



build success JitPack 1.2 License Apache 2.0

Dieses GitHub-Repository von Nils Polek (Matrikelnummer 5428131) enthält eine Java-Bibliothek namens `com.github.nilspolek.chess-java`, die die Implementierung eines Schachspiels bietet.

## Installation

---

### Maven

---

```
<repositories>
  <repository>
    <id>jitpack.io</id>
    <url>https://jitpack.io</url>
  </repository>
</repositories>

<dependency>
  <groupId>com.github.nilspolek</groupId>
  <artifactId>chess-java</artifactId>
```

```
<version>v1.2</version>  
</dependency>
```

## Verwendung

---

### Schachbrett erstellen

```
Board board = new Board();
```

### Schachstellung einrichten

Das Schachbrett kann durch einen **FEN**-string (Forsyth-Edwards-Notation) aufgebaut werden.

```
board.setFEN("rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR");
```

### Zugberechnung

```
board.findBestMove();  
while (board.getLastBestMove() == null) Thread.sleep(1000);  
Move move = board.getLastBestMove();
```

Dieser Aufruf berechnet den besten Zug für das aktuelle Schachbrett. Der berechnete Zug wird als `Move`-Objekt bei der `getLastBestMove` methode zurückgegeben.

### Zug ausführen

```
boolean success = board.move(move);
```

Mit diesem Aufruf wird der berechnete Zug auf dem Schachbrett ausgeführt. Der Rückgabewert `success` gibt an, ob der Zug erfolgreich war.

## Spiel speichern und laden

```
board.saveGame("game.txt");  
board.loadGame("game.txt");
```

Mit diesen Methoden können Sie den aktuellen Spielstand speichern und aus einer Datei laden.

## Weitere Methoden

Das `Board` -Objekt bietet auch andere nützliche Methoden, z. B. zum Abrufen der aktuellen Schachbrettstellung (`getBoard()`), zur Bewertung des Schachbretts (`evaluate()`) und zur Überprüfung, ob das Spiel beendet ist (`isCheckMate()`).

## Beispiel

---

```
import com.github.nilspolek.Board;  
  
public class Main {  
    public static void main(String[] args) throws InterruptedException {  
        Board board = new Board();  
        board.setFEN("rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR");  
  
        board.findBestMove();  
        while (board.getLastBestMove() == null) Thread.sleep(1000);  
        boolean success = board.move(board.getLastBestMove());  
  
        if (success) {  
            System.out.println("Bester Zug: " + board.getLastBestMove());  
            System.out.println("Neue Stellung:");  
            System.out.println(board);  
        } else {  
            System.out.println("Kein gültiger Zug gefunden.");  
        }  
    }  
}
```

Dieses Beispiel zeigt, wie das `Board` -Objekt verwendet werden kann, um den besten Zug für das gegebene Schachbrett zu finden und auszuführen.

## MinMax Algorithmus

---

Es kommt ein Min-Max Algorithmus zum einsatz der benutzt wird um herauszufinden was der beste zug ist.

Dieser zählt zur Evaluation die Figuren.

Wenn die Zeit beendet ist oder die `stopProcessing` methode aufgerufen wird, wird keine weitere Tiefe evaluiert

Figur	Punkte
Bauer	1
Leufer	3
Springer	3
Turm	5
Dame	9

```
void findMateIn1() throws InterruptedException {
    Board b = new Board();
    b.setFEN("k4r2/ppp5/8/8/8/8/PPP5/K7");
    b.setWhite(false);
    b.TIME_LIMIT = 10000;
    b.start();
    while (b.isAlive()) {
        Thread.sleep(1000);
        System.out.println("Waiting");
    }
    System.out.println(b.lastBestMove);
}
```

# UI

---

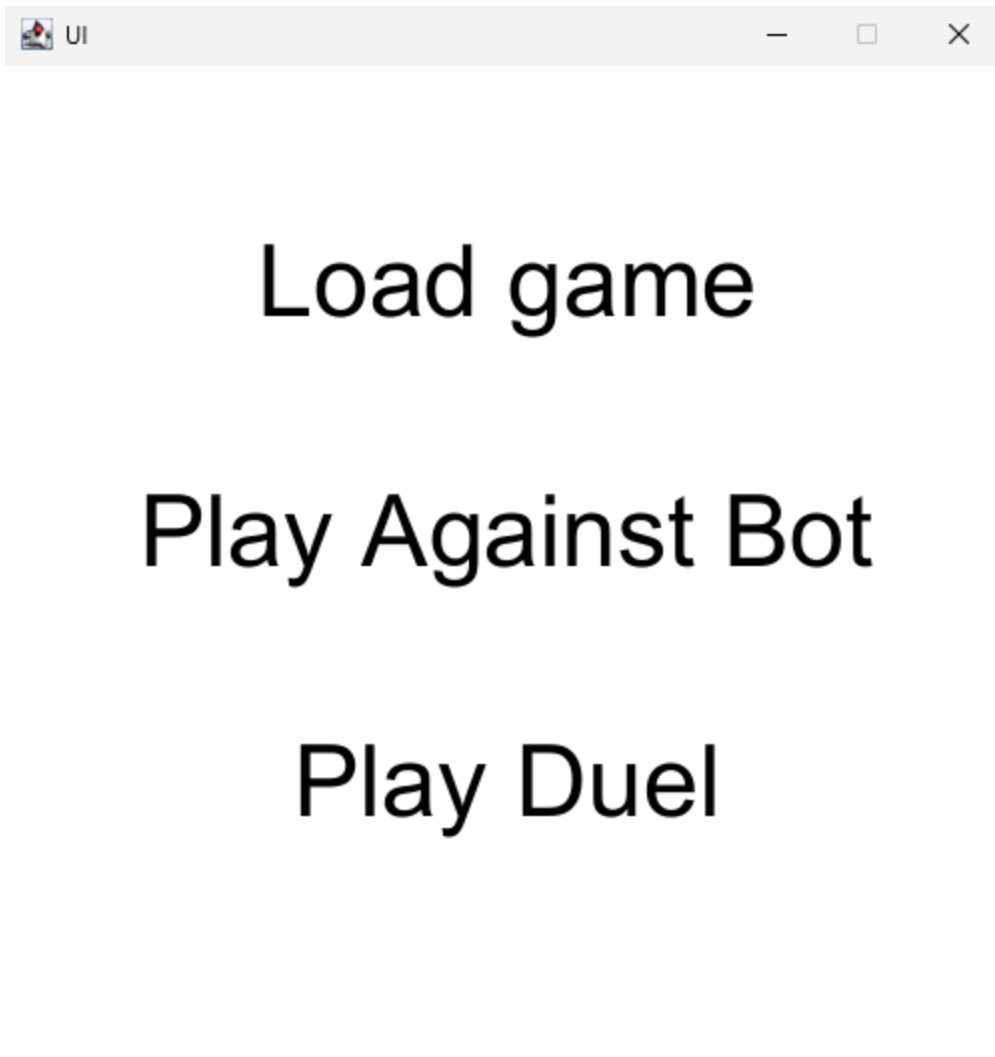
Es wird eine Ui mit ausgeliefert die Processing benutzt

Diese kann mit dem command gestartet werden

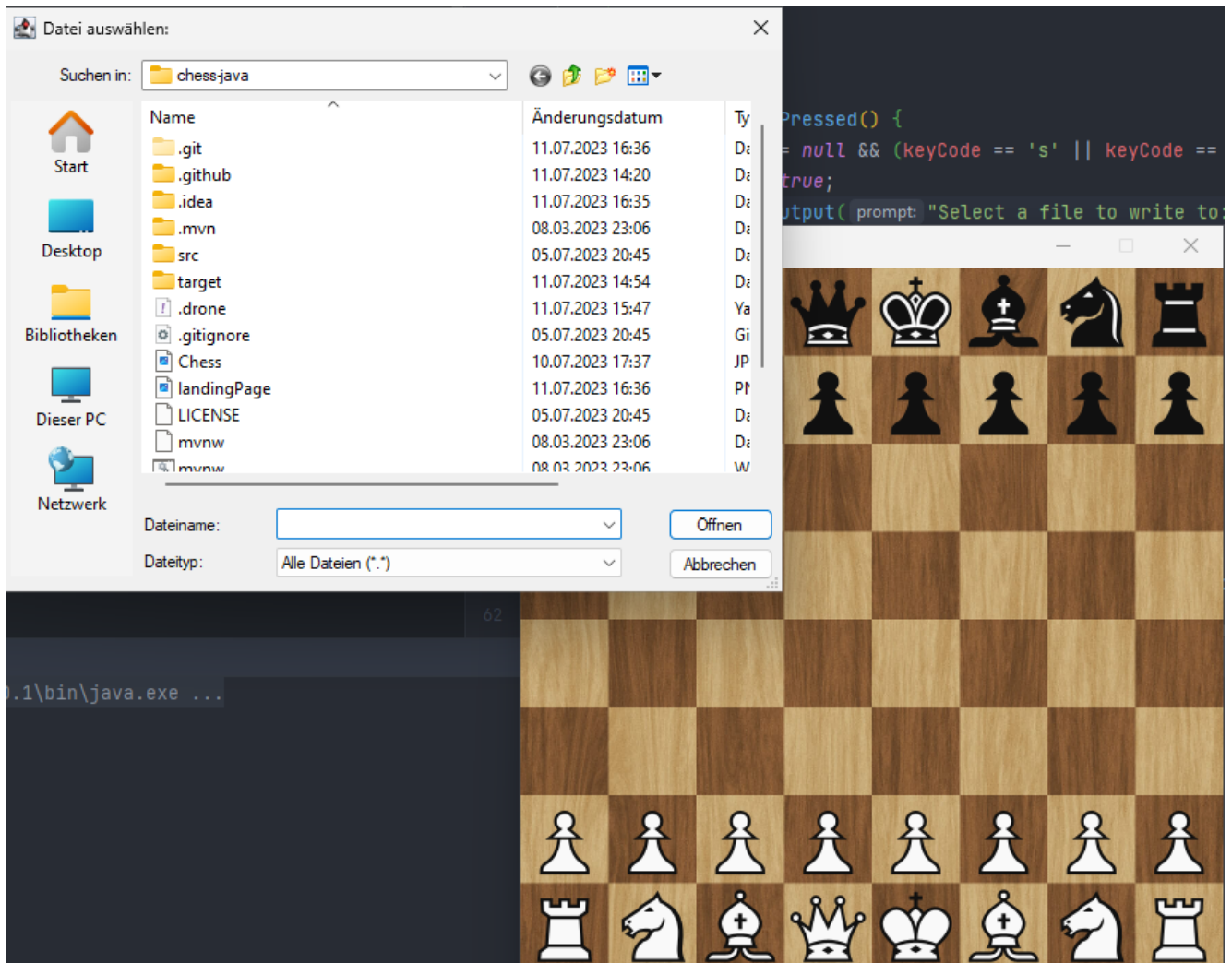
```
wget -o chess.zip https://github.com/nilspolek/chess-java/releases/download/v1.2
unzip chess.zip
cd Schachprojekt
java -jar ./chess-java.jar
```

Taste	Event
z	Geht einen Zug zurück
s	speichert das Spiel
x	Startet ein neues Spiel
y	Stoppt die Berechnung von MinMax

## Landing Page



**Load Game**



## Play Duel (Fried Liver)



## Lizenz

---

Dieses Projekt ist unter der Apache License lizenziert. Weitere Informationen finden Sie in der [Lizenzdatei](#).