

Java Au Naturel

Guide to Object Oriented Design

Java 2 with Swing

**Dr. William C. Jones, Jr.
email jonesw@ccsu.edu**

**Department of Computer Science
Central Connecticut State University
web page: www.javabook.org**

Fourth Edition published 5/26/04 copyright 2004

Java Au Naturel

Table Of Contents

Chapter 1 Objects 34 pages

1.1	Using Turtle objects to draw pictures.	1-1
1.2	A complete Java application program using Turtle methods.	1-3
1.3	A first look at inheritance: defining instance methods in Turtle subclasses.	1-7
1.4	Additional Turtle methods; identifiers versus keywords.	1-11
1.5	Compiling and running an application program.	1-17
1.6	Sending messages to objects.	1-19
Part B Enrichment And Reinforcement		
1.7	Three application programs using other kinds of objects.	1-21
1.8	Program development: Analysis, Logic design, Object design, Coding.	1-26
1.9	Placing Java in history.	1-28
1.10	Fractal Turtles.	1-29
-----	Chapter review and answers to exercises.	1-31

Chapter 2 Conditionals and Boolean Methods 36 pages

2.1	Using Vic objects to control appliances.	2-1
2.2	Defining a subclass containing only instance methods.	2-6
2.3	The if statement.	2-9
2.4	Using class methods and javadoc comments in a program.	2-13
2.5	The if-else statement and the block statement.	2-15
2.6	Boolean methods and the not-operator.	2-17
2.7	Boolean variables and the assignment operator.	2-21
2.8	Boolean operators and expressions; crash-guards.	2-24
2.9	Getting started with UML class diagrams and object diagrams.	2-26
Part B Enrichment And Reinforcement		
2.10	Analysis and Design example: complex conditionals.	2-28
-----	Chapter review and answers to exercises.	2-32

Chapter 3 Loops and Parameters 32 pages

3.1	The while statement.	3-1
3.2	Using the equals method with String variables.	3-4
3.3	More on UML diagrams.	3-7
3.4	Using private methods and the default executor.	3-9
3.5	A first look at declaring method parameters.	3-14
3.6	Returning object values.	3-18
Part B Enrichment And Reinforcement		
3.7	More on the Analysis and Design paradigm.	3-19
3.8	Analysis and Design example: finding adjacent values.	3-24
3.9	Turing machines.	3-27
3.10	Javadoc tags.	3-28
-----	Chapter review and answers to exercises.	3-29

Interlude: Integers and For-loops **6 pages****Chapter 4 Instance Variables** **42 pages**

4.1	Analysis and Design of basic games.	4-1
4.2	Input and output with JOptionPane dialog boxes.	4-3
4.3	Declaring instance variables: a first look at encapsulation.	4-6
4.4	Defining constructors; inheritance.	4-8
4.5	Integer instance variables.	4-13
4.6	Making Random choices.	4-16
4.7	Overloading, overriding, and polymorphism.	4-19
4.8	The rules of precedence for operators.	4-23
Part B Enrichment And Reinforcement		
4.9	Analysis and Design example: the game of Nim.	4-25
4.10	Analysis and Design example: the game of Mastermind.	4-28
4.11	Using BlueJ with its debugger.	4-31
4.12	A first look at event-handling: buttons and textfields.	4-35
----	Chapter review and answers to exercises.	4-39

Chapter 5 Class Methods and Class Variables **34 pages**

5.1	Defining class methods.	5-1
5.2	Declaring class variables; encapsulation and scope.	5-4
5.3	Final local, instance, and class variables.	5-8
5.4	Two new String methods.	5-9
Part B Enrichment And Reinforcement		
5.5	Complete String implementation of a Vic simulator.	5-12
5.6	Case Study: introduction to networks.	5-15
5.7	Extending the Network class.	5-20
5.8	Analysis and Design example: the Reachability Problem.	5-22
5.9	Recursion.	5-27
5.10	More on JOptionPane (*Sun library).	5-31
----	Chapter review and answers to exercises.	5-32

Review: Overall Java Language So Far **6 pages****Chapter 6 Basic Data Types and Expressions** **43 pages**

6.1	Double values, variables, and expressions.	6-1
6.2	Creating your own library classes.	6-5
6.3	Basic String methods; the Comparable interface.	6-9
6.4	Character values and String's charAt method.	6-14
6.5	Long integers; casts and conversions; the Math class.	6-20
Part B Enrichment And Reinforcement		
6.6	Formatted output to a JTextArea in a JScrollPane.	6-23
6.7	Analysis and Design example: the RepairOrder class.	6-26
6.8	Analysis and Design example: Model/View/Controller.	6-29
6.9	More on debugging your program: tracing and type-checking.	6-34
6.10	More on Random, NumberFormat, and DecimalFormat (*Sun library).	6-37
----	Chapter review and answers to exercises.	6-39

Chapter 7 Arrays 41 pages

7.1	Analysis and Design of the Worker class.	7-1
7.2	Analysis and Design example: finding the alphabetically first.	7-6
7.3	An array of counters, an array of Strings.	7-9
7.4	Implementing the Worker class with arrays.	7-13
7.5	Analysis and Design example: finding the average in an array.	7-16
7.6	Implementing the WorkerList class with arrays.	7-20
Part B Enrichment And Reinforcement		
7.7	A first look at sorting: the insertion sort.	7-24
7.8	A first look at two-dimensional arrays: implementing the Network classes.	7-27
7.9	Implementing a Vic simulator with arrays.	7-30
7.10	Command-line arguments.	7-32
7.11	Implementing Queue as a subclass of ArrayList.	7-33
7.12	More on System, String, and StringBuffer (*Sun library).	7-35
----	Chapter review and answers to exercises.	7-38

Chapter 8 Elementary Graphics 40 pages

8.1	The JApplet, Color, and Graphics2D classes.	8-1
8.2	Five Shape classes: rectangles, lines, and ellipses.	8-5
Part B Enrichment And Reinforcement		
8.3	Analysis and Design example: the Flag software.	8-8
8.4	Iterative Development of the Flag software.	8-10
8.5	Animation in a JApplet.	8-15
8.6	Review of the software development paradigm.	8-18
8.7	Common looping patterns.	8-19
8.8	Case Study in animation: Colliding Balls.	8-22
8.9	Implementing the Turtle and Turtlet classes.	8-28
8.10	Elements of JavaScript.	8-32
8.11	About Font, Polygon, and Point2D.Double (*Sun library).	8-36
----	Chapter review and answers to exercises.	8-37

Chapter 9 Exception-Handling 38 pages

9.1	Problem description for the Investor software.	9-1
9.2	Handling RuntimeExceptions; the try/catch statement.	9-2
9.3	Basic text file input; handling checked Exceptions.	9-7
Part B Enrichment And Reinforcement		
9.4	Throwing your own Exceptions.	9-11
9.5	Analysis, Test Plan, and Design for the Investor software.	9-14
9.6	Version 1 of Iterative Development.	9-19
9.7	Version 2 of Iterative Development.	9-23
9.8	Version 3 of Iterative Development.	9-25
9.9	Additional Java statements: switch and break.	9-30
9.10	About Throwable and Error (*Sun library).	9-34
----	Chapter review and answers to exercises.	9-36

Chapter 10 Event-Driven Programming 44 pages

10.1	JFrames, Components, and WindowListeners.	10-1
10.2	JPanels, Containers, and LayoutManagers.	10-4
10.3	JLabels, JTextFields, and ActionListeners.	10-8
10.4	Inner classes.	10-11
10.5	JButtons and EventObjects.	10-12

Part B Enrichment And Reinforcement

10.6	Model/View/Controller pattern applied to Car Rental software.	10-16
10.7	JSliders, JTextAreas, and ChangeListeners.	10-19
10.8	Swing Timers.	10-23
10.9	JComboBoxes using arrays of Objects.	10-26
10.10	JCheckBoxes, JRadioButtons, and Keystrokes.	10-29
10.11	Of Mice and Menus.	10-33
10.12	More on LayoutManager and JList (*Sun library).	10-36
10.13	More on JComponent, ImageIcon, and AudioClip (*Sun library).	10-38
----	Chapter review and answers to exercises.	10-39

Chapter 11 Abstract Classes and Interfaces**40 pages**

11.1	Analysis and Design of the Mathematicians software.	11-1
11.2	Abstract classes and interfaces.	11-4
11.3	More examples of abstract classes and polymorphism.	11-7
11.4	Double, Integer, and other wrapper classes.	11-11
Part B Enrichment And Reinforcement		
11.5	Implementing the Fraction class.	11-14
11.6	Implementing the Complex class.	11-18
11.7	Implementing the VeryLong class using arrays.	11-20
11.8	Implementing the NumericArray class with null-terminated arrays.	11-23
11.9	Too many problems, not enough solutions.	11-28
11.10	Threads: producers and consumers.	11-30
11.11	More on Math and Character (*Sun library).	11-35
----	Chapter review and answers to exercises.	11-37

Chapter 12 Files and Multidimensional Arrays**36 pages**

12.1	Analysis and Design of the Email software.	12-1
12.2	FileReader and BufferedReader for input.	12-2
12.3	FileWriter and PrintWriter for output.	12-6
12.4	The StringTokenizer and StreamTokenizer classes.	12-9
12.5	Defining and using multidimensional arrays.	12-12
Part B Enrichment And Reinforcement		
12.6	Implementing the Email software with a two-dimensional array.	12-16
12.7	Using a two-dimensional array of airline data.	12-20
12.8	The RandomAccessFile class.	12-22
12.9	How buffering is done.	12-24
12.10	Additional Java language features.	12-26
12.11	Java bytecode commands.	12-27
12.12	About networking using Sockets (*Sun library).	12-31
12.13	About File and JFileChooser (*Sun library).	12-33
----	Chapter review and answers to exercises.	12-33

Chapter 13 Sorting and Searching**32 pages**

13.1	The SelectionSort Algorithm for Comparable objects.	13-1
13.2	The InsertionSort Algorithm for Comparable objects.	13-4
13.3	Big-oh and Binary Search.	13-8
Part B Enrichment And Reinforcement		
13.4	The recursive QuickSort Algorithm for Comparable objects.	13-11
13.5	The recursive MergeSort Algorithm for Comparable objects.	13-17
13.6	More on big-oh.	13-22
13.7	Additional sorting algorithms: bucket, radix, and shell.	13-26
13.8	About the Arrays utilities class (*Sun library).	13-29
----	Chapter review and answers to exercises.	13-30

Chapter 14 Stacks, Queues, and Linked Lists 42 pages

14.1	Applications of stacks and queues.	14-1
14.2	Implementing stacks with arrays.	14-8
14.3	Implementing queues with arrays.	14-11
14.4	Implementing stacks and queues with linked lists.	14-14
14.5	The ListADT abstract class.	14-19
Part B Enrichment And Reinforcement		
14.6	Additional linked list operations using recursion.	14-24
14.7	Sorting and shuffling with linked lists.	14-28
14.8	Doubly-linked lists and lists with header nodes.	14-32
14.9	Applications: Josephus and the many-colored WebLinks.	14-36
14.10	About Stack and Vector (*Sun library).	14-39
----	Chapter review and answers to exercises.	14-40

Chapter 15 Collections and Linked Lists 40 pages

15.1	Analysis and Design of the Inventory software.	15-1
15.2	Implementing the Collection interface with arrays.	15-3
15.3	Linked lists with a nested private Node class.	15-6
15.4	Implementing the Collection interface with linked lists.	15-10
15.5	Recursion with linked lists.	15-13
Part B Enrichment And Reinforcement		
15.6	Implementing the Iterator interface for an array-based Collection.	15-16
15.7	Implementing the Iterator interface for a linked-list-based Collection.	15-20
15.8	Implementing a modifiable Collection with linked lists.	15-22
15.9	Implementing the ListIterator interface for a linked list.	15-26
15.10	Implementing the ListIterator interface for a doubly-linked list.	15-30
15.11	About AbstractList and AbstractCollection (*Sun library).	15-35
----	Chapter review and answers to exercises.	15-36

Chapter 16 Maps and Linked Lists 42 pages

16.1	Basic Scheme language elements.	16-1
16.2	Design of the Scheme interpreter.	16-4
16.3	The Map interface and the Mapping interface.	16-8
16.4	Implementing the Mapping interface with arrays.	16-13
16.5	Implementing the Mapping interface with linked lists.	16-17
Part B Enrichment And Reinforcement		
16.6	Implementing Mapping Iterators; ordered tables.	16-23
16.7	Hash tables.	16-28
16.8	Further development of Scheme's List class.	16-34
16.9	About Map, AbstractMap, HashMap, and TreeMap (*Sun library).	16-38
----	Chapter review and answers to exercises.	16-39

Chapter 17 Binary Trees 42 pages

17.1	Analysis and Design of the Genealogy software.	17-1
17.2	Implementing the Genealogy software with binary trees.	17-3
17.3	Searching through a binary tree.	17-7
17.4	Implementing the Mapping interface with binary search trees.	17-11
17.5	Implementing the Iterator interface for a binary search tree.	17-17
Part B Enrichment And Reinforcement		
17.6	More tree traversals.	17-21

17.7	Red-black and AVL binary search trees.	17-24
17.8	Inductive reasoning about binary trees.	17-29
17.9	2-3-4 trees and B-trees.	17-32
17.10	Data Flow Diagrams.	17-35
-----	Chapter review and answers to exercises.	17-40

Chapter 18 Priority Queues, Heaps, and More Sorting **38 pages**

18.1	File compression using Huffman codes.	18-1
18.2	Comparator objects.	18-5
18.3	Implementing priority queues with arrays.	18-8
18.4	Implementing priority queues with linked lists.	18-11
Part B Enrichment And Reinforcement		
18.5	Implementing priority queues with linked lists of queues.	18-15
18.6	Sorting using priority queues; the TreeSort Algorithm.	18-18
18.7	Implementing priority queues using heaps; the HeapSort Algorithm.	18-23
18.8	The MergeSort Algorithm for a linked list; recurrence relations.	18-28
18.9	External sorting: file merge using a priority queue.	18-32
-----	Chapter review and answers to exercises.	18-37

Chapter 19 Graphs **38 pages**

19.1	The Hamiltonian Software.	19-1
19.2	The adjacency matrix implementation of Graphs.	19-6
19.3	The adjacency list implementation of Graphs.	19-9
19.4	Topological sorting; big-oh for Graph algorithms.	19-12
19.5	Graph traversals.	19-15
Part B Enrichment And Reinforcement		
19.6	Union-Find structures.	19-20
19.7	Weighted graphs; algorithms for Minimum Spanning Trees.	19-23
19.8	The shortest path problem: Dijkstra's algorithm.	19-28
19.9	Dynamic programming.	19-31
-----	Chapter review and answers to exercises.	19-37

Chapter 20 Models of Computation **36 pages**

20.1	Decision problems and finite automata.	20-1
20.2	Grammars and pushdown automata.	20-7
20.3	Turing machines.	20-12
20.4	Unsolvable problems and uncomputable functions.	20-16
Part B Enrichment And Reinforcement		
20.5	Extended example of a context-free LL(1) grammar.	20-21
20.6	Comparing programming languages -- Smalltalk	20-30
-----	Chapter review and answers to exercises.	20-36

Appendix A Guidelines for Program Style	APP-1
Appendix B Glossary of Terms	APP-5
Appendix C Common Compile-Time Errors	APP-17
Appendix D Java Reserved Words	APP-22
Appendix E Sun Library Classes	APP-23
Appendix F Major Programming Projects	APP-29

total 848 pages = 10 + 352 (ch1..9) + 234 (ch10..15) + 196 (ch16..20) + 56 (appendix)

Preface for Students

Software development with an object-oriented approach is the fundamental subject of this book. Java is a programming language used to create the animations you see when you browse the web. Of all the programming languages whose use is wide-spread, Java is the best for learning and doing object-oriented software development. That is why Java is used in this computer science book.

You only learn by doing. So in this book, you will work with many different situations where software development is necessary. That way, you can see how to apply newly-learned techniques in a variety of contexts. When you need a language feature to accomplish a certain purpose, that is when you learn it. If it has several alternatives and you do not need them at that point or in the near future, you do not learn them then; you wait until you need them.

When you learn to speak a natural language such as German or Spanish, you start with a few sentences that are useful in key situations. You gradually expand your repertoire of sentences. You learn grammatical principles that apply to several sentences you already know how to use.

You do not start by learning all the conjugations of verbs and all the declensions of nouns. Incremental development is far more effective: Add items a few at a time, mastering those before you add more. And, most important, learn what you can immediately use in realistic conversations.

You should learn a programming language the same way. You should not start by learning Java's eight different primitive types or its five different control statements. You should start by seeing how to develop software for a particular realistic situation. And you should learn the language features you need for the situation as the need arises.

This book starts by introducing you to objects that make drawings. In this context you learn to send messages to objects to carry out simple tasks. More important, you learn how to "teach" those objects to put together a sequence of simple actions to perform a complex task. These "better-trained" objects have new capabilities in addition to the ones they inherit from the original specifications. This **inheritance** technique of course makes your job much easier. You can do bigger jobs with less work and less chance of getting it wrong if you have objects take over most of the work. When objects are used this way, they should be thought of as your agents or executive assistants.

Chapter One explains the details of compiling and executing a program. It introduces much of the vocabulary you need. It establishes the framework for Java programs. And it gives an overview of the book with a look at other contexts where objects are used to perform useful tasks. This is a foretaste of what is to come, so you are not expected to fully understand at this point everything it mentions.

Chapter Two introduces a quite different software context. The software provided to you defines objects that control the basic physical actions of electronic equipment. Your job is to develop additional software that puts these basic actions together in combinations that perform tasks that the purchaser of the electronic equipment finds useful. You learn to use inheritance in this new context. Then you see how to have objects select between two courses of action depending on the circumstances. You also learn a simple method of diagramming the relationships among classes of objects. It is a widely used technique that is part of the Unified Modeling Language (UML). The UML is the industry standard for modeling software.

Chapter Three shows you how to teach objects to repeat a sequence of actions many times until a task is accomplished. This ability leads to more complex programs, so we discuss a reliable process for developing the logic to solve a problem and translate it into a Java program. This is the one chapter that does not introduce a new major context for developing software.

Chapter Four switches to the context of game-playing programs. In this context, you learn how to build objects "from scratch", specifying what they know as well as what they can do. By this time you can write interesting and useful programs using only the standard library of objects that comes with every Java installation. The game-playing software interacts with the outside world through the keyboard and screen rather than via signals to and from electronic equipment or to a drawing surface.

Chapter Five completes the presentation of basic language features you need for working with objects in your programs. At this point you can construct a complete string-based simulation of the electronic equipment you worked with in earlier chapters. A case study on networks lets you see the interaction of all of the object-oriented concepts and most of the language features applied in another context. The situation that the networking material describes is key to many important real-life problems. The analysis and solution of some of these problems can be quite complex; it is the subject of more advanced courses in computer science. But the discussion here is quite elementary.

Chapter Six expands your arsenal of basic types of values to include characters and numbers with decimal points (heretofore you only had the whole-number and true-false kinds of values). It also gives you a full set of methods for dealing with strings of characters. These language features are introduced in the context of software to schedule work orders at a car repair shop using the Model/View/Controller approach and several kinds of objects at once.

Chapter Seven gives you the tools you need to work with large masses of data. This is in the context of software to handle a database of people working for a particular company. A re-implementation of a simulation of the electronic-equipment software from Chapters Two and Three helps you solidify your understanding of the key concept of arrays.

The first seven chapters present the features of Java most frequently used in this book, together with a moderate number of examples. Many of the concepts, especially the various uses of arrays, cannot be learned well enough without a great deal of practice. The remaining eleven chapters give you that practice. Their primary purpose is to (a) improve your understanding of principles and techniques of software design and development, and (b) reinforce the concepts in the basic first seven chapters.

Each of the next twelve chapters presents a different software design and development situation. The emphasis is on techniques for creating quality software.

In summary

- Chapter One: Turtle objects can do simple tasks such as draw a line or move to another position. You use Java to combine these simple actions into complex tasks such as having a Turtle draw a flower garden.
- Chapters Two and Three: Stick figures called Vic objects can do simple tasks such as put a compact disk in or out of the current slot and move to the slot before or after the current slot. You use Java to combine these simple actions into complex tasks that move several CDs where you want them.
- Chapters Four and Five: You use Java to define the simple tasks that objects can do. You apply these language skills to define how game-playing objects interact with a human player, and to define how the stick figure objects perform their simple tasks. You may also, if you wish, skip directly to Chapter Eight on graphics to learn to define how Turtle objects perform their simple tasks.
- Chapter Six: You use Java to develop commercial software that defines how several classes of objects perform simple tasks and then has these several classes of objects cooperate to accomplish a complex task.
- Chapter Seven: You use Java for software handling many objects at once.

Re-read this summary once or twice as you progress through the textbook, whenever you get the feeling that you cannot see the forest for the trees.

You cannot avoid heavy technical vocabulary in a first course in computer science. But as a student you have a right to short paragraphs and simple sentences to explain the vocabulary and the difficult concepts accurately. This book makes a special effort to give you just that.

You should work out most of the unstarred exercises in the book. Many of the ones marked "harder" might take you 10 or 15 minutes, so you could leave out some of those. But the others should only take two minutes. The answers to all unstarred exercises are at the end of the corresponding chapter. If you do not have a good answer after five minutes, either continue working until you do or at least study the given answer until you see why it is a good answer and where you went wrong. This will help you gain a thorough understanding of the concepts rather than a superficial one.

This book does not require any knowledge of programming or any mathematics beyond elementary algebra and (in a few places) a bit of trig. But it does require that you practice what you read about. Reading alone will not suffice -- you only learn by doing!