



Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)

<http://db.in.tum.de/teaching/ss22/ei2/>

Blatt Nr. 5 $\frac{1}{2}$

In diesem Blatt erklären wir rekursive Methoden in Java. Nach einer kleinen Einführung folgen dann Aufgaben, die Sie selbst lösen sollen. Wie alle Blätter ist auch dieses Blatt **klausurrelevant**.

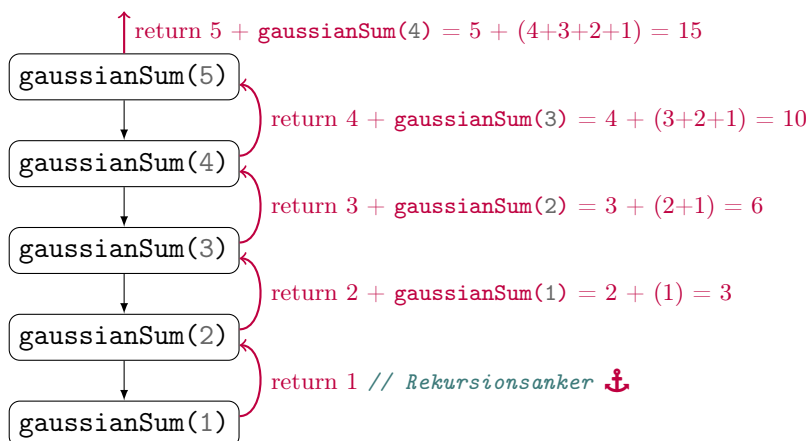
Jedoch wird dieses Blatt aufgrund des Feiertags nicht gesondert besprochen. Fragen stellen Sie bitte in der folgenden Zentralübung am 13.06.2022.

Rekursion in Java

Rekursive Methoden in Java sind Methoden, die sich selbst aufrufen. Bei rekursiver Programmierung kann es schnell zu sogenannten Stackoverflows kommen – dies passiert beispielsweise, wenn die Methode sich unendlich oft selbst aufrufen würde. Um dies zu verhindern, muss es einen sogenannten Rekursionsanker geben, an dem die Funktion returned, ohne sich selbst aufzurufen.

Beispiel: Gaußsche Summenformel

```
int gaussianSum(int n) { // für n > 0
    if (n==1) return 1; // Rekursionsanker
    return n + gaussianSum(n-1);
}
```



1 Fakultät

Vervollständigen Sie die *rekursive* Methode, die für eine Zahl n die Fakultät berechnet.

```
int factorial(int n) { // für n > 0
    // todo
}
```

2 Fibonacci Zahlen

Vervollständigen Sie die *rekursive* Methode, die die n -te Fibonacci Zahl berechnet.

```
int fibonacci(int n) { // für n > 0
    // todo
}
```

3 Stufen

Angenommen, ein Kind kann eine Treppe mit s Stufen erklimmen, indem es entweder 1, 2, ..., k Stufen in einem einzigen Schritt nimmt. Berechnen Sie alle unterschiedlichen Schrittfolgen, mit denen das Kind genau die s Stufen der Treppe erklimmt. Vervollständigen Sie den Programmcode:

```
int stufen(int verbleibendeStufen, List<Integer> bisherigeSchritte, int k) {
    if (verbleibendeStufen == 0) { // mögliche Lösung gefunden
        for (Integer i : bisherigeSchritte)
            System.out.print(i + " - ");
        System.out.println("");
    }
    // todo
}
```

Tipp: In Java werden Listen genauso wie alle anderen Objekte mit Call by Reference aufgerufen – was bedeutet das für Ihre Methode?

4 Türme von Hanoi [anspruchsvoll]

In dieser Aufgabe implementieren wir das bekannte Knobelspiel „Die Türme von Hanoi“¹.

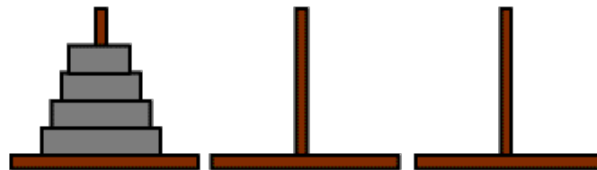


Abbildung 1: Die Türme von Hanoi mit vier Scheiben

Überlegen Sie sich einen rekursiven Algorithmus um alle Scheiben von Position 1 auf Position 3 zu verschieben. Dabei darf immer nur die oberste Scheibe eines Turms auf einen anderen Turm gelegt werden, wenn dessen oberste Scheibe größer ist (oder an der Zielposition gar keine Scheiben sind). **Tipp:** Damit die unterste Scheibe zur Position 3 bewegt werden kann, muss der darüber liegende Turm erstmal auf Position 2 verschoben werden. Dann kann man die größte Scheibe verschieben und dann den kleineren Turm von Position 2 auf die große Scheibe verschieben. Überlegen Sie sich wie dieses Prinzip auch für das Verschieben der kleineren Türme zum Tragen kommt.

¹Für eine ausführliche Beschreibung siehe Wikipedia: http://de.wikipedia.org/wiki/Türme_von_Hanoi. Das ganze kann auch auf unzähligen Seiten ausprobiert werden, z.B.: <http://vornlocher.de/tower.html>

```
// Diese Methode nimmt die oberste Plate von source und fugt sie oben zu dest  
↳ hinzu  
void moveDisk(Stack<Integer> source, Stack<Integer> dest) {  
    assert(source.peek() < dest.peek());  
    dest.push(source.pop());  
}  
  
void hanoi(int numPlates, Stack<Integer> source, Stack<Integer> buffer,  
↳ Stack<Integer> destination) {  
    if (numPlates == 0) return;  
    // todo  
}
```