



Objects First with Java

A Practical Introduction Using BlueJ

Third Edition



David J. Barnes
Michael Kölling

Contents

Foreword	xvii
Preface to the instructor	xviii
Guided tour	xxvi
List of projects discussed in detail in this book	xxviii
Acknowledgements	xxx
Part 1 Foundations of object orientation	1
Chapter 1 Objects and classes	3
1.1 Objects and classes	3
1.2 Creating objects	4
1.3 Calling methods	5
1.4 Parameters	6
1.5 Data types	7
1.6 Multiple instances	8
1.7 State	8
1.8 What is in an object?	9
1.9 Object interaction	10
1.10 Source code	11
1.11 Another example	13
1.12 Return values	13
1.13 Objects as parameters	13
1.14 Summary	15
Chapter 2 Understanding class definitions	17
2.1 Ticket machines	17
2.1.1 Exploring the behavior of a naïve ticket machine	18
2.2 Examining a class definition	19
2.3 Fields, constructors, and methods	21
2.3.1 Fields	23
2.3.2 Constructors	25
2.4 Passing data via parameters	26

2.5	Assignment	27
2.6	Accessor methods	28
2.7	Mutator methods	31
2.8	Printing from methods	32
2.9	Summary of the naïve ticket machine	34
2.10	Reflecting on the design of the ticket machine	35
2.11	Making choices: the conditional statement	36
2.12	A further conditional-statement example	39
2.13	Local variables	41
2.14	Fields, parameters, and local variables	42
2.15	Summary of the better ticket machine	43
2.16	Self-review exercises	43
2.17	Reviewing a familiar example	45
2.18	Summary	48
Chapter 3	Object interaction	52
3.1	The clock example	52
3.2	Abstraction and modularization	53
3.3	Abstraction in software	54
3.4	Modularization in the clock example	54
3.5	Implementing the clock display	55
3.6	Class diagrams versus object diagrams	56
3.7	Primitive types and object types	58
3.8	The <code>ClockDisplay</code> source code	59
3.8.1	Class <code>NumberDisplay</code>	59
3.8.2	String concatenation	61
3.8.3	The modulo operator	62
3.8.4	Class <code>ClockDisplay</code>	63
3.9	Objects creating objects	66
3.10	Multiple constructors	67
3.11	Method calls	68
3.11.1	Internal method calls	68
3.11.2	External method calls	69
3.11.3	Summary of the clock display	70
3.12	Another example of object interaction	70
3.12.1	The mail system example	71
3.12.2	The <code>this</code> key word	72
3.13	Using a debugger	74
3.13.1	Setting breakpoints	74
3.13.2	Single stepping	76
3.13.3	Stepping into methods	77
3.14	Method calling revisited	78
3.15	Summary	78

Chapter 4	Grouping objects	81
4.1	Grouping objects in flexible-size collections	81
4.2	A personal notebook	82
4.3	A first look at library classes	82
4.3.1	An example of using a library	83
4.4	Object structures with collections	84
4.5	Generic classes	86
4.6	Numbering within collections	87
4.7	Removing an item from a collection	88
4.8	Processing a whole collection	89
4.8.1	The for-each loop	90
4.8.2	The while loop	92
4.8.3	Iterating over a collection	95
4.8.4	Index access versus iterators	96
4.9	Summary of the notebook example	97
4.10	Another example: an auction system	98
4.10.1	The <code>Lot</code> class	98
4.10.2	The <code>Auction</code> class	99
4.10.3	Anonymous objects	102
4.10.4	Using collections	103
4.11	Flexible collection summary	105
4.12	Fixed-size collections	105
4.12.1	A log-file analyzer	106
4.12.2	Declaring array variables	108
4.12.3	Creating array objects	109
4.12.4	Using array objects	110
4.12.5	Analyzing the log file	111
4.12.6	The for loop	111
4.13	Summary	116
Chapter 5	More sophisticated behavior	119
5.1	Documentation for library classes	120
5.2	The <code>TechSupport</code> system	120
5.2.1	Exploring the <code>TechSupport</code> system	121
5.2.2	Reading the code	122
5.3	Reading class documentation	126
5.3.1	Interfaces versus implementation	127
5.3.2	Using library-class methods	128
5.3.3	Checking string equality	130
5.4	Adding random behavior	131
5.4.1	The <code>Random</code> class	131
5.4.2	Random numbers with limited range	132
5.4.3	Generating random responses	133
5.4.4	Reading documentation for parameterized classes	135

5.5	Packages and import	136
5.6	Using maps for associations	137
5.6.1	The concept of a map	138
5.6.2	Using a <code>HashMap</code>	138
5.6.3	Using a map for the <code>TechSupport</code> system	139
5.7	Using sets	141
5.8	Dividing strings	142
5.9	Finishing the <code>TechSupport</code> system	144
5.10	Writing class documentation	146
5.10.1	Using <code>javadoc</code> in <code>BlueJ</code>	146
5.10.2	Elements of class documentation	146
5.11	Public versus private	148
5.11.1	Information hiding	149
5.11.2	Private methods and public fields	149
5.12	Learning about classes from their interfaces	150
5.13	Class variables and constants	153
5.13.1	The <code>static</code> key word	153
5.13.2	Constants	154
5.14	Summary	155
Chapter 6	Well-behaved objects	158
6.1	Introduction	158
6.2	Testing and debugging	159
6.3	Unit testing within <code>BlueJ</code>	159
6.3.1	Using inspectors	164
6.3.2	Positive versus negative testing	165
6.4	Test automation	166
6.4.1	Regression testing	166
6.4.2	Automated checking of test results	169
6.4.3	Recording a test	171
6.4.4	Fixtures	173
6.5	Modularization and interfaces	174
6.6	A debugging scenario	176
6.7	Commenting and style	177
6.8	Manual walkthroughs	178
6.8.1	A high-level walkthrough	178
6.8.2	Checking state with a walkthrough	180
6.8.3	Verbal walkthroughs	182
6.9	Print statements	183
6.9.1	Turning debugging information on or off	185
6.10	Choosing a test strategy	186
6.11	Debuggers	187
6.12	Putting the techniques into practice	187
6.13	Summary	188

Chapter 7	Designing classes	189
7.1	Introduction	190
7.2	The <i>world-of-zuul</i> game example	191
7.3	Introduction to coupling and cohesion	193
7.4	Code duplication	194
7.5	Making extensions	197
7.5.1	The task	197
7.5.2	Finding the relevant source code	198
7.6	Coupling	199
7.6.1	Using encapsulation to reduce coupling	200
7.7	Responsibility-driven design	204
7.7.1	Responsibilities and coupling	204
7.8	Localizing change	206
7.9	Implicit coupling	207
7.10	Thinking ahead	210
7.11	Cohesion	211
7.11.1	Cohesion of methods	211
7.11.2	Cohesion of classes	212
7.11.3	Cohesion for readability	213
7.11.4	Cohesion for reuse	213
7.12	Refactoring	214
7.12.1	Refactoring and testing	215
7.12.2	An example of refactoring	215
7.13	Refactoring for language independence	218
7.13.1	Enumerated types	219
7.13.2	Further decoupling of the command interface	220
7.14	Design guidelines	222
7.15	Executing without BlueJ	223
7.15.1	Class methods	224
7.15.2	The main method	224
7.15.3	Limitations of class methods	225
7.16	Summary	225
Part 2 Application structures		227
Chapter 8	Improving structure with inheritance	229
8.1	The DoME example	229
8.1.1	DoME classes and objects	230
8.1.2	DoME source code	232
8.1.3	Discussion of the DoME application	238
8.2	Using inheritance	239
8.3	Inheritance hierarchies	240

8.4	Inheritance in Java	241
8.4.1	Inheritance and access rights	242
8.4.2	Inheritance and initialization	242
8.5	DoME: adding other item types	244
8.6	Advantages of inheritance (so far)	246
8.7	Subtyping	247
8.7.1	Subclasses and subtypes	248
8.7.2	Subtyping and assignment	249
8.7.3	Subtyping and parameter passing	250
8.7.4	Polymorphic variables	251
8.7.5	Casting	251
8.8	The <code>Object</code> class	253
8.9	Autoboxing and wrapper classes	254
8.10	The collection hierarchy	255
8.11	Summary	255
Chapter 9	More about inheritance	258
9.1	The problem: DoME's print method	258
9.2	Static type and dynamic type	260
9.2.1	Calling <code>print</code> from <code>Database</code>	261
9.3	Overriding	262
9.4	Dynamic method lookup	264
9.5	Super call in methods	267
9.6	Method polymorphism	268
9.7	Object methods: <code>toString</code>	268
9.8	Protected access	271
9.9	Another example of inheritance with overriding	273
9.10	Summary	276
Chapter 10	Further abstraction techniques	278
10.1	Simulations	278
10.2	The foxes-and-rabbits simulation	279
10.2.1	The <i>foxes-and-rabbits</i> project	280
10.2.2	The <code>Rabbit</code> class	282
10.2.3	The <code>Fox</code> class	285
10.2.4	The <code>Simulator</code> class: setup	288
10.2.5	The <code>Simulator</code> class: a simulation step	291
10.2.6	Taking steps to improve the simulation	292
10.3	Abstract classes	292
10.3.1	The <code>Animal</code> superclass	293
10.3.2	Abstract methods	294
10.3.3	Abstract classes	296

10.4	More abstract methods	298
10.5	Multiple inheritance	300
10.5.1	An Actor class	300
10.5.2	Flexibility through abstraction	302
10.5.3	Selective drawing	302
10.5.4	Drawable actors: multiple inheritance	302
10.6	Interfaces	303
10.6.1	An Actor interface	303
10.6.2	Multiple inheritance of interfaces	305
10.6.3	Interfaces as types	306
10.6.4	Interfaces as specifications	306
10.6.5	A further example of interfaces	308
10.6.6	Abstract class or interface?	309
10.7	Summary of inheritance	309
10.8	Summary	310
Chapter 11	Building graphical user interfaces	312
11.1	Introduction	312
11.2	Components, layout, and event handling	313
11.3	AWT and Swing	313
11.4	The ImageViewer example	314
11.4.1	First experiments: creating a frame	315
11.4.2	Adding simple components	317
11.4.3	Adding menus	318
11.4.4	Event handling	319
11.4.5	Centralized receipt of events	319
11.4.6	Inner classes	322
11.4.7	Anonymous inner classes	323
11.5	ImageViewer 1.0: the first complete version	325
11.5.1	Image-processing classes	326
11.5.2	Adding the image	327
11.5.3	Layout	328
11.5.4	Nested containers	331
11.5.5	Image filters	334
11.5.6	Dialogs	337
11.6	ImageViewer 2.0: improving program structure	338
11.7	ImageViewer 3.0: more interface components	343
11.7.1	Buttons	343
11.7.2	Borders	346
11.8	Further extensions	347
11.9	Another example: SoundPlayer	349
11.10	Summary	352

Chapter 12	Handling errors	354
12.1	The <i>address-book</i> project	355
12.2	Defensive programming	359
12.2.1	Client–server interaction	359
12.2.2	Argument checking	360
12.3	Server error reporting	361
12.3.1	Notifying the user	362
12.3.2	Notifying the client object	362
12.4	Exception-throwing principles	365
12.4.1	Throwing an exception	366
12.4.2	Exception classes	366
12.4.3	The effect of an exception	368
12.4.4	Unchecked exceptions	369
12.4.5	Preventing object creation	370
12.5	Exception handling	371
12.5.1	Checked exceptions: the throws clause	371
12.5.2	Catching exceptions: the try statement	372
12.5.3	Throwing and catching multiple exceptions	374
12.5.4	Propagating an exception	375
12.5.5	The finally clause	376
12.6	Defining new exception classes	377
12.7	Using assertions	378
12.7.1	Internal consistency checks	378
12.7.2	The assert statement	379
12.7.3	Guidelines for using assertions	380
12.7.4	Assertions and the BlueJ unit testing framework	381
12.8	Error recovery and avoidance	382
12.8.1	Error recovery	382
12.8.2	Error avoidance	383
12.9	Case study: text input/output	384
12.9.1	Readers, writers, and streams	385
12.9.2	The <i>address-book-io</i> project	385
12.9.3	Text output with <code>FileWriter</code>	388
12.9.4	Text input with <code>FileReader</code>	389
12.9.5	<code>Scanner</code> : reading input from the terminal	390
12.9.6	Object serialization	391
12.10	Summary	392
Chapter 13	Designing applications	393
13.1	Analysis and design	393
13.1.1	The verb/noun method	394
13.1.2	The cinema booking example	394

13.1.3	Discovering classes	394
13.1.4	Using CRC cards	395
13.1.5	Scenarios	396
13.2	Class design	399
13.2.1	Designing class interfaces	400
13.2.2	User interface design	401
13.3	Documentation	401
13.4	Cooperation	402
13.5	Prototyping	402
13.6	Software growth	403
13.6.1	Waterfall model	403
13.6.2	Iterative development	404
13.7	Using design patterns	405
13.7.1	Structure of a pattern	406
13.7.2	Decorator	406
13.7.3	Singleton	407
13.7.4	Factory method	408
13.7.5	Observer	408
13.7.6	Pattern summary	410
13.8	Summary	410
Chapter 14	A case study	413
14.1	The case study	413
14.1.1	The problem description	413
14.2	Analysis and design	414
14.2.1	Discovering classes	414
14.2.2	Using CRC cards	415
14.2.3	Scenarios	416
14.3	Class design	418
14.3.1	Designing class interfaces	418
14.3.2	Collaborators	418
14.3.3	The outline implementation	419
14.3.4	Testing	423
14.3.5	Some remaining issues	423
14.4	Iterative development	424
14.4.1	Development steps	424
14.4.2	A first stage	425
14.4.3	Testing the first stage	428
14.4.4	A later stage of development	429
14.4.5	Further ideas for development	431
14.4.6	Reuse	431

14.5	Another example	432
14.6	Taking things further	432
Appendices		433
A	Working with a BlueJ project	433
B	Java data types	435
C	Java control structures	438
D	Operators	443
E	Running Java without BlueJ	445
F	Configuring BlueJ	448
G	Using the debugger	450
H	JUnit unit-testing tools	454
I	Javadoc	456
J	Program style guide	459
K	Important library classes	463
Index		467

Website resources

For Students:

- Program style guide for all examples in the book
- Links to further material of interest

For Lecturers:

- PowerPoint slides that can be downloaded and used as OHTs
- Solutions to exercises
- Additional activities, exercises and projects for use in teaching