Nim Turtle View

Dennis Diener, Technische Hochschule Mittelhessen

Methode 1

possibleMoves()-Methode. Diese soll alle möglichen Züge in einer eingegeben Spielsituation durchgehen und wiedergeben.

```
public static List<Move> possibleMoves(Nim nim) {
    List<Move> possibleMoves = new ArrayList<>();
    for (int row = 0; row < nim.rows.length; row++) {
        for (int number = 1; number <= nim.rows[row]; number
            possibleMoves.add(Move.of(row, number));
        }
    }
    return possibleMoves;
}
```

Methode 2

Die zweite Methode ist eine simulation(). Dort werden alle möglichen Züge eines Spiels durchgegangen und in Win und Lose gezählt. Danach wird in einer List MoveResults wiedergegeben, welche move, win, lose speichert für jeden einzelnen Zug.

```
public List<MoveResults> simulation(List<Move> possibleMoves
    final int N = 100;
    List<MoveResults> moveResults = new ArrayList<>();

    for (Move move : possibleMoves) {
        int win = 0;
        int lose = 0;
        int moveCounter = 0;
        for (int i = 0; i < N; i++) {
            Nim simNim = Nim.of(rows);
            simNim = simNim.play(move);
            moveCounter++;
```

```
                boolean simWinner = NimGame.isWinning(simNim.row
                if (simWinner) {
                    win++;
                } else {
                    lose++;
                } moveResults.add(new MoveResults(move, win, los
            } return moveResults;
        }
    }
```

Methode 3

Des weiteren wird eine Klasse MoveResults mit den Variablen move, win, lose festgelegt.

```
class MoveResults {
    Move move;
    int w; //sieg
    int l; //verloren

    MoveResults(Move move, int win, int lose) {
        this.move = move;
        this.w = win;
        this.l = lose;
    }

    @Override
    public String toString() {
        return "Move: " + move + "Win | Lose: " + w + " | " + l
    }
}
```

Methode 4

mcMove soll alle vorherigen Methoden anwenden, um dann das Ergebnis der besten Moves ausrechnet und zurückgibt.

```
    public Move mcMove(){
        assert !isGameOver();

        Move mcMove;
```

```
        double a = 0;

        List<MoveResults> moveResults = simulation(possibleMoves

        for (MoveResults results : moveResults) {
            double x = results.w / results.l;
            if (x < a) {
                x = a;
                results.move = mcMove;
            }
        }
        return mcMove;
    }
```