# Praktikum 1 zu KMPS

Bei diesem Praktikumsversuch werden Sie ein vorliegendes imperatives Java-Programm in ein funktionales Java-Programm überführen. Dieses funktionale Programm bildet den Grundstein für die spätere Implementierung in einer funktionalen Sprache.

Alle Methoden, mit Ausnahme der Main-Methode, sollen als *reine Funktionen* implementiert werden. D.h. sie dürfen nur von Argumenten und lokalen Variablen lesen und das Programmverhalten nur durch ihre Rückgabewerte beeinflussen. Außerdem sind Iterationen nicht erlaubt (diese sollen durch Rekursion gelöst werden).

**Aufgabe 1:** (Imperatives Programm)
Das vorliegende Programm liest eine XML-Datei ein, die Informationen über Musik-Alben enthält und konvertiert diese in eine Liste von Java-Objekten.

a. Schauen Sie sich hierfür die XML-Datei `alben.xml` an und versuchen Sie die Syntax zu verstehen. Achten Sie darauf wo welche Attribute wie oft vorkommen können.
b. Schauen Sie sich die Java-Dateien des imperativen Programms `Imperativ.java`, `Album.java` und `Track.java` an.
c. Führen Sie das imperative Programm aus und beobachten Sie dessen Funktionsweise. Achten Sie insbesondere auch darauf in welche Datenstrukturen das Programm die XML-Datei transformiert.

**Aufgabe 2:** (Token-Liste erstellen)
Erstellen Sie eine neue Java-Klasse `Funktional`, in der Sie das Verhalten der Klasse `Imperativ` funktional umsetzen werden.

a. Erstellen Sie zunächst eine Main-Methode, in der Sie die XML-Datei in ein Byte-Array mit `Files.readAllBytes(Paths.get("alben.xml"))` einlesen.
b. Schreiben Sie nun eine rekursive Methode `createTokenList`, die den Inhalt der XML-Datei entgegennimmt und eine Liste von Token (hier: Liste von Strings) zurückgibt. Sie können aus dem folgenden Beispiel entnehmen, was in unserem Kontext als „Token" bezeichnet werden soll.

*Beispiel*:

Input:

```
<album>
  <title>Thriller</title>
  <track>
    <title>Billie Jean</title>
    <length>4:54</length>
  </track>
  <artist>Michael Jackson</artist>
</album>
```

# Praktikum 1 zu KMPS

Output (Token-Liste; hier: Liste von Strings):

```
[album, title, Thriller, /title, track, title, Billie Jean, /title,
length, 4:54, /length, /track, artist, Michael Jackson, /artist,
/album]
```

*Hinweis*:
Leerzeichen in Token müssen erhalten bleiben!

**Aufgabe 3** (Parsing der Token)
Schreiben Sie eine Methode `parseFile`, die die Token-Liste aus Aufgabe 2 entgegennimmt und eine Liste aller Alben mit den korrekten Attributen erzeugt und zurückgibt, die in der XML-Datei enthalten sind.

a. Überlegen Sie hierzu, wie Sie es schaffen ohne Iteration (Schleifen) auf der Token-Liste zu operieren.
b. Überlegen Sie sich zusätzlich, ob Sie weitere Untermethoden verwenden können, um die Implementierung zu erleichtern.
c. Implementieren Sie die Methode gemäß Ihrer Überlegungen aus a) und b).
d. Rufen Sie die Methode von Ihrer Main-Methode aus auf. Geben Sie die resultierende Alben-Liste auf der Standardausgabe aus. Ihre Ausgabe sollte gleich der Ausgabe des imperativen Programms sein.

*Hinweis*:
Um komplett auf globale Variablen verzichten zu können ist es unter Umständen nötig mehrere Werte von einer Methode zurückzugeben. Dazu können sie in Java *z.B.* folgenden Workaround verwenden:

```
public static Object[] return1AndHallo(){
     return new Object[]{1, "Hallo"};
}
```

Aufruf:

```
Object[] returnValues = return1AndHallo();
int one = (int)returnValues[0];
String hallo = (String)returnValues[1];
```