Just Java™ 2 SIXTH EDITION

By PETER van der LINDEN

Publisher: Addison Wesley
Pub Date: June 21, 2004
ISBN: 0-13-148211-4
Pages: 848

The #1 introduction to J2SE 1.5 and enterprise/server-side development!

An international bestseller for eight years, ***Just Java(TM) 2*** is the complete, accessible Java tutorial for working programmers at *all* levels. Fully updated and revised, this sixth edition is more than an engaging overview of Java 2 Standard Edition (J2SE 1.5) and its libraries: it's also a practical introduction to today's best enterprise and server-side programming techniques. ***Just Java(TM) 2, Sixth Edition,*** reflects both J2SE 1.5 *and* the latest Tomcat and servlet specifications. Extensive new coverage includes:

- New chapters on generics and enumerated types
- New coverage of Web services, with practical examples using Google and Amazon Web services
- Simplified interactive I/O with printf()
- Autoboxing and unboxing of primitive types
- Static imports, foreach loop construct, and other new language features

Peter van der Linden delivers expert advice, clear explanations, and crisp sample programs throughout—including *dozens* new to this edition. Along the way, he introduces:

- **The core language:** syntax, objects, interfaces, nested classes, compiler secrets, and much more
- **Key libraries:** date and calendar, pattern matching, network software, mapped I/O, utilities and generic collections
- **Server-side technology:** network server systems, a complete tiny HTML Web server, and XML in Java
- **Enterprise J2EE:** Sql and JDBC(TM) tutorial, servlets and JSP and much more

- **Client-side Java:** fundamentals of JFC/Swing GUI development, new class data sharing details

Companion Web Site

All the book's examples and sample programs are available at http://afu.com.

Copyright

Printed in the United States of America.
4150 Network Circle, Santa Clara, California
95054 U.S.A.

Library of Congress Control Number: 2004107483

Prentice Hall PTR offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact U.S. Corporate and Government Sales, 1-800-382-3419, corpsales@pearsontechgroup.com**. For sales outside of the U.S., please contact International Sales, 1-317-581-3793,** international@pearsontechgroup.com.

Acquisitions Editor: *Gregory G. Doench*

Editorial Assistant: *Raquel Kaplan*

Production Supervision: *Julie B. Nahil*

Editor: *Solveig Haugland*

Cover Designer: *Nina Scuderi*

Art Director: *Gail Cocker-Bogusz*

Manufacturing Manager: *Carol Melville*

Marketing Manager: *Chris Guzikowski*

Sun Microsystems Press Publisher: *Myrna Rivera*

Text printed on recycled paper

1 2 3 4 5 6 7 8 9 10—CRW—0807060504

First printing, June 2004

Dedication

I used to dedicate these books to specific brands of real ale; it was always good for lunch with the chairman when I toured the brewery. Here's something different. I've worked in the computer industry all my life, and in the last few years started teaching programming classes too. You learn a lot when you get immediate feedback about topics that need to be expressed in easier pieces. I would like to dedicate this book to all my students past and present, and all my teachers, young and old.

Preface

The first edition of *Just Java* was one of the earliest books to accompany the original release of Java in 1996. The launch of Java coincided with the explosion of interest in the web and the net which, in turn, drove technology forward at a frantic pace. People talked about "Internet time," which meant three things to me in Silicon Valley: there was immense pressure to rapidly create new hardware and software products; everyone wrote software to display stock prices on their desktops and cell phones; you were forgiven for not showering if you fell asleep at your desk after midnight and woke up there the next morning. Times have changed, but software productivity remains a big reason behind Java's popularity.

Over the last eight years Java has had six major releases, averaging one about every 18 months. With each of these releases, there has been a new edition of *Just Java* to describe and explain the technology. Table 1 shows how the language and libraries have improved.

Table 1. Java changes from JDK 1.0.2 to Java 2 v1.4

Release	Date	Content	See Just Java 6th ed.
JDK 1.0.2	Jan 1996	First general release of the language and libraries	Throughout the book
JDK 1.1	Feb 1997	*Language changes*:	
		Instance initializers	Chapter 5
		Array initializers	Chapter 9
		Nested classes	Chapter 12
		Library changes:	
		Delegation based event-handlers	Chapter 20
		I/O Readers and Writers	Chapter 17
		Object serialization	Chapter 18
JDK 1.2 (rebadged to Java 2)	Dec 1998	*Language changes*:	

to Java 2)	1998	strictfp	Chapter 7
		Weak references	Chapter 10
		Library changes:	
		Java Foundation and Swing	Chapter 21
		Collection classes, JDBC enhancements	Chapter 16, 23-24
		Thread local storage	Chapter 14
Java 2 v1.3	May 2000	Performance and bug fixes, no significant changes	Throughout the book
Java 2 v1.4	Dec 2001	*Language changes*:	
		Assert statement	Chapter 10
		Library changes:	
		Regular expressions	Chapter 19
		New I/O (third attempt)	Chapter 18

This is a remarkable pace of development for a programming system, particularly when Sun keeps such an emphasis on backward compatibility and portability. The Java 1.2 release was a significant one, bundling major functionality improvements like the collection classes and the Swing GUI library. Java 1.3 and 1.4 were comparatively smaller, although 1.4 did bring a new statement ("assert") into the language.

Two and a half years in the making, Java 1.5 is the biggest version yet. It is bigger and more significant than JDK 1.2. Sun will probably rename Java 1.5 to some awkward and confusing name using two sets of numbers, like "Java 2 Mega-edition v1.5 fab-o-lux". Whatever they call it, think of Java 1.5 as "Java 3". Table 2 shows some of the substantial language additions.

Table 2. Java 2 v1.5

Release	Date	Content	See Just Java 6th ed.
Java 2 v1.5	Jun 2004	*Language changes*:	
		Autoboxing and unboxing	Chapter 3
		Enum types	Chapter 6
		Generic types	Chapter 15, 16
		Variable-arity methods	Chapter 5
		Static import	Chapter 6
		Enhanced for loop	Chapter 4
		Covariant return types	Chapter 11
		Library changes:	
		printf (like C's printf)	Chapter 17
		java.util.scanner (fourth attempt at fixing I/O)	Chapter 17
		java.util.concurrent thread utilities	Chapter 14
		javax.xml XML support bundled	Chapter 27, 28
		Class data sharing	Chapter 2
		Can add Swing components directly to a Container!	Chapter 21

There are also the traditional bug-fix, library and performance improvements, including some exciting optimizations for desktop applications.

Over the years, I've put a lot of hard work into unlocking the changes in Java, so you don't have to. You're looking at the results of that effort: the sixth edition of *Just Java*.

I'm confident you'll find it easy to read, and packed with the information you need.

I hope that you'll want a copy for yourself.

But if not, I want you to put it back on the shelf, only (as my friend Alan Abel suggested) in a more prominent position.

— P.

Acknowledgments

I'm very grateful to the following people, who are some of the most talented and creative individuals you'll find:

Gilad Bracha

Michael Davidson

Jane Erskine

Marcus Green

Roedy Green

Trey Harris

Karsten Lentzsch

Bob Lynch

Aleksander Malinowski

Simon Roberts

Rick Ross, who provides first-class leadership at http://javalobby.org

Kerry Shetline

Robin Southgate

Lefty Walkowiak

All the cowboys and cowgirls down on the Java Ranch http://javaranch.com

The Limewire team

The unsung heroes and heroines of Sun's Java and Solaris groups

The editorial, marketing, and production teams at Prentice-Hall and Sun Microsystems deserve full appreciation:

Greg Doench, Chris Guzikowski, Julie Nahil, Raquel Kaplan, Nina Scuderi, and Solveig Haugland and her magic editing pixies.

Thanks too, to my wife, family, and friends—hey, if I *wanted* my study to look organized, I'd *keep* it that way, OK?

Part 1: Language

Chapter 1. What Can Java Do for Me?

Java has become a very popular programming language for the kind of modern software people want to write. Java began as a research project inside Sun Microsystems. The results were posted on the web, and Java took off like a Titan rocket. Sun wisely decided to nurture the market by sharing, and licensed the technology across the computer industry. Today, Java compilers and source code are available for free download from many different organizations. Just about the entire computer industry is backing Java enthusiastically with products and support. In this chapter we'll look at the reasons behind Java's popularity, and summarize the key features of the language.

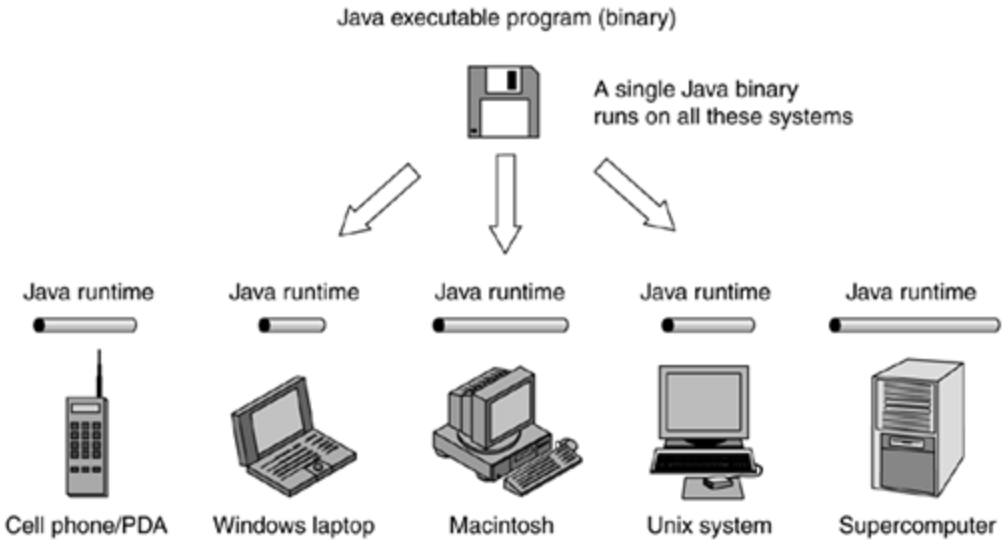
Java works well for web and server-based applications. It has great features like object-oriented programming, easy database access, and well-designed GUI support. The latest release, Java 2 version 1.5, has performance tweaks to speed up desktop programs. There are libraries for communicating across networks, and for encryption. It has strong security built in. Most programmers pick up Java quickly. Behind all this, Java is more than just the latest, most popular programming language. It is a way of creating applications that are independent of all hardware and all operating systems.

What Java Does for You: Software Portability

What is meant by "applications that are independent of all hardware and operating systems?" It means you can compile a Java program on any system and run the resulting binary executable file on the same or any other system—on a Macintosh, on Windows 98, NT, 2K, XP, on Solaris, Linux, BSD or any of the varieties of Unix, on IBM's mainframe operating systems, on cell phones, Personal Digital Assistants (PDAs), embedded processors, and even on smart cards (credit cards with a microprocessor and memory), as shown in Figure 1-1.

Figure 1-1. Future-proof software: Your Java application runs on every system. No more "requires Windows XP" or "Linux PPC only" , or "compiled for MacOS X version 10.5"; just "built with Java," and you're done.

[View full size image]



You will even be able to run on future operating system releases, like Microsoft's Longhorn. Java will be ported to all future operating systems of significance. It's also a fast way for new hardware to get software applications that will run on it.

Java unlocks software from being coupled to any specific OS and hardware platform. This new flexibility has made Java very popular with users, IT departments, and software vendors. All three benefit enormously from software portability.

Why Portability Matters

You might think that software portability does not affect you: your application software runs fine on your PC today and that's all you use. And that's true, right up until the time you want to consider a new or different system.

Say you're in the market to buy a new desktop system, and a friend shows you the video-editing, music, virus resistance, and digital picture capabilities of his Apple Macintosh. You consider switching to a Mac. Switching GUIs is a no-brainer—GUIs all do essentially the same things, and it only takes a day or two to re-train your fingers. The problem is the apps. You are faced with the "choice" of walking away from your investment in your existing PC-only software, or buying yet another Windows PC that has some compatibility with your previous system. You can easily switch hardware from Dell to H-P or IBM, but there's a software barrier to switching from Windows to something with fewer security problems, like Linux or MacOS. You've been locked in.

When your application programs are written in Java, you can upgrade OS and applications independently. You can try Linux and still use your familiar applications. You can move your existing Java programs to any new system, and carry on using them. This is why software portability matters to home users.

For businesses, the problem is worse and far more expensive. Even if your whole organization has standardized on, say, Microsoft Windows XP, there have been numerous releases over just the last decade and a bit: MS-DOS, Win 3.1, Win3.11, Win95A, Win95B, Win98, 98 SE, ME, NT 3.1, NT3.5, NT3.51, NT4, 2K, multiple service packs and required hot-fixes, XP, and Longhorn on the horizon. These

platforms have subtle and different incompatibilities amongthem. Even applications running on a single platform have limited interoperability. Older versions of Microsoft Office cannot read files produced by default from the latest Microsoft Office, even when the files don't use any of the new features.

This is done deliberately, to force upgrades. If even one person in an office upgrades, everyone has to (or risk being cut off from reading new files).

Software portability is all about "future-proofing" your software investment. Rewrite it in Java, and that's the last port you'll ever need to do. Portability is the Holy Grail of the software industry. It has long been sought, but never before attained. Java brings the industry closer to true software portability than any previous system has.

Software portability for office applications

There is today an excellent alternative to costly and incompatible MS Office updates. You can download the free OpenOffice.orgsoftware and use it instead of MS Office.

MS Office Professional Edition 2003 costs $499 in the USA for the basic product—for one computer. OpenOffice.org has the same look and features, and is free for any number of computers. OpenOffice.org can read and save files in MS Office formats. You can even get the source code. Over twenty million users have downloaded it, and you can get your copy at:

www.openoffice.org

There are OpenOffice.org versions available for Linux, Windows, MacOS, and Solaris, in many different national languages. OpenOffice.org is not written in Java, so it needs an executable for each platform. OpenOffice.org (like Java) is based on source code made freely available from Sun. There is a programmers guide and SDK at http://www.openoffice.org/dev_docs/source/sdk/

Java and jobs

Software portability has a *wonderful* side-effect: skills portability for programmers. Many companies today are out-sourcing programming jobs to countries with low direct labor costs. It's a short-term cost-saving that looks good on paper, until you look at the wider implications.

The beneficiaries of jobs exported from the West are Asian countries with poor performance in the annual global survey of corruption (see www.transparency.org). It's a risky long-term bet to move strategic expertise to countries that combine widespread corruption, unproven business privacy and intellectual property laws, with scant free market experience, and no effective environmental or workplace regulation. The bigger picture needs to be thought through when choosing to export jobs and assets from our home economies. As people with a stake in the computer industry, we have a duty to make our views (whatever they are) known to politicians.

Looking at this another way, China (a nation of 1.3 billion) has already standardized on Linux, and India (1 billion people) is reviewing it. Java offers the only viable way to create software targeted at both the West and at emerging markets.

If you're a programmer in the USA or Western Europe affected by jobs moving offshore, Java is a big plus for your career. Employers used to advertise for very specific hardware and OS experience ("must have 2 years of MVS on OS/390") and ignore other resumes. Today, your Java experience gained on any OS is directly transferable to other hardware and jobs.

Java is in demand by employers. An April 2004 review of one of the US's largest job sites reflected these hiring needs. Microsoft's C# was mentioned in about 1400 postings, while C++ was a requirement in about 4000 postings. That's what you would expect. C++ is much more widespread than C#, and runs on many more computers. But Java was a surprise: more than 6800 postings sought Java skills. More employers wanted Java experience than those who wanted C++ and C# combined. This is just an anecdotal datapoint, but it is consistent with other surveys. The *Software Development Times* paper reported in December 2003 that nearly three-quarters of enterprise software development managers are using Java and another 11% plan to start in the next year. Investing some of your time in Java is good for your career.

Java and Microsoft

Java portability poses a real threat to Microsoft's monopoly. Software that can run on any operating system has a larger market than software that is limited to Windows only. Over time ISVs will move their products away from Windows-only to Java—unless Java can somehow be spoiled or broken.

It is unfortunate for you, me, and all computer users that Microsoft uses its monopoly to try to undermine Java. At first, Microsoft introduced deliberate incompatibilities into the Java product it licensed from Sun. Microsoft paid $20Mto Sun to settle the resulting court case. In April 2004, Microsoft paid Sun a further $1.9 billion to settle Sun's litigation over other monopoly abuse.

The current Microsoft plan is to push the C# language, which is Microsoft's barely different copy of Java. But the core C# libraries will be only ever be available on Windows (there's an open source effort to duplicate some C# libraries on Linux, but few believe it will lead anywhere). The C# initiative will last only until Redmond wants to push the Next Incompatible Big Thing. Java is shaped by the computer industry as a whole, and will be around until industry reaches consensus that there is something better to replace Java.

Language and Libraries

Let's spend a minute to review some software terms that are often taken for granted. If you are already familiar with this, just skip ahead until you reach something new. The terminology is spelled out in detail here to give a solid basis for the material that follows in the rest of the book.