Java Secrets
by Elliotte Rusty Harold
IDG Books, IDG Books Worldwide, Inc.
ISBN: 0764580078 **Pub**
Date: 05/01/97

Table of Contents

Preface

There are more than 100 books about Java on bookstore shelves today, and at least 90 of them are completely predictable and more or less interchangeable. It's as if they had all been written from the same outline but by different authors.

Each book begins with a chapter about what's special about Java and how it differs from other programming languages. Each book shows how to write Hello World and other command-line applications to teach Java's syntax. There is a chapter or two on object-oriented programming, a chapter on threads, a chapter on exceptions, and a few chapters on the AWT. I know. I wrote one of these books.

Why *Java SECRETS*?

This book is different. It starts where the other books stop. This book assumes that you already know Java's syntax and what an object is. This book assumes that you're comfortable with the AWT. Instead of rehashing these topics, this book delves into the parts of Java that are not documented by Sun, that are not generally accessible to anyone with a Web browser, and that are not already in a hundred other books.

I had some reservations about writing this book. I still do. This is a dangerous book. It reveals knowledge that can easily be abused. Improper use of the secrets revealed herein can easily tie Java programs to specific platforms or implementations. As a longtime Mac user, I know the agony of watching all the best software come out on Windows first and the Mac much later, if at all. I do not want to extend this trend to Java-based software.

Nonetheless, I have come to the conclusion that a book like this is necessary if Java is to move out of its niche of creating applets for Web pages and into the broader software development market. There are many applications for which Java is ideal, but which cannot be written without more information than Sun has chosen to reveal. These include stand-alone executable applications. HotJava and javac are stand alone applications, so it must be possible to write them, but until now, Sun has not revealed how. This book reveals that secret among others.

There are other reasons programmers want to know these details. Just as in the early days of DOS when you needed to use undocumented functions to load a program without executing it so you could write a debugger, so too will you need to use undocumented parts of Java if you're working on development or runtime environments.

However, rationalize though I might (and I'm quite good at rationalizing, I admit), the real reason this book is being written is that it seemed like a neat thing to do at the time. This is far and away the most exciting book I've ever written. The sheer number of "Aha!" experiences I've had while researching and writing it is phenomenal. I hope you'll get the same feeling while reading it. I know the information I present here will be misused. I accept that. Nonetheless, I firmly believe that in the long run, more knowledge is a good thing, dangerous though it may be; and that secrets are meant to be revealed.

What's in This Book?

There are three different ways a Java program can become dangerous. It can rely on the internal structure of Java objects; it can use classes it isn't supposed to know about; or it can be platform-specific. This book covers all three.

Part I: How Java Works

After a brief introduction, Part I begins with six chapters on Java internals. You learn how objects and primitive data types are laid out in memory, how arguments are passed to and values returned from methods, what a variable really is, and more. Java's implementation of arrays and strings will be explored. Different possible models for threads and algorithms for garbage collection are discussed and compared, shedding some light on why Java uses the data structures and algorithms it does and why it sometimes behaves in unexpected ways. This is all tied to the Java .class file format in Chapter 5, where you learn how to read and disassemble Java byte code. You also learn some details about Java's thread model and garbage collection algorithms.

Finally, you learn how an applet runs and what really happens when a Web browser loads an applet.This section is dangerous because none of it is guaranteed. Tomorrow, Sun could change Java's thread model from cooperative to preemptive or make strings null-terminated. Worse yet, Java might be implemented one way on one system and another way on another. Writing code that depends on implementation issues is always dangerous but sometimes necessary.

Nonetheless, it often helps to know what's going inside a class or method even if you don't explicitly use that information. For example, knowing whether the Vector class is implemented with a growable array or a linked list has a lot to do with whether you choose to use it in a program that performs thousands of insertions in the middle of a list. You can drive a car without knowing the first thing about carburetors or transmissions, but it certainly doesn't hurt to know about them, especially when things go wrong. Knowing what goes on the under the hood but ignoring it when it isn't relevant is a good technique for both programmers and drivers. Not knowing isn't.

Some may object that this goes against the philosophy of object-oriented programming. Objects are supposed to be black boxes into which data is sent and out of which a result flows. You aren't supposed to need to know what happens inside the box. However, objects aren't everything, and practical experience shows that time and time again, the black box doesn't do exactly what it's supposed to and you need to open it up and fix it. Part I opens up many black boxes to expose their inner workings.

Part II: The Sun Classes

Part II delves into the sun classes, a group of undocumented packages that add considerable power to Java programs. The following are just a few of the undocumented classes that will be covered in this section:

- More LayoutManagers
- Communicating with ftp, mail and news servers
- Data encoding and decoding
- Character set conversion
- Protocol and content handlers

As you can see, Sun has hidden a lot of functionality inside the Sun classes. This book reveals it.

Part II is dangerous because these classes may not be present in future releases of Java. They may not even be present in Java implementations not written by Sun. If they are present, their public methods may not have the same signatures. Nonetheless, they provide too much additional power to be ignored, and there are some very simple techniques that allow one to use these packages safely in even non-conforming implementations.

Part III: Platform-Dependent Java

Part III explores the possibilities opened by platform-dependent code. It demonstrates how to call the native API and how to create stand-alone executable programs.

This part is dangerous because it limits the audience of a program. It's also dangerous because it violates many of the security restrictions normally imposed on Java programs. Nonetheless, not all programs are applets on Web pages. Many programs can benefit from taking advantage of native code, either for speed or to add additional functionality not present in the AWT. There are ways to use platform-dependent code to enhance your application without making your program inaccessible to users on all other platforms. This section will explore these possibilities.

Part IV: Appendixes

Part IV provides several appendixes to help supplement those skills you learn from Parts I through III. You can use these appendixes as handy references to relevant information as you learn. Included is an appendix that describes the contents of the accompanying CD-ROM.

Icons used in this book

You'll notice some special icons sprinkled throughout this book to draw your attention to the information at hand. The following briefly describes the use of these icons:

Note: This icon identifies information that is particularly noteworthy or helpful.

Note: This icon alerts you to information that, for one reason or another, is undocumented or is not common knowledge. This information can contain time-saving tricks and techniques or nifty facts that will enhance your understanding and learning of Java.

Who You Are

This is not an introductory book. It is for the programmer who has learned enough about Java to be frustrated by its limitations. You should have a solid grasp of the fundamentals of both the Java language and the AWT, including advanced topics like threads. Although every effort has been made to make this book accessible to as broad a range of readers as possible, this is not an introductory book and does require more of its reader than most books on the market.

On the other hand, this book does not assume prior experience with assembly language, Java byte code, compiler design, or even pointers. In fact, this book may serve as a first taste of some of these to a reader who's never seen them before, in Java or any other language. Nonetheless, low-level programmers who are familiar with pointers, assembly language and compiler design should find the discussion of Java's implementation of these topics to be useful. They'll simply find the book easier going than a programmer encountering these topics for the first time.

How to Use This Book

As mentioned earlier, this book is broken into four main parts. I recommend that you begin by reading or at least skimming Part I more or less in its entirety. This section introduces many deep concepts you'll need later and that the rest of the book depends on. These include bit-shift operators, Unicode, the nature of strings, the virtual machine, the class file format, and Java byte code. These are the tools you'll need to understand the internals of Java.The remainder of the book (Chapters 6 through 20) can be read in pretty much any order that interests you. As a general rule, these chapters are pretty much independent of each other. While each chapter should probably be read from start to finish, the chapters themselves are mostly self-contained.

Bugs

This book is so far out on the bleeding edge, I've got a personal account rep at the New York Blood Bank. I've done my best to try to provide useful and accurate information. All the code in his book has been verified on at least one virtual machine (VM). Most of the code has been tested on two or more. However, because Java runs on so many different platforms and because it is changing in Internet time, it is impossible to be completely precise and accurate in all instances. Furthermore, precisely because the material in this book is secret, it's been extremely hard to verify.

Please use this information carefully and read it with a critical eye. If you do find mistakes or inaccuracies, let me know by sending e-mail to `elharo@sunsite.unc.edu`, and I'll correct them in future editions. I will also post corrections and updates on my Web site at http:// sunsite.unc.edu/javafaq/secrets/, so you may wish to look there first before sending me e-mail. When you communicate with me about a problem you've found, please let me know the VM, version of Java, vendor, processor, and operating system you're testing with. By early 1997, there were already more than 100 slightly different virtual machines is use, so it's important to be as precise as possible.