Aufgabe 4

Für Binomialkoeffizienten $\binom{n}{k}$ gelten neben den grundlegenden Beziehungen $\binom{n}{0} = 1$ und $\binom{n}{n} = 1$ auch folgende Formeln:

> **Exkurs: Binomialkoeffizient**
> Der Binomialkoeffizient ist eine mathematische Funktion, mit der sich eine der Grundaufgaben der Kombinatorik lösen lässt. Er gibt an, auf wie viele verschiedene Arten man k bestimmte Objekte aus einer Menge von n verschiedenen Objekten auswählen kann (ohne Zurücklegen, ohne Beachtung der Reihenfolge). Der Binomialkoeffizient ist also die Anzahl der k-elementigen Teilmengen einer n-elementigen Menge. [a]
>
> [a] `https://de.wikipedia.org/wiki/Binomialkoeffizient`

A $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$

B $\binom{n}{k} = \binom{n-1}{k-1} \cdot \frac{n}{k}$

(a) Implementieren Sie unter Verwendung von Beziehung (A) eine rekursive Methode `binRek(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

Zuerst verwandeln wir die Beziehung (A) geringfügig um, indem wir n durch $n-1$ ersetzen:
$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

```java
public static int binRek(int n, int k) {
  if (k == 0 || k == n) {
    return 1;
  } else {
    return binRek(n - 1, k - 1) + binRek(n - 1, k);
  }
}
```

Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

(b) Implementieren Sie unter Verwendung von Beziehung (B) eine iterative Methode `binIt(n, k)` zur Berechnung des Binomialkoeffizienten in einer objektorientierten Programmiersprache oder entsprechendem Pseudocode!

```java
public static int binIt(int n, int k) {
  // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
  // Zwischenergebnisse ganze Zahlen sind.
  double ergebnis = 1;
  while (k > 0) {
    ergebnis = ergebnis * n / k;
    n--;
    k--;
  }
  // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
  // umgewandelt werden.
```

```
46        return (int) ergebnis;
47      }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

(c) Geben Sie die Laufzeitkomplexität der Methoden `binRek(n, k)` und `binIt(n, k)` aus den vorhergehenden beiden Teilaufgaben in O-Notation an!

Komplette Java-Klasse

```
3  /**
4   * <a href="https://www.studon.fau.de/file2889270_download.html">Angabe:
    ↪  PUE_AUD_WH.pdf</a>
5   * <a href="https://www.studon.fau.de/file3081306_download.html">Lösung:
    ↪  PUE_AUD_WH_Lsg.pdf</a>
6   */
7  public class Binomialkoeffizient {
8
9    /**
10    * Berechnet rekursiv den Binominalkoeffizienten „n über k". Dabei muss gelten:
11    * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
12    *
13    * @param n Ganzzahl n
14    * @param k Ganzzahl k
15    *
16    * @return Eine Ganzzahl.
17    */
18   public static int binRek(int n, int k) {
19     if (k == 0 || k == n) {
20       return 1;
21     } else {
22       return binRek(n - 1, k - 1) + binRek(n - 1, k);
23     }
24   }
25
26   /**
27    * Berechnet iterativ den Binominalkoeffizienten „n über k". Dabei muss gelten:
28    * n &#x3E;= 0, k &#x3E;= 0 und n &#x3E;= k.
29    *
30    * @param n Ganzzahl n
31    * @param k Ganzzahl k
32    *
33    * @return Eine Ganzzahl.
34    */
35   public static int binIt(int n, int k) {
36     // Das Ergebnis wird als Kommazahl deklariert, da nicht alle
37     // Zwischenergebnisse ganze Zahlen sind.
38     double ergebnis = 1;
39     while (k > 0) {
40       ergebnis = ergebnis * n / k;
41       n--;
42       k--;
43     }
44     // Vor dem Zurückgeben kann das Ergebnis nun in eine ganze Zahl
45     // umgewandelt werden.
46     return (int) ergebnis;
47   }
48 }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/Binomialkoeffizient.java`

Test

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BinomialkoeffizientTest {

  public void testeRek(int n, int k, int ergebnis) {
    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
  }

  public void testeIt(int n, int k, int ergebnis) {
    assertEquals(ergebnis, Binomialkoeffizient.binIt(n, k));
  }

  public void teste(int n, int k, int ergebnis) {
    testeRek(n, k, ergebnis);
    testeIt(n, k, ergebnis);
  }

  @Test
  public void teste() {
    teste(0, 0, 1);

    teste(1, 0, 1);
    teste(1, 1, 1);

    teste(2, 0, 1);
    teste(2, 1, 2);
    teste(2, 2, 1);

    teste(3, 0, 1);
    teste(3, 1, 3);
    teste(3, 2, 3);
    teste(3, 3, 1);

    teste(4, 0, 1);
    teste(4, 1, 4);
    teste(4, 2, 6);
    teste(4, 3, 4);
    teste(4, 4, 1);
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_46115/jahr_2014/fruehjahr/BinomialkoeffizientTest.java`