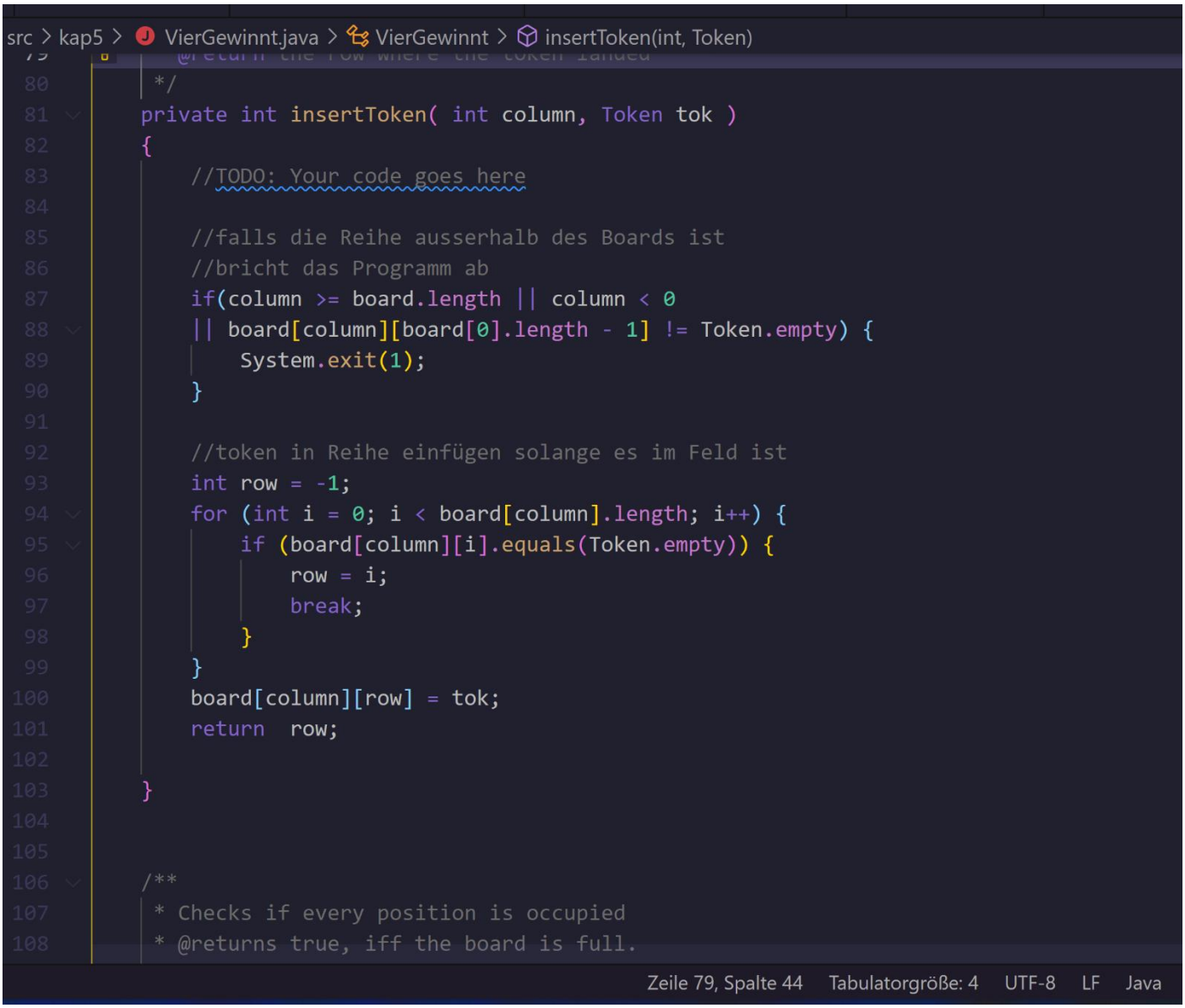
Aufgaben Serie 05 - Programmieren 1

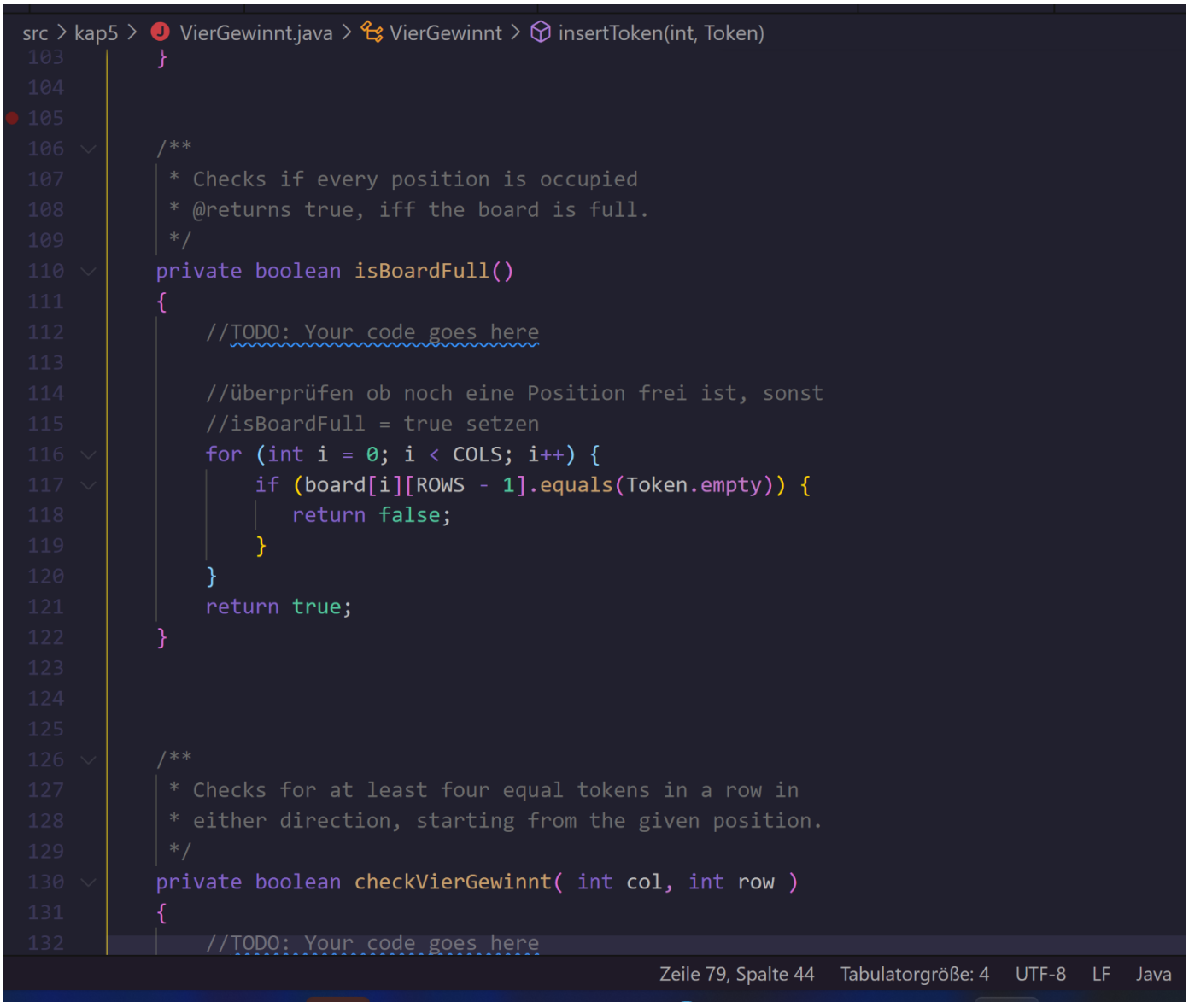
Jara Zihlmann(20-117-032)
Vithusan Ramalingam (21-105-515)
Jan Ellenberger (21-103-643)

Implementationsaufgabe 1. A)



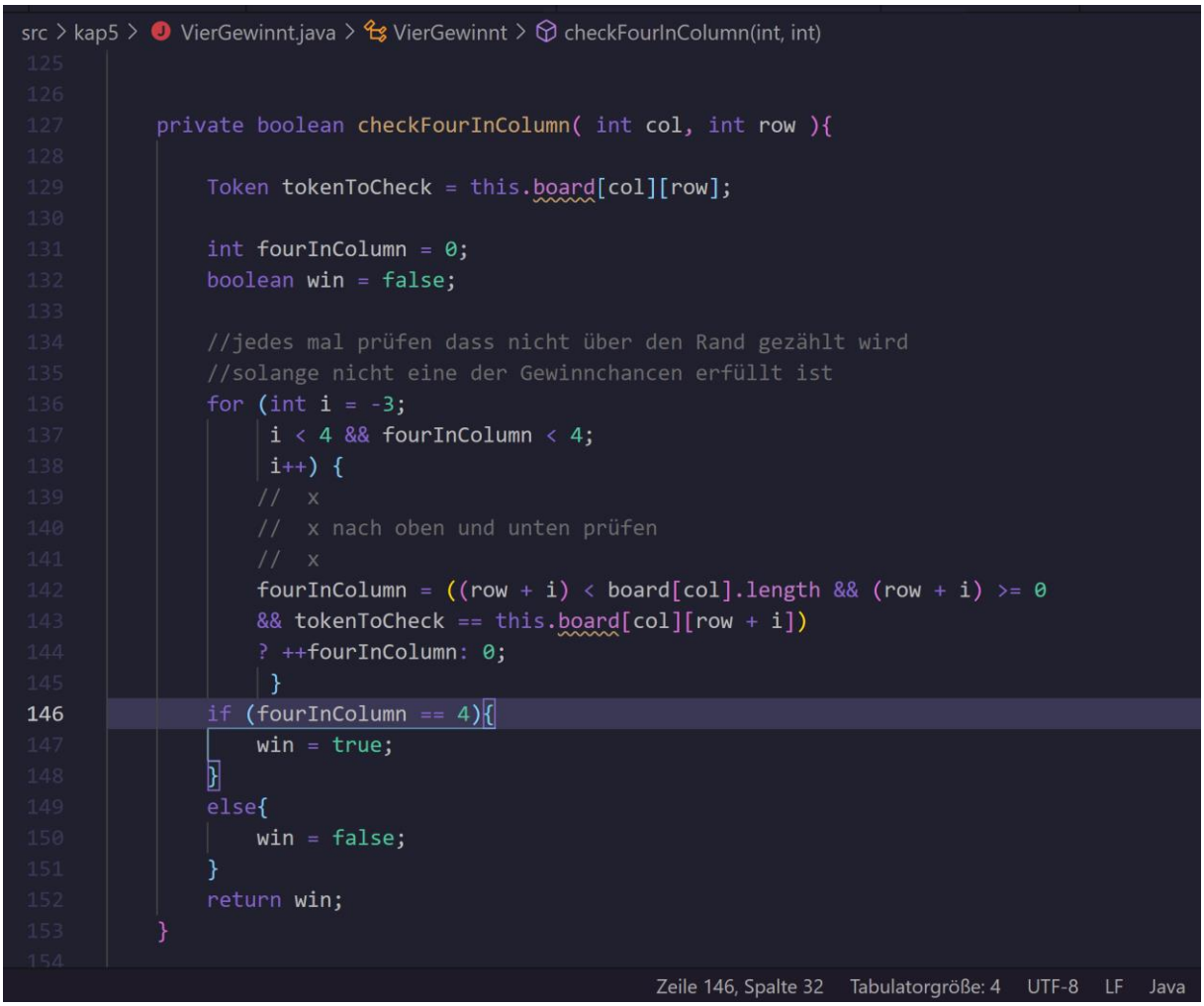
Jara Zihlmann(20-117-032) Vithusan Ramalingam (21-105-515) Jan Ellenberger (21-103-643)

Implementationsaufgabe 1. b)

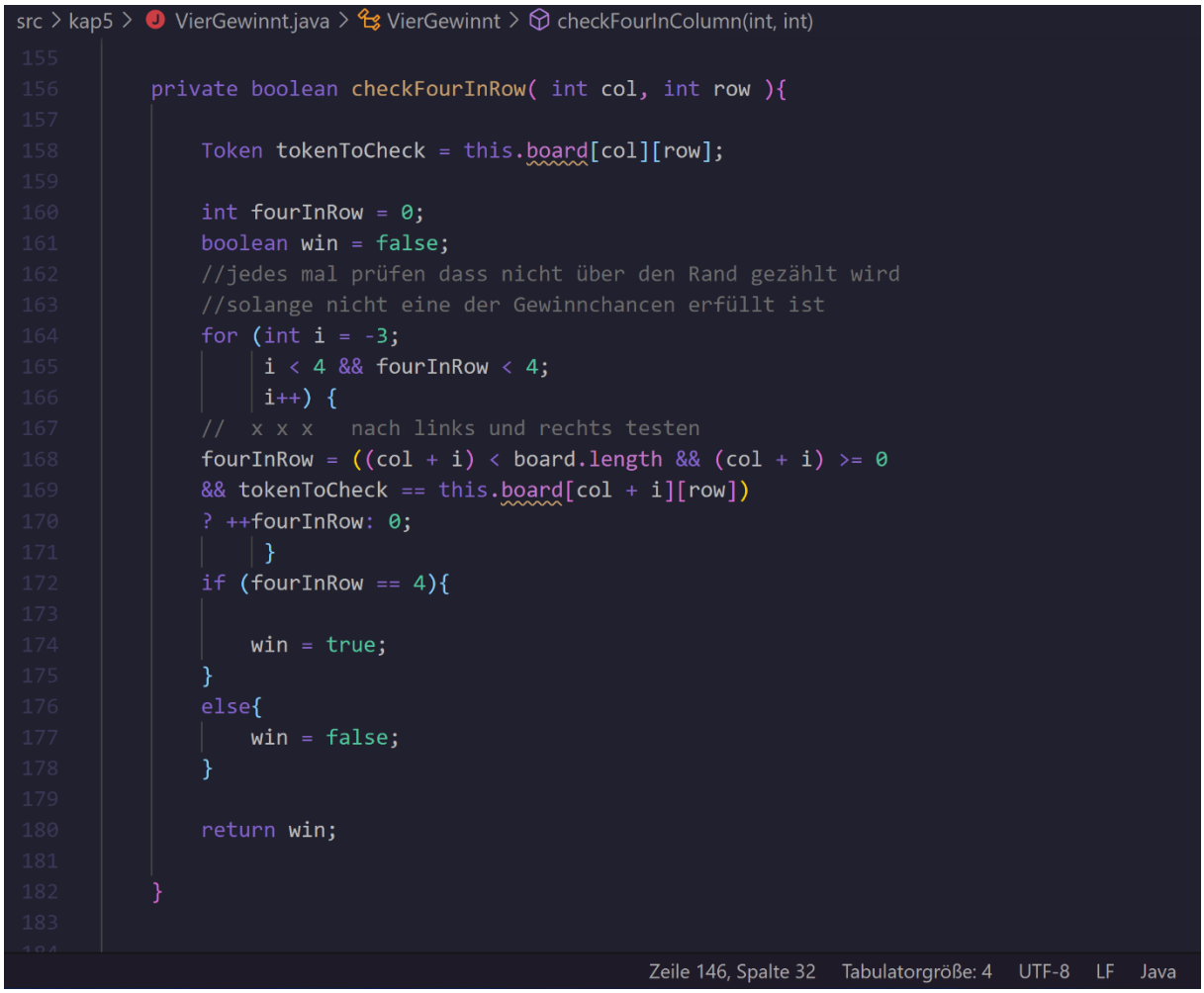


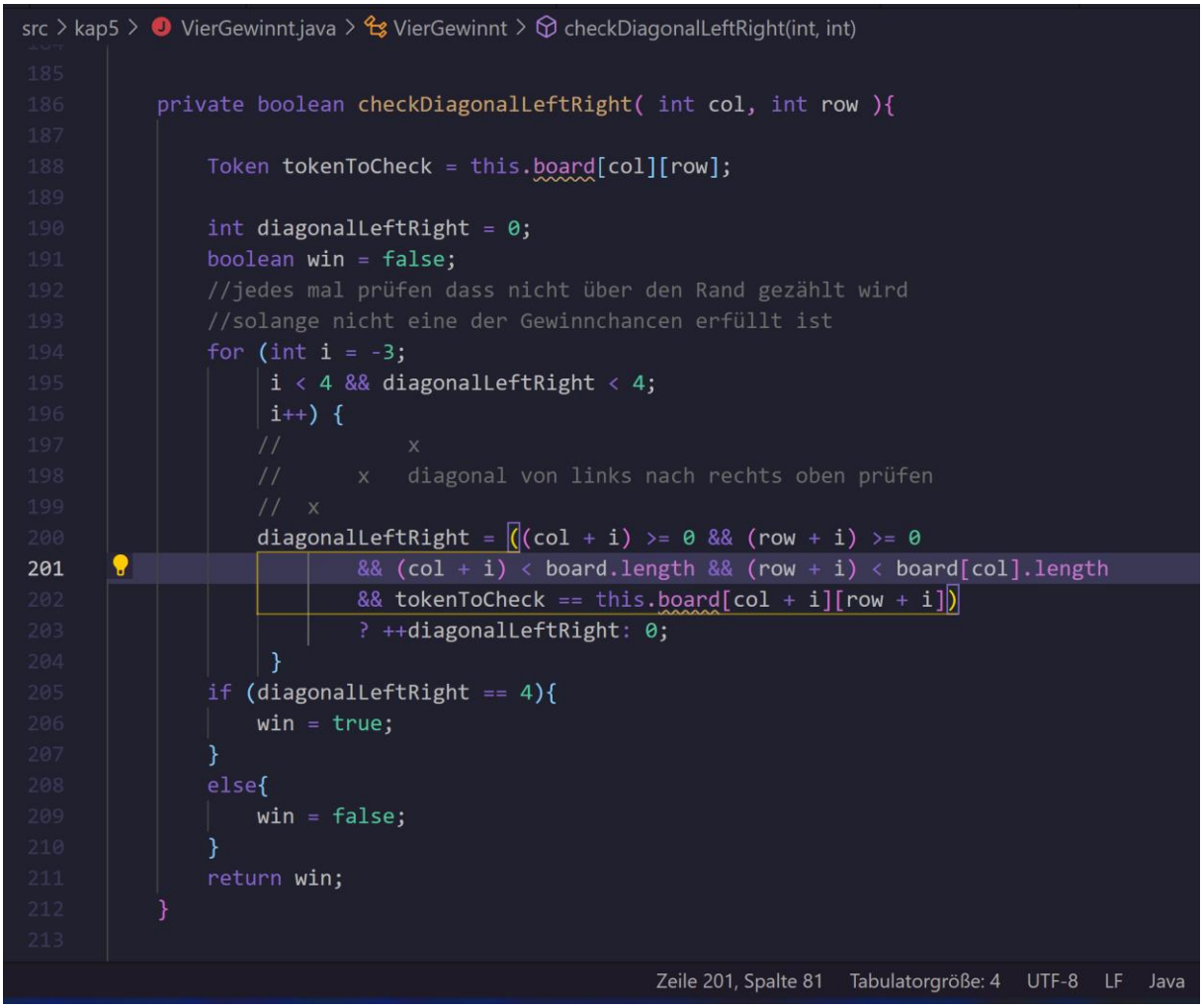
Implementationsaufgabe 1. c)

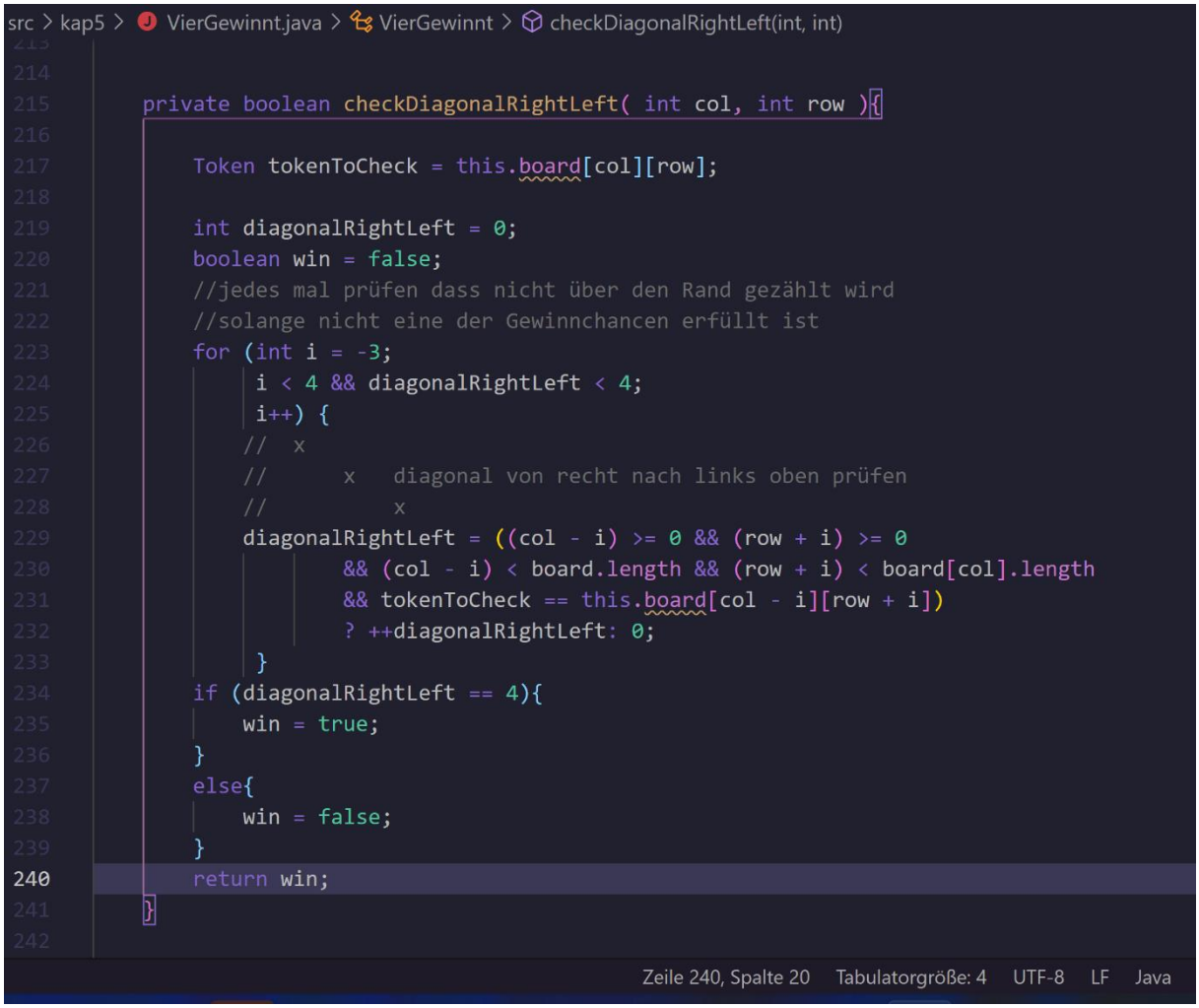
Die 4 Hilfsmethoden:



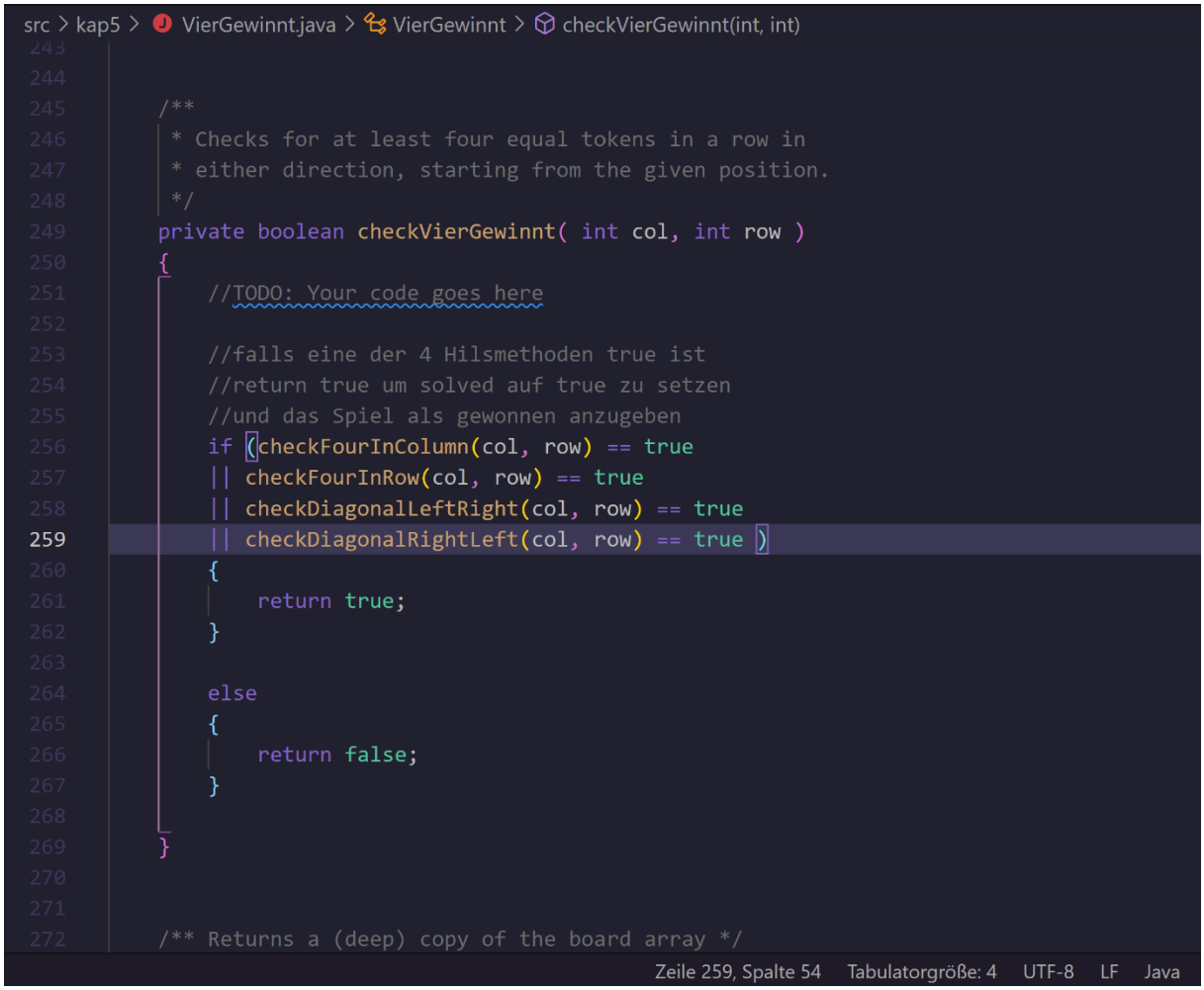

```java
private boolean checkFourInColumn( int col, int row ){

    Token tokenToCheck = this.board[col][row];

    int fourInColumn = 0;
    boolean win = false;

    //jedes mal prüfen dass nicht über den Rand gezählt wird
    //solange nicht eine der Gewinnchancen erfüllt ist
    for (int i = -3;
         i < 4 && fourInColumn < 4;
         i++) {
        //  x
        //  x nach oben und unten prüfen
        //  x
        fourInColumn = ((row + i) < board[col].length && (row + i) >= 0
        && tokenToCheck == this.board[col][row + i])
        ? ++fourInColumn: 0;
         }
    if (fourInColumn == 4){
        win = true;
    }
    else{
        win = false;
    }
    return win;
}
```


src > kap5 > VierGewinnt.java > VierGewinnt > checkFourInColumn(int, int)
155
156 private boolean checkFourInRow(int col, int row){
157
158 Token tokenToCheck = this.board[col][row];
159
160 int fourInRow = 0;
161 boolean win = false;
162 //jedes mal prüfen dass nicht über den Rand gezählt wird
163 //solange nicht eine der Gewinnchancen erfüllt ist
164 for (int i = -3;
165 i < 4 && fourInRow < 4;
166 i++) {
167 // x x x nach links und rechts testen
168 fourInRow = ((col + i) < board.length && (col + i) >= 0
169 && tokenToCheck == this.board[col + i][row])
170 ? ++fourInRow: 0;
171 }
172 if (fourInRow == 4){
173
174 win = true;
175 }
176 else{
177 win = false;
178 }
179
180 return win;
181
182 }
183
Zeile 146, Spalte 32 Tabulatorgröße: 4 UTF-8 LF Java


src > kap5 > VierGewinnt.java > VierGewinnt > checkDiagonalLeftRight(int, int)
185
186 private boolean checkDiagonalLeftRight(int col, int row){
187
188 Token tokenToCheck = this.board[col][row];
189
190 int diagonalLeftRight = 0;
191 boolean win = false;
192 //jedes mal prüfen dass nicht über den Rand gezählt wird
193 //solange nicht eine der Gewinnchancen erfüllt ist
194 for (int i = -3;
195 i < 4 && diagonalLeftRight < 4;
196 i++) {
197 // x
198 // x diagonal von links nach rechts oben prüfen
199 // x
200 diagonalLeftRight = ((col + i) >= 0 && (row + i) >= 0
201 && (col + i) < board.length && (row + i) < board[col].length
202 && tokenToCheck == this.board[col + i][row + i])
203 ? ++diagonalLeftRight: 0;
204 }
205 if (diagonalLeftRight == 4){
206 win = true;
207 }
208 else{
209 win = false;
210 }
211 return win;
212 }
213
Zeile 201, Spalte 81 Tabulatorgröße: 4 UTF-8 LF Java


src > kap5 > VierGewinnt.java > VierGewinnt > checkDiagonalRightLeft(int, int)
214
215 private boolean checkDiagonalRightLeft(int col, int row){
216
217 Token tokenToCheck = this.board[col][row];
218
219 int diagonalRightLeft = 0;
220 boolean win = false;
221 //jedes mal prüfen dass nicht über den Rand gezählt wird
222 //solange nicht eine der Gewinnchancen erfüllt ist
223 for (int i = -3;
224 i < 4 && diagonalRightLeft < 4;
225 i++) {
226 // x
227 // x diagonal von recht nach links oben prüfen
228 // x
229 diagonalRightLeft = ((col - i) >= 0 && (row + i) >= 0
230 && (col - i) < board.length && (row + i) < board[col].length
231 && tokenToCheck == this.board[col - i][row + i])
232 ? ++diagonalRightLeft: 0;
233 }
234 if (diagonalRightLeft == 4){
235 win = true;
236 }
237 else{
238 win = false;
239 }
240 return win;
241 }
242
Zeile 240, Spalte 20 Tabulatorgröße: 4 UTF-8 LF Java

Die Methode checkVierGewinnt:



In der Klasse ComputerPlayer ist bereits implementiert, dass der Bot keine ungültige Züge macht, hier haben wir also nichts verändert.

Implementationsaufgabe 2.a)

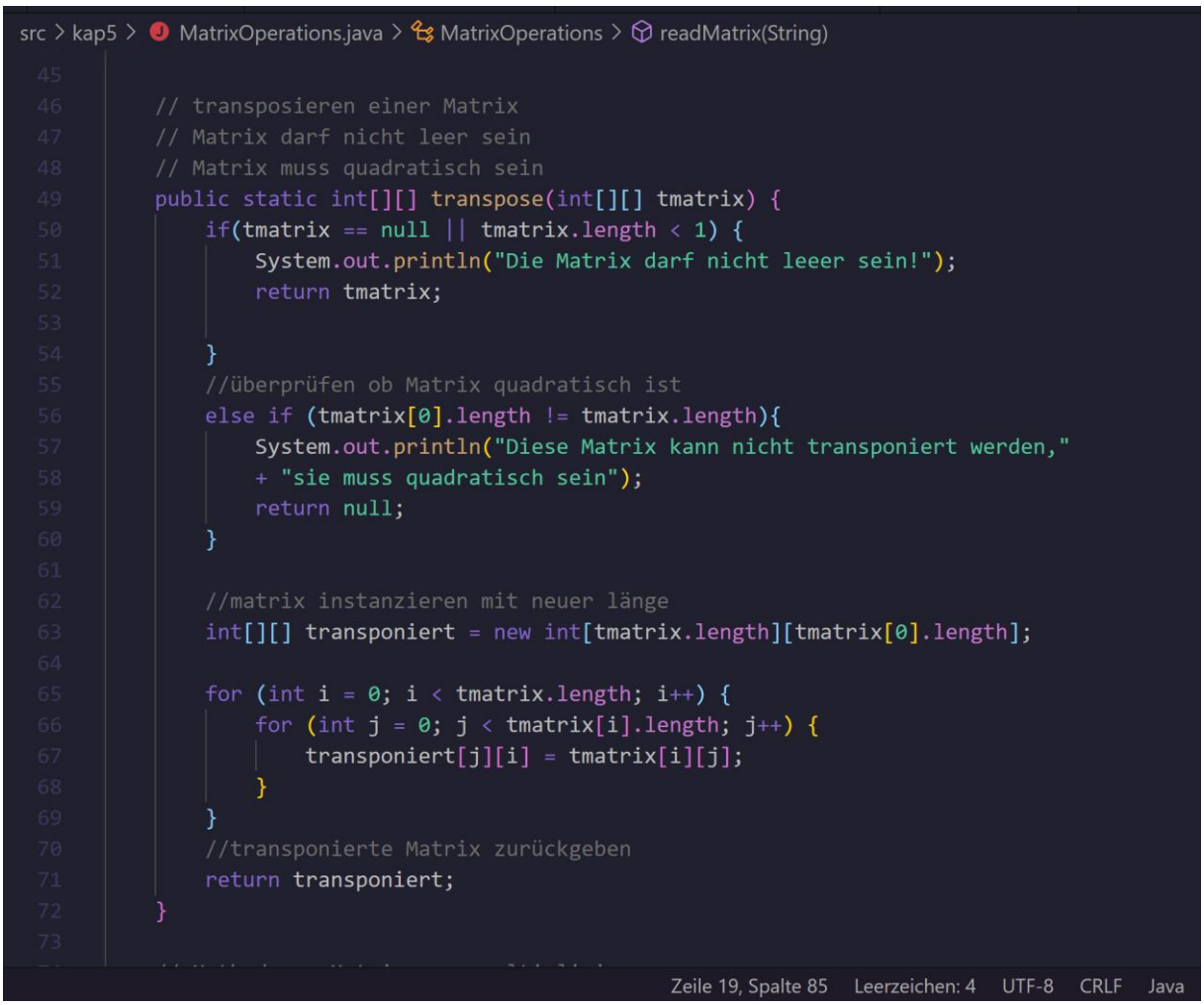
src > kap5 > MatrixOperations.java > MatrixOperations > readMatrix(String)

```java
public class MatrixOperations {

    private final static String PATH_MATRICES = "src/kap5/res/";
    // throws FileNotFoundException falls kein file für die Matrix angelgt wurde
    public static int[][] readMatrix(String fileName) throws FileNotFoundException {
        //ArrayList erstellen  und
        //Lines der Matrizen in Arrayliste einfügen
        //um die grösse zu erhalten
        Scanner scan = new Scanner(new File(PATH_MATRICES + fileName));
        ArrayList<String> matrixLines = new ArrayList<>();
        int[][] matrix = null;
        while (scan.hasNextLine()) {
            matrixLines.add(scan.nextLine());
        }
        scan.close();

        for (int i = 0; i < matrixLines.size(); i++) {
            String[] elements = matrixLines.get(i).split(" ");

            //instanzieren und alle Zahlen dem Array hinzufügen
            if (matrix == null)
                matrix = new int[matrixLines.size()][elements.length];
            for (int j = 0; j < elements.length; j++) {
                matrix[i][j] = Integer.parseInt(elements[j]);
            }
        }
        //Array der Matrix zurückgeben
        return matrix;
    }
```

Zeile 19, Spalte 85 Leerzeichen: 4 UTF-8 CRLF Java

```java
    //Methode um Matrix in einen String umzuwandeln und zurückgeben
    public static String matrixToString(int[][] matrix) {
        String matrixString = "";
        if (matrix == null)
            return "";

        for (int[] line: matrix) {
            for (int entry: line) {
                matrixString += entry + " ";
            }
            matrixString += "\n";
        }
        return matrixString;
    }
}
```

Aufgabe 2. B)


src > kap5 > MatrixOperations.java > MatrixOperations > readMatrix(String)
45
46 // transposieren einer Matrix
47 // Matrix darf nicht leer sein
48 // Matrix muss quadratisch sein
49 public static int[][] transpose(int[][] tmatrix) {
50 if(tmatrix == null || tmatrix.length < 1) {
51 System.out.println("Die Matrix darf nicht leeer sein!");
52 return tmatrix;
53
54 }
55 //überprüfen ob Matrix quadratisch ist
56 else if (tmatrix[0].length != tmatrix.length){
57 System.out.println("Diese Matrix kann nicht transponiert werden,"
58 + "sie muss quadratisch sein");
59 return null;
60 }
61
62 //matrix instanzieren mit neuer länge
63 int[][] transponiert = new int[tmatrix.length][tmatrix[0].length];
64
65 for (int i = 0; i < tmatrix.length; i++) {
66 for (int j = 0; j < tmatrix[i].length; j++) {
67 transponiert[j][i] = tmatrix[i][j];
68 }
69 }
70 //transponierte Matrix zurückgeben
71 return transponiert;
72 }
73
Zeile 19, Spalte 85 Leerzeichen: 4 UTF-8 CRLF Java

Implementationsaufgabe 2. C)

src > kap5 > MatrixOperations.java > MatrixOperations > readMatrix(String)

```
        }
        //transponierte Matrix zurückgeben
        return transponiert;
    }

    // Methode um Matrizen zu multiplizieren
    public static int[][] product(int[][] matrixA, int[][] matrixB) {
        int rowsA = matrixA.length, columnA = matrixA[0].length;
        int rowsB = matrixB.length, columnB = matrixB[0].length;
        int[][] resultMatrix = new int[rowsA][columnB];

        if (columnA != rowsB) {
            System.out.println("Die beiden Matrizen sind nicht kompatibel");
            return null;
        }

        for (int i = 0; i < resultMatrix.length; i ++) {
            for (int j = 0; j < resultMatrix[i].length; j ++) {
                int sum = 0;

                for (int k = 0; k < columnA; k++) {
                    sum += matrixA[i][k] * matrixB[k][j];
                }
                resultMatrix[i][j] = sum;
            }
        }
        return resultMatrix;
    }
```

Zeile 19, Spalte 85 Leerzeichen: 4 UTF-8 CRLF Java

Ergänzend zu Aufgabe 2 a, b, c)

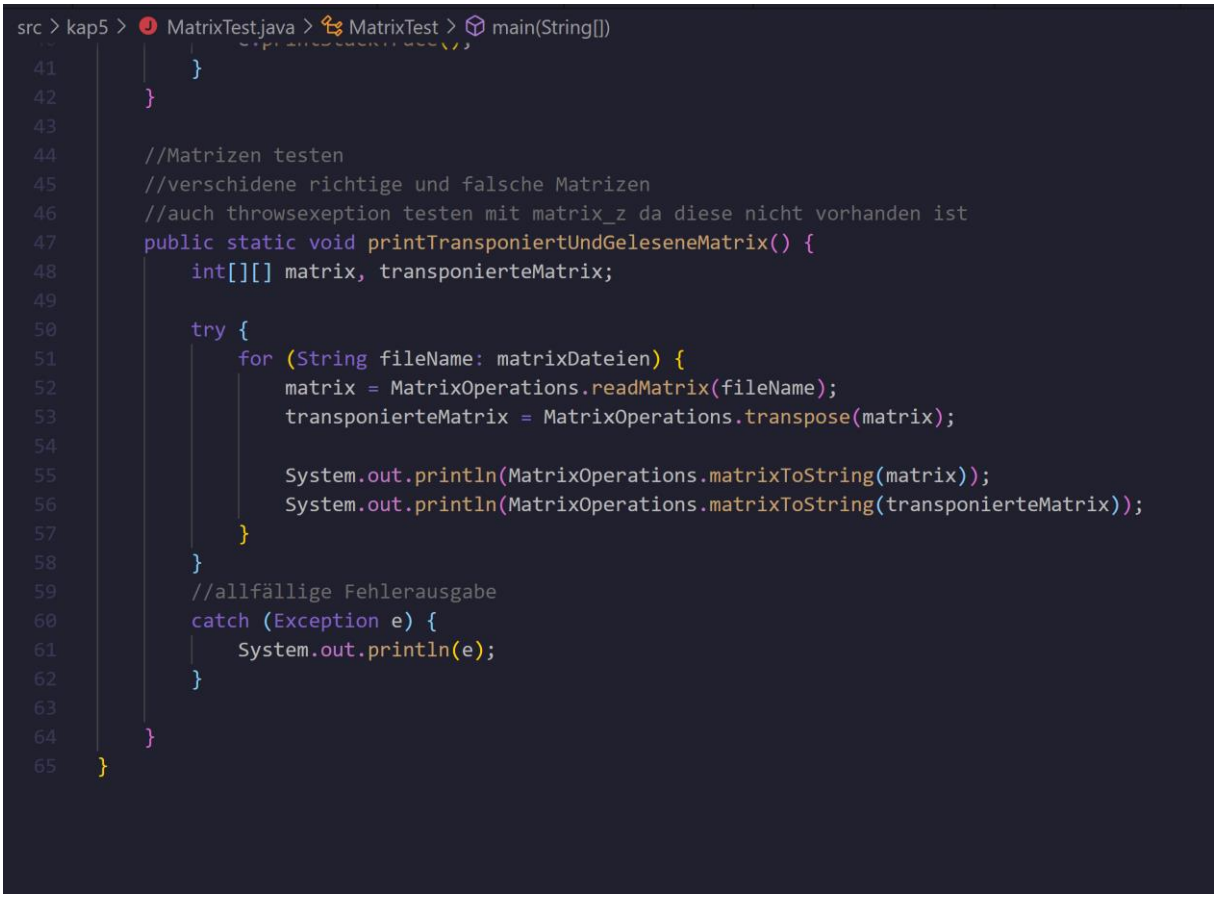
Die Klasse MatrixTest

src > kap5 > MatrixTest.java > MatrixTest > main(String[])

```java
/*
Jara Zihlmann(20-117-032)
Vithusan Ramalingam (21-105-515)
Jan Ellenberger (21-103-643)
*/
package kap5;

public class MatrixTest {

    public static String[] matrixDateien = {"matrix_a", "matrix_b",
    "matrix_c", "matrix_d", "matrix_e", "matrix_z",
    "matrix_product_a", "matrix_product_b"};

    Run | Debug
    public static void main(String[] args) {
        System.out.println("Gelesene und transponierte Matrizen:");
        //Test um gelesene Matrizen wenn möglich Transponiert ausgeben
        printTransponiertUndGeleseneMatrix();
        System.out.println("Test: Matrix Multipliziert");
        //Test multiplizierte Matrizen ausgeben
        printMultiplizierteMatrix();

    }

    //print der Multiplizierten Matrix
    public static void printMultiplizierteMatrix() {
        int[][] matrixA, matrixB, matrixProduct;

        //Matrizen multiplizieren und ausgeben
        try {
```

Zeile 16, Spalte 71 Leerzeichen: 4 UTF-8 CRLF Java


src > kap5 > MatrixTest.java > MatrixTest > main(String[])
printTransponiertUndGeleseneMatrix();
System.out.println("Test: Matrix Multipliziert");
//Test multiplizierte Matrizen ausgeben
printMultiplizierteMatrix();
}
//print der Multiplizierten Matrix
public static void printMultiplizierteMatrix() {
int[][] matrixA, matrixB, matrixProduct;
//Matrizen multiplizieren und ausgeben
try {
matrixA = MatrixOperations.readMatrix("matrix_product_a");
matrixB = MatrixOperations.readMatrix("matrix_product_b");
matrixProduct = MatrixOperations.product(matrixA, matrixB);
System.out.println(MatrixOperations.matrixToString(matrixA));
System.out.println(MatrixOperations.matrixToString(matrixB));
System.out.println(MatrixOperations.matrixToString(matrixProduct));
}
//sonst ausgabe des Fehlers
catch (Exception e) {
e.printStackTrace();
}
}
//Matrizen testen
//verschidene richtige und falsche Matrizen
//auch throwsexeption testen mit matrix z da diese nicht vorhanden ist
Zeile 16, Spalte 71 Leerzeichen: 4 UTF-8 CRLF Java



src > kap5 > MatrixTest.java > MatrixTest > main(String[])

```
41          }
42      }
43
44      //Matrizen testen
45      //verschidene richtige und falsche Matrizen
46      //auch throwsexeption testen mit matrix_z da diese nicht vorhanden ist
47      public static void printTransponiertUndGeleseneMatrix() {
48          int[][] matrix, transponierteMatrix;
49
50          try {
51              for (String fileName: matrixDateien) {
52                  matrix = MatrixOperations.readMatrix(fileName);
53                  transponierteMatrix = MatrixOperations.transpose(matrix);
54
55                  System.out.println(MatrixOperations.matrixToString(matrix));
56                  System.out.println(MatrixOperations.matrixToString(transponierteMatrix));
57              }
58          }
59          //allfällige Fehlerausgabe
60          catch (Exception e) {
61              System.out.println(e);
62          }
63
64      }
65  }
```