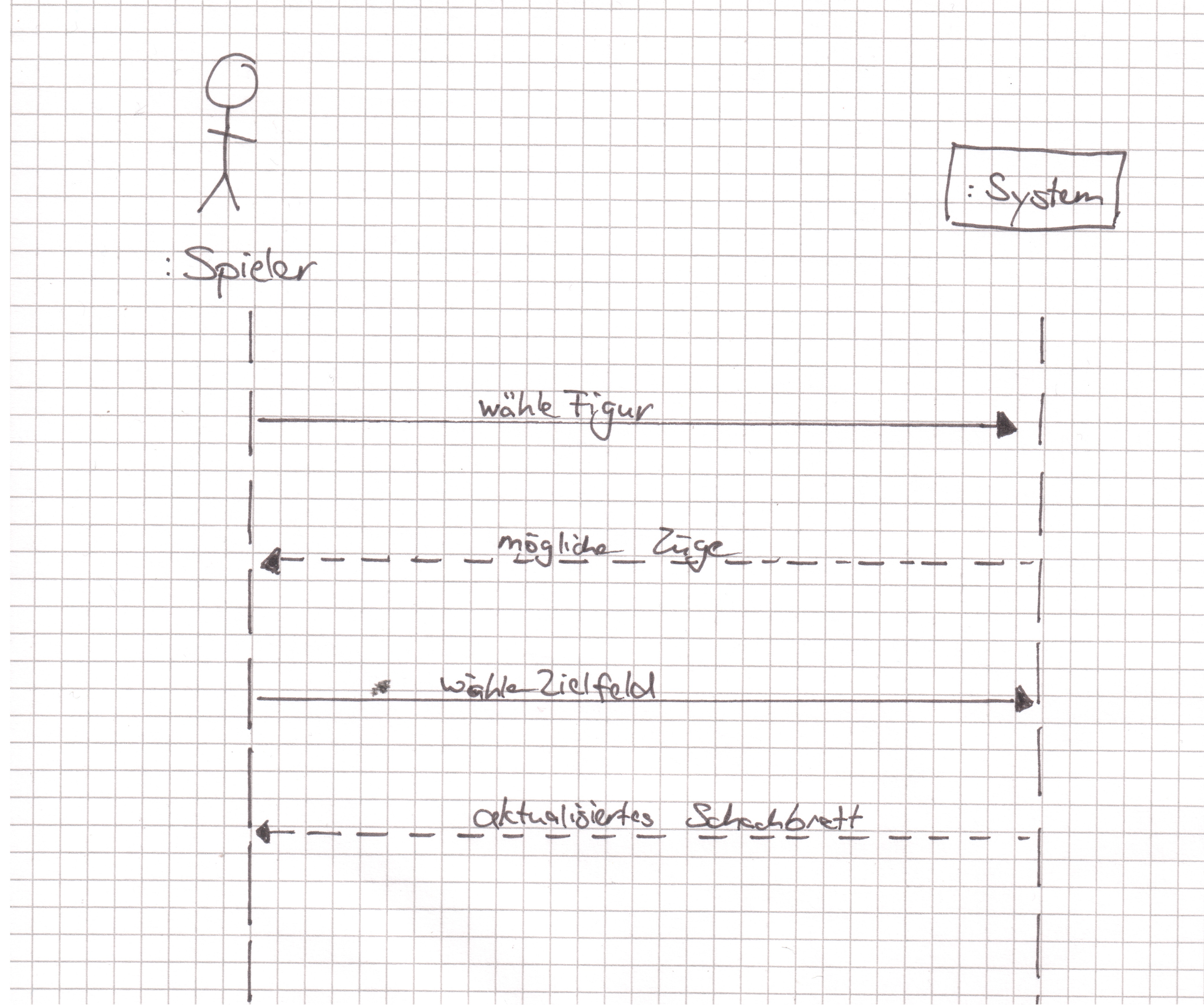
#### Interfaces:

* Das Interface von „necaREx“ soll grafisch sein und ist für normale Bildschirmgrössen ausgelegt (keine Handheld-Geräte).

## Anwendungsfalldiagramm



## System-Sequenzdiagramm



## Systemoperationen

### Systemoperation wähleFigur

**Verweise**

Use Case „Zug ausführen“

**Vorbedingungen**

* Es sind spielbare Figuren auf dem Schachbrett vorhanden
* Der Spieler ist am Zug.

**Nachbedingungen**

* Die Figur die im aktuellen Zug fährt ist bestimmt.
* Die ausgewählte Figur hat mögliche Spielzüge.

### System-Operation wähleZielfeld

**Verweise**

Use-Case „Zug ausführen“

**Vorbedingungen**

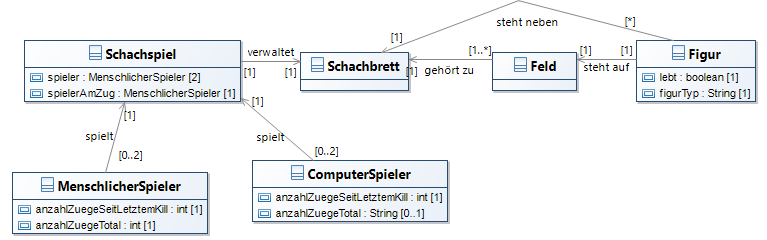
Entsprechen den Nachbedingungen der System-Operation „wähle Figur“:

* Die Figur die im aktuellen Zug fährt ist bestimmt.
* Der ausgewählte Figur hat mögliche Spielzüge

**Nachbedingungen**

* Der Zug wurde durchgeführt und wird auf dem Schachbrett angezeigt
* Der Zug entspricht den Schachregeln und ist gültig.

## Domänenmodell



Die Pfeile beschreiben die Leserichtung und nicht etwaige Abhängigkeiten.

### Beschreibung

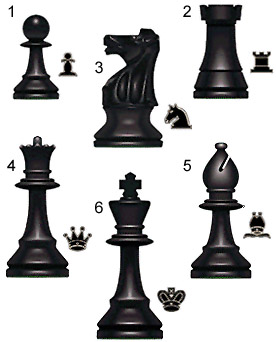
Ein Schachspiel hat zwei Spieler. Einer dieser Spieler ist jeweils am Zug. Die Spieler können menschlich oder künstlich (Computer Spieler) sein. Jeder Spieler hält seine Statistik, diese besteht aus den Anzahl Zügen seit der letzten Tötung einer gegnerischen Figur und der totalen Anzahl der Züge.

Eine Spielfigur steht, falls lebendig, auf einem Feld. Getötete Figuren stehen neben dem Schachbrett. Das Schachbrett hält die verschiedenen Felder. Das Schachspiel verwaltet das Schachbrett.

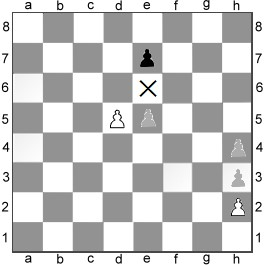
## Glossar

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

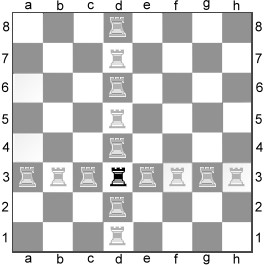
**

**Bauer (1)**

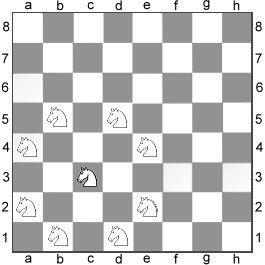
Die schwächste Figur im Schach und hat folgendes Zugverhalten:

* 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.

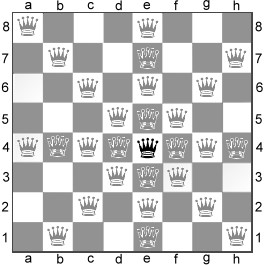
**Turm (2)**

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

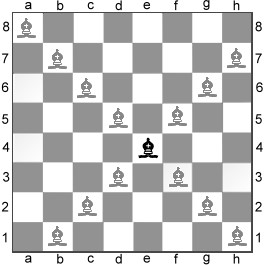
**Springer (3)**

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.

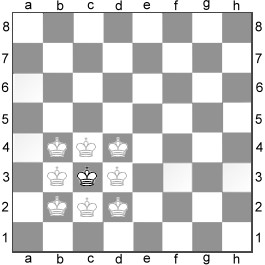
**Dame (4)**

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)**

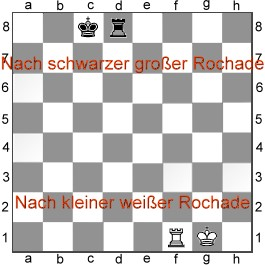
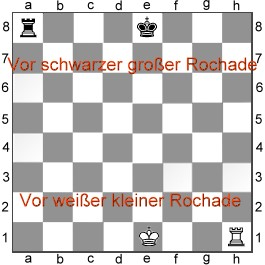
Schachfigur, die auf ein beliebiges Feld entlang der Diagonale ziehen darf, auf welcher er steht.

**König (6)**

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

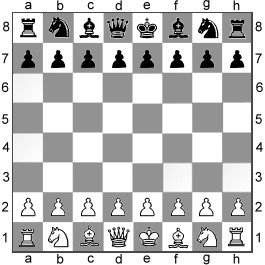
Wird der König in einem gegnerischen Zug angegriffen, so ist der Gegner dazu verpflichtet darauf mit dem Wort „**Schach!**“ hinzuweisen. Der Spieler muss dann die Schachsituation aufheben, beispielsweise durch einen Zug des Königs aus dem Schach.

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