



Steps in Scala

An Introduction to
Object-Functional Programming

Christos K.K. Loverdos and
Apostolos Syropoulos

CAMBRIDGE

Contents

<i>Preface</i>	<i>page</i> xiii
1 Introduction	1
1.1 Object orientation	1
1.2 An overview of functional programming	7
1.3 Extendable languages	9
1.4 Scala: beyond the Java programming language	14
2 Core features	16
2.1 “Hello World!” in Scala	16
2.2 Scala’s basic types	18
2.3 Classes and objects	24
2.4 Some basic operators	29
2.5 Basic built-in control structures	32
2.6 Subclasses and inheritance	38
2.7 Functions	42
2.8 Arrays and tuples	48
2.9 Command line arguments	53
2.10 Sets	56
2.11 Hash tables	59
2.12 Memo functions	62
2.13 Lists	64
2.14 Strings	74
2.15 Regular expressions	76
2.16 Scientific computation with Scala	84
2.17 Inner classes	87
2.18 Packages	88
2.19 Documentation comments	90
2.20 Annotations	92

3	Advanced features	95
3.1	Playing with trees	95
3.2	More about pattern matching	103
3.2.1	Types of patterns	103
3.2.2	Sealed classes	107
3.2.3	Optional values	108
3.3	Traits and <i>mix-in</i> composition	109
3.4	Sorting objects	116
3.5	More on functions	118
3.6	Polymorphism	125
3.6.1	Types of polymorphism	125
3.6.2	Overloading	127
3.6.3	Implicit conversion: a form of coercion	128
3.6.4	Parametric polymorphism	131
3.6.5	More on implicit parameters	136
3.6.6	Inclusion polymorphism	137
3.6.7	Covariance, contravariance and invariance	138
3.6.8	Bounded polymorphism	140
3.6.9	Views and view bounds	143
3.6.10	Existential types	144
3.6.11	Type projections	147
3.6.12	Type erasure	147
3.7	Nominal and structural typing	148
3.8*	Higher order polymorphism	150
3.9	Streams are “infinite” lists!	156
3.10*	More on memo functions	158
3.11	Assertions	159
3.12	Setters and getters	161
3.13*	Monads	163
4	Parser builders	171
4.1	Language parsers	171
4.2	Scala’s parser builders	174
4.3	An interpreter for a toy language	177
4.4	Domain-specific languages	184
4.5	Monadic parsing	185
5	XML processing	187
5.1	What is XML?	187
5.2	Basic XML content manipulation	189
5.3	Producing XHTML content with Scala	193
5.4	XML input and output	196

5.5	XML searching à la Scala	197
5.6	XML pattern matching	199
6	GUI programming	202
6.1	“Hello World!” again!	202
6.2	Interactive GUI programming	207
6.3	Building a desktop calculator	212
6.4	Simple graphics with Scala	216
6.5	Creating pictorial data	225
6.6	Dialogs	231
6.7	Menus	238
6.7.1	Radio buttons	238
6.7.2	Check boxes	240
6.7.3	Combo boxes	243
6.7.4	Building a text editor with a menu bar and menus	250
6.8	Tabs	257
6.8.1	Simple tabs	257
6.8.2	User-disposable tabs	258
6.8.3	GUI lists, sliders, and split panes	263
6.9	More on text components	266
6.10	Tables	271
6.11	Applets	275
6.12	Functional graphics	280
7	Concurrent programming	283
7.1	Programming with threads: an overview	283
7.2	Animation with threads	289
7.3	Using mailboxes	293
7.4	Actors: basic ideas	295
7.5	Message passing with actors	298
7.6	Computing factorials with actors	303
8	On paths and a bit of algebraic abstraction	307
8.1	Path requirements	308
8.2	Path API	310
8.3	Empty paths	311
8.4	Unix paths	312
8.5	Windows paths	314
8.5.1	Simple paths	315
8.5.2	UNC paths	315
8.5.3	Drive absolute paths	315
8.6	Path factory	318
8.6.1	A few more utility methods	320

8.6.2	The factory method	322
8.6.3	Canonical paths	324
8.6.4	Combining paths	326
8.7	Notes on representation	326
8.8	Notes on visibility	327
8.9	Testing paths	327
8.9.1	User-friendliness	328
8.10	Algebraic abstractions	329
8.10.1	Semigroups	330
8.10.2	Monoids	331
9	Virtual files coming into existence	334
9.1	Types, requirements and API	335
9.1.1	Types	335
9.1.2	Design goals	335
9.1.3	VFS API	336
9.1.4	VFile API	339
9.2	Native file system	342
9.3	Memory file system	346
9.3.1	Memory VFS	347
9.3.2	Memory files and folders	348
9.4	Zip file system	351
9.4.1	Preliminaries	351
9.4.2	Zip VFS	354
9.4.3	Zip VFS factory object	357
9.4.4	A VFile that does not exist	357
9.4.5	Zip VFile	358
10	Compositional file matching	360
10.1	Matching files	360
10.2	A less procedural approach	362
10.3	Glob-style matching implementation	367
10.3.1	Remarks on a (non) pure-Scala implementation	370
10.4	Using glob-style matching	371
10.5	Going boolean	376
10.5.1	Less redundancy	377
10.6	Any level down the hierarchy	379
11	Searching, iterating, traversing	380
11.1	Traditional knowledge	380
11.1.1	Iterables	380
11.1.2	Traversables	381
11.1.3	Test trees and expected search results	382

11.2	Iterating the hierarchy	385
11.2.1	The shape of our data	385
11.2.2	Abstracting the ingredients	389
11.3	Traversing the hierarchy	397
11.4	Going on further	399
12	The expression problem	402
12.1	Introduction	402
12.2	Data and operations	403
12.3	Data-centric approach with subclassing	407
12.4	Operation-centric approach with subclassing	410
12.5	Generic operation-centric approach	412
12.6	Generic data-centric approach	415
12.7	OO decomposition with abstract types	417
12.8	Operation-centric decomposition with abstract types	421
12.9	Summary	424
13	A computer algebra system	426
13.1	Mechanical symbol manipulation	426
13.2	The grammar	427
13.3	Basic data model	428
13.4	Experimenting with the data model	429
13.5	Basic operations	430
13.5.1	Finding the derivative of a function	430
13.5.2	Simplifying an expression	432
13.5.3	Pretty-printing expressions	433
13.6	Putting it all together	436
13.7	Functions of more than one variable	437
13.8	Summary and further reading	437
Appendix A:	Multimedia processing	439
Appendix B:	Distributing a Scala application along with Scala itself	441
Appendix C:	Working with the compiler and the interpreter	449
Appendix D:	Scala's grammar	463
<i>References</i>		470
<i>Name index</i>		474
<i>Subject index</i>		475