

Bauernschach

Java Projekt



May 20, 2022

Mohammad Ramadan

[Glossar 3](#_Toc103952544)

[1.0 Einleitung 5](#_Toc103952545)

[1.1 Ziel dieses Dokument 5](#_Toc103952546)

[1.2 Anforderungen 5](#_Toc103952547)

[1.3 Geschichte 5](#_Toc103952548)

[2.0 Entwurf 6](#_Toc103952549)

[2.1 Klassendiagramm 6](#_Toc103952550)

[2.2 Beschreibung den Klassen 6](#_Toc103952551)

[Realisierung 7](#_Toc103952552)

[666 Dokumentation 7](#_Toc103952553)

[Entwicklerdokument 7](#_Toc103952554)

***Ein Wort von dem Autor***

Das Bauerschach Projekt wurde während des 3. Ausbildungsjahres dem Autor erstellt, es wurde in diesem Zeitrahmen immer an der Schule oder an den Wochenenden programmiert, diese führte häufig zu unterschiedliche Versionen, es wurde aber nur die Letzte davon dargestellt.

Das Projekt ist unter MIT lizenziert, die Quellcode davon darf gelesen, bearbeitet und veröffentlicht werden, eine Garantie für "vollständige" Funktonalität ist nicht vorhanden.

Aufgrund Beschäftigung der Autor mit seiner Ausbildung zumindest 5 Tagen an der Woche und die Bearbeitung an einem anderen Projekt durch Unity Engine, ist dieses Dokument und das Projekt noch immer in Bearbeitungsphase.

Dieses Dokument wurde auf Deutsch geschrieben, eine Zweite Version auf Englisch wird in der Zukunft erstellt.

# 1.0 Einleitung

In diesem Kapital beschreibt der Autor Grundlagen des Projektes, wann und wieso es gestartet wurde.

## 1.1 Ziel dieses Dokument

Das Wichtigste im Projekt zu beschreiben, die Gescheite davon und die aufgetretenen Probleme und ihre Lösungen.

Das Dokument wurde nach Abschluss des Projektes erstellt. Wobei der Autor sich nicht mehr an der Schule befindet, möchte er sein Wissen in Java praktischerweise an andere Betriebe darstellen, dazu auch seine Fähigkeit in einer Fremdsprache (Deutsch) dokumentieren zu können.

## 1.2 Anforderungen

Die folgende sind die Anforderungen aus der entsprechende Fachlehrer:

* funktionierendes Spiel mit GUI (graphische Oberfläche)
* Klasse Game bildet die Logik des Spiels ab
  + Collection/ArrayList (Schwarz) spielt gegen Collection/ArrayList (Weiss)
* Abstrakte Klasse (Figur)
* Interface (Schnittstelle zur GUI) - Framework Java Swing oder AWT
  + new Game() - erstellt ein neues Spiel
  + move() - bewegt den Bauern
  + getWinner() - gibt Sieger zurück (kann auch in gameOver gelöst werden)
  + gameOver() - Weiss oder Schwarz gewinnt oder unentschieden
    - Sieg, wenn ein Bauer eines Spielers die Grundlinie des anderen Spielers erreicht hat
  + nächsterSpieler() - gibt Auskunft, wer als nächstes ziehen muss
  + + beliebige weitere Methoden
* Zwei Spieler spielen gegeneinander (Weiss gegen Schwarz)

## 1.3 Geschichte

Das Bauernschach Projekt wurde am Ende 2021 als Schulprojekt entwickelt, In dieser Zeit sollte der Autor die gelernten Lektionen in Java in einem praktischen Fall darstellen, damals existierte noch keine moderne GUI, deswegen wurden die Steine durch die Konsole Interface Als ‚W‘ für White ‚B‘ für Black angezeigt, die Programmierung davon existierte in Shell Klasse.

Nach Beschaffung der Funktionalität wurde festgestellt, dass eine grafische Oberfläche super wäre und sogar nötig für die spätere Erweiterung ist.

Im Jahr 2022 wurde die Klasse GUI entwickelt, das Ziel davon war die grafische Oberfläche darzustellen; bei der Programmierung von Händler Methoden wurde festgestellt, dass für jeden Button eine Methode nötig ist und somit hätte die Klasse über 60 Methoden und das ist noch immer ein Teil des Programmes!

Später wurde eine neue Klasse erstellt, um das Ziel in eleganterer Form erreichen zu können, das Problem mit den Händler Methoden wurde durch das sog. ActionListener gelöst.

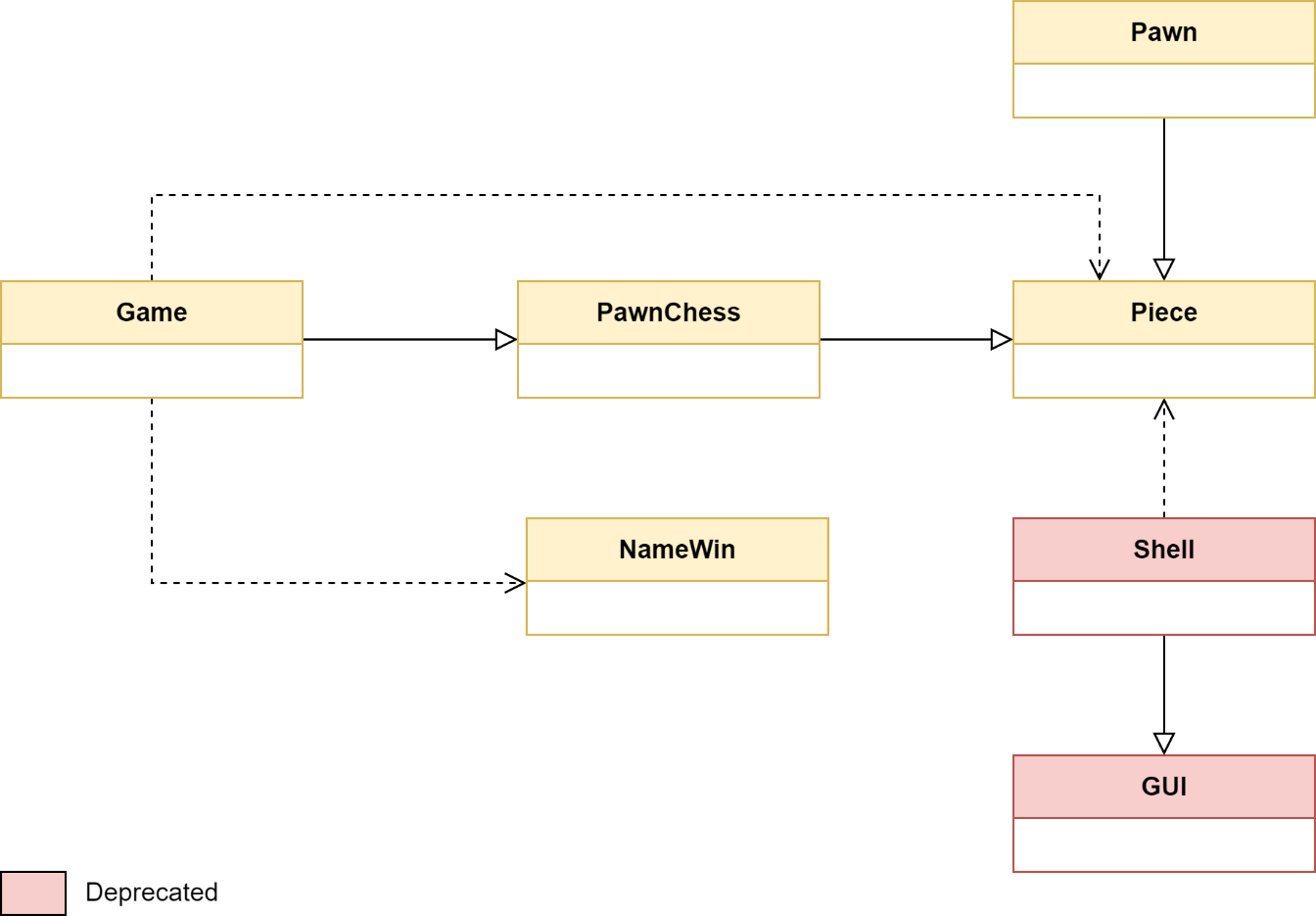
Im Moment befindet sich das Programm in einem sehr guten Zustand, im Prinzip hat das Projekt alles was von einem Bauernschach Spiel erwartet wird.

## 2.0 Entwurf

Hier geht’s um die UML-Diagramme und kurze Beschreibung den Klassen.

## 2.1 Klassendiagramm

Um die Beziehungen zwischen die Klassen besser darzustellen, wurde das folgende Klassendiagramm dargestellt, die Attributen und die offizielle Klasse wie JFrame werden hier nicht gezeigt.



## 2.2 Beschreibung den Klassen

|  |  |
| --- | --- |
| Game | Die Hauptklasse des Programmes, die Grafischeoberfläche wurde hier entwickelt, ActionListener wurde hier im Einsatz gebracht. |
| PawnChess | Temporäre Variablen werden rein gespeichert, verfügt auch über veraltete Methoden. |
| NameWin | Stellt ein separates Fenster für die Eingabe von Benutzernamen. |
| Pawn | Hat die spezifischen Funktionen für Bauern wie Bewegung. |
| Piece | Allgemeine Funktionen und Attributen, die für mehrere Steintypen anwendbar sind. |
| Shell | Veraltete Klasse für die Konsole Interface. |
| GUI | Veraltete Klasse, verfügt über Moderne GUI. |

# 3.0 Realisierung

Das Programm wurde zuerst aufgrund der Anforderungen auf Deutsch programmiert, durch die Refactoring-Methode wurden die Namen später auf Englisch umgestellt, Ziel der Autor ist das Projekt in der Zukunft unter MIT Lizenzen zu veröffentlichen.

Am Anfang des Projektes waren wenige Klassen vorhanden, mit der Zeit wurden neu Methoden programmiert und dafür wurden neu Klassen erstellt. Die veralteten Klassen wurden nicht gelöscht, sondern als „Deprecated“ betrachtet, das wurde auch in Entwicklungsdokument dank Javadoc klar dargestellt.

Veraltete Klassen, die noch immer über lauffähige Funktionen verfügen wie Pawn, wurden nicht ausgelöscht, in solche Fehle werden nur die veralteten Funktionen als veraltet betrachtet, sonst darf die Klasse weiter benutzt werden.

Im Moment befindet sich der Quellcode alle wichtigen Klassen in guten Zustand, in der Zukunft sollen nur die veralteten Klassen völlig gelöscht werden.

## 3.1 Versionen

Das Projekt verfügt über 2 Versionen mit GUI und eine dritte nur mit Konsole Interface, hier wird nur die letzte Version mit einem Verbesserte GUI dokumentiert.

Versionen von:

JDK 🡪 17

NetBeans 🡪 13 und 12

## 4.0 Qualitätsmangament

Das Projekt wird von mehreren IT-Fachleuten gelesen, deswegen muss der Quellcode lesbar sein, es wurde festgestellt, dass die häufigen benutzten Objekte deutliche Namen haben, z.B. p1\_name ist Name dem ersten Spieler, arL\_white ist "ArrayList for white pawns" usw. die gleiche gilt auch für die Funktionen.

Die klare Beschreibung den Quellcode lässt die Klassen von einer Anwendung zur andere übertragen werden, jedoch sollten paar Änderungen ausgeführt werden (Einfache Copy-and-paste geht's nicht immer :) ).

Zuverlässigkeit des Projektes wurde durch mehrere Abfragen gesichert, jeder möglicher Fall wird betrachtet und geprüft, der entsprechende Befehl darf nur angewendet werden, solange das Programm er als zulässig betrachtet.

# 4.0 Dokumentation

Das folgende Kapital beschreibt, wie das Projekt dokumentiert wurde

## 4.1 Entwicklerdokument

Der Quellcode alle nutzbaren Eigenschaften/Funktionen ist im Moment so dokumentiert, dass jede Informatiker/in im Projekt hereinschauen kann und seine Änderungen implementiert.

Die Nutzbare Funktionen/Attributen haben Kommentare, die mithilfe von javadoc erzeugt werden können, die unklaren Zeilen davon werden auch direkt im Quellcode kommentiert. Alles anders was veraltet ist, wird direkt als "Deprecated" in Entwicklungsdokument betrachtet.

# 5.0 Ausbilck

Veraltete Klassen sollen gelöscht werden, sogar sie nicht als „schädlich Teile“ betrachtet werden sollen, es wäre aber besser, wenn nur die moderne Klasse existieren, diese führte noch zum Klare Darstellung von dem Quellcode und macht das Programm noch einfacher zu übertragen.