



AN INTRODUCTION TO PROGRAMMING  
AND OBJECT ORIENTED DESIGN

USING JAVA 1.5

SECOND EDITION

WILEY  
INTERNATIONAL  
EDITION

**RESTRICTED!**  
not for sale in  
the United States

JAIME NIÑO / FREDERICK A. HOSCH

# CONTENTS

---

<b>preface</b>		<b>vii</b>
<b>Purpose and approach</b> .....		vii
<b>Overview</b> .....		viii
<b>Supporting material</b> .....		x
<b>To the instructor</b> .....		xi
<b>Changes since the first edition</b> .....		xv
<b>Finally</b> .....		xvi
<b>contents</b>		<b>xvii</b>
<b>chapter 0</b>	<b>Introduction to object-oriented software design</b>	<b>1</b>
<b>0.1</b>	<b>What is a software system?</b> .....	<b>3</b>
0.1.1	Dealing with complexity: composition and abstraction .....	3
0.1.2	Two aspects of a system: data and functionality .....	6
<b>0.2</b>	<b>Object-oriented systems</b> .....	<b>8</b>
<b>0.3</b>	<b>Summary</b> .....	<b>10</b>
<b>chapter 1</b>	<b>Data abstraction: introductory concepts</b>	<b>15</b>
<b>1.1</b>	<b>Objects</b> .....	<b>16</b>
1.1.1	Data and state .....	16
1.1.2	Functionality: queries and commands .....	19
1.1.3	Some cautions .....	20
<b>1.2</b>	<b>Values and types</b> .....	<b>22</b>
1.2.1	Values and types in Java .....	22
1.2.2	Types and variables .....	24
<b>1.3</b>	<b>Classes</b> .....	<b>26</b>
<b>1.4</b>	<b>An example</b> .....	<b>27</b>
<b>1.5</b>	<b>Reference values: objects as properties of objects</b> .....	<b>32</b>
<b>1.6</b>	<b>Overview of a complete system</b> .....	<b>38</b>
1.6.1	An example: testing a <i>Counter</i> .....	39

1.7	Java in detail: primitive types	43
1.8	Java in detail: identifiers and literals	44
1.8.1	Identifiers	44
1.8.2	Literals	47
1.8.3	Java in detail: literals, the rest of the story	50
1.9	Java in detail: lexical structure	51
1.10	Summary	53

## chapter 2

	<b>Defining a simple class</b>	<b>61</b>
2.1	Object interaction: clients and servers	62
2.1.1	Specification and implementation	63
2.2	Defining a class: a simple counter	65
2.2.1	Specifying the class features	66
2.2.2	Static diagrams	70
2.2.3	Invoking a method or constructor	71
2.2.4	Interaction diagrams	72
2.3	Implementing the class <i>Counter</i>	72
2.3.1	Implementing data	74
2.3.2	Implementing functionality	75
2.3.3	Documenting specification	81
2.4	Simple arithmetic expressions	83
2.5	The class <i>TrafficSignal</i>	84
2.5.1	Named constants	85
2.5.2	Implementing the class <i>TrafficSignal</i>	87
2.6	The class <i>PlayingCard</i>	92
2.6.1	Constructor with parameters	93
2.7	Java in detail: arithmetic expressions	99
2.8	Java in detail: basic organizational structure	104
2.9	Java in detail: referring to classes in a different package, <i>import</i> statements and the class <i>java.lang</i>	106
2.10	Java in detail: arithmetic expressions, the rest of the story	108
2.11	Summary	112

## chapter 3

	<b>Designing interacting classes</b>	<b>125</b>
3.1	Designing with objects	126
3.2	A nim game example	127
3.2.1	Implementing the class <i>Pile</i>	132
3.2.2	Implementing the class <i>Player</i>	134
3.3	A maze game example	140

	3.3.1	Implementing the class <i>Explorer</i> . . . . .	146
<b>3.4</b>		<b>Local variables: another kind of method variable</b> . . . . .	148
<b>3.5</b>		<b>Putting together a complete system</b> . . . . .	152
	3.5.1	Getting it started . . . . .	155
	3.5.2	The method <i>toString</i> . . . . .	159
<b>3.6</b>		<b>Testing an implementation</b> . . . . .	159
	3.6.1	A class to test . . . . .	160
	3.6.2	The <i>TrafficSignalTest</i> class . . . . .	161
	3.6.3	The initiating class . . . . .	162
	3.6.4	Writing the tests . . . . .	163
<b>3.7</b>		<b>Java in detail: <i>static</i> and <i>final</i> features</b> . . . . .	169
	3.7.1	<i>Static</i> methods and the class <i>Math</i> . . . . .	169
	3.7.2	<i>Final</i> features . . . . .	171
<b>3.8</b>		<b>Summary</b> . . . . .	173
<b>chapter 4</b>		<b>Conditions</b> . . . . .	<b>179</b>
<b>4.1</b>		<b>Conditional statements</b> . . . . .	180
	4.1.1	The <i>if-then</i> statement . . . . .	183
	4.1.2	The <i>if-then-else</i> statement . . . . .	184
	4.1.3	Compound statements . . . . .	190
<b>4.2</b>		<b>Boolean expressions</b> . . . . .	192
<b>4.3</b>		<b>Handling multiple cases</b> . . . . .	193
	4.3.1	Dangling <i>else</i> . . . . .	199
<b>4.4</b>		<b>Example: a combination lock</b> . . . . .	200
	4.4.1	A digit-by-digit lock . . . . .	211
<b>4.5</b>		<b>Java in detail: Boolean expressions</b> . . . . .	216
	4.5.1	Reference equality and object equality . . . . .	219
	4.5.2	Boolean operators . . . . .	221
<b>4.6</b>		<b>Java in detail: logical operators and conditional expressions</b> . . . . .	224
<b>4.7</b>		<b>Java in detail: the <i>switch</i> statement</b> . . . . .	225
<b>4.8</b>		<b>Summary</b> . . . . .	228
<b>chapter 5</b>		<b>Programming by contract</b> . . . . .	<b>239</b>
<b>5.1</b>		<b>Programming by contract</b> . . . . .	240
	5.1.1	Verifying preconditions: the <i>assert</i> statement . . . . .	243
<b>5.2</b>		<b>Further examples</b> . . . . .	247
<b>5.3</b>		<b>Preconditions and postconditions: a summary</b> . . . . .	252
<b>5.4</b>		<b>Summary</b> . . . . .	253

<b>chapter 6</b>	<b>Testing</b>	<b>261</b>
	6.1 Functional testing and unit testing . . . . .	262
	6.2 Developing a test plan . . . . .	264
	6.3 An example . . . . .	265
	6.4 Summary . . . . .	274
<b>chapter 7</b>	<b>Building a text-based user interface: iteration and composition</b>	<b>281</b>
	7.1 Relating the user interface and the model . . . . .	282
	7.2 Stream-based I/O . . . . .	284
	7.2.1 Writing standard output and reading standard input . . . . .	285
	7.3 An example: building an interface for <i>Rectangle</i> . . . . .	291
	7.3.1 Repeating actions: the <i>while</i> statement . . . . .	293
	7.3.2 Completing the <i>createRectangle</i> method . . . . .	295
	7.3.3 Finishing the user interface . . . . .	296
	7.4 A second example: using composition . . . . .	302
	7.4.1 Constructing a class by composition . . . . .	304
	7.4.2 Implementing the user interface . . . . .	307
	7.5 Java in detail: the <i>do</i> statement . . . . .	310
	7.6 Summary . . . . .	312
<b>chapter 8</b>	<b>Putting a system together</b>	<b>323</b>
	8.1 Software life cycle . . . . .	324
	8.2 Design of a system . . . . .	325
	8.2.1 Functional specification . . . . .	326
	8.2.2 Preliminary design . . . . .	327
	8.2.3 Relations between objects . . . . .	331
	8.2.4 Class specification . . . . .	333
	8.3 Implementing the system . . . . .	338
	8.3.1 The top level . . . . .	339
	8.3.2 The class <i>Pile</i> . . . . .	339
	8.3.3 The class <i>Player</i> . . . . .	339
	8.3.4 The class <i>Game</i> . . . . .	344
	8.3.5 The class <i>NimTUI</i> . . . . .	346
	8.4 Summary . . . . .	351
<b>chapter 9</b>	<b>Interfaces</b>	<b>359</b>
	9.1 Modeling alternative implementations . . . . .	360
	9.2 Interfaces . . . . .	362
	9.2.1 Interface definition . . . . .	362

9.2.2	Interface implementation	364
9.2.3	Subtyping	366
9.2.4	Putting it together	372
<b>9.3</b>	<b>Clients and interfaces</b>	<b>373</b>
<b>9.4</b>	<b>Multiple inheritance and interface extension</b>	<b>376</b>
<b>9.5</b>	<b>Modifying the system: user vs. computer</b>	<b>378</b>
9.5.1	Player classes	379
9.5.2	The user interface	380
9.5.3	User interface – model interaction	381
<b>9.6</b>	<b>Reclaiming lost ground: the strategy pattern</b>	<b>387</b>
<b>9.7</b>	<b>Java in detail: casting and <i>instanceof</i></b>	<b>390</b>
<b>9.8</b>	<b>Summary</b>	<b>395</b>

## **chapter 10 Inheritance 407**

<b>10.1</b>	<b>Abstraction and classes</b>	<b>408</b>
10.1.1	Class inheritance is single inheritance	409
<b>10.2</b>	<b>Extension and inheritance</b>	<b>410</b>
<b>10.3</b>	<b>Constructors and subclasses</b>	<b>415</b>
<b>10.4</b>	<b>Overloading, overriding, and polymorphism</b>	<b>418</b>
10.4.1	Method overloading	418
10.4.2	Method overriding	422
10.4.3	Overriding and contracts	429
<b>10.5</b>	<b>Java in detail: feature accessibility</b>	<b>431</b>
10.5.1	Protected features	434
10.5.2	Package private features	436
10.5.3	Inner classes	437
<b>10.6</b>	<b>Java in detail: scoping rules</b>	<b>441</b>
<b>10.7</b>	<b>Summary</b>	<b>445</b>

## **chapter 11 Modeling with abstraction 459**

<b>11.1</b>	<b>Abstract classes</b>	<b>460</b>
11.1.1	Interfaces, abstract classes, and concrete classes	461
<b>11.2</b>	<b>Specifying a class for extension</b>	<b>464</b>
11.2.1	Planning for extension	467
<b>11.3</b>	<b>Composition revisited</b>	<b>472</b>
11.3.1	Extension and composition	473
11.3.2	Extension, composition, and reuse	475
11.3.3	Extension, composition, and modifying functionality	478

11.4	<b>Extension and state</b> . . . . .	480
11.4.1	State as an object . . . . .	481
11.5	<b>Summary</b> . . . . .	482
<b>chapter 12</b>	<b>Lists</b>	<b>491</b>
12.1	<b>Containers</b> . . . . .	492
12.2	<b>Lists</b> . . . . .	493
12.2.1	Structuring lists . . . . .	494
12.2.2	Using generics to capture “list-ness” . . . . .	499
12.3	<b>List specification</b> . . . . .	501
12.4	<b>Iteration</b> . . . . .	509
12.4.1	<i>while</i> loops and lists . . . . .	510
12.5	<b>Examples</b> . . . . .	511
12.5.1	Summing items on a list . . . . .	511
12.5.2	Summing selected elements of a list . . . . .	514
12.5.3	Finding the minimum . . . . .	515
12.5.4	Determining if an object is on a <i>List</i> : searching the <i>List</i> . . . . .	517
12.5.5	Removing duplicates . . . . .	520
12.6	<b>Loop structure: a summary</b> . . . . .	523
12.7	<b>Java in detail: the <i>for</i> statement</b> . . . . .	524
12.8	<b>What does “equal” mean?</b> . . . . .	526
12.8.1	Overriding equals . . . . .	526
12.8.2	Problems when overriding equals . . . . .	527
12.9	<b>Implementing the interface <i>List</i></b> . . . . .	534
12.9.1	The class <i>AbstractList</i> . . . . .	534
12.9.2	The class <i>DefaultList</i> . . . . .	536
12.10	<b>Generic type checking is done at compile time</b> . . . . .	538
12.11	<b>Summary</b> . . . . .	539

<b>chapter 13</b>	<b>Implementing lists: array implementations</b>	<b>549</b>
13.1	<b>Arrays</b> . . . . .	550
13.2	<b>Java in detail: arrays with array components and initializing arrays</b> . . . . .	553
13.3	<b>An array-based list implementation</b> . . . . .	557
13.3.1	Copies and clones . . . . .	561
13.3.2	Advantages and limitations of an array implementation . . . . .	567
13.4	<b>Dynamic arrays</b> . . . . .	568
13.5	<b>Summary</b> . . . . .	571

<b>chapter 14</b>	<b>Sorting and searching</b>	<b>577</b>
	14.1 Ordering lists	578
	14.2 Two simple sorts	579
	14.2.1 Selection sort	580
	14.2.2 Analysis of selection sort	585
	14.2.3 Bubble sort	586
	14.2.4 Generalizing the sort methods	591
	14.3 Ordered lists	598
	14.4 Binary search	600
	14.4.1 Completing the search	605
	14.4.2 Sequential search and binary search	607
	14.5 Verifying correctness: using a loop invariant	608
	14.6 Summary	612
<b>chapter 15</b>	<b>Failures and exceptions</b>	<b>619</b>
	15.1 Failures	620
	15.2 The Java exception mechanism	620
	15.2.1 Exceptions as objects	621
	15.2.2 Catching exceptions	622
	15.2.3 Propagated exceptions	625
	15.2.4 Checked and unchecked exceptions	625
	15.3 Dealing with failure: using exceptions	627
	15.3.1 Dealing with exceptions	629
	15.3.2 Application defined exceptions	632
	15.3.3 Dealing with logical errors	633
	15.4 Summary	635
<b>chapter 16</b>	<b>Stream i/o</b>	<b>641</b>
	16.1 Data Streams	642
	16.2 The java.io library	642
	16.3 Input streams	643
	16.3.1 Input byte streams	643
	16.3.2 Input character streams	649
	16.3.3 Input examples	651
	16.4 Output streams	660
	16.4.1 Output byte streams	660
	16.4.2 Output character streams	663
	16.5 The class <i>File</i>	666

16.6	Summary	668
------	---------	-----

chapter 17	<b>Building a graphical user interface</b>	<b>673</b>
------------	--	------------

17.1	Event-driven interfaces	674
17.2	An introduction to Swing	675
17.2.1	Components	675
17.2.2	Containers	680
17.3	Creating a display	683
17.3.1	The top-level frame	683
17.3.2	The event dispatching thread	685
17.3.3	Adding components: layout	686
17.4	Events: programming the user interface	690
17.4.1	An example	693
17.5	A more complex example	704
17.6	Menus, dialogs, fonts, and graphics	713
17.6.1	Menus and menu bars	713
17.6.2	Basic dialogs	714
17.6.3	Fonts	720
17.6.4	Graphics	721
17.7	Some class features	723
17.7.1	<i>Component</i>	723
17.7.2	<i>Container</i>	724
17.7.3	<i>Window</i>	724
17.7.4	<i>Frame</i>	725
17.7.5	<i>JComponent</i>	725
17.7.6	<i>JFrame</i>	726
17.8	Summary	726

chapter 18	<b>Integrating user interface and model: the Model-View-Controller pattern</b>	<b>735</b>
------------	--	------------

18.1	Model-View-Controller	736
18.2	Implementing MVC in Java	737
18.2.1	The model	737
18.2.2	The class <i>Observable</i> and interface <i>Observer</i>	737
18.2.3	An <i>Observer</i>	741
18.2.4	Observer, Observable, and subtyping	743
18.2.5	A simple view and controller	743
18.2.6	A graphic view	752
18.2.7	A logger as a view	757
18.3	Adding a graphical interface to the nim game	758
18.4	The MVC pattern and Swing components	764

	<b>18.5 Summary</b> .....	<b>767</b>
<b>chapter 19</b>	<b>Recursion</b> .....	<b>771</b>
	<b>19.1 Recursion and iteration</b> .....	<b>772</b>
	19.1.1 Iteration .....	772
	19.1.2 Recursion .....	772
	19.1.3 Some simple examples .....	773
	<b>19.2 Example: the tower puzzle</b> .....	<b>790</b>
	<b>19.3 Quick sort</b> .....	<b>794</b>
	<b>19.4 An inefficient algorithm</b> .....	<b>801</b>
	<b>19.5 Indirect recursion</b> .....	<b>802</b>
	<b>19.6 Backtracking</b> .....	<b>804</b>
	<b>19.7 Object recursion</b> .....	<b>809</b>
	<b>19.8 Summary</b> .....	<b>819</b>
<b>chapter 20</b>	<b>Implementing lists: linked implementations</b> .....	<b>827</b>
	<b>20.1 A linked <i>List</i> implementation</b> .....	<b>827</b>
	20.1.1 Implementing <i>LinkedList</i> methods .....	830
	<b>20.2 Linked list variations</b> .....	<b>835</b>
	<b>20.3 Doubly-linked lists</b> .....	<b>838</b>
	<b>20.4 Limitations of linked structures</b> .....	<b>840</b>
	<b>20.5 Dynamic storage allocation</b> .....	<b>842</b>
	<b>20.6 Summary</b> .....	<b>843</b>
<b>chapter 21</b>	<b>Iterators</b> .....	<b>851</b>
	<b>21.1 Iterators</b> .....	<b>851</b>
	21.1.1 Iterator classes .....	854
	21.1.2 Creating an iterator .....	859
	<b>21.2 <i>List&lt;Element&gt;</i> methods with iterators as arguments</b> .....	<b>862</b>
	21.2.1 Improving <i>LinkedListIterator&lt;Element&gt;</i> .....	864
	21.2.2 <i>Iterator</i> extensions .....	864
	21.2.3 Iterators and list modification .....	865
	21.2.4 Internal iterators: <i>forEachDo</i> .....	867
	<b>21.3 Comparing implementations</b> .....	<b>869</b>
	<b>21.4 The <i>java.util Collection&lt;Element&gt;</i> hierarchy</b> .....	<b>870</b>
	<b>21.5 Summary</b> .....	<b>872</b>

<b>supplement a</b>	<b>Systems and software</b>	<b>877</b>
	<b>a.1 A model of a computer system</b>	<b>877</b>
	<b>a.2 Software tools</b>	<b>880</b>
<b>supplement b</b>	<b>Programming errors</b>	<b>885</b>
	<b>b.1 Errors in the programming process</b>	<b>885</b>
<b>supplement c</b>	<b>Applets</b>	<b>889</b>
	<b>c.1 Applets</b>	<b>889</b>
	c.1.1 The class <i>JApplet</i>	890
	c.1.2 Applets as applications	893
	c.1.3 An example: a simple clock	894
	c.1.4 An example: an animated box	896
<b>supplement d</b>	<b>Additional Java 1.5 features</b>	<b>901</b>
	<b>d.1 Generics</b>	<b>901</b>
	d.1.1 Wildcard types	904
	<b>d.2 Autoboxing and unboxing</b>	<b>911</b>
	<b>d.3 Enumeration types</b>	<b>912</b>
	<b>d.4 Enhanced <i>for</i> statement</b>	<b>919</b>
	<b>d.5 Importing static methods and named constants</b>	<b>920</b>
	<b>d.6 The class <i>java.util.Scanner</i></b>	<b>921</b>
<b>appendix i</b>	<b>Compiling, executing, and documenting</b>	<b>925</b>
	<b>i.1 Compiling and running an application</b>	<b>925</b>
	i.1.1 Compilation units	925
	i.1.2 Package name and directory hierarchy	926
	i.1.3 The environment variable <i>CLASSPATH</i>	926
	i.1.4 Compiling and running	927
	<b>i.2 Generating documentation</b>	<b>928</b>

<b>appendix ii</b>	<b>DrJava</b>	<b>931</b>
<b>appendix iii</b>	<b>Controls and basic Latin: the first 128 Unicode characters</b>	<b>933</b>
<b>references</b>		<b>935</b>
<b>index</b>		<b>937</b>