Novus ordo seclorum, cum grano salis
— A new order of ages, with a grain of salt

Network computing, Java, the Internet, client/server, distributed systems, *ad infinitum*—technologies and techniques that promise a new, cheaper, more efficient and easier way of computing, that we must approach with caution and understanding.

Java is a modern programming language that supports development and deployment of network computing applications. There are literally hundreds of books on Java, some specialized, many generic. Almost all neglect to provide a sound basis for understanding the fundamentals of network computing and the basic reasoning behind why you should use Java for developing and deploying your applications. The focus *ad nauseum* has been on language syntax, which is fine in
isolation. Often missing are the nontechnical issues, the helicopter view of the network computing domain that makes it clear where to position Java, how to integrate it with existing technologies, and what you need to consider to allow for a rapidly changing future.

When a “normal” monolithic application runs on a computer, all the components of the application work within the same memory space, and the computer manages the interplay of program modules in a well-understood and time-honored fashion.

Writing an application for a network computing environment involves the factoring of the application’s functionality into components and with the deployment of these components on devices that are most appropriate for their requirements. An application deployed across multiple, (perhaps geographically) separated systems is referred to as a *distributed* application.

Distribution brings with it a number of questions that must be answered before the system can be fully implemented: how can the components of the application communicate, interact with other programs, and make use of external facilities? What are the new “rules” for developing applications for a potentially unlimited number and class of users? How can the new failure modes introduced by the distribution process be handled?

Java is still very young, but under the impetus of the Internet and the World Wide Web, it is maturing rapidly. No programming language is perfect, and in this regard, Java is certainly not unique! Java has a number of very good points, and some that are currently the subject of some debate. As a modern language designed from the ground-up to be an object-oriented language for network computing, it does represent a powerful weapon in the armory of the distributed systems warrior.

The Network Computer (NC) is another arrow in the quiver for all those who are charged with building complex, enterprise-wide systems. It offers a powerful, lower-cost alternative to today’s networked desktop hardware. Even more importantly, the network computer in its various guises represents choice and freedom: system designers are now free to choose their hardware and software systems according to the idea of *fitness for use*, rather than attempting to mould and shoe-horn inappropriate devices into their designs.

Predicting the future is always a risky proposition, but one thing seems clear: both Java and the idea of network computing are now here to stay. We hope this book will help you find your way through the battlefield.
A Thin Client Application—The Lunar Medical Center Story

To illustrate the techniques and technologies discussed in this book, gaze into the proverbial crystal ball, and imagine a future moonbase medical center—The Lunar Medical Center (LMC)—which needs to provide services to the lunar community.

The LMC system designers have chosen to adopt a network computing strategy and need to implement numerous applications—administrative, device-monitoring, public access, and so forth—using a network computing-centric programming language. Having conducted a comprehensive study of the technology options, Java has emerged as the only serious contender for the job.

In the LMC’s medical departments, access terminals are expected to support work 656.7 hours a day, 365 days a year and allow for easy replacement in case of failure. The Network Computer is the solution chosen by the Lunar Medical Center.

Because many people have to share information, the data must be stored on a system that can be accessed by multiple departments. Java provides several communication and access solutions, such as Java Data Base Connectivity (JDBC), Sockets, and Remote Method Invocation (RMI). It is up to the LMC designers to investigate the pros and cons of each method and to determine the most appropriate solution for their needs. To achieve this goal, they have chosen to prototype two aspects of the LMC system—the patient check-in facility and the patient record view/edit facility—using the various options available to them.


About the Book

This book is about developing thin-client Java applications for a network computing environment, such as your local network, whole enterprise, or even the Internet. Thin-client is a technique that, among other things, attempts to minimize the resource requirements of the application at the desktop. Thin-client computing aims to help you to build a system that doesn’t require a system configuration that NASA would be proud of!

This book’s intention is to furnish Java knowledge with a foundation in network computing. In the chapters that follow, you will gain a clear understanding of network computing, the features and issues related to distributed systems, the role of Java as one of the key technologies in this domain, and developing distributed client/server applications with Java. The focus is on architecting thin-client 100 percent pure Java applications by example, both generically and through a packaged Java development environment: IBM’s VisualAge for Java.

Several different communication mechanisms for implementing a distributed Java application are considered: Hypertext Transfer Protocol (HTTP), Remote Method Invocation (RMI), Sockets, and Java Data Base Connectivity (JDBC). The new class of Java-centric network computing desktop devices—the Network Computer (NC), in particular—is discussed, along with the nature of these devices, and the issues associated with deployment of Java applications for such technology. IBM’s Network Station is presented as an NC case study.

This book, in combination with some Java language reference documentation such as that provided in the Java Development Toolkit (JDK) electronic document set, will give you the knowledge required to implement thin-client, distributed Java applications and to be able to put what you’ve created in context.

Structure of the Book

Chapter 1, Introduction

This chapter is the introduction and highlights the remaining chapters in this book.

Chapter 2, Network Computing and Network Computers

This chapter positions Java and Network Computers (NCs) within the broader framework of network computing, by discussing what network computing is, its implications, benefits and problems, along with an overview of the Network Computer Technical Standard, NC products, variations, and usage.
Chapter 3, Distributed Applications
This chapter looks at some of the various issues associated with the development of systems based on the network computing paradigm. It also examines the features and facilities of Java that make it such an excellent platform for the development of distributed systems.

Chapter 4, HTML-Based Clients
This chapter investigates the implementation of simple client applications using HTML and JavaScript. This facilitates using browsers as the data entry interface, as they are readily and inexpensively available and deployable.

Chapter 5, Java-Based Clients
This chapter has three aims. The first is to introduce Java and take a brief look at its features and facilities.

A second aim is to introduce the Java-based LMC patient record retrieval and update application that is used as the vehicle for discussion in the remainder of the book.

The final aim of the chapter is to provide a short overview of some Java facilities supporting enterprise-wide network computing—the “ecology” of Java: its features, properties and related technologies.

Chapter 6, Design Decisions
This chapter looks at the design of the object model underlying many of the applications and also examines how the LMC’s network designers investigated the various network architectures and technologies to see which were most appropriate for their environment.

Chapter 7, Java Servlets and HTTP Communication
Chapter 4 examined the implementation of very thin clients using HTML and JavaScript. Chapter 5 examined a Java Applet implementation of the same client application. This chapter examines the corresponding server application for the clients, which takes the syntactically valid data from the client, stores it to our chosen storage mechanism (either the filing system or a database), and provides some feedback to the browser.

Chapter 8, Java Servers Using JDBC
This chapter introduces Java Database Connectivity (JDBC), the Java API for standardized SQL-based database access. JDBC provides a uniform interface to a wide range of database systems and a common foundation on which higher-level tools
and interfaces can be built. Java-based middleware services that utilize the powerful data storage and retrieval facilities of database systems can be developed for our distributed applications.

**Chapter 9, Java Servers and Socket Communication**

This chapter deals with methods available in Java to slice monolithic applications into functional units which communicate through a network, thus making it a distributed application. The chapter introduces the Internet Protocol (IP) and its API using sockets. Sockets provide the functionality to transport data blocks or data streams over the network and thus represent a very elementary service for distributed computing.

**Chapter 10, Java Servers and RMI Communication**

This chapter introduces Remote Method Invocation (RMI). As its name suggests, this technique offers a way to invoke Java methods remotely. Using RMI allows development of easily distributed applications that can execute and communicate on multiple systems on the network.

**Chapter 11, Client Diets**

This chapter examines techniques and issues that need to be considered in order to produce optimized code—in terms of both speed and size—to maximize the “thinness” of Java solutions and to ensure that they make best use of the available resources.

**Chapter 12, Tasty Additions**

This chapter introduces Java programming for accessing smart cards and the serial port on the Network Computer.

**Chapter 13, NC Deployment: Using IBM Network Stations**

This chapter analyzes the IBM Network Station as an NC solution in the Network Computing world and investigates how it should be configured for commercial use. Performance tips and techniques are also provided.

**Chapter 14, Java in the IBM Network Station**

This chapter discusses how to successfully run Java in an NC, using the IBM Network Station as an example, along with some tips and techniques for avoiding problems.

**Appendix A, Brief HTML Reference**

This appendix provides an overview of some of the more important HTML tags used in the book.
Appendix B is a useful introduction to VisualAge for Java. It covers the VisualAge family, an overview of VisualAge Java, the Integrated Development Environment (IDE) and the Enterprise Access Builders (EAB). The chapter discusses various processes and windows that are used in the development of windows and applications using VisualAge for Java. A self study example is furnished as an exercise.

Who Should Read This Book

This book should prove an excellent resource for many information technology professionals, including programmers, architects, consultants, system administrators, and others. It is meant for those wanting a solid grounding in Java for network computing applications.

Anyone developing software for the Internet and the World Wide Web, for their enterprise networks (or intranets), or for any networking situation will find the material timely and relevant.

Although the discussion is centered on Java and the network computer, the fundamental concepts are applicable to any modern distributed system programming language and associated technology. The book collects a wide body of knowledge and presents it in a cohesive and realistic manner.

Assumptions

This book assumes a basic knowledge of the Java language, HTML, object orientation, and window-based GUI concepts.

Although the book provides a case study in Java development using an integrated development package, any other package, or indeed Version 1.1 of the “vanilla” Java Development Toolkit, may be utilized. The book does not provide a detailed syntax of commands, JDK classes and methods; so the reader will find a Java reference very useful. Many reference books exist, but the online documentation is perfectly adequate.

How to Use This Book

This book is not intended to be read in a linear fashion. It is anticipated that the reader will find certain chapters more relevant to their work than others and thus may want to “dip into” the book in their own order.

Readers are recommended to read this chapter, and especially “A Thin Client Application—The Lunar Medical Center Story” on page 3. This will provide the background that ties many of the chapters together.

Certain groupings of chapters may suggest themselves, however.
Readers new to network computing should first read Chapter 2, to establish a “feel” for the area. “Client-Server Computing with Java” on page 61 also provides a useful starting point.

Readers charged with implementing Java-based, network-aware systems should read Chapter 3 through Chapter 10. Chapter 6 also covers the design of applications for a network-computing-based enterprise.

The details of programming for the IBM Network Station are examined in Chapter 2, and Chapter 12 to Chapter 14. Chapter 9 also contains a relevant discussion. Chapter 14 looks at the new Lotus eSuite Java-based software suite for the Network Station. Programmers of the Network Station should also examine Chapter 11, which looks at how to optimize code to achieve peak performance.