More Java Gems
SIGS Reference Library

1. Object Methodology Overview CD-ROM ♦ Doug Rosenberg
2. Directory of Object Technology ♦ edited by Dale J. Gaumer
4. Next Generation Computing: Distributed Objects for Business ♦ edited by Peter Fingar, Dennis Read, and Jim Stikeleather
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.
13. The Patterns Handbook: Techniques, Strategies, and Applications ♦ edited by Linda Rising

Additional Volumes in Preparation
CONTENTS

INTRODUCTION IX

GETTING STARTED WITH JAVA

WHEN IN ROME: A GUIDE TO THE JAVA PARADIGM • Richard Deadman 3

ENUMERATING TYPES WITH CLASS • Eric White 15

MIGRATING TO JAVA

HOW TO SUCCESSFULLY MIGRATE TO JAVA
• Martin Schellbauer and David Udin 23

KISSIN’ COUSINS: COMMUNICATING BETWEEN JAVA AND JAVASCRIPT • Steven W. Disbrow 31

TECHNIQUES 101

JAVA PERMUTATIONS AND COMBINATIONS • Steven J. Metsker 51

PREVENTION IS BETTER THAN A CURE • Mike Mannion and Roy Phillips 69

INTERNATIONAL TEXT IN JDK 1.2
• Mark Davis, Doug Felt, and John Raley 87
VI CONTENTS

MODELING AND PATTERNS

The Conceptual Model: What’s the Object? • Craig Larman 105

Refining Skills for Expressive Conceptual Models
  • Craig Larman 115

Distributed Observer Chains • Andrew Smith 121

Event Notifier: A Pattern for Event Notification
  • Suchitra Gupta, Jeffrey M. Hartkopf, and Suresh Ramaswamy 131

Analysis Patterns • Martin Fowler 155

JAVA IN A DISTRIBUTED WORLD

Mobile Agent System Architecture • Steven R. Farley 169

Deployment Strategies for Java Client Applications • Ron Repking 179

Locating CORBA Objects from Java • John O’Shea 191

THREADS

Effective Multithreading • Peter Bosch 205

Multithreaded Assignment Surprises
  • John Miller Crawford and Steve Ball 215

Multithreaded Exception Handling in Java
  • Joe De Russo III and Peter Haggar 223

Writing More Complex Synchronization in Java • Bill Lewis 241

USER INTERFACES

Constructing Modular User Interfaces in Java • John Hunt 255

JFC’s Swing, Part 1: Model/View/Controller • David Geary 265

Panel States: A User Interface Design Pattern • Roger Spall 281
## Contents

### SECURITY

**Using the JavaSoft Security Package** • *Steve Burnett*  
295

**Using the Java Cryptographic Extensions** • *Steve Burnett*  
305

**Java Application Server Security Using Capabilities** • *Greg Frasca*  
323

### TESTING

**Effective Test Strategies for Enterprise-Critical Applications** • *Kamesh Pemmaraju*  
339

**Putting JavaBeans to the Test** • *David Houlding*  
351

**Test-Infected: Programmers Love Writing Tests**  
• *Kent Beck and Erich Gamma*  
357

### PERFORMANCE

**Complex Java Applications: Breaking the Speed Limit**  
• *Allen Wirfs-Brock*  
379

**Are Distributed Objects Fast Enough?**  
• *Matjaz B. Juric, Ales Zivkovic, and Ivan Rozman*  
391

**Efficient Text Searching in Java** • *Laura Werner*  
405

**Enterprise Applets and Archive Functionality** • *John Cerwin*  
421

### REALITY CHECK

**Primitive Types Considered Harmful** • *Sherman R. Alpert*  
435

**Getting the Numbers Right, a Cautionary Tale** • *Ken Dickey*  
455

**Index**  
465
INTRODUCTION

TO JAVA REPORT—NOW AND BEYOND

THE JAVA REPORT published its first issue in March 1996. This was a real accomplishment when you consider that it was only on May 23, 1995, that John Gage, director of the Science Office for Sun Microsystems, announced Java to the world at SunWorld. Later in 1995, Sun released the initial Java Development Kit (JDK) and a Java enabled Web Browser called HotJava. The rest is history, as Java is now firmly entrenched in the computing industry. Many of us who saw demonstrations of Java in 1995 knew that it was something new, something different, and something not to be ignored. Those behind the Java Report knew this too, and we have been reporting on Java ever since.

In his first editorial for Java Report, the original Editor-In-Chief, David Fisco, wrote:

The Java community is becoming broader every day, encompassing CIOs, information technologists, market professionals, programmers, multimedia designers, educators, managers, and even hobbyists. … However, many CIOs, developers, and even software experts are having a hard time getting a handle on Java. Some have said that it’s just a neat way to run animations on the Web, others note that Java enables Web-based electronic transaction, and still others tout Java as the Holy Grail that will bring about the $500 PC and change the world of computing as we know it.

David’s comments are as relevant today as they were back in 1996. To quote the lyrics of Pearl Jam, “it’s evolution baby.” This year’s JavaOne conference
— Sun’s original Java conference — had 20,000 attendees. I don’t think anyone left JavaOne without being moved. The feeling started the moment you entered Halls A and B at the Moscone Center in San Francisco for the opening keynote. Imagine going to a technical conference and thinking that you were at a rock concert. With the huge conference attendance, the keynote was overflowing, the remaining people standing in lines that went for blocks waiting to get their conference materials or Palm V’s selling for $199 — Java virtual machine included. As the lights went down and the main characters took the stage, one felt like holding up a lighter.

The message at the conference was clear: Java has arrived, it’s real, it’s not boring, it is not going away anytime soon, and it’s a key ingredient for the enterprise, the desktop, and for small devices. These were more than just statements to take on faith; the proof was there for all to see. Endless companies demonstrated how they were using Java and how it was helping them to be successful in their specific lines of business.

So what’s the evolution? Part of it is the fact that Java grants developers endless possibilities. Here is a language that works everywhere and on everything, opening a new world of applications limited only by the imaginations of developers and entrepreneurs. Not only does Java help with the development of applications for the Networked Economy. It is being used to develop applications that connect us at a personal level never seen before, as we beam information and applications back and forth. After all, it’s great to do business with people and companies around the world, but it’s even better to see a face behind the machine.

The Java Report’s mission from the beginning has been to be your independent, objective source of information about Java. Its intent is to inform, enlighten, and entertain. Initially a bimonthly publication, Java Report has been a monthly publication since September 1997. Over the years, its page count has increased from 64 in the first issue to 80 in the latest one. This year, with the release of The Java 2 Platform (JDK 1.2), Java continues to grow in functionality and use. To keep pace, the Java Report is intending to do the same. Look for more news, reviews, interviews, columns, and feature articles as we reach the millennium. In one of my editorials, I asked, will anything derail the growth of Java? I believe now the answer is no. Sun is delivering on its Java promises and, provided people use it correctly, Java’s future is very bright. The Java Report will continue to help technical managers and developers use Java correctly by providing up-to-date information on topics and issues that they face now and in the near future.
Introduction

I officially took on the responsibility of Editor-In-Chief of Java Report in June 1996. If I knew then what I know now about the job, I think I still would have taken the position. For the most part, the job is fun, although in the magazine business deadlines are hard – something the software industry could learn from. Being Editor-In-Chief involves a number of tasks. I meet with companies or visit their homepages regularly to keep informed of what they are doing and what they plan to do next. There are product reviews and interviews to schedule, meetings with people who want to talk about the status of Java, and the magazine to edit. Then there is the task of getting developers to write for the magazine. Getting someone to say he will write an article is often easier than getting them to complete it. You would be surprised at how long it takes some people to write a 2000–3000 word article and the excuses for not doing so. I know everyone is busy. If you are not, you’re not working hard enough. However, the problems and the issues that developers are dealing with now are ones of great interest to many in the Java community. More than a million people are new to Java and to object-oriented programming. If you have discovered, developed, used, or solved something interesting relating to Java, write about it. The pattern community has learned the benefits of this. What will it take to convince you to do the same? Along with the above tasks, many are surprised to hear that I am also an assistant professor at Carleton University in Ottawa, Canada. You don’t think you can be the editor of a technical magazine without actually working with the technology, do you? For me this is a real benefit. Companies may be able to gloss over important technical details with some of the media. However, it does not take me long to get by the PR people and talk with the ones who are really doing the work: the developers.

At any one time, I am dealing with three different Java Report issues. There is the issue that just hit the streets where I am talking with people about its contents. There is the issue that is in production where I am checking its quality. Then, there is the next issue where I am looking for good articles that fit its theme. Thanks to your submissions, the number of quality articles available for publication is growing. I am always asking colleagues and leading members in the community to write for the Java Report. It may take a while to get them to agree, but I can be a very convincing person.

Java has grown up since I took over the helm of Java Report and so have the people that use it. To keep pace, the Java Report is written for and by colleagues in the Java community. Only you know the problems, solutions, details, and issues that are important. For example, you know why your Java
projects succeed or fail. You know the types of projects being developed. Moreover, only you can help to keep \textit{Java Report} continually focused on your needs. Aimed at knowledgeable Java users, feature articles explore different areas of the language, such as ways to use Java effectively, traps and pitfalls, novel uses, or arguments for and against particular language features. Articles containing descriptions of commercial products are normally excluded. In addition, in each issue an expert group of columnists explores in detail particular aspects of Java. Each issue provides book and product reviews, corporate profiles, one-on-one interviews with industry leaders and Java gurus, product announcements, and a survey of current trends in the industry.

This book is the second collection of articles from \textit{Java Report} and covers the last twenty-four issues, from July 1997 to June 1999. Selecting articles proved difficult again because the \textit{Java Report} had presented a wide range of interesting articles, including primer-style articles, enterprise articles, and those for Java power users. In addition, there were numerous great articles from columnists dealing with themes such as Java Means Business, Madison Avenue Java, ODMG, Effective Java, Durable APIs, Business Objects, Scott’s Solutions, Modeling and Java, 2D Graphics and Distributed Computing. My task was to select 34 articles. I found it analogous to being in a candy store with only fifty cents and wondering where to start. Excellence alone was not a sufficient criterion to help prune the articles available to my target number. I used the following criteria to base my final decision:

- **Developer as intended reader.** I decided not to include any piece whose primary interest was business or managerial related, with the possible exception of Martin Schedlbauer and David Udin’s article, “How to Successfully Migrate to Java.” I wanted this collection to offer insights into Java for the developer.

- **Relevant issues for today’s developer.** With the changes and additions made to Java over the last year, I decided not to include any piece that was dependent on releases of Java before 1.1. I wanted the topics addressed by the collection to match the concerns and issues of developers today.

- **Not readily available from other sources.** I also decided not to include introductory material easily found in other sources. The \textit{Java Report} has had introductory pieces on APIs and techniques. However, many Java books deal with these topics at great length. If your bookshelf is not already filled with these types of books, I am sure it soon will be.
Using the above criteria, my first pass through the back issues left me with a list of 80 articles – more than twice as many as I required. I was proud of the quality of the articles I had selected and of the Java Report for its content, realizing it was going to be difficult to get that number to 34. On my second pass, I arranged the articles into sections relevant for today’s developers and from that managed to get the number down to 50. Finally, I set a goal of no more than three articles per section and pruned the number of articles to 34. I could cut no more! I had not reached my goal, but I felt that to cut any more would take away from the overall flow of the collection.

The collection has eleven sections: “Getting Started with Java,” “Migrating to Java,” “Techniques 101, "Modeling and Patterns,” “Java in a Distributed World,” “Threads,” “User Interfaces,” “Security,” “Testing,” “Performance” and “Reality Check.”

The collection begins with the section “Getting Started with Java,” and it is only fitting that Richard Deadman’s article on “A Guide to the Java Paradigm” leads the way. For many, Java was not their first object-oriented programming language. It was either C++ or Smalltalk. It is no secret that Java has a syntax similar to C++ and many classes similar to those in Smalltalk. However, that’s where the similarities stop. As Richard puts it,

Moving to a new language, even if it’s one object-oriented language to another, involves some paradigm shifts in how you think about structuring and solving problems.

For example, you can’t escape a discussion on pointers or cleaning up memory when working with C++. You can to a greater extent in Java, although even with it you can hold onto objects forever. Richard’s article describes several good examples of what he calls “conceptual confusion” that you can avoid in order to write well-architected Java programs.

Known in the other languages, such as C and C++ as enumeration constants, the enum keyword permitted programmers to define new integer based types. However, there is no such keyword in Java. This does not imply that you can’t build a similar mechanism into Java. The answer is always the same: build a class. Eric White’s article on “Enumerating Types with Class” completes the section describing how to add Java type enumerations as object-oriented constructs to abstract the unsafe type enumerations of legacy procedural code.

Java has rapidly emerged for many as the language of choice for the enterprise, the desktop and small devices. To take advantage of Java’s many capabilities, companies are faced with the task of migrating people that have
never done any object-oriented programming or design. The collection’s second section on “Migrating to Java” examines two different approaches to getting people to use Java.

The first article, by Martin Schidlauer and David Udin, on “How to Successfully Migrate to Java” examines three different migration patterns addressing the unique training requirements of different groups of people, such as mainframe developers, system and application developers, and those with VisualBasic and C experience.

Rather than have developers go directly to using Java, another approach is to have them start with JavaScript. If you are already familiar with Web browsers and HTML, JavaScript is a natural step in your evolution to use Java. However, Java and JavaScript are not completely disjoint from one another, and as Steven Disbrow shows in his article, “Kissin’ Cousins: Communicating Between Java and JavaScript,” developers have a choice to use either one or both.

The next section in the collection, called “Techniques 101” contains three articles on techniques for building a developer’s toolbox, designing by contract and internationalization. At the heart of any application a company has successfully built using object-oriented programming is a set of classes that developers trust and are willing to use over and over again. I am a firm believer of developing classes not only for today, but also for tomorrow. Since browsing source code is an essential task for an object-oriented developer, it should not be surprising to a developer that their classes will be scrutinized. Therefore, when you get the chance to build a class, do a good job. This is exactly what Steven Metsker did in his article on “Java Permutations and Combinations.” He provides us with two excellent implementations of permutations and combinations and shows how to put them to good use by solving a logic puzzle using a generate-and-test technique.

When one thinks of Java interfaces, one thinks of a promise by a class to provide a body for the methods declared in the interface. However, this is only a syntactic construction. As Mike Mannion and Roy Phillips say in their article called “Prevention is Better than a Cure,” “Design by contract is a specification that allows the designer or programmer to specify the semantics of a class’s interface.”

They describe how the addition of assertions to Java and designing by contract bring tremendous benefits, particularly for developers in the component market, by increasing the overall precision in handling pre-, post- and invariant conditions in your software.

The section’s final article addresses the technique of developing software
that can be deployed easily across a variety of world markets, with different languages and writing systems. The Network Economy is here, and if you do not think of your Java applications as being used internationally, you should. Mark Davis, Doug Felt, and John Raley discuss several problems posed by the different writing systems and describe how you can use the new TextLayout class in Java to handle these problems easily and efficiently.

The next section is called “Modeling and Patterns,” and it is only appropriate that Craig Larman, Java Report’s resident modeling columnist, leads off the section. Any type of modeling activity, unless it can be of real value, is arguably not productive. As Craig says, “Modeling and diagramming should practically aid the development of better software.”

In the section’s first two articles, Craig describes how a conceptual model can help you succinctly visualize the vocabulary of a new or large problem space and can help you illustrate the important relationships between the key ideas and information it contains.

Two of the key goals of modeling are to help save time and to develop better designs. The use of software patterns is another excellent way to improve your designs. By applying them, you build into your software well-thought-out, proven designs that developers have used successfully in the past. The section concludes with two software patterns. The first by Andrew Smith, called “Distributed Observer Chains,” extends the Model-View pattern to make distributed objects observable. The second pattern by Sachitra Gupta et al., called “Event Notifier,” describes how to enable components to react to the occurrence of particular events in other components without knowledge of one another, while still allowing the dynamic participation of components and dynamic introduction of new events.

To develop software for a distributed environment forces one to consider several different architectures and components. You can make a great deal of money these days if you can present companies with end-to-end solutions for their applications. With so many options, the problem is to determine what items should be put together and when it is appropriate to do so.

In the section called “Java in a Distributed World,” we give you advice on three such options. The section’s first article by Steven Farley, called “Mobile Agent System Architectures,” provides a new way of thinking about distributed computing: a mobile agent. Rather than sending messages between distributed objects, a mobile agent in a mobile agent system is an object that can autonomously move and execute where and when required.

The section’s second article by Ron Repking, called “Deployment Strategies for Java Client Applications,” compares and contrasts options available
for client-side deployment of Java applications. The three strategies discussed
include Web deployment, stand-alone application deployment, and using
broadcast and push technology.

Not forgetting the server-side of distributed computing, the section’s last
article by John O’Shea from Iona, called “Locating CORBA Objects from
Java,” describes how your client applications and applets should bootstrap
into the CORBA system and how CORBA servers should distribute CORBA
object references so that clients can easily and efficiently find them.

The ability to do multithreaded programming in Java is a feature that most de-
developers use to build robust applications. Conceptually, it is easy to under-
tand Java threads and how to use them to build multithreaded applications. However,
effectively using threads is often difficult. In the next section, called “Threads” we
have four articles chosen from Java Report to help you use threads better.

The first article by Peter Bosch, called “Effective Multithreading,” de-
scribes a framework to thread-separate servers from the clients that call them.
Thread separation is useful for many reasons, including allowing the client
to do other things while waiting on a response from the server and limiting
the number of threads in use. As Steve Ball and John Crawford say in the sec-
tion’s second article, “A volatile brew is formed by mixing assignment and
threads.”

In their article, called “Multithreaded Assignment Surprises,” the two ex-
amine the perils and surprises lurking within the most innocent-looking state-
ments that use assignment. And the article called “Multithreaded Exception
Handling in Java,” Joe De Russo III and Peter Haggard describe how to use
threads and exceptions together. As they say, “How effective is writing a mul-
tithreaded Java program if it is incapable of properly handling exceptions
occurring on secondary threads?”

Bill Lewis completes the discussion on threads with his article called “Writ-
ing More Complex Synchronization in Java.” In his article, he covers syn-
chronization situations that are not obvious and shows you how to extend
the synchronization primitives.

Good software is usually the result of good architecture. Therefore, it
should not be difficult to understand that user interfaces should be easy to
develop and maintain if they too are constructed with this principle in mind.
However, for those early adopters of Java, building modular object-oriented
user interfaces with the AWT was tough. With the addition of the delegation
event model in JDK 1.1, the picture changed, providing the potential of sep-
arating the views for the control parts of the user interface. This separation,
know and the model-view-controller (MVC) architecture was well known
in the Smalltalk community. It just took Java a little longer to catch on.

In the next section, called “User Interfaces,” we begin with two articles on MVC. The first by John Hunt, called “Constructing Modular User Interfaces in Java,” discusses why MVC is so important to Java and how to apply it to build user interfaces that are robust, principled, and reusable. In David Geary’s article, “JFC’s Swing,” he examines how MVC is used by Swing, Java’s collection of lightweight components.

MVC is not the only architecture you can use for constructing user interfaces. The section’s last article, by Roger Spall, “Panel States,” describes an efficient architecture for presenting a group of user interface panels in a specific sequence to a user.

The next section examines the issue of security in Java. I still believe security is one of the most overlooked areas in application development. When I ask developers if they believe security is important for their software, they all answer “yes.” When asked if they are adding it to their application, most say “no.” With more education on security, I believe this situation will change.

The section’s first two articles are by Steve Burnett, a crypto engineer at RSA Data Security, Inc.. In his first article, called “Using the JavaSoft Security Package,” he describes how to sign or verify data using Digital Signal Algorithm (DSA) with Java’s security package. His example will help you to get a better understanding of Java’s Cryptography architecture.

In his second article, called “Using the Java Cryptographic Extensions,” he demonstrates how to encrypt and decrypt data using DES and how to create an RSA digital envelop with the extensions package.

Signing, encryption, and decryption are security features that are provided by Sun. Developers can also add their own security mechanisms. In the section’s final article by Greg Frascadore, called “Java Application Server Security Using Capabilities,” he describes an approach to guarding sensitive server data. Often accomplished by access lists, his capability approach is less cluttered, more efficient, and more fail-safe.

There is no doubt that the first product to market in a given area captures a good slice of the market share. However, the only way it can keep or gain in its share of the market is if it has quality. There is no better way to achieve quality in a product than by making sure it is well tested BEFORE it hits the market. I don’t know how many times I have told people that I would gladly use an application that has fewer features and works than one with hundreds of features that doesn’t.

In the “Testing” section, we have three articles that examine different aspects of testing. The first article by Kamesh Pemmaraju takes a holistic ap-
**Introduction**

In his “Effective Test Strategies for Enterprise-Critical Applications” article, he presents a case-study of how testing was done on a Java-based financial application.

At the other side of the testing scale, we have David Houlding’s article called “Putting Java Beans to the Test.” He illustrates strategies for testing Java Beans, describing how to ensure that the beans you develop are verifiable through testing.

The final article in the section, by Kent Beck and Erich Gamma, describes how to structure your test cases and run them in a framework called JUnit. As they put it, “Code a little, test a little, code a little, . . . .” With their framework you can put these words to the test.

The one capability that Java must exceed in everyone’s mind to become the premier language for development is performance. This is not to say that Java does not perform well. In fact, in most applications I have seen, Java performs fine. I am just tired of hearing that it is not as fast as C++. The point is that Java only has to be fast enough to run your applications and no faster. An analogy to this is that we don’t drive cars that can go 200 m.p.h.; we don’t need them to go that fast.

In the section titled “Performance,” the articles look at different aspects of performance that Java developers have control over. It is up to Sun and the other virtual machine (VM) companies to make VMs execute Java code at the speed of light. However, there are many techniques you can use to make your code go faster. Allen Wirfs-Brock’s article, called “Breaking the Speed Limits,” describes ways to change your code to attack VM inefficiencies. Laura Werner’s article, called “Efficient Text Searching in Java,” shows you how to use Java’s Collator class to help with searching. Finally, John Cerwin’s article, called “Enterprise Applets and Archive Functionality,” describes how to use Java’s Zip classes to improve performance over the network. By using the right class for the right job and in the right way, developers can get the performance they need today.

There does not seem to be a feature or API that Java doesn’t offer to developers. However, for all of the advantages and benefits of Java, it is still up to the developer to make sure that he uses Java correctly and when appropriate. In the last section, called “Reality Check,” we look at two such areas: primitive types and math support.

Sherman Alpert describes the advantages and disadvantages of using types such as int, boolean, float, long, and short. Included with the language due to its similarity with C and C++, and for performance and verification rea-
Introduction xix

sons, primitive types break with Java’s object-oriented nature. Sherman’s discussion will help you decide whether they are worth using.

Related to primitive types is the issue of how accurate numerical computations are in Java. In the article, called “Getting the Numbers Right, A Cautionary Tale,” Ken Dickey reflects on how math is done in Java, using a specific example of interval arithmetic. In sharing their experiences, both writers will help you to get the right answers, using the best classes for the job.

After reflecting on my task of selecting and introducing the articles in this collection, I have the same regret as I did with the previous collection. I wish that I were not limited as much by space. The Java Report has been fortunate in having an excellent group of writers and columnists from its conception. However, no one person represents its voice; it is their collective voices that make the magazine a success.

Our goal for the Java Report is to help you do it right. We want to keep you informed of the latest developments and products in the industry. We want to help you learn and use Java’s new APIs. We want to provide you with tips, techniques, and patterns for Java. We want to help you understand Java and object-oriented programming issues, provide answers to difficult questions, and provide you with solutions to problems you will encounter. We want to help you and your projects succeed!

To help us achieve our goal, we need to hear from you. We need to know why your Java projects succeed or fail. We need to know the types of projects you are developing. We want your help keeping Java Report continually focused on your needs. Java Report is committed to you in being your premier source of information on Java. In return, I want to hear your views and any contributions you would like to make.

Dwight Deugo, editor-in-chief
Java Report