# The Definitive Guide to Java Swing

Third Edition

JOHN ZUKOWSKI

Apress®

**The Definitive Guide to Java Swing, Third Edition**



ISBN (pbk): 1-59059-447-9

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1



Lead Editor: Steve Anglin
Technical Reviewer: Robert Castaneda
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Assistant Publisher: Grace Wong
Project Manager: Beth Christmas
Copy Edit Manager: Nicole LeClerc
Copy Editor: Marilyn Smith
Production Manager: Kari Brooks-Copony
Production Editor: Ellie Fountain
Compositor: Susan Glinert
Proofreaders: Linda Seifert, Liz Welch
Indexer: Michael Brinkman
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders@springer-ny.com`, or visit `http://www.springer-ny.com`. Outside the United States: fax +49 6221 345229, e-mail `orders@springer.de`, or visit `http://www.springer.de`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.



The source code for this book is available to readers at `http://www.apress.com` in the Downloads section.

# Contents at a Glance

# Contents

# About the Author

**JOHN ZUKOWSKI** has been involved with the Java platform since it was just called Java, pushing ten years now. He currently writes a monthly column for Sun's Core Java Technologies Tech Tips (`http://java.sun.com/developer/JDCTechTips/`) and IBM's developerWorks (`http://www-136.ibm.com/developerworks/java/`). He has contributed content to numerous other sites, including jGuru (`http://www.jguru.com`), DevX (`http://www.devx.com/`, Intel (`http://www.intel.com/`), and JavaWorld (`http://www.javaworld.com/`). He is the author of many other popular titles on Java, including *Java AWT Reference* (O'Reilly and Associates), *Mastering Java 2* (Sybex), *Borland's JBuilder: No Experience Required* (Sybex), *Learn Java with JBuilder 6* (Apress), *Java Collections* (Apress), and *Definitive Guide to Swing for Java 2* (Apress).

# About the Technical Reviewers

This book was technically reviewed by Daren Klamer, David Vittor, Hido Hasimbegovic, Charlie Castaneda, and Robert Castaneda, who are all part of the CustomWare Asia Pacific team working on numerous Java and integration-based projects in Australia and the Asia Pacific region. Their web site is `http://www.customware.net`.

# Acknowledgments

This book has been a long time coming, with various starts and stops, and getting sidetracked a few times along the way. Now that it is all done, I need to thank those who helped.

For starters, I want to thank everyone at Apress who hung in there and had patience when dealing with me throughout the project, especially project manager Beth Christmas, who I'm sure I drove nuts at times, and editor Steve Anglin, who kept nudging me along. On the production side, I'd like to thank Marilyn Smith for all the input and updates, Ellie Fountain for her hard work at ensuring little changes got done right, and, of course, my technical reviewer Rob Castaneda and the team at CustomWare for all the input on my rough work. Congrats on that marriage thing.

Some of the images used in the sample programs were made by Deb Felts, who ran a web site called the Image Addict's Attic. The site doesn't seem to be online any more, but the images are used with permission and she does retain copyright on them. Sun also maintains the Java Look and Feel Graphics Repository at `http://java.sun.com/developer/techDocs/hi/repository/`, with its own set of images to be used for Java applications.

For all the readers out there, thanks for asking me to do the update. Without your continued support, you wouldn't be holding this book in your hands.

For their continued encouragement along the way, I'd like to personally thank the following: Joe Sam Shirah, thanks for doing that long drive to visit while I was in Florida for the conference; my Aunt Mary Hamfeldt, congrats on your first grandchild; our Realtor Nancy Moore, thanks for putting up with us for so long; Miguel Muniz, thanks for all the bug reports at SavaJe; Matthew B. Doar, thanks for JDiff (`http://www.jdiff.org/`), a great little Java doclet for reporting API differences. Happy tenth birthday, Duke and Java.

I am forever grateful to my wife, Lisa, for her support, and our dog , Jaeger, for his playfulness. Thanks to Dad, too. Good luck at the casinos.

# Introduction

Welcome to Learn Java 5.0 Swing in a Nutshell for Dummies in 21 Days. Since the beginning of Java time (1995), the component libraries have been actively evolving. What began as a small set of nine AWT components, plus menus and containers, has grown to a more complete and complex set of around 50 Swing components—all just to create graphical user interfaces (GUIs) for your Java client-side programs. That's where this book comes in. Its purpose is to make your life easier in creating those GUIs.

Earlier editions of this book took the approach that if the class wasn't found in the `javax.swing` package, it wasn't covered in the book. This third edition takes a more complete view of creating GUIs. For instance, instead of just describing the Swing layout managers, there is also material on the AWT layout managers, since you're likely to be using them.

The first edition of this book was written for a mix of the Java 1.1 and 1.2 developer. The second edition hit the 1.3 platform. This edition is wholly for the 5.0 developer. Almost all the programs will not work on a 1.4 platform, though with a little tweaking, they can be made to do so.

In this book, you'll find a tutorial-like approach to learning about the Swing libraries and related capabilities. It is not an API reference book, nor is it a primer that describes how to install the Java Development Kit (JDK), compile your programs, or run them. If you need help in those areas, consider using an integrated development environment (IDE)—such as IntelliJ IDEA, Eclipse, or Borland's JBuilder—or get one of Apress's other books, such as *Beginning Java Objects*, by Jacquie Barker.

Is this book for you? If you are new to the Java platform, you might want to start with a more introductory text first, before jumping on the Swing bandwagon. On the other hand, if you've been working with Java for a while and have decided it's time to start using the Swing component set, you'll find this book extremely useful. With this book, you won't have to drudge through the countless Swing classes for a way to accomplish that impossible task. You'll become much more productive more quickly, and you'll be able to make the most of the many reusable components and techniques available with Swing.

## Book Structure

This book can be read from cover to cover, but it doesn't have to be done that way. It's true that later sections of the book assume you've absorbed knowledge from the earlier sections. However, if you want to find something on a topic covered in a later chapter, you don't need to read all the chapters that precede it first. If you come across something that's unfamiliar to you, you can always go back to the earlier chapter or search the index to locate the information you need.

The contents of this book are grouped into three logical sections:

Chapters 1 through 4 provide general knowledge that will prove to be useful as you read through the remainder of the book. In Chapter 1, you'll find an overview of the Swing component set. Chapter 2 details event handling with the Swing component set. It describes the delegation-based event model and focus management policies used by Swing. In Chapter 3, you'll learn about the Model-View-Controller (MVC) architecture. You can avoid using MVC if you wish, but to take full advantage of everything that Swing has to offer, it helps to have a good grasp of MVC concepts. In Chapter 4, you'll find the beginning coverage of the specific Swing components. All Swing components share many of the same attributes, and in Chapter 4, you'll learn the foundation for those common behaviors.

In Chapters 5 through 15, you'll discover the many aspects of the reusable Swing components. You'll find out about menus, toolbars, borders, high-level containers, pop-up dialogs, layout managers, advanced Swing containers, bounded range components, toggle components, list model components, spinners, and text components. Most of what you'll want to accomplish with the Swing libraries is discussed in these chapters.

In Chapters 16 through 22, some of the more advanced Swing topics are covered. These tend to be the areas that even the experienced developers find the most confusing. Chapter 16 goes beyond the basics of text component handling found in Chapter 15. Chapters 17 and 18 deal with the Swing tree and table components. These components allow you to display hierarchical or tabular data. In Chapter 19, you'll learn about drag-and-drop support in Swing. Chapter 20 explores how to customize the appearance of your application. Because the Swing libraries are completely Java-based, if you don't like the way something is done or how it appears, you can change it. In Chapter 21, you'll learn about the undo framework, which offers undo and redo support for your applications. Finally, in Chapter 22, you finish off with a look into the accessibility framework offered by Swing, such as support for screen readers and magnifying glasses to help those needing assistive technologies.

The Appendix contains a list of about 1,000 settable properties the user interface manager employs to configure the appearance of the Swing components for the current look and feel. The Swing components manage various defaults, such as colors and fonts applied to components, so you don't need to subclass a component in order to customize its appearance. Appendix A gathers all of the property settings listed throughout the chapters into one comprehensive list for easy reference.

# Support

You can head to many places online to get technical support for Swing and answers to general Java questions. Here's a list of some of the more useful places around:

- The Java Ranch at `http://www.javaranch.com/` offers forums for just about everything in the Big Moose Saloon.
- Java Forums at `http://forums.java.sun.com/` are Sun's online forums for Java development issues.

- developerWorks at `http://www.ibm.com/developerworks/java/` is the IBM's developer community for Java with forums and tutorials.
- jGuru at `http://www.jguru.com` offers a series of FAQs and forums for finding answers.
- Marcus Green's Java Certification Exam Discussion Forum at `http://www.jchq.net/discus/` provides support for those going the certification route.

While I would love to be able to answer all reader questions, I get swamped with e-mail and real-life responsibilities. Please consider using these resources to get help.

# About Java

Java is one of 13,000 islands that makes up Indonesia, whose capital is Jakarta. It is home to about 120 million people with an area about 50,000 square miles (132,000 square kilometers). While on the island, you can hear traditional music such as gamelan or angklung and enjoy Java's main export, a coffee that is considered spicy and full-bodied, with a strong, slightly acidic flavor. The island also has a dangerous volcano named Merapi, which makes up part of the Pacific "Ring of Fire." In 1891, on the island, Eugene Dubois discovered fossils from Pithecanthropus erectus, better known as Java man (homo javanensis).

For more information, see `http://encyclopedia.lockergnome.com/s/b/Java_(island)`.

CHAPTER 1

# Swing Overview

According to *Encyclopedia Britannica*, Swing was a popular music in the United States, circa 1930–1945. Okay, maybe not in the Java sense. Instead, on May 23, 1995, John Gage, then director of the Science Office for Sun, introduced Java to the world. With its birth came something called the Abstract Window Toolkit, or AWT. In turn, with AWT came native widgets, and with this early native widget set came . . . trouble.

The original component set that came with the Java platform, AWT, was dependent on too many idiosyncrasies of the underlying platform. Instead of providing a mature-looking component set, Java offered the lowest common denominator version. If a feature wasn't available on all Java platforms, it wasn't available on any Java platform. And then you had to deal with all the browser/platform differences. Each Java runtime environment relied on how the component set was connected with the underlying platform-specific native widget set. If there were issues with the connection, first, they were specific to the platform (and/or browser) and second, you had to code around these problems so your programs could be write-once, run anywhere (WORA), the Java mantra of the time.

As Java technologies became more popular, users realized AWT was extremely slow and unreliable, and you couldn't really do much with the provided components. Very few of them were available, and you couldn't use them in a visual programming environment. So, new technologies were introduced, such as just-in-time (JIT) compilers to improve performance and, with Borland's help, JavaBeans for a component-based development.

With these new technologies came more and more widget sets, for the AWT component set itself was very basic. So, applet download times grew and grew, because these new widget sets weren't part of the core Java platform, and Java archive (JAR) files were introduced to improve delivery time. Eventually, each of the major browser vendors added its favorite component library to its virtual machine—AFC, IFC, and WFC, to name just a few. Yet all the libraries used different design models, and there were no true cross-browser standards.

Eventually, Sun Microsystems teamed up with Netscape Communication and other partners to create yet another library called the Java Foundation Classes, or JFC. Part of JFC is something called the Swing component set. This Swing component set is what this book is all about.

■ **Note** Later technologies were introduced to help people use the Swing components within a browser and with web-based application delivery. These include the Java Plug-in (`http://java.sun.com/products/plugin/`) and Java Web Start (`http://java.sun.com/products/javawebstart/`). Alternatives to Swing, like the SWT component set with Eclipse (`http://www.eclipse.org/swt/`), have also been created. These are not discussed here.

This chapter will familiarize you with the various Swing pieces. For starters, there is the component set. Without these, there is no Swing. Next, you'll peek at the world of event handling and layout management common to both AWT and Swing components. After that, you'll take a quick look at the undo/redo framework available within the Swing architecture. Then you'll explore the SwingSet2 demonstration provided with the Java 2 Platform Standard Edition 5.0 Development Kit (JDK 5.0) so that you can see some of the capabilities. Lastly, I'll point out where in the book all these capabilities are discussed in detail.

# Getting to Know the Swing Components

The book will serve as a guide to development using the Swing component set. Over the course of its pages, you'll look at every package in the `javax.swing` package hierarchy, as shown in Figure 1-1.

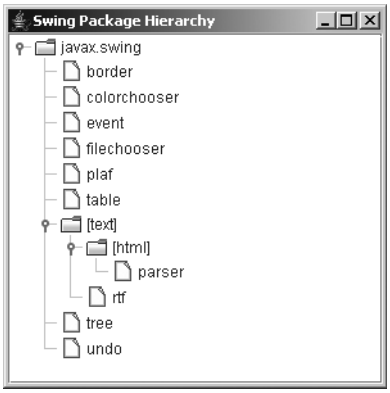


**Figure 1-1.** *The Swing package hierarchy*

■ **Note** The `javax.swing.plaf` package contains several subpackages and related packages, some of which are located outside the `javax.swing` package hierarchy. *Plaf* stands for pluggable look and feel—a Swing concept that will be described more fully in Chapter 20.

The Swing component set is one big group of components. While the JDK 5.0 release didn't add any new Swing components to the mix, logically, you can think of them as those with duplicate components within AWT and those without.

## AWT Component Replacements

The Swing component set was originally created because the basic AWT components that came with the original version of the Java libraries were insufficient for real-world, forms-based applications. All the basic components were there, but the existing set was too small and far too restrictive. For instance, you couldn't even put an image on a button. To alleviate this situation, the Swing component set offers replacements for each of the AWT components. The Swing components support all the capabilities of the original set and offer a whole lot more besides. As such, you should never need to deal with any of the basic AWT components.

---

■ **Note** Although the Swing components replace the AWT components, you'll still need to understand several basic AWT concepts, such as layout managers, event handling, and drawing support. In addition, you'll need to grasp the concept that all of Swing is built on top of the core AWT libraries.

---

The basic distinction between the Swing and equivalent AWT components is, in most cases, the Swing component class names begin with a *J* and the AWT ones don't. Swing's `JButton` is a replacement for the AWT `Button` component. One exception is the `JComboBox`, which replaces the AWT `Choice` component.

At the application programming interface (API) level, the Swing components are almost always a superset of the features the AWT components support. While they support additional capabilities, the basic AWT capabilities are there for everything but the `JList` component, whose API is completely unlike that of the AWT `List` component. Table 1-1 maps the original AWT components to their replacement Swing components.

**Table 1-1.** *AWT to Swing Component Mapping*

| AWT Component | Nearest Swing Replacement |
|---|---|
| `Button` | `JButton` |
| `Canvas` | `JPanel` |
| `Checkbox` | `JCheckBox` |
| `Checkbox` in `CheckboxGroup` | `JRadioButton` in `ButtonGroup` |
| `Choice` | `JComboBox` |
| `Component` | `JComponent` |
| `Container` | `JPanel` |
| `Label` | `JLabel` |
| `List` | `JList` |

**Table 1-1.** *AWT to Swing Component Mapping (Continued)*

| AWT Component | Nearest Swing Replacement |
|---|---|
| Menu | JMenu |
| MenuBar | JMenuBar |
| MenuItem | JMenuItem |
| Panel | JPanel |
| PopupMenu | JPopupMenu |
| Scrollbar | JScrollBar |
| ScrollPane | JScrollPane |
| TextArea | JTextArea |
| TextField | JTextField |

■ **Note** For most people, the fact that the Swing components replace AWT components is irrelevant. Just treat the Swing components as an *independent* component set, and you'll be perfectly okay.

To help you understand how to use the Swing components, you'll examine each of the components in this book. For instance, Chapter 4 looks at how the `JButton` component works, with just a single line of text as its label, like an AWT `Button`, but adds capabilities, such as using image icons on buttons and working with multiple lines of text. To find out where each component is discussed in this book, see the "Swing Component to Chapter Mapping" section later in this chapter.

In addition to replacing each of the basic components, the Swing component set has a replacement for the higher-level window objects. Although the only change in most of the components' names is the beginning *J*, you'll discover in Chapter 8 how the high-level container objects are much different in the Swing world. Swing's replacement for the old `FileDialog` object differs even more and is discussed in Chapter 9. Table 1-2 maps the high-level window objects from the AWT component world to the Swing universe.

**Table 1-2.** *AWT to Swing Window Mapping*

| AWT Window | Nearest Swing Replacement |
|---|---|
| Applet | JApplet |
| Dialog | JDialog |
| FileDialog | JFileChooser |
| Frame | JFrame |
| Window | JWindow |

Whereas the AWT components rely on the user's operating system to provide the actual component to a Java program, Swing components are all controlled from within the Java runtime. The AWT approach is called either the *heavyweight* or the *peered* approach; most Swing components are *lightweight* or *peerless*. You'll explore the basics of this approach in Chapter 4 with the `JComponent`. Additional features for customizing the look and feel of components are discussed in Chapter 20.

## Non-AWT Upgraded Components

In addition to offering replacements for all the basic AWT components, the Swing component set includes twice as many new components.

---

■ **Note** If you're new to Java, just think of all of these components—both the AWT component replacements and those that were not in the AWT—as one big set of components, versus two distinct sets.

---

Here's a look at those components that didn't originate in the AWT world:

- `JPasswordField`: This specialized text field is for password entry, as shown in Figure 1-2. You cannot use cut or copy operations within the component, but you can paste text into it.

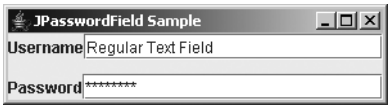


**Figure 1-2.** *The Swing JPasswordField*

- `JEditorPane` and `JTextPane`: These two components provide support for displaying and editing multiple-attributed content, such as an HTML and RTF viewer. Figure 1-3 shows a `JEditorPane` component.

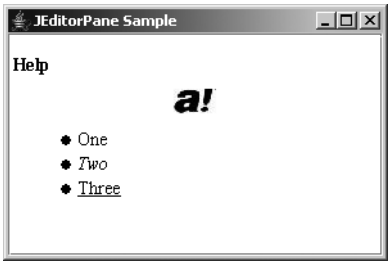


**Figure 1-3.** *The Swing JEditorPane*

- `JSpinner`: This component, shown in Figure 1-4, provides selection from an ordered set of predefined values, offering arrows to scroll through the next and previous choices. The predefined values can be an array of strings, a sequential set of numbers, or a date.

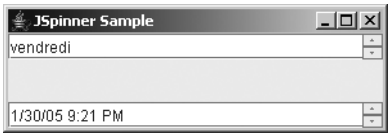


**Figure 1-4.** *The Swing JSpinner*

- `JToggleButton`: This component offers a button that stays depressed when selected. In the example shown in Figure 1-5, the North, East, and South buttons are depressed.

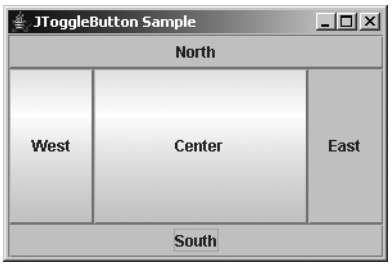


**Figure 1-5.** *The Swing JToggleButton*

- `JSlider`: This component is like the `Scrollbar` component of AWT (or `JScrollBar` in the Swing component set). However, its purpose in Swing is for user input. It offers various clues to help the user choose a value. Figure 1-6 shows an example of a `JSlider` component.

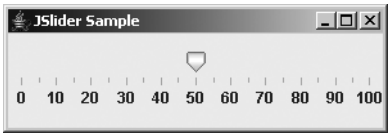


**Figure 1-6.** *The Swing JSlider*

- `JProgressBar`: This component allows the user to visually see the progress of an activity. Some options available include showing static text or percentage done, as shown in Figure 1-7.

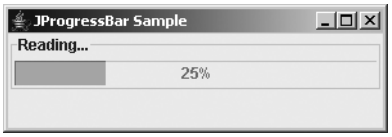


**Figure 1-7.** *The Swing JProgressBar*