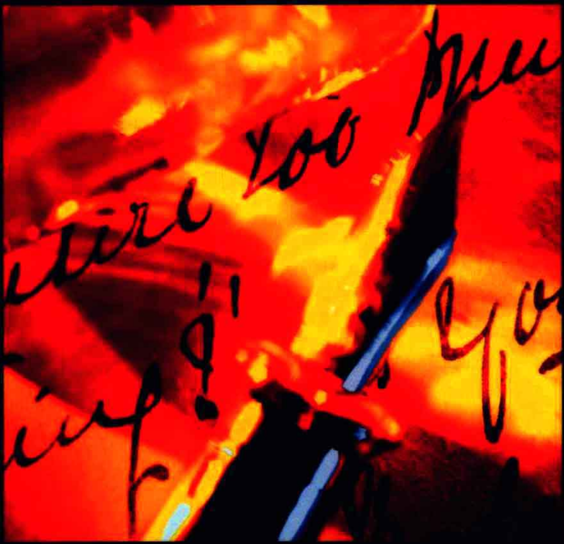
essential

JNI

JAVA™ NATIVE INTERFACE



- Integrate Java applications with C/C++ legacy code
- Invoke Java methods & create Java objects from a native application
- Create a Java VM within a native application
- Includes up-to-date coverage of Java 1.2

ROB GORDON

essential

JNI

JAVA™ NATIVE INTERFACE

R O B G O R D O N

Prentice Hall PTR
Upper Saddle River, NJ 07458
http://www.phptr.com

Library of Congress Cataloging-in-Publication Data

Gordon, Rob
Essential JNI: Java Native Interface / Rob Gordon
p. cm.
Includes index.
ISBN 0-13-679895-0 (paper)
1. Java (Computer program language) 2. Application software.
Title.
QA76.73.J38G665 1998
005.13'3--dc21 97-44105
CIP

Editorial/Production Supervision: *James D. Gwyn*
Acquisitions Editor: *Gregory Doench*
Series Editor: *Alan McClellan*
Manufacturing Manager: *Alexis R. Heydt*
Marketing Manager: *Kaylie Smith*
Editorial Assistant: *Mary Treacy*

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-679895-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

essential

JNI

JAVA™ NATIVE INTERFACE

Series Editor Alan McClellan is coauthor of the best-selling *Java by Example*, *Graphic Java: Mastering the AWT*, and *Automating Solaris Installations: A JumpStart Guide* (SunSoft/Prentice Hall). He is an award-winning software technical writer with over ten years of experience in the computer industry.

For Kathy, Matthew, Kilian and Kieran
who define my place in the world.

PREFACE

Before we plow an unfamiliar patch
It is well to be informed about the winds,
About the variations in the sky,
The native traits and habits of the place,
What each locale permits, and what denies

Virgil
The Georgics

What This Book is About

The subject of this book is the Java Native Interface (JNI) Application Programming Interface (API). The JNI was introduced in release 1.1 of the Java Development Kit (JDK) as distributed by JavaSoft. This book covers the entire API for the JNI including the enhancements introduced in release 1.2 of the JDK. Where there are minor differences between various 1.1 point releases, these are discussed.

Who Should Read This Book?

This book is written for the software engineer who needs to make Java and C or C++ talk to one another. Experience with C/C++ and Java is assumed. This book also assumes some familiarity with both UNIX and Win32 platforms.

If you are a Java programmer who needs to step outside the Java Virtual Machine to take advantage of some platform-specific functionality, this book will show you how. If you are a C programmer responsible for putting a Java font-end on a legacy application, this book will show you how. If you are a

C++ programmer wanting to take advantage of an existing C++ class library, this book will show you how.

Further, this book covers these topics for both UNIX and Win32 platforms.

Okay, now turn around and walk to the sales counter.

Structure of This Book

This book can be thought of as having three distinct parts. The first part, roughly the first eight chapters, covers the JNI API in great detail. The second part, the remaining chapters, covers some general issues involved with native method programming. The third part, a series of appendices, contains reference material, both for the JNI and for tools introduced in this book. There is also an appendix that compares the JNI with the old-style native method programming model introduced in JDK 1.0. The last appendix offers a brief discussion of native methods, applets and security issues.

The early chapters contain plenty of simple examples intended to highlight the essential features of the API. No attempt is made to place JNI function calls into large, complex examples that obscure their salient features.

The first part of the book takes a walk-before-you-run approach. After an overview of the JNI in Chapter 1, Chapter 2 presents a JNI version of the classic "Hello World" example. Chapter 3 then follows with examples of some of the more common JNI operations before plunging into the syntactical details of the JNI in Chapter 4.

With all that work behind you, Chapter 5 through Chapter 8 provides detailed coverage of the remaining JNI functions.

The second half of the book deals with a series of general topics on using the JNI to integrate Java code with non-Java code. Chapter 9 presents an approach for mirroring existing C++ classes in Java. Chapter 10 introduces a tool for the automatic conversion of C structures into Java classes and an accompanying set of adapter functions for copying data between an instance of a C structure and a Java object.

Chapter 11 starts with a collection of Java classes that provide a high-level interface to serial and parallel ports. Throughout this chapter the native code for targetting these classes for both POSIX and Win32 platforms is presented and discussed. The Java package used in this chapter is the *portio* package which is freely-available from Central Data (`www.cd.com`) as well as being included with the examples at the Prentice-Hall ftp site.

Chapter 12 and Chapter 13 deal with the Invocation API. Chapter 12 is a broad discussion of the mechanics involved with starting a Java Virtual Machine from a C/C++ application. Chapter 13 provides a very specific example of this facility, namely, starting the Java Virtual Machine (JVM) as an NT service. This chapter is for NT developers who wish to use Sun's JVM as an engine for their Java applications.

Chapter 14 presents some approaches to debugging a Java application that includes native code. Finally, Chapter 15 is dedicated to changes and enhancements to the JNI in the JDK 1.2.

Whose Java?

This book covers the Java Native Interface API. To use the Java Native Interface API, you will need to run your Java code on a Java Virtual Machine that supports the JNI. The list of JVM vendors supporting the JNI is growing, but a sure bet is the JVM distributed by Sun. The Sun JVM, the Java Core classes and the JNI are available as part of the Java Development Kit (JDK). The JNI is supported starting in release 1.1 of the JDK.

To download the JDK, surf to the JavaSoft download site

`http://java.sun.com/products/jdk/1.1/index.html`

and follow the instructions. The JDK 1.2 release is available from:

`http://java.sun.com/products/jdk/1.2/index.html`

Downloading Example Source

The source code to the examples in this book is available at the Prentice Hall `ftp` site, `ftp.prenhall.com`. This site is available via anonymous `ftp`. The following sequence of steps illustrates how to download the *Essential JNI* examples. You type the bold.

```
% ftp ftp.prenhall.com
Connected to iq-ss3.prenhall.com.
220 iq-ss3 FTP server(UNIX(r) System V Release 4.0) ready.
User (iq-ss3.prenhall.com:(none)): anonymous
331 Guest login ok, send ident as password.
Password: <type your user name, e.g. username@someisp.net>
230 Guest login ok, access restrictions apply.
(1) ftp> cd /pub/ptr/professional_computer_science.w-022/
gordon/essential_jni
250 CWD command successful.
ftp> get ejni_ex.tar <or ejni_ex.zip>
200 PORT command successful.
150 ASCII data connection for ejni_examples.tar
    (38.11.232.6,1099) (19818 bytes).
226 ASCII Transfer complete.
20184 bytes received in 5.93 seconds (3.40 Kbytes/sec)1
ftp> bye
Goodbye.
```

1. Do not take these file sizes and transfer times at face value. A test run generated these values.

The `ftp` directory named in line [1] contains two examples files described in the following table.

File Name	**Format**	**Size**
`ejni_ex.zip`	zip	452KB
`ejni_ex.tar`	tar	1.55MB

These two files include makefiles for both Win32 and Solaris.

After downloading the examples, unzip or untar the file in a directory of your choice. Users of a UNIX zip utility should be careful to use the `-a` flag with unzip so that `CR/NL` sequences are properly converted.

Extracting the files will create a directory named `ejni`. Within the `ejni` directory will be another directory, `examples`.

The directory `ejni/examples` is known as the *examples directory* throughout this book.

Building the Examples

Before the examples can be built and run, some environment must be set up.

Setting Up the Environment

Two files are included with the example files that describe the appropriate environment settings. The file `csh.env` is provided for UNIX csh users. It is easily modified for other shells. The file `win32.bat` is provided for Win32 users. Both files must be modified to specify the location of your JDK and the location of your examples directory. Specifically, two environment variables are provided, `JDK_HOME` and `EJNI_HOME`, which point to these locations.

Additionally, these files define an appropriate setting for the `CLASSPATH` environment variable, adding the directories required to run the examples.

Finally, there are some platform-specific environment variables that need to be set.

Solaris Environment

The environment variable `LD_LIBRARY_PATH` is set within the `csh.env` file. The directory in which the example native libraries are installed is added to your existing value.

The Solaris build assumes you are using Sun's compilers. Be certain to modify the `CC` settings in `EJNI_HOME/Makefile.master` to point to the proper installation directory, or comment it out and use your environment settings.

The examples in Chapter 9 require Rogue Wave's Tools.h++ class library. If you have this software, you will need to set the relevant macros in

EJNI_HOME/Makefile.master to the appropriate values. If you are a UNIX user and don't have the Tools.h++ software, remove the target chap9 from the SUBDIRS macro in EJNI_HOME/Makefile. Your enjoyment of this example will be necessarily limited to viewing the source.

Win32 Environment

Win32 users can use the batch file win32.bat located in EJNI_HOME to set up their environment for building the examples. The file win32.bat batch assumes you are building using Microsoft's compiler, specifically Visual C++ and cl.exe. The appropriate environment for running cl from the command line is set using the vcvars32.bat batch file shipped with Visual C++. This file is located in your Visual C++ bin directory, typically \Program Files\DevStudio\VC\bin. vcvars32.bat should be run before running win32.bat.

If you get the message "Out of environment space," you will need to use the command command from the DOS prompt as shown below.

• User Input **Increasing DOS Environment Space**

```
C:\> command /e:8192
```

This setting will only be in effect for the current DOS shell. To set this value permanently, add the following line to your CONFIG.SYS file.

```
shell=command.com /e:8192 /p
```

How To Build

Once the environment is set up you are ready to build. On Solaris, the following make commands, executed from the EJNI_HOME directory, build the Java class files and native libraries.

• User Input **Building on Solaris**

```
% make all; make install
```

On Win32 systems, the nmake command is used to build the example Java class files and native DLLs. This command must be executed from the EJNI_HOME directory.

• User Input **Building on Win32**

```
% nmake TARGET=all /f nmake.mak all
```

When the build is complete, all the class files live in `$EJNI_HOME/classes` and all the native libraries live in `$EJNI_HOME/`*platform*`/lib` where *platform* is either `sparc` or `win32`. Tilt the slashes appropriately for Win32 platforms.

A Word About the Examples

All of native coding examples are written in either C or C++. As you will learn, the syntax for calling a JNI function differs across the two languages.

In the first part of the book, through Chapter 8, all of the examples are written in C++. Therefore, whenever a reference is made to the *first* argument to a JNI function, it is understood not to count the `JNIEnv` pointer argument that is present in a C call of a JNI function.

The examples directory contains a directory for each chapter. Within the chapter directory is a directory for each example. Example 1 in Chapter 5 resides in `ejni/examples/chap5/example1`.

A `README` file in each `exampleN` directory describes how to run the example since the book does not actually show all the examples running.

Some Conventions

All code listings, output, all variable, method and function names that appear in the text, command names, directory names, and all URLs appear in `courier`.

User input appears in **`bold courier`**.

Special terms, upon their introduction, and parameters to be substituted for appear in *italics*.

Pseudo-code and labels within figures appear in helvetica.

Help From Fellow Travellers

There are at least two list-servers available for JNI programmers. The first deals directly with native method programming. On the jnative-l list you will find discussions of JNI as well as old-style JDK native method programming, Microsoft's Runtime Native Interface (RNI), Netscape's Java Runtime Environment (JRE) and, to a lesser extent, problems with incorporating native code with various browsers.

To subscribe to this list, send an e-mail message to majordomo@lists.teleport.com with the text

```
subscribe jnative-l
```

in the body of the message.

It is not unusual for questions related to native method programming to appear on the Advanced Java list. To subscribe to this list, send mail to majordomo@xcf.berkeley.edu with an empty subject and the following body:

```
subscribe advanced-java your_email_address
end
```

Do not use this latter list injudiciously. It is heavily patrolled by self-appointed experts who spend as much time telling people their question is not sufficiently “advanced” as they do providing answers to those question that they deem worthy of their consideration.

To stay in step with updates and enhancements to the JDK and the JNI, you may want to register with the *Java Developer’s Connection (JDC)*. To do so, follow the link off the JavaSoft home page (`java.sun.com`). As a member of the JDC, you are eligible for Early Access APIs. This is also the place to submit and track bugs. The JDC is a free service from JavaSoft but you do have to register.

A Word About Borrowed Words

Reading the quotes at the beginning of each chapter is a requirement for successful completion of this book. Too many late nights were spent finding them and too many phone calls were made acquiring permission to use them for you to ignore them.

Why include them at all? A guy who takes writing seriously has to do something to catch the eye of the reviewers at *The New York Times Review of Books*. Besides, it was fun trying to decorate the often bland minutia of a programming API with the distinguished dress of great ideas. I don’t know if it worked, but I highly recommend all the quoted authors. Well, I can take or leave Nietszche. As for Hobbes, I think he had it backwards. We are heading toward where he thinks we began.

ACKNOWLEDGMENTS

When I grow up I want to work on computers and I am going to write a book but I won't have any kids so I can finish it.

Matthew Gordon
6 years old

That these words are being written is proof that a father of three very high-energy, as the politically correct, early childhood educators are wont to say, boys, as the not so politically correct, exhausted parent is willing to acknowledge, can finish a book.

What the presence of these words on paper can not even begin to hint at is the extent to which finishing a book depends on the patience, perseverance, forbearance and tolerance of, no, not the author, but the people closest to him: his wife and his three sons.

Kathy deserves a big thanks because, well, if she had a nickel for every time somebody asked her how she puts up with me, she would be a rich woman. What more need I say?

Then there are little ones: Matthew, Kilian and Kieran. If they were not in my life, I may not have undertaken this. It is a long story so you will have to take me at my word. When they get older, I will explain it to them, although as intuitive as they are, I suspect they have an idea even now.

Beyond those close to me who helped by their forbearance, there are many more who helped by their active participation. Steve Talley was with me as every chapter came off the printer providing critique both fair and fast. Peter Rivera, who must now hold some sort of record for pre-publication review of

Java books, also lent his sharp eye to reviewing an early draft of many chapters. Dave Geary provided spiritual support to the very last day.

Four additional technical reviewers, Buzzy Brown of the IBM Retail Store Solutions Group, John Gray of the University of Hartford, Francoise Duclos of Kodak, and Jeff Barr all provided valuable input after thorough readings of the manuscript.

There are numerous people at Prentice Hall, many nameless, who helped get this book from my computer to your bookstore. Greg Doench provided a steady hand from beginning to end. Jim Gwyn took care of many details I did not pretend to understand. Gail Cocker-Bogusz and Lisa Iarkowski helped minimize document processing-related headaches and contributed much to the layout of this book. Julie Bettis was a painfully thorough copy editor and went beyond the call of duty in the home stretch.

The boys at QuickStream Software (`www.quickstream.com`), Shawn Bertini and Eric Westerkamp, besides having a pretty slick 100% Pure Java web-based document management product, offered their native method programming experience gained from their forays into not-so-pure Java. Chapter 10 and Chapter 12 owe a lot to Shawn's skills. However, in spite of his brainpower, Chapter 12 would not have happened without the kind and timely help of Richard Brame.

Jeff Roloff and Emilio Millán at Central Data (`www.cd.com`) were generous enough to review Chapter 11 and allow me to distribute their *portio* package with the example code available with this book. Dr. Val Veirs of Colorado College gets credit for helping me understand that such a chapter would be useful.

Aaron Naumann of the Software Productivity Consortium (`www.software.org`) planted the seed which grew into Chapter 14. Aaron was also kind enough to review a draft of this chapter. Appendix E was subject to last minute reviews by Bill "Billy" Birnbaum of Sun Microsystems and Raman Tenneti and Sailesh Krishnamurthy of Netscape Communications. Besides having the most lyrical names of all the people who helped, these three agreed to a very short turnaround time for providing their input.

A handful of people expert at adjusting fonts, aligning text, numbering paragraphs and importing raster files into FrameMaker deserve a tip of the hat. Without Paul Kasper, Scott Hudson, Kris Trierweiler and Lou Ordorica (rhymes with Topeka) I would have been more frustrated at times than I already was.

Leah Kral, Richard Willems, and Craig Lindley helped in small but very measurable ways, all of which fall under the heading of helping me out in a fix. John Kelin's contribution, the compilation of Appendix A, falls under this heading also, but was far from small.

There were three people who were there when I (and my family) needed them: Patrick Hamilton, Susan Gordon and Rebecca Felts. Their contribution to this book is indirect, yet considerable. And then, there is my peripatetic brother, John. He dropped from the sky two days before my deadline and entertained my boys while I put the finishing touches on this beast.

I am also deeply grateful to a few people who, although not directly contributing to this book, have been important in the evolution of my thinking of myself as a writer. Barbara Huber, S.C., Michael Gardner, Richard Skorman and Kathryn Eastburn have supported my writing in other, non-technical venues over the past five years. There is also my father-in-law, Dr. R. B. (Bob) Sullivan, who, perhaps more than anyone else, convinced me I could arrange words on a page.

Finally, I owe a big thanks to Alan McClellan who, besides being the editor of the Essential Java series, is a friend and mentor. It is an immeasurable boost when a writer as talented as Alan is supportive of one's own writing. I am grateful for his invitation to be a part of this series.

Contents