Wegberechnung im Gitter

Betrachten Sie das folgende Gitter mit $m+1$ Zeilen und $n+1$ Spalten ($m \geq 1$ und $n \geq 1$): [1] geeksforgeeks [2]

Angenommen, Sie befinden sich zu Beginn am Punkt $(0,0)$ und wollen zum Punkt (m,n).

Für die Anzahl $A(i,j)$ aller verschiedenen Wege vom Punkt $(0,0)$ zum Punkt (i,j) lassen sich folgende drei Fälle unterscheiden (es geht jeweils um die kürzesten Wege ohne Umweg!):

- $1 \leq i \leq m$ und $j = 0$:

  Es gibt genau einen Weg von $(0,0)$ nach $(i,0)$ für $1 \leq i \leq m$.

- $i = 0$ und $1 \leq j \leq n$:

  Es gibt genau einen Weg von $(0,0)$ nach $(0,j)$ für $1 \leq j \leq n$.

- $1 \leq i \leq m$ und $1 \leq j \leq n$:

  auf dem Weg zu (i,j) muss als vorletzter Punkt entweder $(i-1,j)$ oder $(i,j-1)$ besucht worden sein.

Daraus ergibt sich folgende Rekursionsgleichung:

$$A(i,j) = \begin{cases} 1 & \text{falls } (1 \leq i \leq m \text{ und } j = 0) \text{ oder } (i = 0 \text{ und } 1 \leq j \leq n) \\ A(i-1,j) + A(i,j-1) & \text{falls } 1 \leq i \leq m \text{ und } 1 \leq j \leq n \end{cases}$$

Implementieren Sie die Java-Klasse `Gitter` mit der Methode

```
public int berechneAnzahlWege(),
```

die ausgehend von der Rekursionsgleichung durch dynamische Programmierung die Anzahl aller Wege vom Punkt $(0,0)$ zum Punkt (m,n) berechnet. Die Überprüfung, ob $m \leq 1$ und $n \leq 1$ gilt, können Sie der Einfachheit halber weglassen.

```
32
33    public int berechneAnzahlWege() {
34      int i, j;
35      for (i = 1; i <= m; i++) {
36        anzahlWege[i][0] = 1;
37      }
38      for (j = 1; j <= n; j++) {
39        anzahlWege[0][j] = 1;
40      }
41      for (i = 1; i <= m; i++) {
42        for (j = 1; j <= n; j++) {
43          anzahlWege[i][j] = anzahlWege[i - 1][j] + anzahlWege[i][j - 1];
44        }
45      }
```

[1] Quelle möglicherweise von `https://www.yumpu.com/de/document/read/17936760/ubungen-zum-prasenzmodul-algorithmen-und-datenstrukturen`

[2] `https://www.geeksforgeeks.org/count-possible-paths-top-left-bottom-right-nxm-matrix/`

```
46      return anzahlWege[m][n];
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_3/Gitter.java`

Die komplette Java-Klasse

```
3   import org.bschlangaul.helfer.Farbe;
4   import org.bschlangaul.helfer.Konsole;
5
6   /**
7    * <a href="https://www.studon.fau.de/file2521908_download.html">Angabe:
     ↪  AB_3 Greedy_DP_Backtracking.pdf</a>
8    * <a href="https://www.studon.fau.de/file2521907_download.html">Lösung:
     ↪  AB_3 Greedy_DP_Backtracking_Lsg.pdf</a>
9    */
10  public class Gitter {
11
12    /**
13     * m + 1: Anzahl der Zeilen
14     */
15    private int m;
16
17    /**
18     * n + 1: Anzahl der Spalten
19     */
20    private int n;
21
22    /**
23     * anzahlWege[i][j]: Anzahl der Wege vom Punkt (0,0) zum Punkt (i,j)
24     */
25    private int anzahlWege[][];
26
27    public Gitter(int m, int n) {
28      this.m = m;
29      this.n = n;
30      anzahlWege = new int[m + 1][n + 1];
31    }
32
33    public int berechneAnzahlWege() {
34      int i, j;
35      for (i = 1; i <= m; i++) {
36        anzahlWege[i][0] = 1;
37      }
38      for (j = 1; j <= n; j++) {
39        anzahlWege[0][j] = 1;
40      }
41      for (i = 1; i <= m; i++) {
42        for (j = 1; j <= n; j++) {
43          anzahlWege[i][j] = anzahlWege[i - 1][j] + anzahlWege[i][j - 1];
44        }
45      }
46      return anzahlWege[m][n];
47    }
48
49    /**
50     * Zeige die Lösung in der Konsole.
51     */
52    public void zeigeLoesung() {
```

```
53      System.out.println(String.format("Anzahl der Wege von %sx%s: %s",
        ↪  Farbe.gelb(m), Farbe.gelb(n), Farbe.grün(berechneAnzahlWege())));
54      System.out.println(Farbe.rot("Gitter:"));
55      Konsole.zeige2DIntFeld(anzahlWege);
56      System.out.println();
57    }
58
59    public static void main(String args[]) {
60      new Gitter(2, 2).zeigeLoesung();
61      new Gitter(3, 3).zeigeLoesung();
62      new Gitter(4, 4).zeigeLoesung();
63      new Gitter(5, 5).zeigeLoesung();
64    }
65  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/ab_3/Gitter.java`

Text-Ausgabe

```
Anzahl der Wege von 2x2: 6
Gitter:
 x 0 1 2
 0 0 1 1
 1 1 2 3
 2 1 3 6

Anzahl der Wege von 3x3: 20
Gitter:
  x  0  1  2  3
  0  0  1  1  1
  1  1  2  3  4
  2  1  3  6 10
  3  1  4 10 20

Anzahl der Wege von 4x4: 70
Gitter:
  x  0  1  2  3  4
  0  0  1  1  1  1
  1  1  2  3  4  5
  2  1  3  6 10 15
  3  1  4 10 20 35
  4  1  5 15 35 70

Anzahl der Wege von 5x5: 252
Gitter:
   x   0   1   2   3   4   5
   0   0   1   1   1   1   1
   1   1   2   3   4   5   6
   2   1   3   6  10  15  21
   3   1   4  10  20  35  56
   4   1   5  15  35  70 126
   5   1   6  21  56 126 252
```

Test-Datei

```
3  import static org.junit.Assert.*;
4  import org.junit.Test;
5
```

```java
public class GitterTest {
  @Test
  public void zweiMalZwei() {
    Gitter gitter = new Gitter(2, 2);
    assertEquals(6, gitter.berechneAnzahlWege());
  }

  @Test
  public void zehnMalZwanzig() {
    Gitter gitter = new Gitter(10, 20);
    assertEquals(30045015, gitter.berechneAnzahlWege());
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/aufgaben/aud/ab_3/GitterTest.java`