The Elements of Java™ Style
SIGS Reference Library

1. Object Methodology Overview CD-ROM • Doug Rosenberg
2. Directory of Object Technology • edited by Dale J. Gaquiot
3. Dictionary of Object Technology: The Definitive Desk Reference • Donald G. Firesmith and Edward M. Eykholt
4. Next Generation Computing: Distributed Objects for Business • edited by Peter Finger, Dennis Read, and Jim Siskenleather
5. C++ Gems • edited by Stanley B. Lippman
6. OMT Insights: Perspectives on Modeling from the Journal of Object-Oriented Programming • James Rumbaugh
7. Best of Booch: Designing Strategies for Object Technology • Grady Booch (Edited by Ed Eykholt)
10. Java™ Gems: Jewels from Java™ Report • collected and introduced by Dwight Deugo, Ph.D
12. Advanced Object-Oriented Analysis and Design Using UML • James J. Odell
13. The Patterns Handbook: Techniques, Strategies, and Applications • edited by Linda Rising
15. The Elements of Java™ Style • Al Vermeulen, et al
16. More Java™ Gems • edited by Dwight Deugo, Ph.D
17. More C++ Gems • edited by Robert C. Martin

Additional Volumes in Preparation
The Elements of Java™ Style

Al Vermeulen
Scott W. Ambler
Greg Bumgardner
Eldon Metz
Trevor Misfeldt
Jim Shur
Patrick Thompson
The authors would like to thank our loved ones for enduring us while we toiled away on this book.
# Table of Contents

**Preface** ........................................... ix  
**Audience** ........................................... x  
**Acknowledgments** ................................. xi  

**Introduction** ......................................... xiii  

1. **General Principles** .............................. 1  
2. **Formatting Conventions** ....................... 5  
3. **Naming Conventions** ............................ 15  
   Package Names ....................................... 18  
   Type Names ......................................... 20  
      Class Names ..................................... 20  
      Interface Names ................................ 22  
   Method Names ...................................... 23  
   Variable Names .................................... 25  
      Field Names .................................... 27  
      Parameter Names ............................... 28  
   Constant Names .................................. 29
Preface

At Rogue Wave, we sell C++ and Java software components. We have always included source code with our products. Customers often browse through the code to get a feeling, not just for how it works, but for how to write good software. As a result, we have always felt pressure—maybe more pressure than most companies—to have good, consistent style throughout our source code.

As the company grew, making sure programmers were all following the same rules became difficult. To address this, our founder and first programmer, Tom Keffer, wrote 35 pages that explained how we write C++ code at Rogue Wave. We passed the document around and made sure new hires got a copy. It worked. When customers asked how we maintained consistency in our coding, we told them about Tom’s “C++ Design, Implementation, and Style Guide,” and sent them a copy. Word spread and we turned Tom’s document into a technical report. We sent out thousands of copies and received terrific positive feedback.

When Java came along, we decided we needed a document like the “C++ Guide.” A note went out to our internal javadev@roguewave.com mailing list soliciting rules for Java use that we should be using. The resulting list of rules became the first draft of the “Rogue Wave Java Style Guide.”

As the list of rules grew, the style guide began to look more and more like a real book. This time, we decided to publish our guide instead of simply issuing another Rogue Wave technical report. To our amazement, the folks at Cambridge University Press thought this was a great idea, and *The Elements of Java Style* was born.
One of the first reviewers of that original draft was Scott Ambler, current president of Ronin International (www.ronin-intl.com). Scott liked the idea of the book and suggested we check out the coding standards for Java he’d been distributing on the Web. We liked his standards a lot and decided we should work as a team. The result of combining Scott’s standards and the Rogue Wave style document is this book.

**Audience**

We wrote this book for anyone writing Java code, but especially for programmers who are writing Java as part of a team. For a team to be effective, everyone must be able to read and understand everyone else’s code. Having consistent style conventions is a good first step!

We assume you already know the basics of Java and object-oriented programming.
Acknowledgments

THIS BOOK was a team effort. The team extends far beyond the seven named authors. We'd like to thank those who reviewed and contributed to the original “Rogue Wave Java Style Guide” and the “Ambysoft Inc. Coding Standards for Java.” This includes Jeremy Smith, Tom Keffer, Wayne Gramlich, Pete Handsman, and Cris Perdue.

This book would certainly never have happened without the help and encouragement of the folks at Cambridge University Press. Our editor, Lothlorien Homet, hooked the Rogue Wave people up with Scott Ambler and made it all happen with her gentle, yet persistent, prodding. Thanks Lothlorien!
Introduction

**style.**

1b. the shadow-producing pin of a sundial.
2c. the custom or plan followed in spelling, capitalization, punctuation, and typographic arrangement and display.

—Webster’s New Collegiate Dictionary

The syntax of a programming language tells you what code it is possible to write—what the machine will understand. Style tells you what you ought to write—what the humans reading the code will understand. Code written with a consistent, simple style will be maintainable, robust, and contain fewer bugs. Code written with no regard to style will contain more bugs. It may simply be thrown away and rewritten rather than maintained.

Our two favorite style guides are classics: Strunk and White’s The Elements of Style and Kernighan and Plauger’s The Elements of Programming Style. These small books work because they are simple—a list of rules, each containing a brief explanation and examples of correct, and sometimes incorrect, use. We followed the same pattern in this book.

This simple treatment—a series of rules—enabled us to keep this book short and easy to understand. The idea is to provide a clear standard to follow, so programmers can spend their time on solving the problems of their customers, instead of worrying about naming conventions and formatting.