**FP-Aufgaben 01**

Aufgabe 1

1. Schreiben Sie eine Java-Funktion (mittels for- oder while-Anweisung), die die n-te Zahl der Fibonacci-Folge zurückliefert

**public** **static** **int** fibonacciIterativ(**int** n) {

**int** numb1 = 0, numb2 = 1;

**while** (n > 1) {

**int** prevSum = numb1 + numb2;

numb1 = numb2;

numb2 = prevSum;

n--;

}

**return** numb1;

}

1. Schreiben Sie eine rekursive Java-Funktion, die die n-te Zahl der Fibonacci-Folge zurückliefert.

**public** **static** **int** fibonacciRecursive(**int** n) {

**return** *fibHelper*(0, 1, n - 1);

}

**public** **static** **int** fibHelper(**int** numb1, **int** numb2, **int** n) {

**if** (n <= 0) {

**return** numb1;

} **else** {

**int** prevSum = numb1 + numb2;

numb1 = numb2;

numb2 = prevSum;

**return** *fibHelper*(numb1, numb2, n - 1);

}

}

1. Was gibt das folgende Programm aus und wie oft wird die rekursive Funktion „fibonacci“ (aus dem Teil-b) aufgerufen?

**public** **static** **void** main(String[] args) {

**int** ergebnis = *fibonacciRecursive*(3);

System.***out***.println(ergebnis);

}

Die **fibonacciRecursvie** Funktion wird einmal aufgerufen, jedoch wird die **fibHelper** n-mal aufgerufen, also 3-mal aufgerufen. Als insgesamt finden 4 Funktionsaufrufe statt.

**Was im Stack passiert:**

1. *Schritt StackFrame*

Operanten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **fibHelper(0, 1, 3-1)** |  |  |  |  |

Operatoren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

1. *Schritt StackFrame*

Operanten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fibHelper(0, 1, 2) | **fibHelper(1, 1, 1)** |  |  |  |

Operatoren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

1. *Schritt StackFrame*

Operanten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fibHelper(0, 1, 2) | fibHelper(1, 1, 1) | **fibHelper(1, 2, 0)** |  |  |

Operatoren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

1. *Schritt StackFrame*

Operanten

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fibHelper(0, 1, 2) | fibHelper(1, 1, 1) | fibHelper(1, 2, 0) | **2** |  |

Operatoren

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Aufgabe 2

Schreiben Sie eine rekursive Java-Funktion, die kontrolliert, ob ein eingegebenes Wort ein Palindrom ist

**public** **boolean** isPalindrome(String palindrom) {

**return** isPalindromeHelper(

palindrom.toLowerCase(),

0,

palindrom.length() - 1);

}

**public** **boolean** isPalindromeHelper(String palindrom, **int** left, **int** right) {

**if** (left >= right) {

**return** **true**;

} **else** {

**boolean** rightEqLeft =

(palindrom.charAt(left) == palindrom.charAt(right));

**return**

rightEqLeft &&

isPalindromeHelper(palindrom, left + 1, right - 1);

}

}

Aufgabe 3

Erwähnen Sie die Vorteile und Nachteile der Rekursion im Vergleich mit der Iteration

Vorteile Rekursion:

* Manche Algorithmen lassen sich mit Rekursion anschaulicher erklären
* Viele Datenstrukturen sind rekursiv aufgebaut z.B. Binärbäume

Nachteile Rekursion:

* Komplizierter nachzuvollziehen
* Nicht sehr intuitiv
* Langsamer und verbraucht mehr Speicher, da für die Funktionsaufrufe Stack Frames reserviert/erstellt werden müssen, um dann die Funktionsaufrufe auf dem Stack zu speichern.
* Kann einfacher einen Stack Overflow Error verursachen