Funktionale Programmierung in Scala

Jan Albert

8. Oktober 2018

Inhaltsverzeichnis

Was ist Funktionale Programmierung?

Idee: Benutzt ausschließlich "reine Funktionen" d. h. Funktionen, welche keine Seiteneffekte haben.

Beispiele für Seiteneffekte:

- Verändern/Modifizieren einer Variable
- Verändern/Modifizieren einer Datenstruktur
- Ein Attribut initalisieren
- Eine Exception werfen
- Konsolen Eingabe/Ausgabe
- Lesen/Schreiben aus/von einer Datei

Definitionen

Definition (Reine Funktionen)

Eine *reine Funktion* mit Eingabetyp A und Ausgabetyp B (Schreibweise: $A \Rightarrow B$) ist eine Berechnung, welche jeden Wert a vom Typ A genau einen Wert b vom Typ B zuordnet, sodass b nur aus dem Wert von a bestimmt wird.

Beispiele:

- Eine Funktion intToString vom Typ Int $\Rightarrow$ String bildet jede ganze Zahl auf einen String ab und macht nichts anderes.
- Die Addition von ganzen Zahlen.

Ausdruck

Definition (Ausdurck)

Jeder Teil eines Programms, welcher zu einem Ergebnis zusammengefasst werden kann d. h. alles was man in den Scala-Interpreter tippen kann und ein Ergebnis liefert, nennen wir einen *Ausdruck*.

Beispiel: $2 + 3$ ist ein Ausdruck, welcher die reine Funktion $+$ vom Typ (Int, Int) $\Rightarrow$ Int auf 2 und 3 anwendet.

Referenziell Transparent (RT)

Definition (Referenziell Transparent (RT))

Ein Ausdruck e ist *Referenziell Transparent (RT)*, wenn für alle Programme p, alle Vorkommnisse von e in p durch das Ergebnis von e ersetzt werden können, ohne die Bedeutung von p zu ändern. Eine Funktion ist rein, wenn der Ausdruck $f(x)$ referenziell transparent für alle referenziell transparenten x ist.

Beispiel mit Seiteneffekte

```
1 class Cafe {
2   def buyCoffee(cc: CreditCard): Coffee = {
3     val cup = new Coffee()
4     cc.charge(cup.price)
5     cup
6   }
7 }
```

Beispiel ohne Seiteneffekte

```
1 class Cafe {
2   def buyCoffee(cc: CreditCard): (Coffee, Charge) = {
3     val cup = new Coffee()
4     (cup, Charge(cc, cup.price))
5   }
6 }
```

Die Klasse Charge

```
1 case class Charge(cc: CreditCard, amount: Double) {
2   def combine(other: Charge): Charge =
3     if (cc == other.cc)
4       Charge(cc, amount + other.amount)
5     else
6       throw new Exception("Can't combine charges to
            different cards")
7 }
```

Quellen

Paul Chiusano, Runar Bjarnason
Functional Programming in Scala
Manning, 2014.

S. Jemand.
On this and that.
Journal of This and That, 2(1): 50–100, 2000.

Vielen Dank
für eure Aufmerksamkeit.