

## Praktikum 3 zu KMPS

WS 2022/2023

Bei diesem Praktikumsversuch werden Sie gebräuchliche Higher-Order-Funktionen selber implementieren und anwenden. Als Basis für dieses Praktikum soll Ihr Scala-Programm aus dem 2. Praktikum dienen.

### Einschränkungen bei der Programmierung:

Alle Funktionen sollen, wie auch beim ersten Praktikumsversuch, als *reine Funktionen* implementiert werden und dürfen keine Iterationen/Schleifen verwenden. Ausnahmen werden in den Aufgabenstellungen angegeben.

Das Benutzen von Funktionen der Scala- oder Java-Standardbibliothek, oder Methoden von Objekten ist nicht erlaubt, es sei denn die Aufgabenstellung macht explizit eine Ausnahme.

So dürfen in ihrem Code nur Konstanten (`val`) verwendet werden, keine Variablen (`var`).

### Allgemeine Hinweise:

Für die Nutzung von Higher-Order-Funktionen sollen anonyme Funktionen als Argumente verwendet werden.

### Aufgabe 1: (`map`)

Implementieren Sie die Higher-Order-Funktion `map`:

- a. `map` nimmt zwei Argumente entgegen:
  1. Eine Liste `input_list` mit Elementen beliebigen Typs
  2. Eine Funktion `func`, die ein Element entgegennimmt und als Rückgabewert ein Element des gleichen Typs zurückgibt`map` wendet `func` auf jedes Listenelement von `input_list` an, und gibt eine Liste aus den Rückgabewerten von `func` zurück.
- Beispiel:  
`func` gebe für jeden Kleinbuchstaben den Großbuchstaben zurück.  
`input_list`: `[a, b, c, d]`  
Rückgabewert: `[A, B, C, D]`
- b. Nutzen Sie `map`, die `copy`-Methode und die String-Methode `toUpperCase`, um aus ihrer Alben-Liste eine neue Liste zu erstellen, in der alle Alben-Titel nur aus Großbuchstaben bestehen.
- c. Erstellen Sie auf die gleiche Weise eine Liste, in der auch alle Track-Titel nur aus Großbuchstaben bestehen.
- d. Implementieren Sie die Higher-Order-Funktion `poly_map`. `poly_map` verhält sich genau wie `map`, allerdings können die Elemente der Ergebnis-Liste einen anderen Typ haben als die Elemente der ursprünglichen Liste. Dementsprechend muss der Rückgabewert der Funktion `func` nicht zwingend den gleichen Typ haben wie das Argument.
- e. Nutzen Sie `poly_map`, um aus ihrer Alben-Liste eine Liste zu erzeugen, die für jedes Album eine Liste der Längen ihrer Tracks enthält.

### Beispiel:

Für ein Album mit 2 Tracks der Länge 3:15 und 4:00 soll die Liste `[3:15, 4:00]` erzeugt werden.

## Praktikum 3 zu KMPS

WS 2022/2023

### Hinweis für Aufgabe 2 und Aufgabe 3:

In den Aufgaben 2 und 3 sollen Sie nur auf dem *Michael-Jackson-Album* arbeiten. Speichern Sie sich dieses Album als separate Konstante, um den Zugriff zu vereinfachen. Sie dürfen dafür den Klammer-Operator der Liste nutzen.

### Aufgabe 2: (filter)

Implementieren Sie die Higher-Order-Funktion `filter`:

- a. `filter` nimmt zwei Argumente entgegen:
  1. Eine Liste `input_list` mit Elementen beliebigen Typs.
  2. Eine Funktion `condition`, die ein Element entgegennimmt und als Rückgabewert Boolean zurückgibt.`filter` wendet `condition` auf jedes Listenelement von `input_list` an und gibt eine Liste zurück, in der nur Elemente enthalten sind, für die `condition` wahr zurückgegeben hat.

#### Beispiel:

`condition` sei für jede gerade Zahl wahr.

`input_list`: [1,2,3,4,4,5,5,6,7]

Rückgabewert: [2,4,4,6]

- b. Nutzen Sie `filter`, um eine Liste von Tracks zu erzeugen, in der nur Tracks mit einer Bewertung von  $\geq 4$  enthalten sind.
- c. Kombinieren Sie `poly_map` und `filter`, um eine Liste der Titel aller Tracks zu erzeugen, die von *Rod Temperton* geschrieben wurden.

### Aufgabe 3: (partition)

Implementieren Sie die Higher-Order-Funktion `partition`:

- a. `partition` nimmt zwei Argumente entgegen:
  1. Eine Liste `input_list` mit Elementen beliebigen Typs.
  2. Eine Funktion `condition`, die ein Element entgegennimmt und als Rückgabewert Boolean zurückgibt.`partition` wendet `condition` auf jedes Listenelement von `input_list` an. Sollte `condition` wahr sein, wird die Liste an diesem Punkt geteilt. Die Liste aller Teillisten wird zurückgegeben.

#### Beispiel:

`condition` sei für jeden Großbuchstaben wahr.

`input_list`: [a, b, c, D, e, f, G, H, i, J]

Rückgabewert: [[a, b, c], [e, f], [], [i], []]

- b. Nutzen Sie `partition`, um zwei Listen zu erzeugen: die Erste Liste enthält alle Tracks vor dem Track mit dem Titel „Thriller“; die zweite Liste enthält alle Tracks danach.
- c. Ersetzen Sie ihre Funktion `createTokenList` durch eine Kombination von `poly_map`-, `filter`- und `partition`-Aufrufen. Sie dürfen die Methode `mkString` benutzen, die aus einer Liste von Chars einen String bildet, und die Funktion `isBlank`, die für einen String zurückgibt, ob dieser nur aus Whitespace-Characters besteht und wie folgt zu implementieren ist:

```
def isBlank(s: String): Boolean = s.trim.isEmpty
```

## Praktikum 3 zu KMPS

WS 2022/2023

### Aufgabe 4:

- a. Implementieren Sie eine Higher-Order-Funktion, die die Funktionen
  - `sum` auf Folie HigherOrderProgrammierung 3
  - `prod` aus der Übungverallgemeinert, insofern, dass die Funktionswerte in einem beliebigen Bereich durch eine beliebige Operation verknüpft werden können. Dabei dürfen keine weiteren Higher-Order-Funktionen verwendet werden. Sämtliche Bereiche sind als inklusiv zu betrachten (d.h. der Bereich 1-5 enthält sowohl 1 als auch 5)
- b. Verwendet Ihre Implementierung aus a. right- oder left-folding? Unter welchen Umständen würde das jeweils Andere die Funktionalität Ihres Programms ändern?
- c. Wie verhält sich Ihre Implementierung aus a. für einen leeren Wertebereich? Welches Verhalten wäre in so einem Fall sinnvoll?
- d. Implementieren Sie die Funktion aus a. ohne Rekursion jedoch unter Verwendung der Higher-Order-Funktionen `map`, `fold` und `range` auf Integer-Listen aus Vorlesung und Übung.