Eindimensionale Arrays

Die Arrays, die wir bisher behandelt haben, nennt man „**ein**dimensional“.
Das bedeutet, es gibt **einen** Index, und das Array lässt sich als Liste darstellen:

Index	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Wert	99	–15	11	123	409	–32	–65	0	4	315

Zweidimensionale Arrays

Das Konzept von Arrays lässt sich verallgemeinern zu Tabellen. Um sich das vorstellen zu können, betrachten wir ein Schachprogramm. In diesem Programm könnte man jede Stellung von Figuren auf einem Schachbrett als Tabelle von Zahlen implementieren, wobei jede Zahl eine Figur codiert:

0 = leeres Feld

1 = Bauer (weiß)	11 = Bauer (schwarz)
2 = Springer (weiß)	12 = Springer (schwarz)
3 = Läufer (weiß)	13 = Läufer (schwarz)
4 = Turm (weiß)	14 = Turm (schwarz)
5 = Dame (weiß)	15 = Dame (schwarz)
6 = König (weiß)	16 = König (schwarz)

Die Anfangsaufstellung würde dann folgendermaßen codiert:

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	14	12	13	15	16	13	12	14
1	11	11	11	11	11	11	11	11
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	1	1	1	1	1	1	1	1
7	4	2	3	5	6	3	2	4

Wie beim eindimensionalen Array kann man auf die Elemente über Indizes zugreifen.
Es braucht beim **zwei**dimensionalen Array allerdings **zwei Indizes**:

Ein Index für die **Zeile**, ein Index für die **Spalte**.

Auf das Element links oben in der Ecke würde man z.B. über schachbrett[0][0] zugreifen, der schwarze König stände momentan auf schachbrett[4][0].

Hinweis: Die Zuordnung, dass der erste Index die Zeile und der zweite die Spalte angibt ist willkürlich und kann auch umgekehrt festgelegt werden. Es sollte daher im Programm durch Kommentare deutlich gemacht werden, wofür der erste bzw. der zweite Index steht. Innerhalb des Programms darf die Zuordnung nicht wechseln.

Zweidimenionale Arrays in Java

```
public class Schachspiel
{
   // Deklaration 2D-Array.  Erster Index x, zweiter Index y
   int[][] schachbrett;

   public Schachspiel()
   {
      // Erzeugung das 2D-Array
      schachbrett = new int[8][8];
   }

   // setzt die Figuren auf die Grundstellung (ohne Bauern)
   public void stelleFiguren()
   {
      schachbrett[0][0] = 14;   // schwarzer Turm
      schachbrett[1][0] = 12;   // schwarzer Läufer
      ...
   }

   // setzt alle Bauern auf die Grundstellung (in Zeile 1 bzw. 6)
   public void stelleBauern()
   {
      int x;
      for (x = 0; x < 8; x++)
      {
         schachbrett[x][1] = 11;
         schachbrett[x][6] =  1;
      }
   }

   // räumt das ganze Brett leer
   public void leereBrett()
   {
      int x, y;
      for (y = 0; y < 8; y++)
      {
         for (x = 0; x < 8; x++)
         {
            schachbrett[x][y] = 0;
         }
      }
   }
}
```

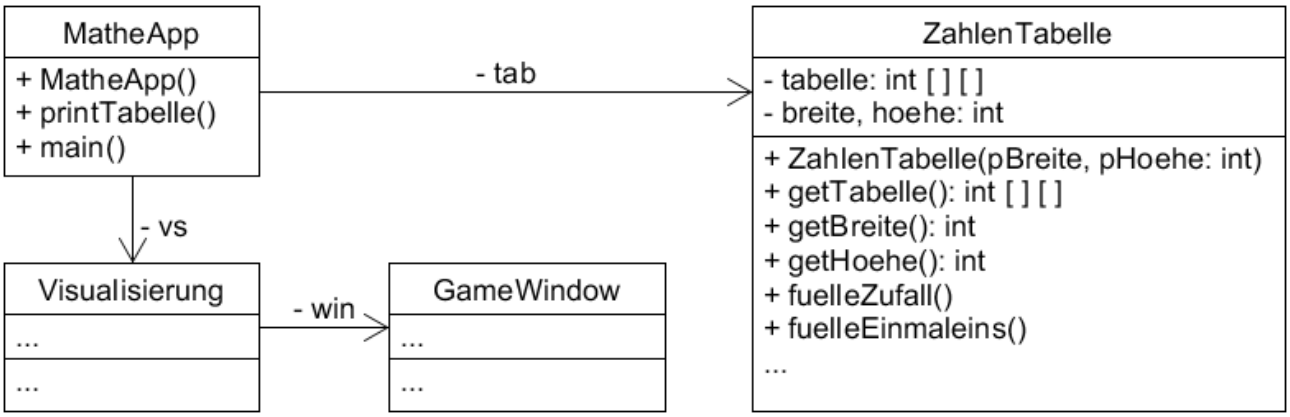
Aufgabe

Hinweis: Die Aufgaben haben keinen Bezug zum Schachspiel.

Nutze die ausgeteilte BlueJ-Vorlage. Die Klasse Visualisierung ist bereits implementiert.

Deine Aufgabe ist, die **Methoden der Klasse ZahlenTabelle** zu implementieren, und diese dann (entsprechend der Vorlage) in die **main-Methode** einzubauen.

Die Klasse ZahlenTabelle enthält ein 2D-Array mit 10 mal 10 Integer-Werten. Die Werte sollten im Bereich 0 – 99 liegen. Die Klasse Visualisierung zeichnet dann ein Muster aus 100 farbigen Quadraten entsprechend dieser Zahlen. 0 entspricht der Farbe schwarz, 99 der Farbe Magenta, und die Zahlen dazwischen den Tönen zwischen Schwarz und Magenta.



a) Die Methode fuelleLinear() füllt die erste Zeile mit 0, die zweite mit 11, die dritte mit 22 usw.

b) Die Methode fuelleZufall() füllt das 2D-Array mit Zufallszahlen zwischen 0 und 99.

c) Die Methode fuelleEinmaleins füllt das 2D-Array wird mit dem kleinen Einmaleins.
Die erste Zeile enthält also die Einer-Reihe 1, 2, 3 usw. bis 10.
Die zweite Zeile dann die Zweier-Reihe 2, 4, 6 usw. bis 20, und so weiter.

d) Weitere Methoden füllen das 2D-Array mit Mustern nach den folgenden Vorlagen (hier nur für 5x5-Tabelle). Das zweite Muster ist schon relativ schwierig…

```
10 20 30 20 10    01 02 03 04 05
20 30 40 30 20    16 17 18 19 06
30 40 50 40 30    15 24 25 20 07
20 30 40 30 20    14 23 22 21 08
10 20 30 20 10    13 12 11 10 09
```

e) Denke dir selbst weitere Füllmuster aus.