

Object-Oriented  
Design Using

JAVA

Dale Skrien

# Contents

## CHAPTER 1 Elegance in Object-Oriented Design and Implementation 1

- Section 1.0 Introduction 1
- Section 1.1 Why Worry? 2
- Section 1.2 Software Engineering 4
- Section 1.3 Criteria for Elegant Software 5
- Section 1.4 Road Map 7

This chapter lays the groundwork for the rest of the text. It motivates the study of object-oriented design and gives examples of software failures that could possibly have been prevented if the software had been designed better. It includes the properties that software should have in order to be called “elegant.” It finishes with a summary of the contents of the other chapters in the text.

## CHAPTER 2 Fundamentals of Object Orientation 9

- Section 2.0 Introduction 9
- Section 2.1 Object-Oriented Programming vs. Non-Object-Oriented Programming 9
  - Overview of OO vs. Non-OO Programming 9
  - Object-Oriented Languages 10
  - Advantages of OO programming 11
- Section 2.2 Classes, Objects, Variables, and Methods in Java 12
- Section 2.3 Aside: Class Methods & Variables in Java 14
  - Introduction to Class Variables and Methods 14
  - Class Variables in Java and Their Uses 15
  - Class Methods in Java and Their Uses 15
  - Summary 16

- Section 2.4 Brief Introduction to UML Class Diagrams 16
- Section 2.5 Implementation Inheritance 18
  - Specialization 18
  - The Object Superclass in Java 20
  - Another Use of Specialization 21
  - Generalization 22
  - Single Inheritance in Java 23
- Section 2.6 Types, Subtypes, and Interface Inheritance 24
  - Type 25
  - Polymorphism 26
  - The Value of Polymorphism 27
- Section 2.7 Interfaces vs. Abstract Classes 30
- Section 2.8 Dynamic Method Invocation 31
- Section 2.9 Overloading vs. Overriding 35
- Section 2.10 Controlling Access to Methods and Data (Public, Private, Protected, Package) 39
- Section 2.11 Summary 41

This chapter reviews the basics of object-oriented programming and the advantages that such programming can provide to the programmer and software designer for making the software more flexible, extendable, reusable, and scalable.

## CHAPTER 3 Elegance and Implementation Inheritance 45

- Section 3.0 Introduction 45
- Section 3.1 Four Perspectives on Inheritance 46
  - Code Reuse Perspective 46
  - Is-A Perspective 46
  - Public Interface Perspective 46
  - Polymorphism Perspective 46
- Section 3.2 Sufficiency of Code Reuse 46

<b>Section 3.3</b>	<b>Sufficiency of Code Reuse and the Is-A Relationship</b>	<b>47</b>
<b>Section 3.4</b>	<b>Sufficiency of Code Reuse, the Is-A Relationship, and Public Interfaces</b>	<b>53</b>
<b>Section 3.5</b>	<b>Has-A Relationships and UML Associations</b>	<b>56</b>
<b>Section 3.6</b>	<b>Sufficiency of Code Reuse, the Is-A Relationship, Public Interfaces, and Polymorphism</b>	<b>56</b>
<b>Section 3.7</b>	<b>Costs of Using Implementation Inheritance</b>	<b>57</b>
<b>Section 3.8</b>	<b>Example: Person, Woman, and Man</b>	<b>60</b>
<b>Section 3.9</b>	<b>Example: Drawing Polygons</b>	<b>61</b>
<b>Section 3.10</b>	<b>Example: Sorting</b>	<b>64</b>
<b>Section 3.11</b>	<b>Subclassing Arrays in Java</b>	<b>74</b>
<b>Section 3.12</b>	<b>Inheritance vs. Referencing Revisited</b>	<b>75</b>
<b>Section 3.13</b>	<b>Summary</b>	<b>78</b>

This chapter expands on the material in the preceding chapter to discuss more thoroughly the role of inheritance in object-oriented software design, including its advantages and disadvantages, and to discuss alternatives to inheritance.

## **CHAPTER 4**

### **Elegance and Methods 82**

<b>Section 4.0</b>	<b>Introduction</b>	<b>82</b>
<b>Section 4.1</b>	<b>Coding Styles and Naming Conventions</b>	<b>83</b>
<b>Section 4.2</b>	<b>Methods and Decomposition</b>	<b>85</b>
<b>Section 4.3</b>	<b>Cohesive Methods</b>	<b>87</b>
<b>Section 4.4</b>	<b>Well-Formed Objects and Class Invariants</b>	<b>90</b>
<b>Section 4.5</b>	<b>Internal Documentation</b>	<b>91</b>
<b>Section 4.6</b>	<b>External Documentation</b>	<b>93</b>
<b>Section 4.7</b>	<b>Case Study: Overriding the Equals Method in Java</b>	<b>98</b>
<b>Section 4.8</b>	<b>Case Study: Overriding the Clone Method in Java</b>	<b>106</b>
<b>Section 4.9</b>	<b>Refactoring</b>	<b>110</b>
<b>Section 4.10</b>	<b>Code Optimization</b>	<b>120</b>
<b>Section 4.11</b>	<b>Summary and Further Reading</b>	<b>121</b>

This chapter looks at issues concerning low-level code, such as readability, modifiability, and reusability. In this chapter, we assume that the classes and their desired behaviors have already been mapped out, and only the implementation of those behaviors remains to be done.

## **CHAPTER 5**

### **Elegance and Classes 128**

<b>Section 5.0</b>	<b>Introduction</b>	<b>128</b>
<b>Section 5.1</b>	<b>Starting Out Finding Classes and Their Relationships</b>	<b>128</b>
	Extract Nouns and Verbs	131
	Use Concepts from the Application Domain	132
	Use CRC Cards	132
	Class Protocols	135
	The Big Picture	137
<b>Section 5.2</b>	<b>Maximizing Cohesion</b>	<b>139</b>
<b>Section 5.3</b>	<b>Separation of Responsibility</b>	<b>140</b>
<b>Section 5.4</b>	<b>Duplication Avoidance</b>	<b>144</b>
<b>Section 5.5</b>	<b>Complete and Consistent Protocols</b>	<b>147</b>
<b>Section 5.6</b>	<b>Mutability vs. Immutability Revisited</b>	<b>151</b>
<b>Section 5.7</b>	<b>Designing for Change</b>	<b>154</b>
<b>Section 5.8</b>	<b>Law of Demeter</b>	<b>161</b>
<b>Section 5.8</b>	<b>Summary and Further Reading</b>	<b>165</b>

This chapter looks at object-oriented software development at a higher level. We discuss how to design classes to solve particular problems. That is, we discuss general principles to follow when deciding what classes to create, what their behavior will be, and with what other objects they will communicate.

## **CHAPTER 6**

### **Simple Case Study of a Money Class 173**

<b>Section 6.0</b>	<b>Introduction</b>	<b>173</b>
<b>Section 6.1</b>	<b>Naive Representations of Money</b>	<b>173</b>
<b>Section 6.2</b>	<b>A USMoney Class</b>	<b>175</b>
<b>Section 6.3</b>	<b>Using Subclasses of Money to Represent Different Currencies</b>	<b>177</b>

<b>Section 6.4</b>	<b>Using One Class of Money with a Currency Attribute</b>	<b>179</b>
<b>Section 6.5</b>	<b>Mixed Currencies vs. Simple Currencies</b>	<b>182</b>
<b>Section 6.6</b>	<b>Converting Between Currencies</b>	<b>184</b>
<b>Section 6.7</b>	<b>MoneyConverter Issues</b>	<b>185</b>
<b>Section 6.8</b>	<b>MixedMoney and SimpleMoney Issues</b>	<b>187</b>
<b>Section 6.9</b>	<b>Mixed Money Only</b>	<b>188</b>
<b>Section 6.10</b>	<b>Alternate Implementation with Binary Trees</b>	<b>189</b>
<b>Section 6.11</b>	<b>Summary</b>	<b>192</b>

This chapter is a small case study. It introduces several implementations of money, each a refinement of the preceding one.

## **CHAPTER 7**

### **Introduction to Design Patterns 196**

<b>Section 7.0</b>	<b>Introduction</b>	<b>196</b>
<b>Section 7.1</b>	<b>The Adapter Pattern</b>	<b>197</b>
<b>Section 7.2</b>	<b>The Singleton Pattern</b>	<b>201</b>
<b>Section 7.3</b>	<b>The Iterator Pattern</b>	<b>204</b>
<b>Section 7.4</b>	<b>The Command Pattern</b>	<b>210</b>
<b>Section 7.5</b>	<b>Factories</b>	<b>214</b>
<b>Section 7.6</b>	<b>Summary</b>	<b>217</b>

This chapter introduces the reader to the topic of design patterns. We present four simple patterns as samples. The discussion of each pattern includes examples from earlier chapters of the book where the pattern was used but not explicitly stated. The succeeding chapters in the text introduce more design patterns in the contexts of their case studies.

## **CHAPTER 8**

### **Figure-Drawing Application Case Study 220**

<b>Section 8.0</b>	<b>Introduction</b>	<b>220</b>
<b>Section 8.1</b>	<b>The User Interface</b>	<b>221</b>
<b>Section 8.2</b>	<b>The Observer Pattern</b>	<b>223</b>
<b>Section 8.3</b>	<b>The Figure Hierarchy</b>	<b>230</b>
<b>Section 8.4</b>	<b>The Model-View-Controller Architecture</b>	<b>234</b>

<b>Section 8.5</b>	<b>The Prototype Pattern</b>	<b>238</b>
<b>Section 8.6</b>	<b>The State Pattern</b>	<b>239</b>
<b>Section 8.7</b>	<b>The Composite Pattern</b>	<b>244</b>
<b>Section 8.8</b>	<b>The Memento Pattern</b>	<b>248</b>
<b>Section 8.9</b>	<b>Summary</b>	<b>254</b>

This chapter gives a case study of a drawing application. We start with a very simple application, and in each section we add enhancements to it, which provides us with a context in which to introduce more design patterns and to use the design principles discussed earlier in the text. The sections include the discussion of several alternative designs and implementations of the application and their advantages and disadvantages.

## **CHAPTER 9**

### **Language Parser Case Study 258**

<b>Section 9.0</b>	<b>Introduction</b>	<b>258</b>
<b>Section 9.1</b>	<b>VSSJ: A Very Simple Subset of Java</b>	<b>258</b>
<b>Section 9.2</b>	<b>Pretty Printing</b>	<b>259</b>
<b>Section 9.3</b>	<b>Scanning</b>	<b>260</b>
<b>Section 9.4</b>	<b>A Simple Pretty Printer</b>	<b>262</b>
<b>Section 9.5</b>	<b>Interpreter Pattern</b>	<b>265</b>
<b>Section 9.6</b>	<b>Design of the AST</b>	<b>267</b>
<b>Section 9.7</b>	<b>Method Finder</b>	<b>274</b>
<b>Section 9.8</b>	<b>Some Problems with These Elegant Implementations</b>	<b>276</b>
<b>Section 9.9</b>	<b>The Visitor Pattern</b>	<b>279</b>
<b>Section 9.10</b>	<b>Visitors and Double-Dispatching</b>	<b>285</b>
<b>Section 9.11</b>	<b>Facade Pattern</b>	<b>286</b>
<b>Section 9.12</b>	<b>Parsers and Builders</b>	<b>287</b>
<b>Section 9.13</b>	<b>Tokens, Visitors, and Polymorphism (Optional Section)</b>	<b>292</b>
<b>Section 9.14</b>	<b>Summary</b>	<b>299</b>

This chapter gives a case study of code manipulators for a subset of the Java language, which provides us with a context in which to introduce more design patterns.

## **APPENDIX A**

### **An Introduction to UML 303**

<b>Section A.0</b>	<b>Introduction</b>	<b>303</b>
<b>Section A.1</b>	<b>Class Diagrams</b>	<b>304</b>

<b>Section A.2</b>	<b>Sequence Diagrams</b>	<b>307</b>
<b>Section A.3</b>	<b>State Machine Diagrams</b>	<b>310</b>
<b>Section A.4</b>	<b>Use Case Diagrams</b>	<b>313</b>
<b>Section A.5</b>	<b>Summary</b>	<b>316</b>

This appendix explains how to understand and use the four UML diagrams that appear in this text.

## **APPENDIX B**

### **Coding Conventions and Javadoc Comments 318**

<b>Section B.0</b>	<b>Introduction</b>	<b>318</b>
<b>Section B.1</b>	<b>Indentation and Spacing</b>	<b>318</b>

<b>Section B.2</b>	<b>Punctuation and Layout</b>	<b>319</b>
<b>Section B.3</b>	<b>Formatting a Loop</b>	<b>322</b>
<b>Section B.4</b>	<b>Incrementing Integer Variables</b>	<b>323</b>
<b>Section B.5</b>	<b>Working with Boolean Variables</b>	<b>324</b>
<b>Section B.6</b>	<b>Line and Block Comments</b>	<b>326</b>
<b>Section B.7</b>	<b>File Layout</b>	<b>326</b>
<b>Section B.8</b>	<b>Javadoc Syntax</b>	<b>328</b>
<b>Section B.9</b>	<b>Summary</b>	<b>331</b>

This appendix covers some of the material in Sun's coding conventions for Java. It covers Javadoc comments in quite a bit of detail.

<b>Index</b>	<b>333</b>
--------------	------------