

# Aufgaben zu Java Functional Interface

## Einleitung:

---

Lesen Sie zunächst 17.1\_JavaFunctionalInterfaces.

Danach lösen Sie die Aufgaben



## Ziel

---

- ⇒ Sie können die Unterschiede verschiedener Functional Interfaces von Java erklären.
- ⇒ Sie können Functional Interfaces von Java spezifisch anwenden.

## Inhalte

---

Einleitung: .....	1
Ziel .....	1
Inhalte .....	1
Aufgabe: Functional Interfaces aus java.util.function.....	2
Aufgabe: Methoden von java.util List, die ein Functional Interface erwarten .....	3
Aufgabe: Methoden von java.util.stream Stream, die ein Functional Interface erwarten .....	3
Beispiel: Einfache Anwendung mit Functional Interface.....	4
Aufgabe: Summe bis n mit Hilfe eines Functional Interface .....	4
Aufgabe: Wachstum und Zerfall mit Hilfe eines Functional Interface.....	5

**Aufgabe: Functional Interfaces aus java.util.function**


---

Es in verschiedensten Packages von Java Funktional Interfaces.

Sie untersuchen diese und halten fest, welche Parameter und Return Wert die durch die lambda Expression zu implementierende Methode verlangt.

ACHTUNG: die functional Interfaces enthalten oft mehrere Methoden. Eine davon ist die zentrale Methode, die durch die lambda Expression implementiert wird.

**java.util.function**

Functional interface	Erklärung	Zentrale Methode des Interfaces
Consumer<T>	Represents an operation that accepts a single input argument and returns no result.	void accept(T t)
Function<T, R>		
DoubleFunction<R>		
BiFunction<T,U,R>		
Supplier<T>		
Predicate<T>		
BinaryOperator<T>		
DoubleUnaryOperator		

**Aufgabe: Methoden von java.util List, die ein Functional Interface erwarten**

Es gibt bei der Java List verschiedene Methoden, die als Parameter eine Methodenreferenz auf ein Functional Interface erwarten.

Das bedeutet, dass bei diesen Methoden lambda Expression eingesetzt werden können.

Methode von List	Erwartetes functional Interface	Methode des functional interfaces
sort	Comparator<T>	int compare(T o1, T o2)
forEach		
removeIf		

**Aufgabe: Methoden von java.util.stream Stream, die ein Functional Interface erwarten**

Und auch bei Stream gibt es verschiedene Methoden, die als Parameter eine Methodenreferenz auf ein Functional Interface erwarten.

Das bedeutet, dass bei diesen Methoden lambda Expression eingesetzt werden können.

ACHTUNG: Es gibt auch Methoden, die erwarten mehrere functional Interfaces oder auch keine.

Methode von Stream	Erwartetes functional Interface	Methode des functional interfaces
forEach	Consumer<T>	void accept(T t)
map		
collect		
filter		
findFirst		
peek		
sorted		
sum		
reduce		

### Beispiel: Einfache Anwendung mit Functional Interface

---

Sie können Functional Interfaces auch direkt anwenden.

Dabei wird der Bezug von Parameter, Return-Wert und Methode sichtbar.

#### Beispiel:

Das erste Beispiel zeigt das Prinzip mit einer sehr einfachen Rechnung.

⇒ Aufgabe: Addieren Sie zu einer Zahl 2 hinzu.

```
//Zahl + 2
Function<Integer, Integer> twoMore = n -> n+2;
System.out.println("Two more then: " + 5 + " " + twoMore.apply(5));
```

#### Erläuterungen:

- Sie brauchen ein Functional Interface mit einem Parameter und einem Return-Wert.  
*Function(T, R)*
- Sie implementieren eine einfache Lambda Expression und weisen diese der Variablen *twoMore* zu.
- Beim Aufruf verwenden Sie die Methode *apply* des Functional Interface, übergeben die Ausgangszahl 5 und erhalten eine 7 zurück.

### Aufgabe: Summe bis n mit Hilfe eines Functional Interface

---

- ⇒ Aufgabe: Berechne die Summer der Zahlen von 1 bis n.  
Beispiel: n=5 dann ist die Summe 1+2+3+4+5 = 15
- Orientieren Sie sich am Beispiel oben.

Variante:

- Erstellen Sie eine zweite Implementierung auf der Basis der *Gaussischen Summenformel*.

### Aufgabe: Wachstum und Zerfall mit Hilfe eines Functional Interface

---

Beim **Wachstum** verdoppelt sich eine Zahl  $v$  genau  $n$ -mal.

Beispiel: Zahl 3 dreimal wachsen lassen: 3, 6, 12, 24

- ⇒ Aufgabe: Berechne das Wachstum einer Zahl.  
Als Parameter übergeben Sie die Zahl und die Anzahl Verdoppelungen.
- Wählen Sie ein geeignetes Functional Interface, das zwei Parameter und einen Return-Wert anbietet.

Beim **Zerfall** halbiert sich eine Zahl  $v$  genau  $n$ -mal.

Beispiel Zahl 24 dreimal zerfallen lassen: 24 12 6 3

- ⇒ Aufgabe: Berechne den Zerfall einer Zahl.  
Als Parameter übergeben Sie die Zahl und die Anzahl Halbierungen.
- Wählen Sie ein geeignetes Functional Interface, das zwei Parameter und einen Return-Wert anbietet.