

# Contents

## Part I Basic Object-Oriented Concepts

<b>1</b>	<b>Introduction</b>	3
1.1	What Is Object-Oriented Development?	4
1.2	Key Concepts of Object-Oriented Design	5
1.3	Other Related Concepts	7
1.3.1	Modular Design and Encapsulation	7
1.3.2	Cohesion and Coupling	7
1.3.3	Modifiability and Testability	8
1.4	Benefits and Drawbacks of the Paradigm	8
1.5	History	9
1.6	Discussion and Further Reading	10
1.7	Exercises	11
	References	11
<b>2</b>	<b>Basics of Object-Oriented Programming</b>	13
2.1	The Basics	13
2.2	Implementing Classes	16
2.2.1	Constructors	20
2.2.2	Printing an Object	22
2.2.3	Static Members	23
2.3	Programming with Multiple Classes	25
2.4	Interfaces	28
2.4.1	Implementation of StudentLinkedList	30
2.4.2	Array Implementation of Lists	33
2.5	Abstract Classes	35
2.6	Comparing Objects for Equality	36
2.7	A Notation for Describing Object-Oriented Systems	37
2.7.1	Class Diagrams	41
2.7.2	Use Cases and Use Case Diagrams	41
2.7.3	Sequence Diagrams	42

2.8	Discussion and Further Reading . . . . .	45
2.9	Exercises . . . . .	48
	References. . . . .	48
<b>3</b>	<b>Relationships Between Classes . . . . .</b>	<b>49</b>
3.1	Association. . . . .	50
3.1.1	Characteristics of Associations . . . . .	51
3.2	Inheritance . . . . .	53
3.2.1	An Example of a Hierarchy . . . . .	53
3.2.2	Inheriting from an Interface . . . . .	58
3.2.3	Polymorphism and Dynamic Binding. . . . .	59
3.2.4	Protected Fields and Methods . . . . .	65
3.2.5	The Object Class . . . . .	67
3.3	Genericity . . . . .	67
3.4	Discussion and Further Reading . . . . .	69
3.4.1	A Generalised Notion of Conformance. . . . .	70
3.5	Exercises . . . . .	73
	References. . . . .	74
<b>4</b>	<b>Language Features for Object-Oriented Implementation. . . . .</b>	<b>75</b>
4.1	Organising the Classes. . . . .	75
4.1.1	Creating the Files . . . . .	76
4.1.2	Packages . . . . .	76
4.1.3	Protected Access and Package Access . . . . .	77
4.2	Collection Classes . . . . .	78
4.3	Exceptions . . . . .	79
4.4	Run-Time Type Identification. . . . .	81
4.4.1	Reflection: Using the Class Object . . . . .	82
4.4.2	Using the instanceof Operator. . . . .	83
4.4.3	Downcasting. . . . .	84
4.5	Graphical User Interfaces: Programming Support. . . . .	85
4.5.1	The Basics . . . . .	85
4.5.2	Event Handling. . . . .	88
4.5.3	More on Widgets and Layouts . . . . .	91
4.5.4	Drawing Shapes . . . . .	93
4.5.5	Displaying a Piece of Text . . . . .	93
4.6	Long-Term Storage of Objects . . . . .	94
4.6.1	Storing and Retrieving Objects . . . . .	96
4.6.2	Issues in Storing and Retrieving Objects. . . . .	97
4.6.3	The Java Serialization Mechanism. . . . .	99
4.7	Discussion and Further Reading . . . . .	101
4.8	Exercises . . . . .	104

## Part II Introduction to Object-Oriented Analysis, Design, Implementation and Refactoring

<b>5</b>	<b>Elementary Design Patterns</b>	109
5.1	Iterator	110
5.1.1	Iterator Implementation	113
5.2	Singleton	116
5.2.1	Subclassing Singletons	117
5.3	Adapter	120
5.4	Discussion and Further Reading	124
5.5	Exercises	126
	References	127
<b>6</b>	<b>Analysing a System</b>	129
6.1	Overview of the Analysis Phase	130
6.2	Stage 1: Gathering the Requirements	131
6.2.1	Case Study Introduction	132
6.3	Functional Requirements Specification	134
6.3.1	Use Case Analysis	134
6.4	Defining Conceptual Classes and Relationships	145
6.5	Using the Knowledge of the Domain	151
6.6	Discussion and Further Reading	153
6.7	Exercises	156
	References	158
<b>7</b>	<b>Design and Implementation</b>	159
7.1	Design	159
7.1.1	Major Subsystems	160
7.1.2	Creating the Software Classes	161
7.1.3	Assigning Responsibilities to the Classes	163
7.1.4	Class Diagrams	173
7.1.5	User Interface	178
7.1.6	Data Storage	179
7.2	Implementing Our Design	180
7.2.1	Setting Up the Interface	180
7.2.2	Adding New Books	181
7.2.3	Issuing Books	182
7.2.4	Printing Transactions	184
7.2.5	Placing and Processing Holds	185
7.2.6	Storing and Retrieving the Library Object	188
7.3	Discussion and Further Reading	192
7.3.1	Conceptual, Software and Implementation Classes	193
7.3.2	Building a Commercially Acceptable System	193
7.3.3	The Facade Pattern	195

7.3.4	Implementing Singletons . . . . .	197
7.3.5	Further Reading . . . . .	197
7.4	Exercises . . . . .	197
	References. . . . .	198
<b>8</b>	<b>How ‘Object-Oriented’ Is Our Design? . . . . .</b>	<b>199</b>
8.1	Introduction . . . . .	199
8.2	A First Example of Refactoring. . . . .	200
8.2.1	A Library that Charges Fines: Initial Solution . . . . .	200
8.2.2	Refactoring the Solution. . . . .	204
8.3	A Second Look at RemoveBooks . . . . .	208
8.4	Using Generics to Refactor Duplicated Code . . . . .	211
8.4.1	A Closer Look at the Collection Classes . . . . .	212
8.4.2	Instantiating Catalog and MemberList. . . . .	216
8.5	Discussion and Further Reading . . . . .	218
8.6	Exercises . . . . .	219
	Reference . . . . .	219

### Part III Advanced Concepts in Object-Oriented Design

<b>9</b>	<b>Exploring Inheritance . . . . .</b>	<b>223</b>
9.1	Introduction . . . . .	223
9.2	Applications of Inheritance. . . . .	224
9.2.1	Restricting Behaviours and Properties . . . . .	224
9.2.2	Abstract Superclass . . . . .	224
9.2.3	Adding Features . . . . .	225
9.2.4	Hiding Features of the Superclass . . . . .	226
9.2.5	Combining Structural and Type Inheritance . . . . .	227
9.3	Inheritance: Some Limitations and Caveats. . . . .	227
9.3.1	Deep Hierarchies. . . . .	227
9.3.2	Lack of Multiple Inheritance. . . . .	228
9.3.3	Changes in the Superclass . . . . .	228
9.3.4	Typing Issues: The Liskov Substitution Principle. . . . .	229
9.3.5	Addressing the Limitations . . . . .	232
9.4	Type Inheritance . . . . .	232
9.4.1	A Simple Example . . . . .	233
9.4.2	The Cloneable Interface . . . . .	234
9.4.3	The Runnable Interface . . . . .	237
9.5	Making Enhancements to the Library Class . . . . .	239
9.5.1	A First Attempt. . . . .	239
9.5.2	Drawbacks of the Above Approach . . . . .	242
9.6	Improving the Design . . . . .	244
9.6.1	Designing the Hierarchy. . . . .	244
9.6.2	Invoking the Constructors. . . . .	246

9.6.3	Distributing the Responsibilities . . . . .	250
9.6.4	Factoring Responsibilities Across the Hierarchy . . .	252
9.7	Consequences of Introducing Inheritance . . . . .	254
9.7.1	Exception Handling . . . . .	255
9.7.2	Adding New Functionality to a Hierarchy. . . . .	256
9.8	Multiple Inheritance. . . . .	260
9.8.1	Mechanisms for Resolving Conflicts . . . . .	263
9.8.2	Repeated Inheritance . . . . .	264
9.8.3	Multiple Inheritance in Java . . . . .	268
9.9	Discussion and Further Reading . . . . .	268
9.9.1	Design Patterns that Facilitate Inheritance. . . . .	269
9.9.2	Performance of Object-Oriented Systems . . . . .	270
9.10	Exercises . . . . .	271
	References. . . . .	272
<b>10</b>	<b>Modelling with Finite State Machines . . . . .</b>	<b>275</b>
10.1	Introduction . . . . .	275
10.2	A Simple Example . . . . .	275
10.3	Finite State Modelling . . . . .	277
10.4	A First Solution to the Microwave Problem . . . . .	279
10.4.1	Completing the Analysis . . . . .	279
10.4.2	Designing the System . . . . .	281
10.4.3	The Implementation Classes . . . . .	283
10.4.4	A Critique of the Above Design . . . . .	287
10.5	Using the State Pattern. . . . .	288
10.5.1	Creating the State Hierarchy . . . . .	289
10.5.2	Implementation . . . . .	295
10.6	Improving Communication Between Objects. . . . .	296
10.6.1	Loosely Coupled Communication . . . . .	296
10.7	Redesign Using the Observer Pattern. . . . .	298
10.7.1	Communication with the User. . . . .	299
10.7.2	The Improved Design . . . . .	301
10.8	Eliminating the Conditionals. . . . .	302
10.8.1	Using the Java Event Mechanism . . . . .	303
10.8.2	Using the Context As a ‘Switchboard’ . . . . .	306
10.8.3	Implementation . . . . .	308
10.9	Designing GUI Programs Using the State Pattern . . . . .	311
10.9.1	Design of a GUI System for the Library. . . . .	311
10.9.2	The Context . . . . .	314
10.10	Discussion and Further Reading . . . . .	315
10.10.1	Implementing the State Pattern . . . . .	315
10.10.2	Features of the State Pattern . . . . .	315
10.10.3	Consequences of Observer . . . . .	316

10.10.4	Recognising and Processing External Events . . . . .	317
10.10.5	Handling the Events . . . . .	318
10.11	Exercises . . . . .	321
	References . . . . .	322
<b>11</b>	<b>Interactive Systems and the MVC Architecture . . . . .</b>	<b>323</b>
11.1	Introduction . . . . .	323
11.2	The MVC Architectural Pattern . . . . .	324
11.2.1	Examples . . . . .	326
11.2.2	Implementation . . . . .	326
11.2.3	Benefits of the MVC Pattern . . . . .	328
11.3	Analysing a Simple Drawing Program . . . . .	328
11.3.1	Specifying the Requirements . . . . .	328
11.3.2	Defining the Use Cases . . . . .	329
11.4	Designing the System . . . . .	331
11.4.1	Defining the Model . . . . .	332
11.4.2	Defining the Controller . . . . .	332
11.4.3	Selection and Deletion . . . . .	338
11.4.4	Saving and Retrieving the Drawing . . . . .	339
11.5	Design of the Subsystems . . . . .	339
11.5.1	Design of the Model Subsystem . . . . .	340
11.5.2	Design of Item and Its Subclasses . . . . .	341
11.5.3	Design of the Controller Subsystem . . . . .	348
11.5.4	Design of the View Subsystem . . . . .	349
11.6	Getting into the Implementation . . . . .	352
11.6.1	Item and Its Subclasses . . . . .	352
11.6.2	Implementation of the Model Class . . . . .	354
11.6.3	Implementation of the Controller Class . . . . .	355
11.6.4	Implementation of the View Class . . . . .	356
11.6.5	The Driver Program . . . . .	359
11.6.6	A Critique of Our Design . . . . .	359
11.7	Implementing the Undo Operation . . . . .	360
11.7.1	Employing the Command Pattern . . . . .	364
11.7.2	Implementation . . . . .	368
11.8	Drawing Incomplete Items . . . . .	371
11.9	Adding a New Feature . . . . .	374
11.10	Pattern-Based Solutions . . . . .	377
11.10.1	Examples of Architectural Patterns . . . . .	379
11.11	Discussion and Further Reading . . . . .	380
11.11.1	Separating the View and the Controller . . . . .	381
11.11.2	The Space Overhead for the Command Pattern . . . . .	381
11.11.3	How to Store the Items . . . . .	382

11.11.4	Exercising Caution When Allowing Undo . . . . .	382
11.11.5	Synchronising Updates . . . . .	383
11.12	Exercises . . . . .	384
	References . . . . .	385
<b>12</b>	<b>Designing with Distributed Objects . . . . .</b>	<b>387</b>
12.1	Client/Server Systems . . . . .	388
12.1.1	Basic Architecture of Client/Server Systems . . . . .	388
12.2	Java Remote Method Invocation . . . . .	390
12.2.1	Remote Interfaces . . . . .	391
12.2.2	Implementing a Remote Interface . . . . .	392
12.2.3	Creating the Server . . . . .	394
12.2.4	The Client . . . . .	395
12.2.5	Setting up the System . . . . .	396
12.3	Implementing an Object-Oriented System on the Web . . . . .	397
12.3.1	HTML and Java Servlets . . . . .	397
12.3.2	Deploying the Library System on the World-Wide Web . . . . .	402
12.4	Discussion and Further Reading . . . . .	424
12.5	Exercises . . . . .	425
	References . . . . .	425
<b>13</b>	<b>The Unified Modelling Language . . . . .</b>	<b>427</b>
13.1	Communication Diagrams . . . . .	429
13.1.1	Specification-Level Communication Diagrams . . . . .	429
13.1.2	Instance-Level Communication Diagrams . . . . .	431
13.2	Timing Diagrams . . . . .	432
13.3	Activity Diagrams . . . . .	436
13.4	Interaction Overview Diagrams . . . . .	439
13.5	Component Diagrams . . . . .	440
13.5.1	Usage . . . . .	442
13.6	Composite Structure Diagrams . . . . .	443
13.7	Package Diagrams . . . . .	446
13.8	Object Diagrams . . . . .	449
13.9	Deployment Diagrams . . . . .	450
13.10	Discussion and Further Reading . . . . .	452
	Reference . . . . .	453
	<b>Appendix: Java Essentials . . . . .</b>	<b>455</b>
	<b>Index . . . . .</b>	<b>467</b>

Object-Oriented Analysis, Design and Implementation

An Integrated Approach

Dathan, B.; Ramnath, S.

2015, XIX, 471 p. 166 illus. in color., Softcover

ISBN: 978-3-319-24278-1