Programmieren und Software-Engineering I
Übung 26

Name: _____ Klasse: _____ Datum: _____

Lernziele:

- Methoden
- Zweidimensionale Arrays

Aufgabe 1: Matrix füllen (Ex_26_01_FillMatrix)
Erstelle ein Java-Programm, welches eine beliebige rechteckige (NxM) Matrix auf verschiedene Arten (aufsteigend von 1 bis NxM) füllt und danach ausgibt.

Beispiel: **Art 1:** (zeilenweise von links nach rechts)

```
1   2   3   4   5
6   7   8   …
```

Art 2: (spaltenweise von oben nach unten)

```
1   4   7   …
2   5
3   6
```

Art 3: (zeilenweise von rechts nach links)

```
5   4   3   2   1
10  9   8   7   6
…
```

Art 4: (spaltenweise von unten nach oben)

```
6   12
5   11
4   …
3   …
2   8   …
1   7   13
```

Verwende **entsprechende Methoden**!!!!

Beispiel:

```
…
fillMatrix(ma, 1);
printMatrix(ma);

fillMatrix(ma, 2);
printMatrix(ma);
…
```

Aufgabe 2: Tabelle bereinigen (Ex_26_02_Table)
Erstelle ein Programm, welches ein 2-dimensionales Array (Tabelle) der Größe 10 X 20 mit Zufallszahlen zwischen 0 und 99 initialisiert und gib anschließend diese Tabelle in übersichtlicher Form aus.
Anschließend „lösche“ (= mit -1 gekennzeichneten Feldinhalt) alle Zahlen, in denen die Zeilennummer (als Ziffer gesehen) nicht vorkommt. Gib ebenfalls zur Kontrolle das Array wieder in übersichtlicher Form aus!

Beispiel:

```
Ausgabe der Tabelle nach dem Initialisieren!
Ausgabe Tabelle:
Zeile 1: 73 96 67 25 80  7 82 25 59 26 67 20 52 78 65 77 16 21 81 71
Zeile 2: 58 45 97 86 23 84 40 59 26  6 50 17 35 56 34 60 96 62  8 40
Zeile 3:  7 68 95 64 67 56  4 68 44 44 14 70 21 59 56 28 98 79  1 95
Zeile 4: 65 23 97 52 74 49 84 95 43  4 80 52 12 40 63 88 48 32 19 66
Zeile 5: 79 71 15 81 91 79 94 74 98  6 61 16 61 12 74 83 70 42 89 17
Zeile 6: 71  8 25 39 62 60 58 80 94 67 78 65 60 34 16 93  8 16 60 94
Zeile 7: 54  9 95 36 42 63 90 98 65 79 18  5 42  1 93 70 45 75 18 30
Zeile 8: 20 14 79 30 27 94 54 26 95 79 75 53 81 93 61 62 11 87 95 98
Zeile 9: 69 58 15 14 49 72 23 58 40 44 47 31 96 30 73 39 28 64  9 47
Zeile 10:  9 63  3 84  4 63 23 47 38 30 59 57 78 19 59 60 98 65 80 53

Ausgabe der Tabelle nach dem Löschen!
Ausgabe Tabelle:
Zeile 1: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 16 21 81 71
Zeile 2: -1 -1 -1 -1 23 -1 -1 -1 26 -1 -1 -1 -1 -1 -1 -1 62 -1 -1
Zeile 3: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Zeile 4: -1 -1 -1 -1 74 49 84 -1 43  4 -1 -1 -1 40 -1 -1 48 -1 -1 -1
Zeile 5: -1 -1 15 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Zeile 6: -1 -1 -1 -1 62 60 -1 -1 -1 67 -1 65 60 -1 16 -1 -1 16 60 -1
Zeile 7: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 79 -1 -1 -1 -1 -1 70 -1 75 -1 -1
Zeile 8: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 81 -1 -1 -1 -1 87 -1 98
Zeile 9: 69 -1 -1 -1 49 -1 -1 -1 -1 -1 -1 -1 96 -1 -1 39 -1 -1  9 -1
Zeile 10: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Programmieren und Software-Engineering I
Übung 26

Aufgabe 3: Matrix (HUE_26_03_Matrix)
Implementiere für eine rechteckige Matrix (zB 4x6) zumindest nachfolgende Methoden:

```
//Matrix mit ZZ befüllen
void fillMatrix(int[][] ma, int ug, int og)

//Matrix zeilenweise ausgeben
void printMatrix(int[][] ma)
```

Ausgabebeispiel:
```
43 46 32 31 43 1
37 28 23 28 3  48
44 39 50 6  23 9
12 2  24 16 1  11
```

```
//Zeilensumme berechnen
int getRowSum(int[][] ma, int row)

//Spaltensumme berechnen
int getColSum(int[][] ma, int col)
```

```
Die 0.Zeilensumme ergibt:  196
Die 1.Zeilensumme ergibt:  167
Die 2.Zeilensumme ergibt:  171
Die 3.Zeilensumme ergibt:  66

Die 0.Spaltensumme ergibt: 136
Die 1.Spaltensumme ergibt: 115
Die 2.Spaltensumme ergibt: 129
Die 3.Spaltensumme ergibt: 81
Die 4.Spaltensumme ergibt: 70
Die 5.Spaltensumme ergibt: 69

Minimale Zeilensumme :    Zeile 3 mit 66
Maximale Zeilensumme :    Zeile 0 mit 196
Minimale Spaltensumme :   Spalte 5 mit 69
Maximale Spaltensumme :   Spalte 0 mit 136
```

```
//Gesamtsumme berechnen
int getTotalSum(int[][] ma)

//Durchschnitt berechnen
double getAverage(int[][] ma)

//Minimum ermitteln
int getMinimum(int[][] ma)

//Maximum ermitteln
int getMaximum(int[][] ma)
```

```
Gesamtsumme:              600
Durchschnitt:             25.0
Maximum:                  50
Minimum:                  1
```

```
//Häufigste Zahl ermitteln
int getMostFrequentNum(int[][] ma, int ug,
int og)

//Anzahl der häufigsten Zahl ermitteln
int getMostFrequentCount(int[][] ma, int ug, int og)

//Zahlen ermitteln, die nicht in der Liste vorkommen
int[] getNumNotInMatrix(int[][] ma, int ug, int og)
```

```
Haeufigste Zahl :          1 (2x)
Nicht enthalten :          4, 5, 7, 8,…
```

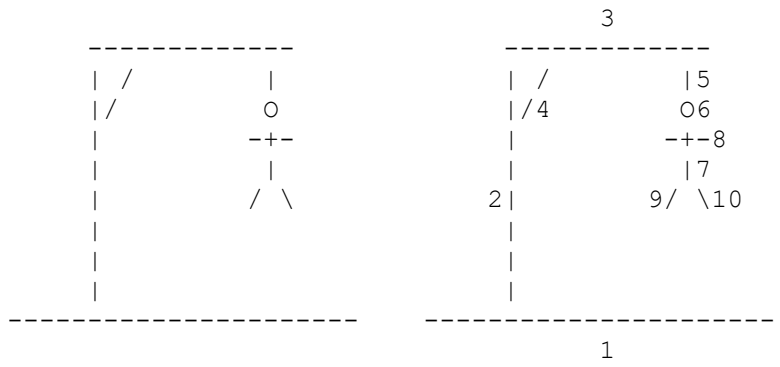
Aufgabe 4: Hangman (HUE_26_04_Hangman)
Beim Spiel **Hangman** geht es darum, einen **Begriff** mit möglichst **wenig Versuchen** zu erraten. (Dieser Begriff kann entweder aus einem fixen Array kommen bzw. wird vom Mitspieler eingegeben). Zu Beginn steht eine Kette von Strichen, die so viele Elemente enthält wie das Wort lang ist. Der Spieler gibt dabei einen beliebigen Buchstaben des Alphabets ein (in der Hoffnung, dass er in dem gesuchten Wort vorkommt).
Falls der Buchstabe vorkommt, wird er über jeden Strich geschrieben, an der der Buchstabe im Wort vorkommt. Falls der Buchstabe vorkommt wird ein Galgen mit einem Strichmännchen um ein weiteres von insgesamt 10 Elementen vervollständigt.

Der **Galgen** besteht aus folgenden **10 Elementen**:

1) Der Grundlinie (dem Galgenhügel)
2) Dem Stamm
3) Dem Querträger
4) Einer zusätzlichen Versteifung
5) Dem Strick
6) Dem Kopf des Gegners
7) Seinem Körper
8) Seinen Armen
9) Seinem linken Bein
10) Seinem rechten Bein

Der **Galgen** kann leicht mit normalen **ASCII-Zeichen** dargestellt werden, wie folgende Abbildung zeigt (rechts mit nummerierten Elementen):

Programmieren und Software-Engineering I
Übung 26



WICHTIG
Alle Programme müssen einen Programmkopf (=Beschreibung) enthalten.
z.B.:

```
/**************************************************************************
*       Name:           Max Mustermann
*       Hü:             4
*       Bsp:            2
*       Datum:          20.10.2015
*       Dateiname:      HUE_04_02_Dreieck.java
*       Beschreibung:   Es wird für gegebene Seitenlängen a, b und c geprüft, ob es ein gleich…
**************************************************************************/
```