Dieses GitHub-Repository van Nils Polek (Matrikelnummer 5428131) enthält eine Java-Bibliothek namens `com.github.nilspolek.chess-java`, die die Implementierung eines Schachbretts und von Schachzügen 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>LATEST</version>
</dependency>
```

Verwendung

Schachbrett erstellen

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

Schachstellung einrichten

Das Schachbrett kann durch die Verwendung der FEN-Notation (Forsyth-Edwards-Notation) eingerichtet werden.

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

Zugberechnung

```
Move move = board.findBestMove();
```

Dieser Aufruf berechnet den besten Zug für das aktuelle Schachbrett. Der berechnete Zug wird als `Move` -Objekt 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 Sie das `Board`-Objekt verwenden können, 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.
Es wird bei einem zug der Schlechter ist als ein anderer nicht witer gerechnet.

Es wird zuerst einmal jeder zug gemacht und dann wird auf die Zeit geschaut.

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/v
unzip chess.zip
java -jar ./chess/chess-java.jar
```

Taste	Event
z	Geht einen Zug zurück

Taste	Event
s	speichert das Spiel
x	Startet ein neues Spiel
y	Stoppt die Berechnung von MinMax

Lizenz

Dieses Projekt ist unter der Apache License lizenziert. Weitere Informationen finden Sie in der Lizenzdatei.