

# Inhaltsverzeichnis

<b>Vorwort</b>	<b>9</b>
<b>1 Warum Scala?</b>	<b>11</b>
1.1 Was ist Scala?	11
1.2 Warum Scala statt Java?	12
1.3 Warum Scala statt Groovy, JRuby & Co?	16
<b>2 Entwicklungsumgebung</b>	<b>17</b>
2.1 Kommandozeilen-Werkzeuge	17
2.1.1 scalac und fsc	18
2.1.2 scala	20
2.1.3 scaladoc	21
2.2 Build-Werkzeuge	21
2.2.1 Ant	22
2.2.2 Maven	23
2.2.3 SBT	24
2.3 IDEs	27
2.3.1 Scala IDE for Eclipse	28
2.3.2 IntelliJ IDEA	29
<b>3 Das Fallbeispiel „ScalaTrain“</b>	<b>31</b>
<b>4 Erste Gehversuche in der REPL</b>	<b>33</b>
4.1 Variablen	33
4.1.1 Unveränderliche Variablen	33
4.1.2 Veränderliche Variablen	35
4.2 Methoden	35
4.2.1 Alles hat ein Ergebnis	35
4.2.2 Unit-Methoden	37
4.3 Funktionen	38

<b>5 OO-Grundlagen</b>	<b>41</b>
5.1 Vorbereitung: Projekt initialisieren	41
5.2 Klassen	42
5.2.1 Klassenparameter und Konstruktoren	43
5.2.2 Felder	45
5.2.3 Methoden	47
5.2.4 Named and Default Arguments	50
5.3 Packages und Sichtbarkeit	51
5.3.1 Verschachtelte Packages	52
5.3.2 Imports	53
5.3.3 Sichtbarkeit	54
5.4 Singleton Objects	55
5.4.1 Companion Objects	56
5.4.2 Predef	56
5.5 Case Classes	57
5.6 Projekt-Code: aktueller Stand	60
<b>6 Testen von Scala-Programmen</b>	<b>61</b>
6.1 Unit-Tests mit specs	61
6.1.1 Vorbereitung: Dependencies verwalten mit SBT	61
6.1.2 Testfälle einfach gemacht	64
6.1.3 Testdaten einfach gemacht	66
6.2 Test Coverage mit scct	68
6.3 Projekt-Code: aktueller Stand	71
<b>7 Erste Schritte mit FP</b>	<b>73</b>
7.1 Scala-Collections	74
7.1.1 Klassenhierarchie	74
7.1.2 Collection-Instanzen erzeugen	75
7.1.3 Typ-Parameter	76
7.1.4 Tupel	77
7.1.5 Unveränderliche und veränderliche Collections	78
7.1.6 Collections in ScalaTrain	80

7.2 Funktionale Collections	81
7.2.1 Funktionslitterale	81
7.2.2 Funktions-Typen	83
7.2.3 Funktionale Collections in ScalaTrain	85
7.2.4 map, flatMap und filter im Detail	90
7.3 For Expressions und For Loops	93
7.3.1 For Expressions	95
7.3.2 For Loops und foreach	98
7.4 Projekt-Code: aktueller Stand	100
<b>8 Vererbung und Traits</b>	<b>101</b>
8.1 Vererbung	101
8.1.1 Sub-Klassen mit extends definieren	101
8.1.2 Member überschreiben	104
8.1.3 Abstrakte Klassen und Member	107
8.1.4 Scala-Typhierarchie	110
8.2 Traits	112
8.2.1 Traits hinein mixen	114
8.2.2 Linearisierung	115
8.2.3 Beispiel: Ordered implementieren	118
8.2.4 Einschub: By-Name Parameters	119
8.2.5 Self Types	121
8.3 Projekt-Code: aktueller Stand	123
<b>9 Pattern Matching</b>	<b>127</b>
9.1 match-Ausdrücke	127
9.2 Welche Pattern gibt es?	128
9.2.1 Wildcard Pattern	128
9.2.2 Constant Pattern	128
9.2.3 Variable Pattern und Typed Pattern	128
9.2.4 Tuple Pattern	129
9.2.5 Constructor Pattern	129
9.2.6 Sequence Pattern	131
9.3 Pattern Guards und Variable Binding	132
9.4 Pattern Matching außerhalb von match-Ausdrücken	133
9.5 Projekt-Code: aktueller Stand	134

<b>10 Scala und XML</b>	<b>137</b>
10.1 XML-Literale	137
10.2 XML-Verarbeitung	138
10.3 XML für ScalaTrain	140
10.4 Projekt-Code: aktueller Stand	141
<b>11 Implicits</b>	<b>143</b>
11.1 Implicit Conversions	143
11.1.1 Implicit Conversions zum Expected Type	144
11.1.2 Implicit Conversions des Receivers	146
11.2 Implicit Parameters	149
11.3 Type Classes	152
11.4 Projekt-Code: aktueller Stand	155
<b>12 Fortgeschrittene Konzepte</b>	<b>159</b>
12.1 Rekursion	159
12.2 Upper Bounds und View Bounds	162
12.2.1 Einschub: Package Objects	162
12.2.2 Einschub: Varianz	163
12.2.3 Upper Bounds	164
12.2.4 View Bounds	165
12.3 Existential Types	166
12.4 Vertiefung objekt-funktionale Programmierung	168
12.4.1 Problemstellung	168
12.4.2 Lösungsansatz	169
12.4.3 Streckenabschnitte ermitteln	171
12.5 Verbindungen ermitteln	174
12.6 Projekt-Code: aktueller Stand	176
<b>13 Scala Libraries</b>	<b>181</b>
13.1 Validieren mit scalaz	181
13.2 Web-Applikationen mit Lift	186
13.2.1 Lift-Konfiguration	187
13.2.2 View First	188
13.2.3 Snippets	191
13.3 Abschluss – Endgültiger Stand	194
<b>Stichwortverzeichnis</b>	<b>201</b>