

Aufgaben 3

Gegeben seien die folgenden Formeln zur Berechnung der *ersten* Fibonacci-Zahlen:

$$\text{fib}_n = \begin{cases} 1 & \text{falls } n \leq 2 \\ \text{fib}_{n-1} + \text{fib}_{n-2} & \text{sonst} \end{cases}$$

sowie der Partialsumme der Fibonacci-Quadrate:

$$\text{sos}_n = \begin{cases} \text{fib}_n & \text{falls } n = 1 \\ \text{fib}_n^2 + \text{sos}_{n-1} & \text{sonst} \end{cases}$$

Sie dürfen im Folgenden annehmen, dass die Methoden nur mit $1 \leq n \leq 46$ aufgerufen werden, so dass der Datentyp `long` zur Darstellung aller Werte ausreicht.

Exkurs: Fibonacci-Folge

Die Fibonacci-Folge beginnt zweimal mit der Zahl 1. Im Anschluss ergibt jeweils die Summe zweier aufeinanderfolgender Zahlen die unmittelbar danach folgende Zahl: 1, 1, 2, 3, 5, 8, 13

Exkurs: Partialsumme

Unter der n -ten Partialsumme s_n einer Zahlenfolge a_n versteht man die Summe der Folgenglieder von a_1 bis a_n . Die immer weiter fortgesetzte Partialsumme einer (unendlichen) Zahlenfolge nennt man eine (unendliche) Reihe. ^a Partialsummen sind das Bindeglied zwischen Summen und Reihen. Gegeben sei die Reihe $\sum_{k=1}^{\infty} a_k$. Die n -te Partialsumme dieser Reihe lautet: $\sum_{k=1}^n a_k$. d. h. wir summieren unsere Reihe nur bis zum Endindex n . ^b

^a<https://www.lernhelfer.de/schuelerlexikon/mathematik/artikel/folgen-partialsummen>

^b<https://www.massmatics.de/merkzettel/index.php#!164:Partialsummen>

[`sos` steht für *Summe of Squares*]

n	fib _n	fib _n ²		$\sum_{k=1}^n \text{fib}_k^2$
1	1	1	1	1
2	1	1	1 + 1	2
3	2	4	1 + 1 + 4	6
4	3	9	1 + 1 + 4 + 9	15
5	5	25	1 + 1 + 4 + 9 + 25	40
6	8	64	1 + 1 + 4 + 9 + 25 + 64	104
7	13	169	1 + 1 + 4 + 9 + 25 + 64 + 169	273
8	21	441	1 + 1 + 4 + 9 + 25 + 64 + 169 + 441	714
9	34	1156	1 + 1 + 4 + 9 + 25 + 64 + 169 + 441 + 1156	1870
10	55	3025	1 + 1 + 4 + 9 + 25 + 64 + 169 + 441 + 1156 + 3025	4895

- (a) Implementieren Sie die obigen Formeln zunächst rekursiv (ohne Schleifenkonstrukte wie `for` oder `while`) und ohne weitere Optimierungen („naiv“) in Java als:

```
long fibNaive (int n) {
```

bzw.

```
long sosNaive (int n) {
```

```
16 public static long fibNaive(int n) {
17     if (n <= 2) {
18         return 1;
19     }
20     return fibNaive(n - 1) + fibNaive(n - 2);
21 }
22
23 public static long sosNaive(int n) {
24     if (n <= 1) {
25         return fibNaive(n);
26     }
27     return fibNaive(n) * fibNaive(n) + sosNaive(n - 1);
28 }
```

Code-Beispiel auf Github ansehen:
[src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java)

- (b) Offensichtlich ist die naive Umsetzung extrem ineffizient, da viele Zwischenergebnisse wiederholt rekursiv ausgewertet werden müssen. Die Dynamische Programmierung (DP) erlaubt es Ihnen, die Laufzeit auf Kosten des Speicherbedarfs zu reduzieren, indem Sie alle einmal berechneten Zwischenergebnisse speichern und bei erneutem Bedarf „direkt abrufen“. Implementieren Sie obige Formeln nun rekursiv aber mittels DP in Java als:

```
long fibDP (int n) {
```

bzw.

```
long sosDP (int n) {
```

```
30 public static long fibDP(int n) {
31     // Nachschauen, ob die Fibonacci-Zahl bereits berechnet wurde.
32     if (fib[n] != 0) {
33         return fib[n];
34     }
35     // Die Fibonacci-Zahl neu berechnen.
36     if (n <= 2) {
37         fib[n] = 1;
38     } else {
39         fib[n] = fibDP(n - 1) + fibDP(n - 2);
40     }
41     return fib[n];
42 }
43
44 public static long sosDP(int n) {
45     // Nachschauen, ob die Quadratsumme bereits berechnet wurde.
46     if (sos[n] != 0) {
47         return sos[n];
48     }
49     // Die Quadratsumme neu berechnen.
50     if (n <= 1) {
51         sos[n] = fibDP(n);
52     } else {
53         long tmp = fibDP(n);
```

```

54     sos[n] = tmp * tmp + sosDP(n - 1);
55 }
56 return sos[n];
57 }

```

Code-Beispiel auf Github ansehen:
[src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen_exam_66115_jahr_2017_fruehjahr/Fibonacci.java)

- (c) Am „einfachsten“ und bzgl. Laufzeit [in $\mathcal{O}(n)$] sowie Speicherbedarf [in $\mathcal{O}(1)$] am effizientesten ist sicherlich eine iterative Implementierung der beiden Formeln. Geben Sie eine solche in Java an als:

```
long fibIter (int n) {
```

bzw.

```
long sosIter (int n) {
```

```

59 public static long fibIter(int n) {
60     long a = 1;
61     long b = 1;
62     for (int i = 2; i < n; i++) {
63         long tmp = a + b;
64         b = a;
65         a = tmp;
66     }
67     return a;
68 }
69
70 public static long sosIter(int n) {
71     long a = 1;
72     long b = 0;
73     long sosSum = 1;
74     for (int i = 2; i <= n; i++) {
75         long tmp = a + b;
76         b = a;
77         a = tmp;
78         sosSum += a * a;
79     }
80     return sosSum;
81 }

```

Code-Beispiel auf Github ansehen:
[src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java](https://github.com/bschlangaul/examen_exam_66115_jahr_2017_fruehjahr/Fibonacci.java)

Additum Kompletter Code

```

3 import java.lang.reflect.InvocationTargetException;
4 import java.lang.reflect.Method;
5
6 /**
7  * <a href="https://www.studon.fau.de/file2861014_download.html">Angabe:
8  * PUE_AUD_3.pdf</a>
9  * <a href="https://www.studon.fau.de/file2893062_download.html">Lösung:
10  * PUE_AUD_3_Lsg.pdf</a>
11  */
12 public class Fibonacci {
13     static long[] fib = new long[47];

```

```
14 static long[] sos = new long[47];
15
16 public static long fibNaive(int n) {
17     if (n <= 2) {
18         return 1;
19     }
20     return fibNaive(n - 1) + fibNaive(n - 2);
21 }
22
23 public static long sosNaive(int n) {
24     if (n <= 1) {
25         return fibNaive(n);
26     }
27     return fibNaive(n) * fibNaive(n) + sosNaive(n - 1);
28 }
29
30 public static long fibDP(int n) {
31     // Nachschauen, ob die Fibonacci-Zahl bereits berechnet wurde.
32     if (fib[n] != 0) {
33         return fib[n];
34     }
35     // Die Fibonacci-Zahl neu berechnen.
36     if (n <= 2) {
37         fib[n] = 1;
38     } else {
39         fib[n] = fibDP(n - 1) + fibDP(n - 2);
40     }
41     return fib[n];
42 }
43
44 public static long sosDP(int n) {
45     // Nachschauen, ob die Quadratsumme bereits berechnet wurde.
46     if (sos[n] != 0) {
47         return sos[n];
48     }
49     // Die Quadratsumme neu berechnen.
50     if (n <= 1) {
51         sos[n] = fibDP(n);
52     } else {
53         long tmp = fibDP(n);
54         sos[n] = tmp * tmp + sosDP(n - 1);
55     }
56     return sos[n];
57 }
58
59 public static long fibIter(int n) {
60     long a = 1;
61     long b = 1;
62     for (int i = 2; i < n; i++) {
63         long tmp = a + b;
64         b = a;
65         a = tmp;
66     }
67     return a;
68 }
69
70 public static long sosIter(int n) {
71     long a = 1;
72     long b = 0;
73     long sosSum = 1;
74     for (int i = 2; i <= n; i++) {
```

```

75     long tmp = a + b;
76     b = a;
77     a = tmp;
78     sosSum += a * a;
79 }
80 return sosSum;
81 }
82
83 public static String fixColumn(long n) {
84     return (n + " ").substring(0, 5);
85 }
86
87 public static long invokeMethod(String methodName, int n) {
88     try {
89         Method method = Fibonacci.class.getMethod(methodName, int.class);
90         return (long) method.invoke(null, n);
91     } catch (NoSuchMethodException | SecurityException |
92             ↪ IllegalAccessException | IllegalArgumentException
93             | InvocationTargetException e) {
94         e.printStackTrace();
95     }
96     return 0;
97 }
98
99 public static void showResult(String methodName) {
100     System.out.println("\n" + methodName);
101     for (int i = 1; i <= 10; i++) {
102         System.out.print(fixColumn(invokeMethod(methodName, i)));
103     }
104 }
105
106 public static void main(String args[]) {
107     showResult("fibNaive");
108     showResult("sosNaive");
109     showResult("fibDP");
110     showResult("sosDP");
111     showResult("fibIter");
112     showResult("sosIter");
113     System.out.println();
114 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/Fibonacci.java)

Test

```

3 import static org.junit.Assert.assertEquals;
4 import org.junit.Test;
5
6 public class FibonacciTest {
7     private void testFib(String methodName) {
8         Fibonacci.fibNaive(1);
9         assertEquals(1, Fibonacci.invokeMethod(methodName, 1));
10        assertEquals(1, Fibonacci.invokeMethod(methodName, 2));
11        assertEquals(2, Fibonacci.invokeMethod(methodName, 3));
12        assertEquals(3, Fibonacci.invokeMethod(methodName, 4));
13        assertEquals(5, Fibonacci.invokeMethod(methodName, 5));
14        assertEquals(8, Fibonacci.invokeMethod(methodName, 6));
15        assertEquals(13, Fibonacci.invokeMethod(methodName, 7));
16        assertEquals(21, Fibonacci.invokeMethod(methodName, 8));

```

```
17     assertEquals(34, Fibonacci.invokeMethod(methodName, 9));
18     assertEquals(55, Fibonacci.invokeMethod(methodName, 10));
19 }
20
21 private void testSos(String methodName) {
22     Fibonacci.fibNaive(1);
23     assertEquals(1, Fibonacci.invokeMethod(methodName, 1));
24     assertEquals(2, Fibonacci.invokeMethod(methodName, 2));
25     assertEquals(6, Fibonacci.invokeMethod(methodName, 3));
26     assertEquals(15, Fibonacci.invokeMethod(methodName, 4));
27     assertEquals(40, Fibonacci.invokeMethod(methodName, 5));
28     assertEquals(104, Fibonacci.invokeMethod(methodName, 6));
29     assertEquals(273, Fibonacci.invokeMethod(methodName, 7));
30     assertEquals(714, Fibonacci.invokeMethod(methodName, 8));
31     assertEquals(1870, Fibonacci.invokeMethod(methodName, 9));
32     assertEquals(4895, Fibonacci.invokeMethod(methodName, 10));
33 }
34
35 @Test
36 public void fibNaive() {
37     testFib("fibNaive");
38 }
39
40 @Test
41 public void fibDP() {
42     testFib("fibDP");
43 }
44
45 @Test
46 public void fibIter() {
47     testFib("fibIter");
48 }
49
50 @Test
51 public void sosNaive() {
52     testSos("sosNaive");
53 }
54
55 @Test
56 public void sosDP() {
57     testSos("sosDP");
58 }
59
60 @Test
61 public void sosIter() {
62     testSos("sosIter");
63 }
64
65 }
```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/FibonacciTest.java](https://github.com/src/test/java/org/bschlangaul/examen/examen_66115/jahr_2017/fruehjahr/FibonacciTest.java)