

Contents

- 1 The Term “Design Pattern” 1**
 - 1.1 What Are Design Patterns? 1
 - 1.1.1 Historical and Intellectual Background 1
 - 1.1.2 Advantages and Disadvantages of Patterns 2
 - 1.1.3 A Pattern Is Not a Code Is Not a Pattern 4
 - 1.1.4 So What Are “Design Patterns”? 5
 - 1.1.5 Patterns in Practice 5
 - 1.2 Categorize and Describe Design Patterns 6
 - 1.2.1 Different Categories of Patterns 6
 - 1.2.2 Defining a Sample Template 7
 - 1.3 Summary 10
- 2 Object-Oriented Programming and Design Principles 13**
 - 2.1 Object-Oriented Programming 13
 - 2.2 Programming Against Interfaces 16
 - 2.3 Single Responsibility Principle (SRP) 18
 - 2.4 Inheritance May Be Evil 19
 - 2.5 Open/Closed Principle (OCP) 20
 - 2.6 Principle of the Right Sense of Proportion 21
- 3 Singleton 23**
 - 3.1 The Task of the Singleton Pattern 23
 - 3.2 A First Version of the Singleton 24
 - 3.3 Synchronize Access 26
 - 3.4 Double Checked Locking 26
 - 3.5 Early Instantiation – Early Loading 28
 - 3.6 Singleton – The UML Diagram 29
 - 3.7 Antipattern 29
 - 3.7.1 Criticism of the Singleton 30
 - 3.7.2 Is Singleton an Antipattern? 30

3.8	Summary	30
3.9	Description of Purpose	31
4	Template Method	33
4.1	How Template Method Works	33
4.1.1	A First Approach	33
4.1.2	The Second Approach	34
4.1.3	The Hollywood Principle	35
4.1.4	Introducing Hook Methods	36
4.2	The “ListModel” Interface	37
4.3	Template Method – The UML Diagram	38
4.4	Summary	38
4.5	Description of Purpose	39
5	Observer.	41
5.1	Introduction.	41
5.2	A First Realization	41
5.3	Extending the Approach	43
5.4	Observer in the Java Class Library	45
5.5	Concurrent Access	45
5.5.1	Synchronize Accesses	47
5.5.2	Copying the Database	48
5.5.3	Use of a Thread-Safe List	49
5.6	Observer as Listener	49
5.7	Listener in GUI Programming	51
5.8	The Model-View-Controller Pattern	56
5.9	Observer – The UML Diagram.	57
5.10	Summary	57
5.11	Description of Purpose	59
6	Chain of Responsibility.	61
6.1	A Real World Example	61
6.2	First Code Example: Grocery Shopping.	61
6.2.1	The Foodstuffs Required	62
6.2.2	The Sellers	62
6.2.3	The Client.	64
6.3	An Example from the Class Library	66
6.4	Chain of Responsibility – The UML Diagram	68
6.5	Summary	69
6.6	Description of Purpose	69
7	Mediator.	71
7.1	Distinction from the Observer Pattern	71
7.2	Task of the Mediator Pattern.	72

7.3	Mediator in Action – An Example	72
7.3.1	Definition of a Consumer.	72
7.3.2	Definition of a Producer.	73
7.3.3	Mediator Interface	74
7.3.4	Testing the Mediator Pattern	76
7.4	Mediator in Action – The Second Example	77
7.4.1	Mediator in GUI Programming	78
7.4.2	Structure of the GUI.	78
7.5	Mediator – The UML Diagram.	80
7.6	Criticism of Mediator	81
7.7	Summary.	81
7.8	Description of Purpose	81
8	State	83
8.1	Excursus: The Enum Pattern.	83
8.1.1	Representing a State by Numerical Values.	83
8.1.2	Representing a State by Objects.	84
8.1.3	Implementation in the Java Class Library	85
8.2	Changing the State of an Object	86
8.2.1	A First Approach	86
8.2.2	A Second Approach	87
8.3	The Principle of the State Pattern	88
8.3.1	Defining the Role of All States.	89
8.3.2	The Project from the Client’s Point of View	90
8.3.3	Changes to the Project	92
8.4	The State Pattern in Practice.	93
8.5	State – The UML Diagram	94
8.6	Summary.	94
8.7	Description of Purpose	95
9	Command.	97
9.1	Encapsulating Commands in Classes	97
9.1.1	Version 1 – Basic Version.	97
9.1.2	Other Suppliers Appearing.	101
9.1.3	Encapsulating a Command.	102
9.2	Command in the Class Library	104
9.2.1	Example 1: Concurrency	104
9.2.2	Example 2: Event Handling	105
9.3	Reusing Command Objects.	106
9.3.1	The “Action” Interface	106
9.3.2	Use of the “Action” Interface.	107
9.4	Undo and Redo of Commands	108
9.4.1	A Simple Example	108
9.4.2	A More Extensive Example	111

9.4.3	Discussion of the Source Code	112
9.4.4	Undo and Redo Considered in Principle	114
9.5	Command – The UML Diagram	114
9.6	Summary	114
9.7	Description of Purpose	115
10	Strategy	117
10.1	A First Approach	117
10.2	Strategy in Action – Sorting Algorithms	118
10.2.1	The Common Interface	119
10.2.2	The Selection Sort	119
10.2.3	The Merge Sort	120
10.2.4	The Quick Sort	120
10.2.5	The Context	121
10.2.6	Evaluation of the Approach and Possible Variations	123
10.3	The Strategy Pattern in Practice	123
10.4	Strategy – The UML Diagram	125
10.5	Distinction from Other Designs	125
10.6	Summary	126
10.7	Description of Purpose	126
11	Iterator	127
11.1	Two Ways to Store Data	127
11.1.1	Storing Data in an Array	127
11.1.2	Storing Data in a Chain	129
11.2	The Task of an Iterator	131
11.3	The Interface Iterator in Java	132
11.3.1	The Iterator of the Class MyArray	132
11.3.2	The Iterator of the Class MyList	134
11.4	The Iterable Interface	135
11.5	Iterator – The UML Diagram	137
11.6	Summary	137
11.7	Description of Purpose	138
12	Composite	139
12.1	Principle of Composite	139
12.2	Implementation 1: Security	140
12.3	Implementation 2: Transparency	143
12.4	Consideration of the Two Approaches	145
12.5	Going One Step Further	147
12.5.1	Creating a Cache	147
12.5.2	Referencing the Parent Components	149
12.5.3	Moving Nodes	152

12.6	Composite – The UML Diagram	154
12.7	Summary	154
12.8	Description of Purpose	154
13	Flyweight	155
13.1	Task of the Pattern.	155
13.2	The Realization	157
13.3	A More Complex Project	158
13.3.1	The First Approach.	158
13.3.2	Intrinsic and Extrinsic State	159
13.4	Flyweight in Practice	162
13.5	Flyweight – The UML Diagram	163
13.6	Summary	163
13.7	Description of Purpose	164
14	Interpreter	165
14.1	The Task in This Chapter	165
14.2	The Scanner	167
14.2.1	The Defined Symbols	167
14.2.2	The Scanner Converts Strings into Symbols	168
14.3	The Parser	171
14.3.1	Abstract Syntax Trees.	171
14.3.2	Expressions for the Parser	172
14.3.3	Parse Stroke Calculation.	173
14.3.4	Parse Point Calculation.	176
14.3.5	Consider Brackets.	178
14.4	Interpreter – The UML Diagram.	179
14.5	Discussion of the Interpreter Pattern.	179
14.6	Summary	182
14.7	Description of Purpose	182
15	Abstract Factory (Abstract Factory).	183
15.1	Create Gardens	183
15.1.1	The First Attempt	183
15.1.2	The Second Attempt – Inheritance	185
15.1.3	The Third Approach – The Abstract Factory	185
15.1.4	Advantages of the Abstract Factory	187
15.1.5	Defining a New Garden	188
15.2	Discussion of the Pattern and Practice	189
15.3	Chasing Ghosts	189
15.3.1	The First Version.	190
15.3.2	The Second Version of the Project	196

15.3.3	Version 3 – Introduction of Another Factory	198
15.3.4	Version 4 – The Haunted House	200
15.4	Abstract Factory – The UML Diagram	202
15.5	Summary	202
15.6	Description of Purpose	202
16	Factory Method	205
16.1	A First Example	205
16.2	Variations of the Example	207
16.3	Practical Application of the Pattern	208
16.3.1	Recourse to the Iterator Pattern	208
16.3.2	At the Abstract Factory	209
16.4	A Larger Example – A Framework	210
16.4.1	And Another Calendar	210
16.4.2	The Interfaces for Entries and Their Editors	211
16.4.3	The Class “Contact” as an Example of an Entry	211
16.4.4	The FactoryMethod Class as a Client	213
16.5	Difference to Abstract Factory	214
16.6	Factory Method – The UML Diagram	214
16.7	Summary	215
16.8	Description of Purpose	216
17	Prototype	217
17.1	Cloning Objects	217
17.1.1	Criticism of the Implementation	218
17.1.2	Cloning in Inheritance Hierarchies	222
17.2	A Major Project	225
17.2.1	Discussion of the First Version	225
17.2.2	The Second Version – Deep Copy	228
17.2.3	Defining Your Own Prototypes	230
17.3	Prototype – The UML Diagram	230
17.4	Summary	230
17.5	Description of Purpose	232
18	Builder	233
18.1	An Object Creates Other Objects	233
18.1.1	Telescoping Constructor Pattern	233
18.1.2	JavaBeans Pattern	235
18.1.3	Builder Pattern	236
18.2	A More Complex Design Process	239
18.2.1	Converting XML Files into a TreeModel	241
18.2.2	Displaying XML Files as HTML	244
18.3	Builder – The UML Diagram	248

18.4	Summary	248
18.5	Description of Purpose	248
19	Visitor	249
19.1	A Simple Example	249
19.1.1	The Power Unit	249
19.1.2	The Visitor	251
19.1.3	The Client	253
19.1.4	Another Visitor	253
19.1.5	Criticism of the Project	254
19.2	Visitor – The UML Diagram	255
19.3	Summary	256
19.4	Description of Purpose	256
20	Memento	257
20.1	Task of the Memento Pattern	257
20.1.1	Public Data Fields	257
20.1.2	The JavaBeans Pattern	258
20.1.3	Default Visibility	258
20.2	A Possible Realization	259
20.3	A Major Project	261
20.4	Memento – The UML Diagram	263
20.5	Summary	264
20.6	Description of Purpose	264
21	Facade	265
21.1	An Example Outside IT	265
21.2	The Facade in a Java Example	266
21.2.1	Introduction of a Facade	268
21.2.2	A Closer Look at the Term “System”	269
21.3	The Facade in the Class Library	270
21.4	The “Law of Demeter”	271
21.5	Facade – The UML Diagram	272
21.6	Summary	272
21.7	Description of Purpose	273
22	Adapter	275
22.1	An Introductory Example	275
22.2	A Class-Based Design	275
22.3	An Object-Based Design	277
22.4	Criticism of the Adapter Pattern	278
22.5	A Labeling Fraud	279
22.6	Adapter – The UML Diagram	280
22.7	Summary	280
22.8	Description of Purpose	281

23	Proxy	283
23.1	Virtual Proxy	283
23.2	Security Proxy	284
23.3	Smart Reference	286
23.3.1	The Basic Version	286
23.3.2	Introduction of a Proxy	287
23.3.3	Introducing a Second Proxy	289
23.3.4	Dynamic Proxy	291
23.4	Remote Proxy	295
23.4.1	Structure of RMI in Principle	295
23.4.2	The RMI Server	295
23.4.3	The RMI Client	298
23.4.4	Making the Project Work	299
23.5	Proxy – The UML Diagram	300
23.6	Summary	300
23.7	Description of Purpose	301
24	Decorator	303
24.1	Build Cars	303
24.1.1	One Attribute for Each Optional Extra	303
24.1.2	Extending with Inheritance	304
24.1.3	Decorating According to the Matryoshka Principle	305
24.2	Practical Examples	307
24.2.1	The “JScrollPane” Class	308
24.2.2	Streams in Java	308
24.2.3	Streams as Non-Decorator	310
24.3	Decorator – The UML Diagram	312
24.4	Summary	312
24.5	Description of Purpose	313
25	Bridge	315
25.1	Two Definitions	315
25.1.1	What Is an Abstraction?	315
25.1.2	What Is an Implementation?	316
25.1.3	A Problem Begins to Mature	318
25.2	The Bridge Pattern in Use	319
25.2.1	First Step	319
25.2.2	Second Step	321
25.2.3	Extending the Implementation	322
25.3	Discussion of the Bridge Pattern	323
25.3.1	Bridge in the Wild	323
25.3.2	Distinction from Other Patterns	325
25.4	Bridge – The UML Diagram	325

25.5	Summary	326
25.6	Description of Purpose	326
26	Combine Patterns	327
26.1	The Example	327
26.2	Singleton: Logging	328
26.3	Abstract Factory: Building the Sensors	330
26.4	Observer: The Trigger	332
26.5	Adapter: Send Notifications	333
26.6	Command: Actuators and Remote Control	334
26.7	A Sensor Triggers	336
26.8	Class Diagram	337
26.9	The Client	337
26.10	Considerations	341
26.11	Summary	341
	Concluding Remarks	343