5.3 Funktionen höherer Ordnung

Eine Eigenschaft funktionaler Programme ist es, dass Funktionen miteinander verkettet werden können. In der Vorlesung haben Sie dazu verschiedene Mechanismen kennengelernt: Funktionen höherer Ordnung sind ein wesentlicher Bestandteil des funktionalen Paradigmas.

a) Was sind Funktionen höherer Ordnung und welchen Vorteil gewinnt man durch Sie?

Funktionen höherer Ordnung sind Funktionen die andere Funktionen als Parameter bekommen. Durch dieses Prinzip gewinnt man, dass nicht alle Logik in einer Funktion stehen muss. Teile der Logik können in der übergebenen Funktion stehen. Dadurch kann eine einzelne Funktion für viele verschiedene Anwendungsfälle verwendet werden.

b) Wie werden in Scala Funktionen höherer Ordnung umgesetzt? Gehen Sie auf die Begriffe partielle Anwendung und Unterversorgung ein!

- Ein möglicher Syntax ist **functionName: (Parameter...) => Rückgabetyp**, wobei die Paramterliste auch leer sein kann. Der Aufruf der Funktion erfolgt im Syntax functioName(Parameter...), wobei Parameter auch leer sein kann _. Hier ein Beispiel:

```
def foo(f1: () => Unit, f2: Double => Unit, f3: () => Double, f4: (Double,
Double) => Double){
 //...
    f1()
    f2(_)
    val f3Result = f3()
 //...
}
```

- Funktionen können partiell berechnet werden, d.h. es werden nicht alle Parameter übergeben (leere Parameter werden mit _ angegeben). Das Ergebnis eines solchen Aufrufs ist eine Funktion. Diese Funktion kann da zu einem späteren Zeitpunkt mit den fehlenden Paramtern aufgerufen werden.

```
def foo(a: Double, b: Double){

}
val partialFoo = foo(10,_)
partialFoo(10)
```