

# Aufgaben Serie 05 - Programmieren 1

Jara Zihlmann(20-117-032)

Vithusan Ramalingam (21-105-515)

Jan Ellenberger (21-103-643)

## Implementationsaufgabe 1. A)

```
src > kap5 > VierGewinnt.java > VierGewinnt > insertToken(int, Token)
80  */
81  private int insertToken( int column, Token tok )
82  {
83      //TODO: Your code goes here
84
85      //falls die Reihe ausserhalb des Boards ist
86      //bricht das Programm ab
87      if(column >= board.length || column < 0
88         || board[column][board[0].length - 1] != Token.empty) {
89          System.exit(1);
90      }
91
92      //token in Reihe einfügen solange es im Feld ist
93      int row = -1;
94      for (int i = 0; i < board[column].length; i++) {
95          if (board[column][i].equals(Token.empty)) {
96              row = i;
97              break;
98          }
99      }
100     board[column][row] = tok;
101     return row;
102
103 }
104
105
106 /**
107  * Checks if every position is occupied
108  * @returns true, iff the board is full.
```

**Implementationsaufgabe 1. b)**

```
src > kap5 > VierGewinnt.java > VierGewinnt > insertToken(int, Token)
103     }
104
105
106     /**
107      * Checks if every position is occupied
108      * @returns true, iff the board is full.
109      */
110     private boolean isBoardFull()
111     {
112         //TODO: Your code goes here
113
114         //überprüfen ob noch eine Position frei ist, sonst
115         //isBoardFull = true setzen
116         for (int i = 0; i < COLS; i++) {
117             if (board[i][ROWS - 1].equals(Token.empty)) {
118                 return false;
119             }
120         }
121         return true;
122     }
123
124
125
126     /**
127      * Checks for at least four equal tokens in a row in
128      * either direction, starting from the given position.
129      */
130     private boolean checkVierGewinnt( int col, int row )
131     {
132         //TODO: Your code goes here
```

Zeile 79, Spalte 44    Tabulatorgröße: 4    UTF-8    LF    Java

**Implementationsaufgabe 1. c)****Die 4 Hilfsmethoden:**

```
src > kap5 > VierGewinnt.java > VierGewinnt > checkFourInColumn(int, int)
125
126
127 private boolean checkFourInColumn( int col, int row ){
128
129     Token tokenToCheck = this.board[col][row];
130
131     int fourInColumn = 0;
132     boolean win = false;
133
134     //jedes mal prüfen dass nicht über den Rand gezählt wird
135     //solange nicht eine der Gewinnchancen erfüllt ist
136     for (int i = -3;
137         i < 4 && fourInColumn < 4;
138         i++) {
139         // x
140         // x nach oben und unten prüfen
141         // x
142         fourInColumn = ((row + i) < board[col].length && (row + i) >= 0
143             && tokenToCheck == this.board[col][row + i])
144             ? ++fourInColumn: 0;
145     }
146     if (fourInColumn == 4){
147         win = true;
148     }
149     else{
150         win = false;
151     }
152     return win;
153 }
154
```

Zeile 146, Spalte 32    Tabulatorgröße: 4    UTF-8    LF    Java

```
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
184
```

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)
213
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:**

```
src > kap5 > VierGewinnt.java > VierGewinnt > checkVierGewinnt(int, int)
243
244
245     /**
246     * Checks for at least four equal tokens in a row in
247     * either direction, starting from the given position.
248     */
249     private boolean checkVierGewinnt( int col, int row )
250     {
251         //TODO: Your code goes here
252
253         //falls eine der 4 Hilsmethoden true ist
254         //return true um solved auf true zu setzen
255         //und das Spiel als gewonnen anzugeben
256         if (checkFourInColumn(col, row) == true
257             || checkFourInRow(col, row) == true
258             || checkDiagonalLeftRight(col, row) == true
259             || checkDiagonalRightLeft(col, row) == true )
260         {
261             return true;
262         }
263
264         else
265         {
266             return false;
267         }
268     }
269
270
271
272     /** Returns a (deep) copy of the board array */
```

Zeile 259, Spalte 54    Tabulatorgröße: 4    UTF-8    LF    Java

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)
15 public class MatrixOperations {
16
17     private final static String PATH_MATRICES = "src/kap5/res/";
18     // throws FileNotFoundException falls kein file für die Matrix angelgt wurde
19     public static int[][] readMatrix(String fileName) throws FileNotFoundException {
20         //ArrayList erstellen und
21         //Lines der Matrizen in Arrayliste einfügen
22         //um die grösse zu erhalten
23         Scanner scan = new Scanner(new File(PATH_MATRICES + fileName));
24         ArrayList<String> matrixLines = new ArrayList<>();
25         int[][] matrix = null;
26         while (scan.hasNextLine()) {
27             matrixLines.add(scan.nextLine());
28         }
29         scan.close();
30
31         for (int i = 0; i < matrixLines.size(); i++) {
32             String[] elements = matrixLines.get(i).split(" ");
33
34             //instancieren und alle Zahlen dem Array hinzufügen
35             if (matrix == null)
36                 matrix = new int[matrixLines.size()][elements.length];
37             for (int j = 0; j < elements.length; j++) {
38                 matrix[i][j] = Integer.parseInt(elements[j]);
39             }
40         }
41         //Array der Matrix zurückgeben
42         return matrix;
43     }
44 }
```

Zeile 19, Spalte 85   Leerzeichen: 4   UTF-8   CRLF   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 // transponieren 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 leer 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)
69     }
70     //transponierte Matrix zurückgeben
71     return transponiert;
72 }
73
74 // Methode um Matrizen zu multiplizieren
75 public static int[][] product(int[][] matrixA, int[][] matrixB) {
76     int rowsA = matrixA.length, columnA = matrixA[0].length;
77     int rowsB = matrixB.length, columnB = matrixB[0].length;
78     int[][] resultMatrix = new int[rowsA][columnB];
79
80     if (columnA != rowsB) {
81         System.out.println("Die beiden Matrizen sind nicht kompatibel");
82         return null;
83     }
84
85     for (int i = 0; i < resultMatrix.length; i++) {
86         for (int j = 0; j < resultMatrix[i].length; j++) {
87             int sum = 0;
88
89             for (int k = 0; k < columnA; k++) {
90                 sum += matrixA[i][k] * matrixB[k][j];
91             }
92             resultMatrix[i][j] = sum;
93         }
94     }
95     return resultMatrix;
96 }
97
98
```

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[])
1  /*
2   Jara Zihlmann(20-117-032)
3   Vithusan Ramalingam (21-105-515)
4   Jan Ellenberger (21-103-643)
5   */
6   package kap5;
7
8   public class MatrixTest {
9
10      public static String[] matrixDateien = {"matrix_a", "matrix_b",
11      "matrix_c", "matrix_d", "matrix_e", "matrix_z",
12      "matrix_product_a", "matrix_product_b"};
13
14      public static void main(String[] args) {
15          System.out.println("Gelesene und transponierte Matrizen:");
16          //Test um gelesene Matrizen wenn möglich Transponiert ausgeben
17          printTransponiertUndGeleseneMatrix();
18          System.out.println("Test: Matrix Multipliziert");
19          //Test multiplizierte Matrizen ausgeben
20          printMultiplizierteMatrix();
21      }
22
23      //print der Multiplizierten Matrix
24      public static void printMultiplizierteMatrix() {
25          int[][] matrixA, matrixB, matrixProduct;
26
27          //Matrizen multiplizieren und ausgeben
28          try {
29              // ...

```

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

```
src > kap5 > MatrixTest.java > MatrixTest > main(String[])
17     printTransponiertUndGeleseneMatrix();
18     System.out.println("Test: Matrix Multipliziert");
19     //Test multiplizierte Matrizen ausgeben
20     printMultiplizierteMatrix();
21
22 }
23
24 //print der Multiplizierten Matrix
25 public static void printMultiplizierteMatrix() {
26     int[][] matrixA, matrixB, matrixProduct;
27
28     //Matrizen multiplizieren und ausgeben
29     try {
30         matrixA = MatrixOperations.readMatrix("matrix_product_a");
31         matrixB = MatrixOperations.readMatrix("matrix_product_b");
32         matrixProduct = MatrixOperations.product(matrixA, matrixB);
33
34         System.out.println(MatrixOperations.matrixToString(matrixA));
35         System.out.println(MatrixOperations.matrixToString(matrixB));
36         System.out.println(MatrixOperations.matrixToString(matrixProduct));
37     }
38     //sonst ausgabe des Fehlers
39     catch (Exception e) {
40         e.printStackTrace();
41     }
42 }
43
44 //Matrizen testen
45 //verschiedene richtige und falsche Matrizen
46 //auch throwsexention 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 //verschiedene 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 }
```