

Funktionale Programmierung (in Scala)

Jan Albert

2. Dezember 2018

Buch

Functional Programming in Scala

Paul Chiusano, Runar Bjarnason

Manning, 2014



Inhaltsverzeichnis

Was ist Funktionale Programmierung?

Definitionen

Beispiele referenzielle Transparenz

Beispiele Seiteneffekt

Zusammenfassung

Was ist Funktionale Programmierung?

Idee: Verwendung "reiner Funktionen" (keine Seiteneffekte)

Beispiele für Seiteneffekte:

- verändern/modifizieren einer Variable
- verändern/modifizieren einer Datenstruktur
- eine Exception werfen
- Konsoleneingabe/-ausgabe
- lesen/schreiben aus/von einer Datei

Reine Funktion

Definition

Eine *reine Funktion* mit Eingabetyp A und Ausgabotyp 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:

- Funktion `intToString`
- Addition ganzer Zahlen

Ausdruck

Definition

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

Beispiel: $2 + 3$

Referenzielle Transparenz

Definition

Ein Ausdruck e ist *referenziell transparent*, 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 für referenzielle Transparenz

```
scala> val x = "Hello, World"  
x: java.lang.String = Hello, World
```

```
scala> val r1 = x.reverse  
r1: java.lang.String = dlroW ,olleH
```

```
scala> val r2 = x.reverse  
r2: java.lang.String = dlroW ,olleH
```


Beispiel für referenzielle Transparenz

```
scala> val r1 = "Hello, World".reverse  
r1: java.lang.String = dlroW ,olleH
```

```
scala> val r2 = "Hello, World".reverse  
r2: String = dlroW ,olleH
```

Gegenbeispiel für referenzielle Transparenz

```
scala> val x = new StringBuilder("Hello")  
x: java.lang.StringBuilder = Hello
```

```
scala> val y = x.append(", World")  
y: java.lang.StringBuilder = Hello, World
```

```
scala> val r1 = y.toString  
r1: java.lang.String = Hello, World
```

```
scala> val r2 = y.toString  
r2: java.lang.String = Hello, World
```

Gegenbeispiel für referenzielle Transparenz

```
scala> val x = new StringBuilder("Hello")  
x: java.lang.StringBuilder = Hello
```

```
scala> val r1 = x.append(", World").toString  
r1: java.lang.String = Hello, World
```

```
scala> val r2 = x.append(", World").toString  
r2: java.lang.String = Hello, World, World
```

Beispiel mit Seiteneffekt

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

- Seiteneffekt: Returntype von `cc.charge(cup.price)` nicht sichtbar
- Überprüfung referenzielle Transparenz:
 $p(\text{buyCoffee}(a\text{CreditCard})) \neq p(\text{new Coffee}())$

Beispiel ohne Seiteneffekt

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

Zusammenfassung

- Idee: Verwendung reiner Funktionen (keine Seiteneffekte)
- Referenzielle Transparenz testet Abwesenheit von Seiteneffekte
- Ergebnisse reiner Funktionen sind kontextunabhängig
- Reine Funktionen: leichter testbar, wiederverwendbar und modular