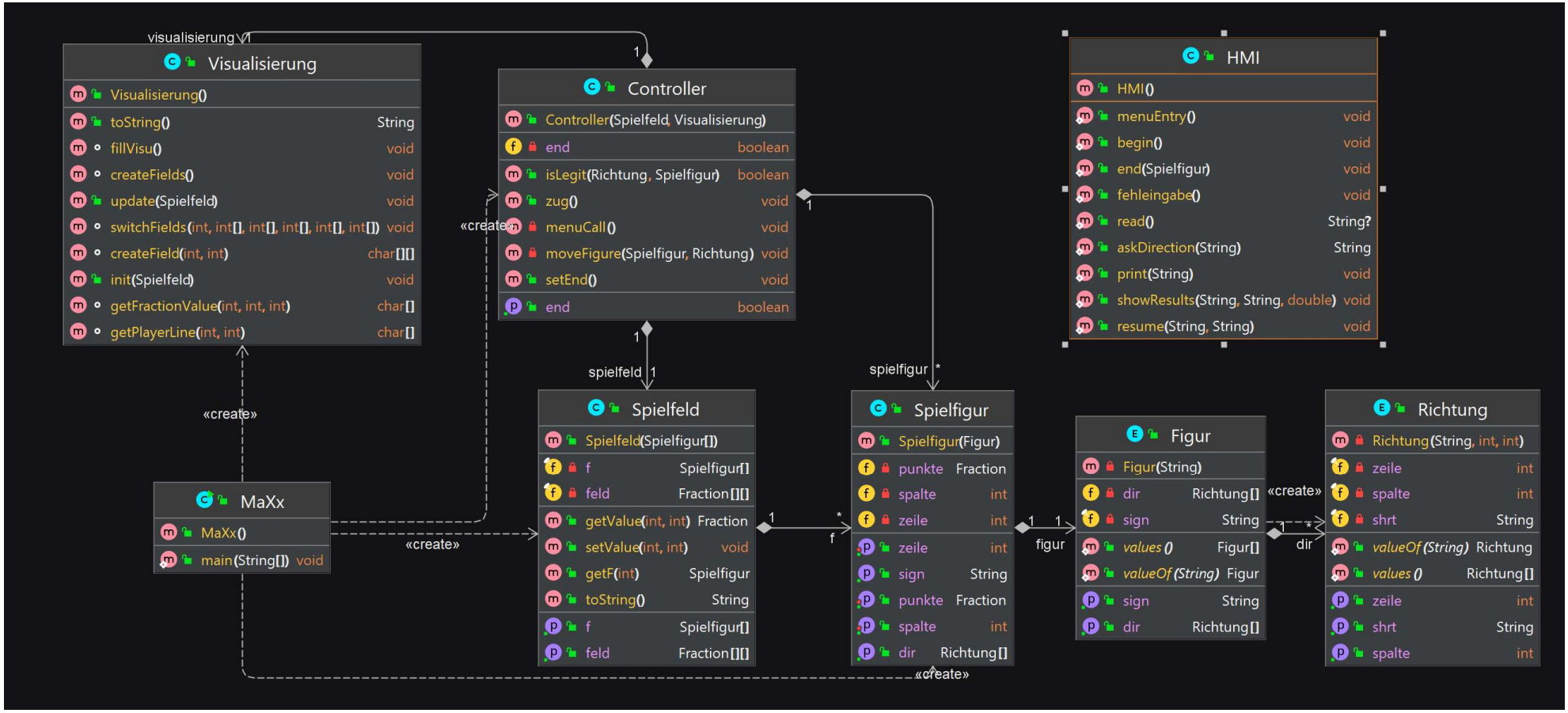
MaXx

BESTES SPIEL

Klassendiagramm



Main Methode

```java
/**
 * @author Jan Obernberger, Kevin Goldmann, Lau Kailany, Florijan Deljija, Benno Dinsch
 * @version X, 11.01.2023
 **/
public class MaXx {
    public static void main(String[] args) {
        Spielfigur[] p = {new Spielfigur(Figur.Weiss), new Spielfigur(Figur.Schwarz)};
        Spielfeld spielfeld = new Spielfeld(p);
        Visualisierung visualisierung = new Visualisierung();
        Controller controller = new Controller(spielfeld, visualisierung);
        while (!controller.isEnd()) {
            controller.zug();
        }
        controller.setEnd();
    }
}
```

Klasse Spielfeld – Instanzvariablen & Konstruktor

```java
public class Spielfeld {
    3 usages
    private final int zeilen = 8;
    3 usages
    private final int spalten = 8;
    18 usages
    private final Fraction[][] feld;
    5 usages
    private final Spielfigur[] f;

    6 usages
    public Spielfeld(Spielfigur[] arr) {
        this.feld = new Fraction[this.zeilen][this.spalten];
        this.f = arr;

        for (int i = 0; i < this.zeilen; i++) {
            for (int j = 0; j < this.spalten; j++) {
                Random r1 = new Random();
                Random r2 = new Random();

                do {
                    this.feld[i][j] = new Fraction(new BigInteger( numBits: 10, r1), new BigInteger( numBits: 10, r2));
                } while (this.feld[i][j] == null || this.feld[i][j].getNumerator().compareTo(this.feld[i][j].getDenominator()) <= 0 || this.feld[i][j].getDenominato

            }
        }
        for (Spielfigur ff : f) {
            setValue(ff.getZeile(), ff.getSpalte());
        }

    }

```

Enums Richtung & Figur, Klasse Spielfigur

```java
public enum Richtung {
    3 usages
    Nord( shrt: "N",  zeile: -1,  spalte: 0), Ost( shrt: "O",  zeile: 0,  spalte: 1), Sued( shrt: "S",  zeile: 1,  spalte: 0),
    3 usages
    West( shrt: "W",  zeile: 0,  spalte: -1), Nordost( shrt: "NO",  zeile: -1,  spalte: 1), Suedwest( shrt: "SW",  zeile: 1,  spalte: -1);

}
```

```java
public enum Figur {
    1 usage
    Schwarz( sign: "B"), Weiss( sign: "W");
    4 usages
    private final String sign;
    3 usages
    private Richtung[] dir;

    2 usages
    Figur(String sign) {
        this.sign = sign;
        if (this.sign.equals("B")) {
            this.dir = new Richtung[]{Richtung.Nord, Richtung.Ost, Richtung.Sued, Richtung.West, Richtung.Suedwest};
        }
        if (this.sign.equals("W")) {
            this.dir = new Richtung[]{Richtung.Nord, Richtung.Ost, Richtung.Sued, Richtung.West, Richtung.Nordost};
        }
    }

}
```

```java
public class Spielfigur {
    5 usages
    private final Figur figur;
    4 usages
    private int zeile;
    4 usages
    private int spalte;
    4 usages
    private Fraction punkte;


    10 usages
    public Spielfigur(Figur figur) {
        this.figur = figur;
        this.punkte = new Fraction( zaehler: 0,  nenner: 0);
        if (this.figur.getSign().equals("W")) {
            this.zeile = 2;
            this.spalte = 3;
        } else if (this.figur.getSign().equals("B")) {
            this.zeile = 5;
            this.spalte = 4;
        }
    }

}
```

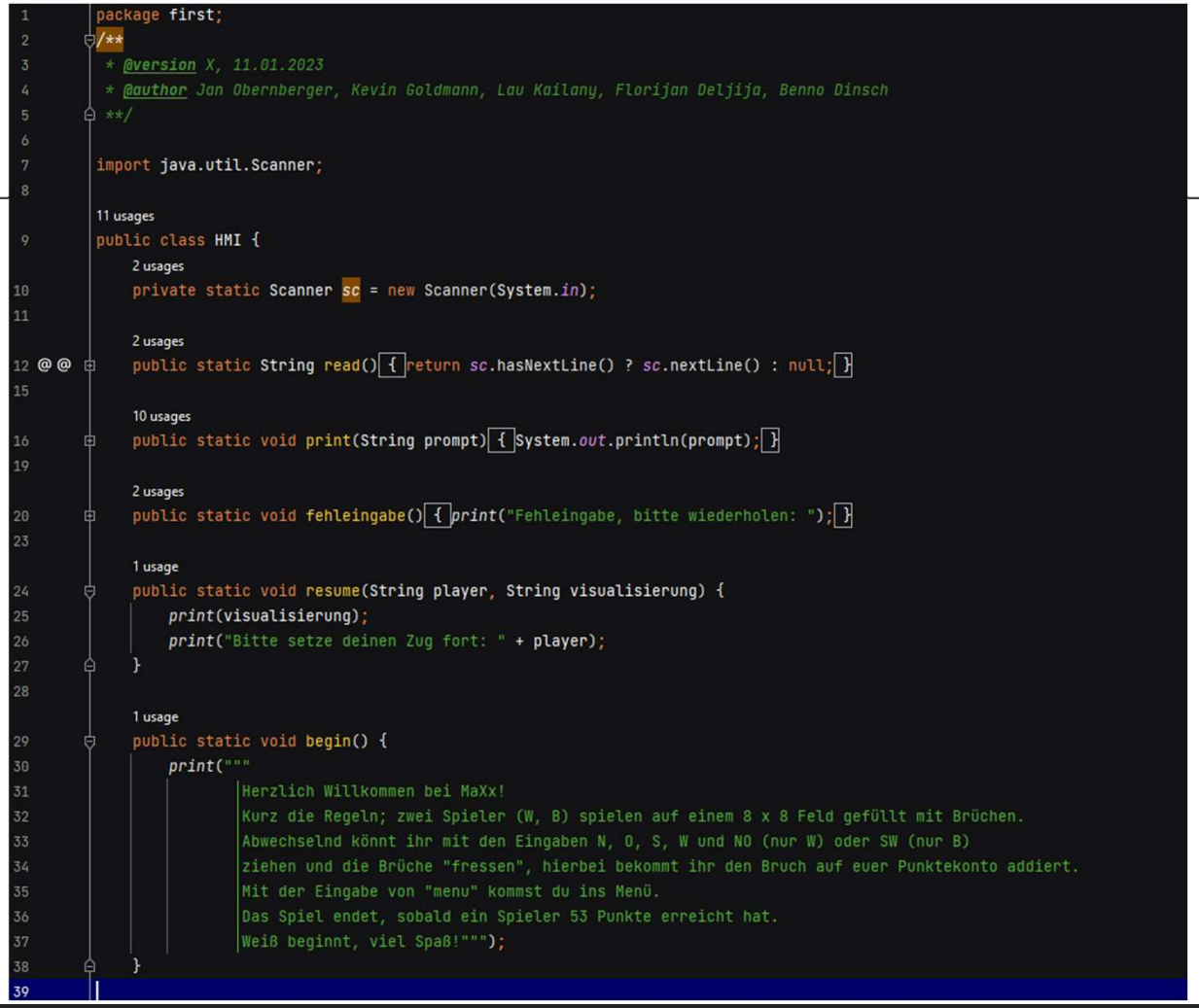
Controller: Zug Methode

```java
public void zug() {
    playerIndex = player ? 0 : 1;
    if (eingabe) {
        HMI.print(visualisierung.toString());
        for (Spielfigur f : spielfigur) {
            Fraction points = f.getPunkte();
            if (!points.getNumerator().equals(BigInteger.ZERO) && !points.getDenominator().equals(BigInteger.ZERO))
                //Anzeige des Punktestands des Spielers nur, wenn Punkte > 0!
                HMI.showResults(f.getSign(), f.getPunkte().toString(), f.getPunkte().doubleValue());
        }
    }
    String s2 = HMI.askDirection(spielfigur[playerIndex].getSign());
    eingabe = false;
    for (Richtung r : spielfigur[playerIndex].getDir()) {
        if (r.getShrt().equalsIgnoreCase(s2)) {
            //Eingabe wird true, wenn der eingegebene String mit einer legitimen Richtung der Spielfigur des
            //spieler übereinstimmt
            eingabe = true;
            break;
        }
    }
    if (eingabe) {
        Richtung richtung = switch (s2.toUpperCase()) {
            case "N" -> Richtung.Nord;
            case "O" -> Richtung.Ost;
            case "S" -> Richtung.Sued;
            case "W" -> Richtung.West;
            case "NO" -> Richtung.Nordost;
            case "SW" -> Richtung.Suedwest;
            default -> throw new IllegalStateException("Unexpected value: " + s2);
        };
        if (!isLegit(richtung, spielfigur[playerIndex])) {
            eingabe = false;
        } else {
            moveFigure(spielfigur[playerIndex], richtung);
            visualisierung.update(spielfeld);
        }

    } else if (s2.equalsIgnoreCase( anotherString: "menu")) {
        menuCall();
        if (recWat || end) {
            recWat = !recWat;
            return;
        }

    }
    if (!eingabe) {
        HMI.fehleingabe();
        zug();
    }
}

```

Controller: Menü

```
93      private void menuCall() {
94          menu = true;
95          HMI.menuEntry();
96          while (menu) {
97              String s3 = HMI.read();
98              switch (s3.toLowerCase()) {
99                  case "resume" -> {
100                     recWat = true;
101                     menu = false;
102                     HMI.resume(spielfigur[playerIndex].getSign(), visualisierung.toString());
103                     zug();
104                 }
105                 case "exit" -> {
106                     end = true;
107                     recWat = true;
108                     return;
109 
110                 }
111                 default -> HMI.fehleingabe();
112             }
113 
114         }
115     }
116 
```

HMI (=Human Machine Interface)



```java
package first;
/**
 * @version X, 11.01.2023
 * @author Jan Obernberger, Kevin Goldmann, Lau Kailany, Florijan Deljija, Benno Dinsch
 **/

import java.util.Scanner;

11 usages
public class HMI {
    2 usages
    private static Scanner sc = new Scanner(System.in);

    2 usages
    public static String read() { return sc.hasNextLine() ? sc.nextLine() : null; }

    10 usages
    public static void print(String prompt) { System.out.println(prompt); }

    2 usages
    public static void fehleingabe() { print("Fehleingabe, bitte wiederholen: "); }

    1 usage
    public static void resume(String player, String visualisierung) {
        print(visualisierung);
        print("Bitte setze deinen Zug fort: " + player);
    }

    1 usage
    public static void begin() {
        print("""
                Herzlich Willkommen bei MaXx!
                Kurz die Regeln; zwei Spieler (W, B) spielen auf einem 8 x 8 Feld gefüllt mit Brüchen.
                Abwechselnd könnt ihr mit den Eingaben N, O, S, W und NO (nur W) oder SW (nur B)
                ziehen und die Brüche "fressen", hierbei bekommt ihr den Bruch auf euer Punktekonto addiert.
                Mit der Eingabe von "menu" kommst du ins Menü.
                Das Spiel endet, sobald ein Spieler 53 Punkte erreicht hat.
                Weiß beginnt, viel Spaß!""");
    }

```

Visualisierung: Aufbau

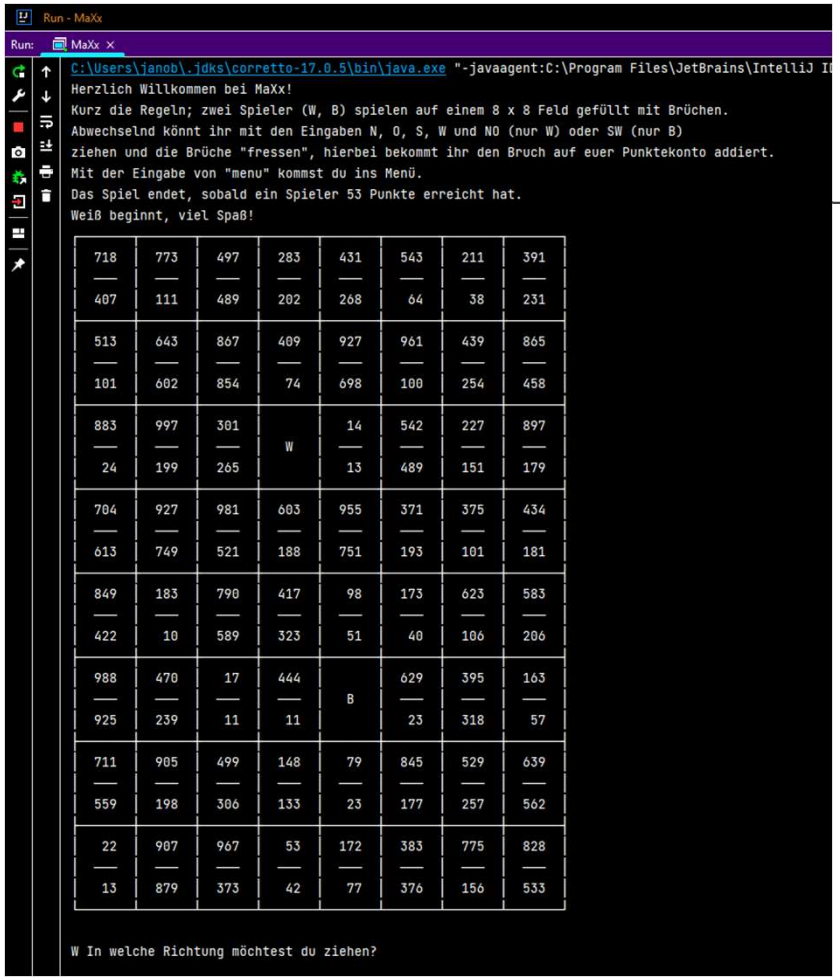
```java
public class Visualisierung {

    6 usages
    private static final int zeilenAussen = 8;
    6 usages
    private static final int spaltenAussen = 8;
    13 usages
    private static final int zeilenInnen = 3;
    13 usages
    private static final int spaltenInnen = 7;

    4 usages
    private static final char horizontalChar = '\u2500';
    1 usage
    private static final char verticalChar = '\u2502';
    1 usage
    private static final char verticalLeft = '\u251C';
    1 usage
    private static final char verticalRight = '\u2524';
    1 usage
    private static final char horizontalTop = '\u252C';
    1 usage
    private static final char horizontalBottom = '\u2534';
    1 usage
    private static final char cornerTopRight = '\u2510';
    1 usage
    private static final char cornerTopLeft = '\u250C';
    1 usage
    private static final char cornerBottomRight = '\u2518';
    1 usage
    private static final char cornerBottomLeft = '\u2514';
    1 usage
    private static final char node = '\u253c';
    4 usages
    private static final char[] emptyLine = {' ', ' ', ' ', ' ', ' ', ' ', ' '};
    1 usage
    public static final char[][] emptyField = {emptyLine, emptyLine, emptyLine};
    1 usage
```

```java
    private final char[][][][] feld;
    8 usages
    private final char[][] visu;
    2 usages
    private Fraction[][] playground;
    9 usages
    private int[] posZeile, posSpalte;


    4 usages
    public Visualisierung() {

        this.feld = new char[zeilenAussen][spaltenAussen][zeilenInnen][spaltenInnen];
        this.visu = new char[zeilenAussen * zeilenInnen + zeilenAussen + 1][spaltenAu
        this.visuZeilen = zeilenAussen * zeilenInnen + zeilenAussen;
        this.visuSpalten = spaltenAussen * spaltenInnen + spaltenAussen;

    }

```

```java
    1 usage
    public void init(Spielfeld spielfeld) {
```

```java
    1 usage
    public void update(Spielfeld spielfeld) {
```

MaXx in der Konsole




```
W hat 409/74Punkte. ca.5.527027027027027
B In welche Richtung möchtest du ziehen?
```

```
W hat 25890/3737Punkte. ca.6.928017126036928
B hat 122303/38301Punkte. ca.3.193206443695987
W In welche Richtung möchtest du ziehen?
n
Fehleingabe, bitte wiederholen:
W In welche Richtung möchtest du ziehen?
```

```
W B
```

```
W In welche Richtung möchtest du ziehen?
menu
Du bist jetzt im Menü
Mit resume kannst du das Spiel fortsetzten
Mit exit kannst du es beenden
```

```
Gewonnen hat: W mit
 2839309024897/52419810828 oder ca.54.16480868680254

Process finished with exit code 0
```