

# JAVA™ DEVELOPER'S JOURNAL

JavaDevelopersJournal.com

Volume: 2 Issue: 12

## What is a Java Application Server?

Scott Dietzen pg. 5

## Making Enterprise

Java a Reality

Tina Lorentz pg. 7

## Under the Sun

JavaBean

Component Reuse

pg.46

## The Grind

The Brass Ring

by Joe S. Valley pg.90

## JDJ Forum Update

Welcome to the

JDJ Forum

by Ashok Ramachandran pg.86

## Java & Business

Wrangling Big Iron

by Eric Lehrfeld pg.32

## Product Reviews

InstallShield Express 2

by Jason Cohen pg.48

CodeBase 6

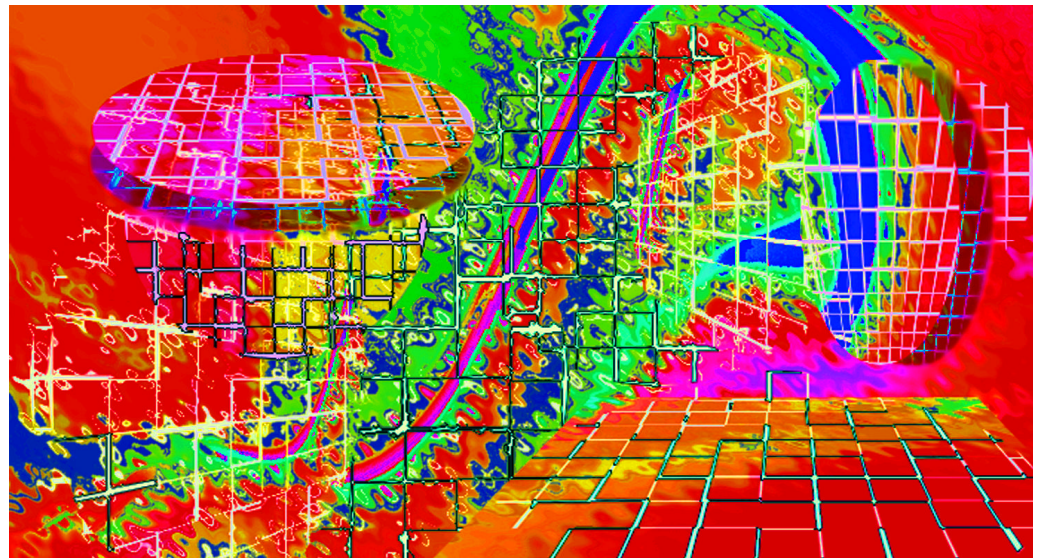
by David Jung pg.58

VisiBroker 3.0

by Khanderao Kand pg.72

## Java News

pg.88



## Developing Collaborative Games Using Active Objects

Simplifying the control flow

Larry Chen

& Todd Busby 8

## JDJ Feature: Java Security: Beyond Code Safety

Practical uses of the cryptographic interfaces of JDK 1.1

Jahan Moreh

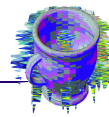
22

## JDJ Feature: Writing Unix Filters in Java

Using Java in place of the more traditional languages

Kenneth Kranz

36



## From the Ground Up: Polymorphism

An object's ability to identify with a parent class

John Tabbone

16



## Implementing Assertions in Java

A valuable mechanism in larger, critical applications

John Hunt & Fred Long

52

## CORBACorner: CORBA Beans

A distributed component model with cross language interoperability

Jeff Nelson

60



## Visual Café: POP goes the Server

Basic functionality needed to implement a POP3 client

Alan Williamson

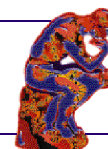
64

## What's All the Fuss About?

What do Java's components really deliver?

Clive Boustred

76



# Web Logic Full Page Ad

# Schlumberger Full Page Ad

# Bristol Full Page Ad



Scott Dietzen

## EDITORIAL ADVISORY BOARD

Ted Coombs, Bill Dunlap, Allan Hess,  
Arthur van Hoff, Brian Maso, Miko Matsumura,  
Kim Polese, Richard Soley, David Spenhoff

Art Director: Jim Morgan  
Executive Editor: Scott Davison  
Managing Editor: Gail S. Schultz  
Editorial Assistant: Christy Wrightington  
Copy Editor: Alix Lowenthal  
Technical Editor: Ed Zebrowski  
Visual J++ Editor: John Fronckowiak  
Visual Café Pro Editor: Alan Williamson  
Product Review Editor: Jim Mathis  
Games & Graphics Editor: Eric Ries  
Tips & Techniques Editor: Brian Maso  
Java Security Editor: Jay Heiser

## WRITERS IN THIS ISSUE

Clive Boustred, Todd Busby, Laurence Cable, Larry Chen,  
Jason Cohen, Scott Dietzen, David Jung, Kenneth Kranz,  
Eric Lehrfeld, Jahan Moreh, Tina Lorentz, Jeff Nelson,  
Eduardo Pelegri-Llopert, Ashok Ramachandran, Steven  
Schwell, John Tabbone, Joe S. Valley, Alan Williamson

## SUBSCRIPTIONS

For subscriptions and requests for bulk orders,  
please send your letters to Subscription Department

Subscription Hotline: 800 513-7111

Cover Price: \$4.95/issue.

Domestic: \$49/yr. (12 issues) Canada/Mexico: \$69/yr.  
Overseas: Basic subscription price plus air-mail postage  
(U.S. Banks or Money Orders). Back Issues: \$12 each

Publisher, President and CEO: Fuat A. Kircaali  
Vice President, Production: Jim Morgan  
Vice President, Marketing: Carmen Gonzalez  
Advertising Manager: Claudia Jung  
Advertising Assistant: Erin O'Gorman  
Marketing Director: Larry Hoffer  
Accounting: Jennifer Patterson  
Senior Designer: Robin Groves  
Web Master: Robert Diamond  
Web Designer: Corey Low  
Customer Service: Patricia Mandaro  
Rae Miranda  
Sian O'Gorman

## EDITORIAL OFFICES

SYS-CON Publications, Inc.  
39 E. Central Ave., Pearl River, NY 10965  
Telephone: 914 735-1900 Fax: 914 735-3922  
Subscribe@SYS-CON.com

JAVA DEVELOPER'S JOURNAL (ISSN#1087-6944) is  
published monthly (12 times a year) for \$49.00 by SYS-CON  
Publications, Inc., 39 E. Central Ave., Pearl River, NY 10965-2306.  
Application to mail at Periodicals Postage rates is pending at  
Pearl River, NY 10965 and additional mailing offices.

POSTMASTER: Send address changes to:

JAVA DEVELOPER'S JOURNAL, SYS-CON Publications, Inc.,  
39 E. Central Ave., Pearl River, NY 10965-2306.

## © COPYRIGHT

Copyright © 1997 by SYS-CON Publications, Inc. All rights reserved. No part of this  
publication may be reproduced or transmitted in any form or by any means, electronic  
or mechanical, including photocopy or any information storage and retrieval system,  
without written permission. For promotional reprints, contact reprint coordinator.  
SYS-CON Publications, Inc. reserves the right to revise, republish and authorize its  
readers to use the articles submitted for publication.

ISSN # 1087-6944

DISTRIBUTED in USA by

**International Periodical Distributors**

674 Via De La Valle, Suite 204, Solana Beach, CA92075 619 481-5928

BPA Membership Applied for August, 1996

Java and Java-based marks are trademarks or registered trademarks of  
Sun Microsystems, Inc. in the United States and other countries.  
SYS-CON Publications, Inc. is independent of Sun Microsystems, Inc.



# What Is a Java Application Server?

Application servers are not new. Many information systems, from mainframe transaction processing environments like CICS to the stored procedures of a DBMS, provide for the server-side execution of business processes. Running business logic on a server can improve security, managability, performance and reusability. With the explosive growth of Intranets and the Internet, developers need a rich, flexible server to host their business applications—a server that complements the content from databases and Web servers.

As a platform, Java raises the bar for application servers. A Java application server carries the benefits of a robust, scalable application server with the expressive power and dynamism of the Java platform. Java's byte-code-based Virtual Machine allows objects, including both code and data, to be exchanged among heterogeneous systems. This means that rather than making an RPC call or sending a message composed of ints and structs to a server, a stock trading application can send a high-level business object like Order to a remote business component like Trader. Moreover, Java's dynamism allows applications to be partitioned and repartitioned at runtime by relocating a service component from one machine to another.

When it comes to deciding how to partition an application between the client and server(s), developers need flexible tools that best meet the needs of the problem at hand. For example, when performing an ad hoc query, you need database access directly from the desktop. But in a "thin-client" application, you want to hide the DBMS operations behind a set of reusable server-side business processes. Certain applications will require synchronous request-response between components (JavaBeans) or objects (RMI), while others are better organized using asynchronous communications like event publication and subscription, or point-to-point messaging. Just as developers use different kinds of collections for different kinds of data (e.g., Hashtable, Vector and Set), they also need a set of off-the-shelf tools (Remote Method Invocation, Events, Distributed JavaBeans) for tackling application communications.

Besides code mobility and partitioning, developers want to use Java-blessed programming models: database access via JDBC, name and directory services via JNDI, distributed objects via RMI and Enterprise JavaBeans, Web server plug-ins via Servlets, and event management and messaging via JMS. Merely wrapping a proprietary C/C++ API with Java won't cut it. Nor will inventing yet another set of non-portable proprietary APIs. Developers want to

use Java industry-standard APIs so that the skills they learn on one project will be transferable to other projects and the code they write will run wherever there's a JVM.

Given that a developer should be able to assemble an application using best-of-breed products, a Java application server must be compatible with a diverse range of complementary technologies. It should work in harmony with leading Web servers and database management systems, and across heterogeneous Java platforms. The server should also work with major Integrated Development Environments (IDEs); it should not require the adoption of an IDE that is proprietary to the application server.

Standards and portability don't stop at the server. GUI building technology used for multi-tier applications should use standard widgets and JavaBeans – not yet another proprietary set of widgets. Any automatically generated Java code should be accessible for further customization so that you can tweak it, should you choose to, and not worry about getting out of sync with the tool.

Finally, a Java application server should provide an integrated management environment that offers a comprehensive view of the application and the server. Transaction semantics must be built in to ensure data integrity even across distributed components, and the Java application server must address network security using SSL and access control lists. The server should also allow access to the broadest number of clients through standard protocols like HTTP and IIOP. And in the end, a Java application server should be written entirely in Java so that it is portable, it can be easily embedded within value-added applications and so that new Java-native capabilities can be delivered rapidly to market.

This is a tall order. Java has been out of beta for about two years and many of the products and applications written in it have been servlet and applet plug-ins to non-Java systems rather than full-blown production applications. The time for the all-Java application server is now, and developers should demand nothing less in the server than the Java API standards, stability and robustness they require in the client. ●

## About the Author

Scott Dietzen is Vice President of Marketing at WebLogic, Inc., developers of Tengah, the industry's first commercial-grade Java application server. Scott formerly was the Principal Technologist of Transarc Corporation. He holds a Ph.D. and M.S. in Computer Science and a B.S. degree in Applied Mathematics from Carnegie Mellon University. Scott can be reached at dietzen@weblogic.com

# Suntest Full Page Ad

CALL FOR SUBSCRIPTIONS  
**1 800 513-7111**International Subscriptions  
& Customer Service Inquiries  
914 735-1900  
or by fax: 914 735-3922E-Mail: [Subscribe@SYS-CON.com](mailto:Subscribe@SYS-CON.com)  
<http://www.SYS-CON.com>MAIL All Subscription Orders or  
Customer Service Inquiries to**DEVELOPER'S  
JAVA JOURNAL**Java Developer's Journal  
<http://www.javaDevelopersJournal.com>**DEVELOPER'S  
VRML JOURNAL****National JAVA  
LEARNING CENTER**

National Java Learning Center, Inc.

**DEVELOPER'S  
JAVA JOURNAL  
1998 JAVA  
Buyer's Guide**

JDJ Buyer's Guide

<http://www.sys-con.com/inetbg/index.html>**WEB-PRO  
DEVELOPER'S SUPPLEMENT**

Web-Pro Developer's Supplement

SYS-CON Publications, Inc.  
39 E. Central Ave.  
Pearl River, NY 10965 - USAEDITORIAL OFFICES  
Phone: 914 735-1900  
Fax: 914 735-3922ADVERTISING & SALES OFFICE  
Phone: 914 735-0300  
Fax: 914 735-7302CUSTOMER SERVICE  
Phone: 914 735-1900  
Fax: 914 735-3922DESIGN & PRODUCTION  
Phone: 914 735-7300  
Fax: 914 735-6547DISTRIBUTED in the USA by  
International Periodical Distributors  
674 Via De La Valle, Suite: 204  
Solana Beach, CA 92075  
Phone: 619 481-5928Worldwide Distribution by  
Curtis Circulation Company  
739 River Road,  
New Milford NJ 07646-3048  
Phone: 201 634-7400

Tina Lorentz

# Making Enterprise Java a Reality

We've all read about the Internet's "endless potential" for redefining the way businesses operate and computers are built. The Internet's astonishing growth is a testament to its ability to live up to at least some of this hype. Yet most corporate Web sites consist strictly of marketing brochures and other static text and pictures.

To truly capitalize on the business potential the Web offers, organizations need to combine the Web's universal access and deployment with their own mission-critical business processes. By making transactions (such as travel planning, stock trading and package shipping) available on-line, companies can achieve benefits such as expanding their markets to a worldwide audience and accepting orders 24 hours a day, all while lowering their administrative costs.

To deliver applications like these on-line, corporations must re-engineer their architectures to support large-scale transaction processing. Enter Java.

Java enables a rich user interface, secure database access and high-volume transaction processing - a combination that has the potential for dramatic advances in the development and deployment of Web-based enterprise applications.

However, the Java language alone is not enough to develop these end-to-end enterprise solutions. For this, a scalable Java-based platform is required to make it easy to not only develop applications, but also to manage and deploy them. This platform is starting to emerge and consists of client, middle and server tiers. It is also becoming widely adopted with the definition of standard Application Programming Interfaces (APIs) and the delivery of comprehensive products, such as development tools and component transaction servers.

Implementing this architecture, users find and launch applications using traditional HTML pages and Web servers. But instead of simply loading a static page, they download a dynamic "applet" into their browser. The applet also contains high-speed protocols that allow it to communicate directly with application servlets or business logic, which exist in the form of components running in the middle tier. The middle-tier server executes and manages most of the application logic and high-speed JDBC-based access to distributed databases. Java is also beginning to appear as the stored procedure language in back-end DBMSs, allowing data-intensive procedures to be written in Java and executed inside of the DBMS.

The combination of each of these elements provides application developers with a single

programming language across all tiers. It removes the artificial barriers between client-, middle- and server-side programming and provides developers the flexibility they need to increase productivity (by focusing on building Java components regardless of where they are deployed).

To effectively deliver Java on all tiers, standards are emerging for each key component in the enterprise Java platform. For graphical development on the client, both JavaSoft and Microsoft have created standard foundation classes: JFC and AFC, respectively. In the middle tier, Enterprise JavaBeans and ActiveX provide a standard way to deploy and manage server-side components. In addition to components, the enterprise Java platform consists of a suite of connectivity APIs: Java Naming Directory Interface (JNDI) for connectivity to enterprise naming and directory services; Java Transaction Service (JTS) for transaction services and Java Message Service (JMS) for enterprise messaging systems. JDBC, a call-level interface similar to ODBC, provides the standard mechanism for accessing relational and legacy data stores. Sybase, IBM, Tandem and Oracle are working with JavaSoft, the ANSI SQL standards committee and the JSQL consortium to develop standards for running Java in the database.

As technology vendors release new tools and servers for building these Java-based architectures, aggressive enterprises can finally reap the full benefits of the Internet. Sybase, for instance, is providing a distributed, end-to-end architecture that answers the call for mission-critical business application development for the Internet. It accomplishes this by delivering products to enable Java in the client, middle and server tiers and by providing open support for emerging standards, so it can be implemented easily throughout the IT organization.

This is "Java for the Enterprise": bringing together the power of the Java language with an end-to-end architecture, in combination with leading-edge tools and technologies. It holds great promise for businesses today. In fact, when implemented correctly, it truly does offer "endless potential".

#### About the Author

Tina Lorentz is the Product Manager for PowerJ. Previously, Tina was part of the marketing team for Power++ and Watcom C/C++. She joined Watcom shortly before the merger with Powersoft in 1994. Prior to that Tina worked for Northern Telecom for two years as an analyst/programmer, after receiving a Bachelor of Science in Math from the University of Waterloo.



## A useful paradigm for implementing collaborative client/server protocols

by Larry T. Chen & Todd A. Busby

### Abstract

This is the first of a two-part series presenting a Java implementation of a real-time multi-user blackjack game based on a collaborative, active object framework. In this article, we will walk through the design of an active object framework for developing collaborative client/server applications. Important concepts, such as synchronous collaboration, active objects, multicasting, sessions and events are defined and discussed.

### Introduction

In the world of collaborative multi-user software, or groupware, there are two main collaboration models: synchronous and asynchronous. The asynchronous model allows multiple users to exchange and share information through asynchronous messages, sent at different times to a central location. Plain e-mail, bulletin boards and Lotus Notes are examples of asynchronous groupware. The synchronous model, on the other hand, allows information sharing on a real-time basis; that is, all participants interact at the same time. Chat systems and multi-user virtual worlds are examples of synchronous groupware. Both models of collaboration are necessary in certain situations, yet most so-called groupware today ignores the synchronous collaboration model. If the goal of collaborative software is to let people work together concurrently across the boundaries of time and space, then they should be able to do so in real-time, in order to allow immediate feedback and achieve faster group consensus.

The applications of groupware are far-reaching, especially those of synchronous groupware, and will revolutionize computing in the next decade. Java has come at the right time to help fuel this revolution.

### Java: An Enabling Technology for Real-time Collaboration

The development of Java has made it easier than ever to create

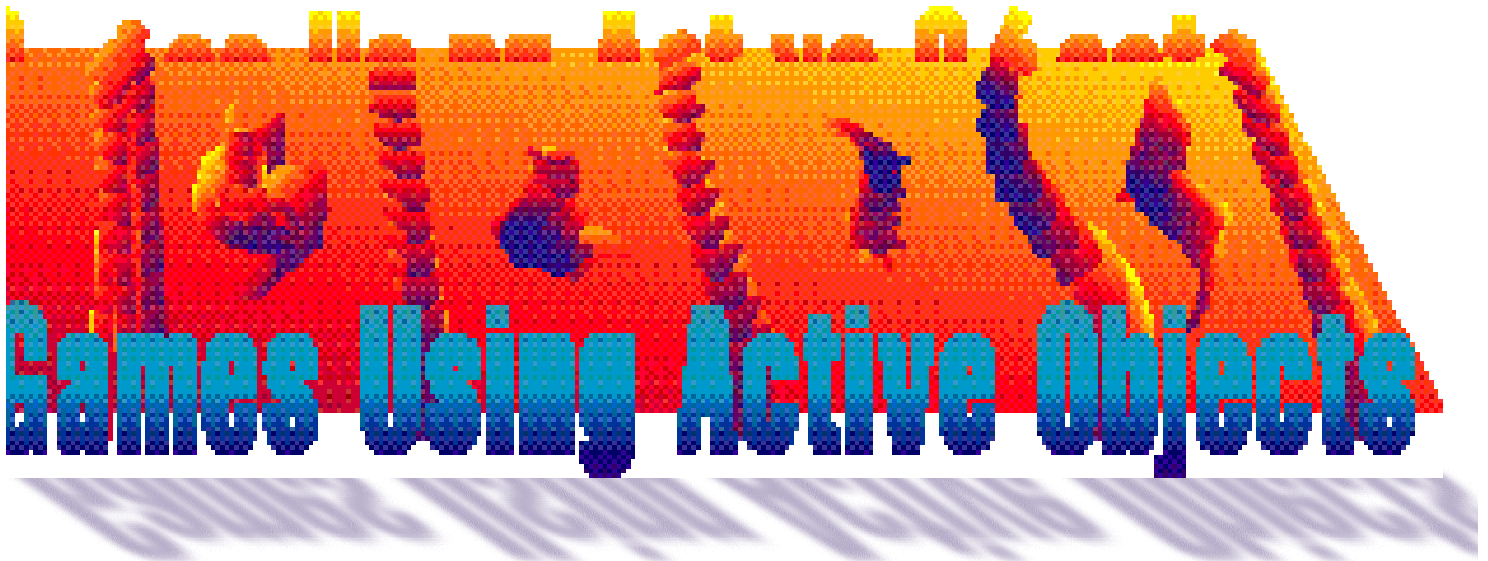
synchronous groupware over Intranets and the Internet. Java is the perfect platform to build groupware due to its built-in cross-platform networking features, and its ease of access through popular browsers. In this article, we will demonstrate one approach to building synchronous collaborative software in Java, by designing around an active object model. This approach considerably simplifies implementation of collaborative client/server protocols. A collaborative client/server protocol specifies control flow among one server and a set of clients. To illustrate the active object approach, we present a simple framework for an active object model. A sample implementation of a multi-user blackjack game using the framework will be presented in Part 2.

### What is an Active Object?

We distinguish active and normal (passive) objects by the following: Active objects possess both a proactive and a reactive event handler, while a passive object uses only reactive event handling. In other words, active objects are characterized by having a thread executing a state-based event protocol, in addition to normal callback-based (reactive) event handling. Most applications today are written only using callback-based event handling. This is where an application registers callback methods with the runtime system, which calls one of the application's callback methods when an event occurs. Java's AWT event handling system (both 1.02 and 1.1) is an example of a callback-based model. A threaded active object, on the other hand, actively waits for a set of events to occur, suspending its thread until one of the desired events occurs. The active object model permits straightforward implementations of client/server control flow, where events may be causally dependent upon one another. Implementing control flow logic is vastly complicated by having to spread the protocol's implementation across many callback methods.

As shown in Figure 1, an active object contains two main components, an active event handler used to implement client/server





control flow, and a passive event handler used to implement status updates. The active event handler is intended to process events with causal dependencies between each other, while the passive event handler is intended to process idempotent, self-contained events (i.e., events that have no causal dependencies between each other). There should also be no causal dependencies between events used by the passive handler and events used by the active handler. For example, in a multi-player blackjack game, the active handler would control the game flow with respect to the client player's game. All other status events relating to other players on the same table would be handled by the passive handler.

There are alternative approaches to building a collaborative system, such as using RMI. However, RMI is practically equivalent to a pure callback model, and RMI is not fully supported in current browsers, making it unusable for deployment of collaborative applications over the Internet today.

Let's define an abstract class `ActiveObject` to represent our concept of an active object.

```
public abstract class ActiveObject extends Thread
{
    public abstract void run();
    public abstract void handleEvent(Object evt);
}
```

Inter-object events are implemented as standard Java objects. Each event type is identified with a unique integer.

The `run()` method is where the active control flow is implemented, whereas `handleEvent()` is used to process callback events.

Note that `handleEvent()`'s thread should never be suspended (i.e., no blocking statements should ever be executed inside `handleEvent()`) because its thread is the thread that the system uses to process all incoming callback events.

The control flow thread in the `run()` method should interact with other objects, either by waiting for events or sending an event to another object. Let's define two basic methods in class `ActiveObject` to serve these purposes.

```
protected synchronized Object waitFor(int[] evts) throws ProtocolException;
```

```
protected void send(Object evt, int destID);
```

`waitFor()` takes an int array containing "hash codes" of events that the control flow thread inside `run()` is currently expecting. It either returns with an arriving expected event, or throws a `ProtocolException` to signify that an event was received which was not expected. This makes the control flow processing deterministic, and results in the automatic checking of client/server protocol correctness.

`send()` transmits an event to the destination object, specified by `destID`, which is a numeric identifier assigned to each member of a session (described later).

Both of these methods are protected to make sure that only the owning active object uses these methods. (Only `run()` should use the `waitFor()` method since it is a blocking call. `send()` is non-blocking and may be used by both `handleEvent()` and `run()`.)

We also want support for callback-based event handling, so we define

```
protected void registerCallback(int[] evts);
```

to allow the active object to specify which events should be handled using the callback model in the `handleEvent()` method.

### Implementing Event Queuing

Active event handling requires that the object possess an event queue, which collects events from other active objects and distributes them in first-in, first-out (FIFO) order to the object's active and passive event handlers. For simplicity, we will implement a FIFO queue using a `Vector`, although more efficient implementations using linked lists or piped streams may also be used. For an implementation using piped streams, see *Tips & Techniques in Java Developer's Journal* (Vol. 2, Issue 8) by Brian Maso.

A queue obviously requires a way of putting arriving events into the queue. We define

```
public synchronized void pushEvent(Object evt);
```

as the means for putting an arriving event into the queue. This method will first check if the arriving event is intended for pro-

cessing by a callback method. If so, it calls `handleEvent()` immediately; otherwise, the arriving event is pushed into the queue. Moreover, if the `run()` method is actively waiting for this arriving event, it immediately calls `notify()` to wake up the thread so that the arriving event may be processed.

Implementing the `waitFor()` method is straightforward. First, we save the array of expected events so that `pushEvent()` may be aware of them when an event arrives. Then, if the queue is empty, the thread waits (using `wait()`) until an event arrives, which is when `pushEvent()` wakes up the thread using `notify()`. If the queue is not empty or the thread is awakened, it proceeds to check that the first event in the queue is an expected event, returning it if it is an expected event, or throwing a `ProtocolException` if it is not.

### Adding Support for Collaboration

To support collaborative client/server protocols, the active object framework must support event communication among a set of objects interacting within a group. Any object wishing to notify all other objects in the group should be able to multicast an event to the rest of the group with one call, without having to track the other group members themselves.

We introduce the concept of a session to represent a group of collaborating active objects. A session is identified by its session name and its session instance number. Since it may be necessary to run multiple sessions concurrently, the session instance number (or `sessionID`) distinguishes between multiple instances of the same session protocol.

In the Java/Internet environment, making collaboration work among applets requires a server program to coordinate all participants in the session. This is because 1) an applet may only connect to its originating server, and may not communicate with other applets directly, and 2) a server is needed to start a daemon for each active session to provide a central location to which clients can connect. Thus, our Java collaborative model resembles Figure 2.

Let's define a class called `ActiveSession` to represent our concept of a session. The "Active" prefix is just to remind us that it represents a group of active objects and has nothing to do with Microsoft's ActiveX.

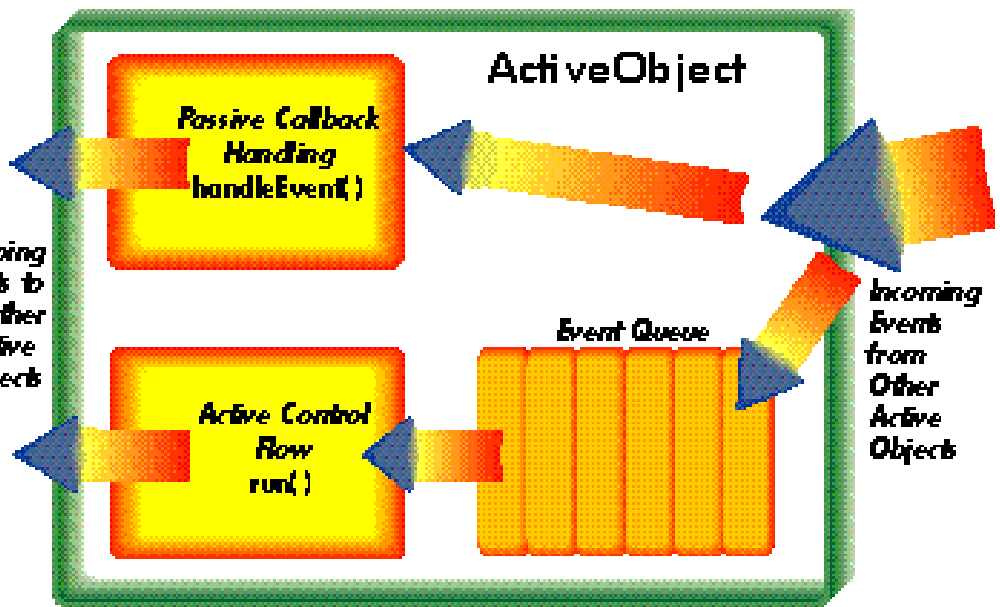


Figure 1: Active object model

```
public class ActiveSession extends Object
{
    public ActiveSession(String sessionName,
        int sessionID);
    public static void join(String sessionName,
        int sessionID)
}
```

A session is started by the server-side active object using

```
new ActiveSession(sessionName, sessionID);
```

In our framework, a server is always identified with a numeric id of 0. Thus, clients always send an event to the current server using with a `destID` of 0, i.e., `send(evt,0);`

Then, clients join the session by calling

```
ActiveSession.join(sessionName, sessionID).
```

A particular session type and instance (`sessionID`) must be started by a server before any client can join it using the same session name and `sessionID`. Any number of parallel sessions, each with a unique `sessionID`, may be started by the server.

Clients are assigned a numeric id starting from 1, up to the maximum number of clients allowed in the session. This numeric id may be chosen by the client, or it may be dynamically assigned by the session entity by finding, for example, the first available slot in the session. The sample implementation given in the code listings lets the clients choose their own numeric id.

Of course, we also need to add a multicast command to our API to send an event to everyone in the group.

```
public void mcast(Object evt, int omitID)
```

The `omitID` allows the multicast sender to omit sending a copy to someone in the group. Usually, the sender will not need a copy of the multicast event, so `omitID` usually specifies the sender itself.

### Implementing Network Communication

Full networking support is not given in the code listings due to space limitations. Instead, we do provide a simulated network in Listing 1b so that the collaborative client/server protocol may be demonstrated and tested using one program, by instantiating a server-side active object and several client-side active objects all in the same program. The clients and server will communicate strictly through event-based communication, as if they were talking across the network. However, we will discuss here, issues in implementing network support for our active object framework.

- `ActiveSession` constructor: Creating an `ActiveSession` object is equivalent to starting a daemon to listen for incoming client connections. However, we need to consider the usage of port numbers. Either all session types use the same port number, or each session type uses a separate port number. If all session types use the same port number, only one server socket (`java.net.ServerSocket`) needs to be created. Otherwise a server socket needs to be created for each session type in use. Note that there may also be multiple sessions instances for a single session type. Communications for multiple session instances of the same session type may be

# RogueWave Full Page Ad

multiplexed through the same socket, but of course the server needs to sort out events as it receives them and forward them to the appropriate sessions. This implies that each event must carry a session name and sessionID.

- `ActiveSession.join()`: A request to join a session from the client translates to a connection request to the server on a certain port--the port associated with the desired session. The server, upon a connection request, should check whether the requested session name and session instance are active (i.e., have already been started by the server).

- `ActiveObject.send()`: Since we've chosen to use class `java.lang.Object` as the base class for our events, we need to figure out a way to serialize the data members, or instance variables, of any arbitrary subclass of `Object`. In Java 1.1, serialization is available to aid in this task. Unfortunately, in Java 1.0.2 which resides in

most browsers in use today, strictly speaking this is impossible. However, we may simulate serialization by manually converting each of the values of the object's instance variables into a serialized form in the `toString()` method. We also need to provide a means of deserializing the values of the instance variables and recreating the event object. This will require adding a `requiredFromString()` method (not part of `java.lang.Object`), so another derived abstract class or interface is required.

- `ActiveObject.mcast()`: An ideal implementation of group multicast would use multicast sockets (`java.net.MulticastSocket`) to communicate efficiently with other group members. However, applets are not allowed to open multicast sockets, and even if they were, operation of a multicast socket through the Internet requires the underlying support of the Multicast Backbone (MBONE), which is not yet widely deployed. Moreover, multicast sockets only support unreliable datagrams, and would thus require designing a reliable protocol. An interim solution, though inefficient, is to use iterative unicasting through multiple TCP sockets to achieve the same effect; that

is, the server sends a copy of the event to each connected client in the session.

Listing 1b implements a simulated network within the `ActiveSession` class. Implementing full networking support is left as an exercise to the reader.

### Putting it to Work

Now, to build a collaborative system using active objects, we may subclass `ActiveObject` to define a server active object, and define another subclass for client active objects. Required event types are defined by subclassing `Object` and implementing `hashCode()` to return a unique integer identifier for each event type. We must carefully design the protocol such that causally dependent events are processed in the `run()` method, while the self-contained, independent events are processed in `handleEvent()`.

To illustrate in detail active objects at work, the second part of this series will present a multi-user blackjack game built using our collaborative active object framework.

### Conclusion

Active objects are a useful paradigm for implementing collaborative client/server protocols. They allow for intuitive coding of

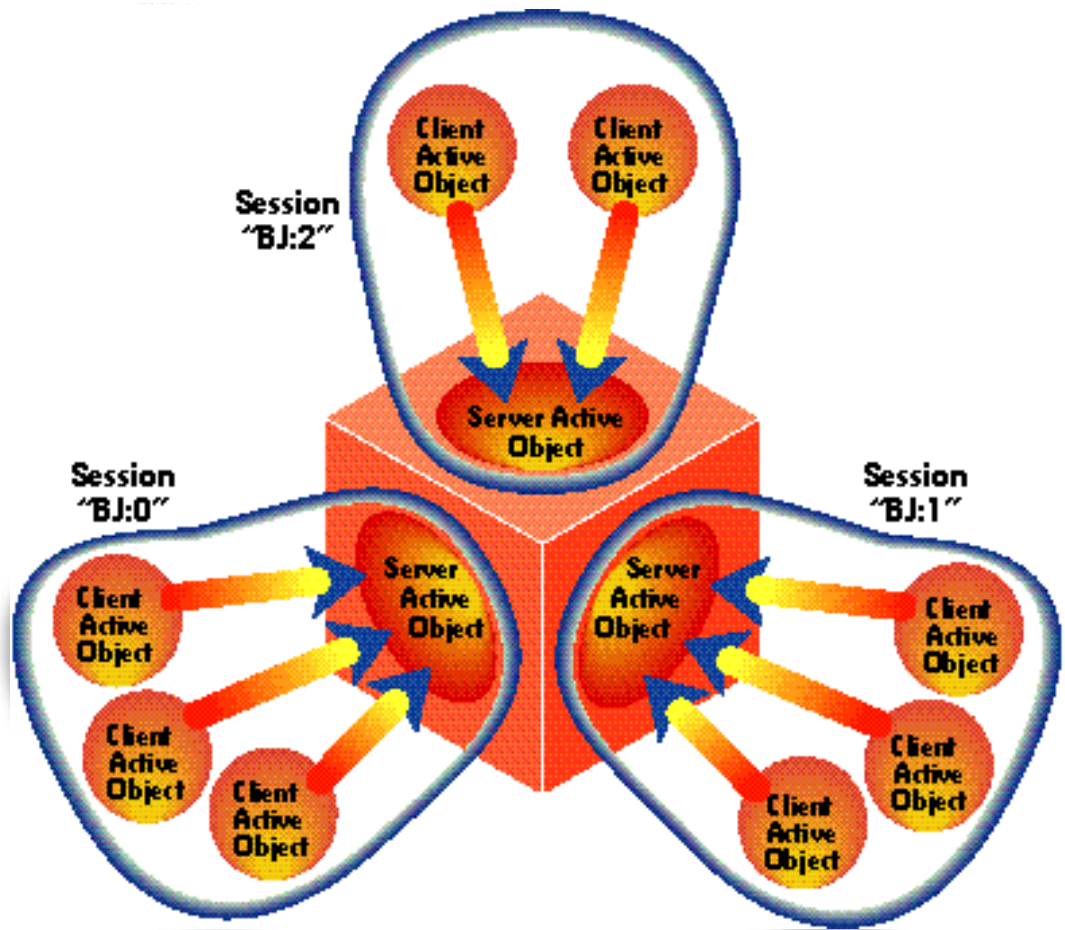


Figure 2: The session concept for collaboration

client/server control flow, rather than being forced to use awkward callback methods to process causally dependent events. Active objects are similar to actors and agent-oriented programming, well-known in the distributed systems research community. The active object paradigm has been used to successfully implement a full-scale multi-user virtual world populated with avatars and collaborative games. This virtual world is named Funtopia, at <http://www.funtopia.com>.

A complete example for the framework presented is available at <http://www.avanteer.com>. ●

#### About the Authors

Larry T. Chen is a co-founder of Avanteer, Inc., a company focusing on developing Java-based collaborative software. He is also pursuing a Ph.D. degree in the area of distributed systems at the University of California, Irvine. He may be reached at [larryc@avanteer.com](mailto:larryc@avanteer.com).

Todd Busby is a project manager at Avanteer, Inc. Todd holds an MS in Computer Science from the California State University, Fullerton. He may be reached at [toddb@avanteer.com](mailto:toddb@avanteer.com).

 [larryc@avanteer.com](mailto:larryc@avanteer.com)

 [toddb@avanteer.com](mailto:toddb@avanteer.com)



### Listing 1a: Active object.

```
import java.util.*;

public abstract class ActiveObject extends Thread
{
    /* for efficiency, use a linked-list implementation */
    private Vector eventQueue = new Vector();
    private BitSet callbackSet = new BitSet();
    private int[] expectedEvs;

    /* current active session */
    ActiveSession session;

    public synchronized void pushEvent(Object evt)
    {
        //check if intended for callback
        if (callbackSet.get(evt.hashCode())) {handleEvent(evt); return;}
        //check beginning of queue
        eventQueue.addElement(evt);
        //notify waiting object
        if (checkQueue(expectedEvs) != null) notify();
    }

    private Object checkQueue(int[] evts)
    {
        if (evts == null) return null;
        Object firstEvent = eventQueue.elementAt(0);
        for (int i=0; i<evts.length; i++)
            if (evts[i] == firstEvent.hashCode())
                return firstEvent;
        return null;
    }

    protected void registerCallback(int[] evts)
    {
        for (int i=0; i<evts.length; i++)
            callbackSet.set(evts[i]);
    }
}
```

```
protected void send(Object evt, int destid)
{
    session.send(evt, destid);
}

protected void mcast(Object evt, int omitid)
{
    session.mcast(evt, omitid);
}

protected Object waitFor(int evt)
{
    int[] evts = new int[1]; evts[0]=evt;
    return waitFor(evts);
}

public synchronized Object waitFor(int[] evts)
{
    //save expected events
    expectedEvs = evts;

    //if queue is empty, wait
    if (eventQueue.size()==0)
        try {wait();} catch (Exception e) {}

    //check front of queue and return if valid
    Object arrivedEvent = checkQueue(evts);
    if (arrivedEvent != null)
    {
        eventQueue.removeElementAt(0);
        return arrivedEvent;
    }

    //throw ProtocolException here!
}

/* implement active control flow here in subclass */
public abstract void run();

/* implement passive callback event handling here in subclass */
protected abstract void handleEvent(Object evt);
}
```

### Listing 1b: Active session.

```
import java.util.*;

public class ActiveSession extends Object
{
    static Hashtable sessions = new Hashtable();

    private Vector members = new Vector(); //Vector of ActiveObjects

    public ActiveSession(ActiveObject server,
        String sessionName, int sessionID)
    {
        ActiveSession.sessions.put(sessionName+sessionID, this);
        members.addElement(server);
    }

    public static ActiveSession join(ActiveObject obj,
        String sessionName, int sessionID)
    {
        ActiveSession s = (ActiveSession)
            (ActiveSession.sessions.get(sessionName+sessionID));
        s.members.addElement(obj);
        return s;
    }

    public void send(Object evt, int clientid)
    {
        ((ActiveObject) members.elementAt(clientid)).pushEvent(evt);
    }

    public void mcast(Object evt, int omitid)
    {
        for (int i=0; i<members.size(); i++)
            if (i!=omitid)
                ((ActiveObject) members.elementAt(i)).pushEvent(evt);
    }
}
```

# Net Guru 1/2 Ad





# Introduction to OO: part 2

## Polymorphism

### A parent class with its own states and behaviors

by John Tabbone

My last column introduced you to object orientation and discussed how some of the principles are expressed in Java. In particular, we were working with a chess example. Also, there was an assignment. You were to think about the classes: Mammal, Human and Canine, and how one might use Java notation and inheritance to describe this small taxonomy system. This column will continue with OO by reviewing the assignment and explain a powerful feature called polymorphism.

Remember that the principal thrust of OO is to model a system. A system is composed of objects and the relationships and interactions between the objects. An object is a structure that exhibits state, identity and behavior. We create a class to describe a definition for a type of object. A class is similar to a blueprint for a house object. It describes the structure of a house, but different houses built from the same blueprint can have different colors, appliances, etc. **Instances** of classes are created that can actually maintain states and do things (behave).

The assignment, discussed previously, was to consider a taxonomy system. A **Mammal** would be a base class that defines some common behaviors of mammals, and classes **Canine** and **Human** would implement the behaviors particular to their own species. The goal is to model this system, identifying the objects and their states, identities and behaviors. To start modeling the system, consider the base class **Mammal**. What is a Mammal? If I remember correctly from elementary school, a mammal is a warm blooded creature that has fur and gives birth to live young. There is probably more to it, but this definition will suffice.

Considering this, what are some possible states of a mammal? The ones mentioned in the definition are fixed. All mammals have fur, warm blood and give birth to live young.

What are some states that can vary among mammals? Well, mammals are either male or female. Mammals have a specific number of legs. And of course, a mammal can be a herbivore, carnivore or omnivore.

What are the identities of a mammal? What else is a mammal besides just being a mammal? We can say that a mammal is also a vertebrae. Just as a human is a mammal, a mammal is a vertebrae. Generally, if you can describe the relationship between two classes using the phrase 'is a' or 'is a kind of', the two classes share an inheritance relationship. In Java, one would denote that

“Inheritance and  
polymorphism are  
intrinsically linked.  
You can't have  
polymorphism  
without inheritance.”

class **Mammal** extends **Vertebrae**, and class **Human** extends **Mammal**. The relationship is transitive, meaning that a **Human** is also ('is a') **Vertebrae**.

What are the behaviors of a mammal? There are many things that a mammal can do, but for this exercise, the **Mammal** model will have only one behavior: walk. All mammals can walk (we assume). Take a look at Listing 1 to see how our **Mammal** is modeled in Java.

Notice the static final state variables. These are constant values that are included

in the class to assist the programmer. Since things like gender and dietHabits can only have certain values, it is common practice to include static final data members to ease the assignment of these states. Later in some other part of our code, a programmer can use these static final data members by making assignments such as: gender = **Mammal.MALE**; or dietHabits = **Mammal.OMNIVORE**.

The next step is to model a **Human**. As with **Mammal**, we are going to keep the model simple to illustrate a point. We can say that a human is a mammal with some added states and behaviors. What other states can a human maintain? In the interest of simplicity, let's say that a human has a name. Also in the interest of simplicity, let's say that the only other behavior that separates a human from a mammal is a human's ability to go to work. The model for this definition of a human would look like Listing 2.

Notice first that a **Human** is not an abstract class. We have designed our system this way to better reflect the real world. There is nothing in the real world that you can point to and say: "That is **ONLY** a mammal." Every mammal is a member of some species. So, since there are no real world objects that are only mammals, there will be no objects in our system that are only mammals. You cannot create a mammal object.

Keep in mind that just as in the real world, you can point to a human or dog and say, "That is a **Mammal**." In our system too, you can refer to an instance of **Human** or **Canine** and say, "That is a **Mammal**." However, in addition to being a mammal, those objects can also be identified as **Humans** and **Canines**. We will examine this in more detail after we model class **Canine**.

Class **Human** extends **Mammal**. Remember from the last article that the first line of a Java class definition expresses the identities of the class. A **Human** can be identified as a **Human** or as a **Mammal**. What does it mean to be a mammal? In our definition (i.e. our **Mammal** class), a **Mammal** has certain states (i.e. gender, dietHabits, etc.) and behaviors. These states and behaviors are **inherited** by class **Human**. This means that every human also has a gender and dietHabits and can also **Walk**. Furthermore,



# Kovisky full

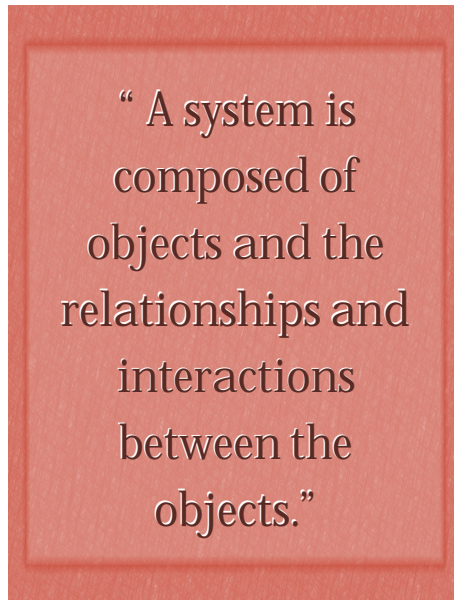
class Mammal extends class Vertebrae, which we have not defined. Had class Vertebrae been defined, a Human would have the behaviors and states defined in that class as well as a Vertebrae identity.

Class Human provides a class specific implementation for the abstract method Walk(). This means that we define the walk method for a human to reflect how a human walks. First of all, a human has two legs. He will throw one foot out in front of him, fall a bit until that foot hits the ground, and then repeat the process for the other foot. The Walk method for a human will be different than the walk method for a four legged creature because a four legged creature walks differently. Focus on the fact that all mammals can walk, regardless of how the method is actually implemented.

Finally, class Human defines the state variable name and the behavior goToWork(). These are things specific to Humans. Since these states and behaviors are ones not shared by all mammals, they are defined in the Human class.

A Canine is also a Mammal. For this exercise, our Canine class will have a bark behavior, and they will be carnivores. Look at Listing 3 for a brief class description.

We now have classes that we can use to create Mammal, Canine and Human objects. Our class definitions are pretty light, but sufficient to demonstrate polymorphism. Polymorphism is an ability to have many shapes. It refers to the ability of an object to assume the identity of one of its parent classes. In this exercise, an object of type human can assume the identity of a Mammal. In other words, a variable of type Mam-



mal can contain an instance of a Human or a Canine or any other descendant of Mammal. Listing 4 demonstrates this.

When a Human is being identified as a Mammal, only methods defined in class Mammal can be called. m.bark() makes no sense because bark() is a behavior of a Canine. When walk() is called on m from Listing 4, the walk method defined in class Human is actually invoked. Similarly, a state variable overridden in class Human will be accessed through m with the same effect. That is, even though we are identifying the object through m (a mammal), the state variable accessed will actually be of class Human. This leads to a more formal definition of polymorphism: Polymorphism is an object's inherent ability to be identified as

one of its parent classes, yet implement its own specific states and behaviors.

Notice that polymorphism is an 'inherent ability'. Inheritance and polymorphism are intrinsically linked. You can't have polymorphism without inheritance. The key here is that a class can have many identities. Remember, the first line of a class definition enumerates the possible identities that the class can assume. In the case of Human and Canine, both can also be identified as Mammals. Even when an object of type Human or Canine is identified as an object of type Mammal, the implementation of methods and data contained in state variables remains. Therefore, if m.walk() is called immediately after m = john in Listing 4, the walk() method executed will be john's walk method. The Human one.

Polymorphism is widely demonstrated in Java, especially in the AWT. The Abstract Windowing Toolkit has an abstract base class Component. Component is the parent class of all of Java's widgets. Things like Buttons, Labels and Textfields all extend class Component. A Panel is an object that can contain other Components. One might add Buttons and other Components to a Panel for screen display. A Panel needs to know the size of every widget that it is holding to make sure that everything is laid out correctly. Thanks to polymorphism, a Panel can maintain an internal list or array of Components. Any descendant of Component can be added to the list, and therefore be displayed on a Panel. So, using one array of type Component, a Panel can keep a list of any kind of widget added to it, as long as the widget descends class Component. Fur-

**Listing 1: Abstract class Mammal extends Vertebrae.**

```
{
    public static final int    MALE          =    0 ;
    public static final      int    FEMALE   =    1 ;
    public static final      int    HERBIVORE=    0 ;
    public static final      int    CARNIVORE=    1 ;
    public static final      int    OMNIVORE =    2 ;

    public final boolean      hasWarmBlood   =
true;
    public final boolean      givesBirthToLiveYoung =
true;
    public final boolean      hasFur        =
true;

    public int    gender;           // 0 for male, 1 for female
    public int    numLegs;         //
    public int    dietHabits;     // 0 for herbivore, 1 for carnivore,
2 for omnivore

    abstract public void walk();
}
```

```
} // end class
```

**Listing 2: Class Human extends Mammal.**

```
{
    String name;

    public Human()
    {
        gender = MALE;
        numLegs = 2 ;
        dietHabits= OMNIVORE;
    }

    public void walk()
    {
        // The code that describes how a human walks goes here.
    }

    public void goToWork()
    {
        // The code that describes how a human goes to work goes here.
    }
}
```



Furthermore, method `getSize()` is defined in class `Component`. The panel can traverse the array and call `getSize()` on every element. Polymorphism will ensure that the actual size returned, is the size stored in the `Button`, `Label`, `Textfield` or whatever the actual `Component` is that has been added to the `Panel`.

There are many more examples of Polymorphism in the Java core packages. Your assignment is to look through the API documentation and find some. Start with any package you like. You will find a lot in the `java.io` package. Remember, polymorphism is an object's inherent ability to be identified as one of its parent classes, yet implement its own specific states and behaviors. Look for abstract base classes. They are typically a dead giveaway for polymorphism. 🍌

---

#### About the Author

John V. Tabbone is a lecturer at New York University's Information Technologies Institute, where he teaches two Java programming courses and advises on curriculum development. He has been a professional Java programmer since early 1996, and continues to consult on and develop systems for a variety of New York based businesses. You may e-mail him with questions and comments at [john.tabbone@nyu.edu](mailto:john.tabbone@nyu.edu).



[john.tabbone@nyu.edu](mailto:john.tabbone@nyu.edu)

```
} // end class
```

---

#### Listing 3: Class `Canine` extends `Mammal`.

```
{  
  
    public Canine()  
    {  
        gender = MALE;  
        numLegs = 4  
        dietHabits= CARNIVORE;  
    }  
  
    public void walk()  
    {  
        // the code that describes how a human walks goes here  
    }  
  
    public void bark()  
    {  
        // Woof!  
    }  
}
```

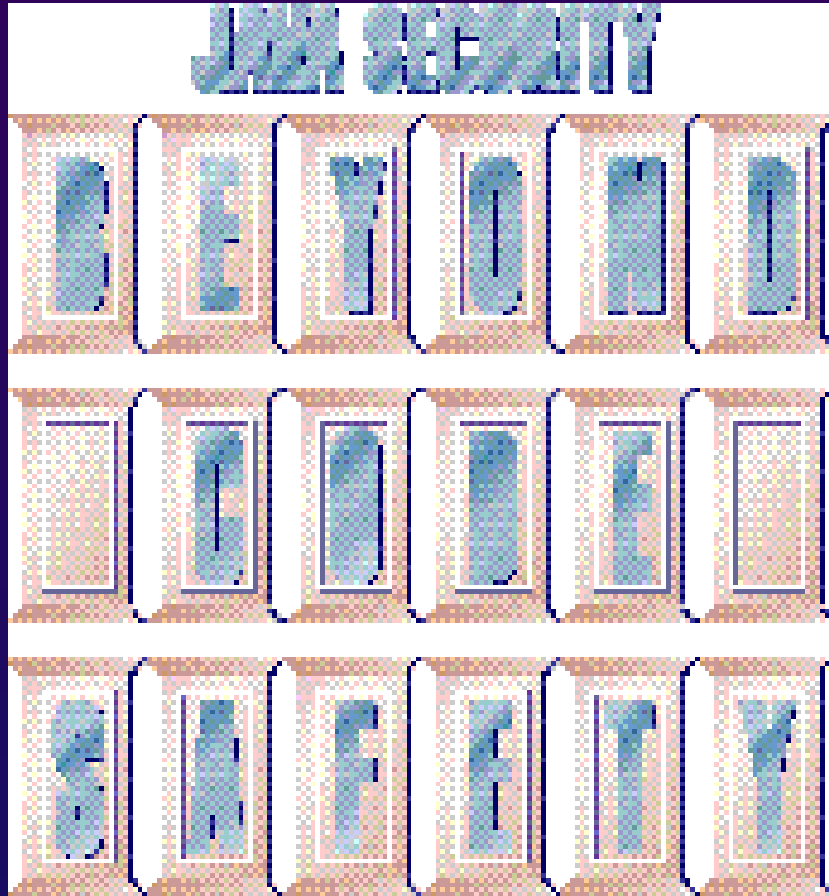
---

#### Listing 4.

```
...  
...  
Human john = new Human();           // create a human object  
Mammal m;           // declare a Mammal variable  
m = john;           // this is legal;
```

1/2 Ad





## Experiences in using JDK 1.1 Security

by Jahan Moreh

Much has been said and written about Java security and developing secure Java applications for deployment on the Internet. However, most available materials deal with Java security in the context of three different but related aspects:

- The Java language code safety mechanism via the sandbox model
- The Java compiler and runtime environments
- The Java SecurityManager class



In an article entitled **Implementing a Security Policy** (Java Developer's Journal, Vol. 2., Issue 8), Qusay Mahmoud wrote on the practical uses of the Java SecurityManager class.

Starting with JDK 1.1, Java provides a number of classes and interfaces for implementing a comprehensive security policy. These include cryptographic interfaces for

Option	Meaning
c	Create a new identity
d	Display certificate for an identity
gc	Generate a certificate for an identity
gk	Generate a key pair for an identity
gs	Generate signature for data stored in a file
ic	Import a certificate for an identity
ik	Import public key for an identity from a file
ikp	Import key pair for a signer identity from a file
l	List all identities in the identity database in succinct form
ld	List all identities in the identity database in detail
li	List an identity in detail
r	Remove an identity from the database
s	The new identity is a signer (has a private key)

Table 1: Javakey options

Option	Meaning
O	Store only; use no ZIP compression
c	Create new archive
f	Use the next argument as the archive file name
m	Include manifest information from specified manifest file
M	Do not create a manifest file for the entries
t	List table of contents for archive
v	Generate verbose output on standard error
x	Extract named (or all) files from archive

Table 2: Jar options

signature production and verification, cryptographic checksums as well interfaces for access control. This article focuses on practical uses of the cryptographic interfaces of JDK 1.1, including their use in creating and verifying the origin of trusted applets.

## Overview

Many organizations are attempting to

harness the connectivity power of the Internet to conduct real business on a global scale and earn more money. In order for electronic business to flourish, the infrastructure must provide a basic framework of trust. The elements of trust include provisions for:

1. Identification and Authentication – The process of ascertaining the identity of one party to another
2. Authorization and Access Control – The process of granting and denying the rights to a party to perform a certain operation
3. Privacy-protection – The assurance that the details of the transaction can not be revealed to a third party.
4. Integrity-protection – the assurance that a receiving party would reject a transaction that is accidentally or maliciously corrupted
5. Non-repudiation – The assurance that an entity cannot deny being a party to a transaction after it voluntarily engages in that transaction
6. Audit – A means of securely keeping a detailed record of events for possible future examination

Java is rapidly becoming the language of choice for developing and deploying Internet-based applications. The longevity of Java depends on many factors, including the availability of built-in tools to implement a security policy that reflects the needs of individual organizations.

## Trust Model before JDK 1.1

From its very inception, the Java language has implemented a simple and somewhat effective trust model. This model – known to Java developers as the sandbox – provides a binary choice within the runtime environment: trust everything that is local, but do not trust anything that is downloaded. In other words, the Java runtime completely trusts each and every Java application and completely restricts each and every downloaded Java applet. The Java runtime restricts applets from accessing local files. This restriction manifests itself in several ways, including the restriction to load a local class, to link with a local library and to read and write local files. Implementations of the JDK within each browser modify the SecurityManager class to provide different degrees of access to downloaded applets. However, the trust model remains binary.

## The Sandbox Dilemma: To Use or Not to Use

Experience shows that from a security policy standpoint, enforcing the sandbox

model is very important. The sandbox protects you from malicious attacks and misbehaving applications that can delete your files, misuse private information or consume your system resources. Simultaneously, most applications need to interact with users in a personalized way. This means that in order for an applet to provide a useful function – especially in the context of electronic business – it does need access to some local resources.

## Solutions to the Sandbox Dilemma

JavaSoft addresses the sandbox dilemma in two releases of the JDK. JDK 1.1 allows the originator of an applet to digitally sign the applet (see the sidebar for a primer on digital signatures). The digital signature provides a highly tamper resistant fingerprint of the applet. The client who downloads the applet can choose to trust the originator and allow it to access local resources (files, network connections, etc.). Therefore, a client can treat each applet differently based on its digital signature. However, the trust model for a given applet remains binary. JavaSoft addresses this problem in JDK 1.2 with fine grain access control. JDK 1.2 introduces the concept of protection domains, which allows a client to specify exactly which resources a given applet may access and for what reason (read, write, connect, etc.).

## JDK 1.1 Tools for Applet Signing

JDK 1.1 provides two tools for applet signing: Javakey and Jar. Use Javakey to create identities and load them into your identity database. Once you create an identity, you can designate it as a trusted identity. Applets that are signed by a trusted identity are trusted applets. JDK 1.1 treats trusted applets as local applications: a trusted applet has access to everything a local application can access. Using Javakey you can also create a signer identity. A signer is an identity with a private key. You create a signer on the server side before signing the applet. Table 1 shows the various options of Javakey.

Even though Jar is not a security tool, it is very useful for packaging a series of files, including Java classes, into a single downloadable entity. The Jar command provides the same function as the UNIX tar command. Table 2 shows the various options of Jar.

## Steps in Creating a Signed Applet

In order to create a signed applet, you need to take the following steps. Listings 1 and 2 show an example of each step. The string ambrosia is the prompt of the system that was used to generate these examples.





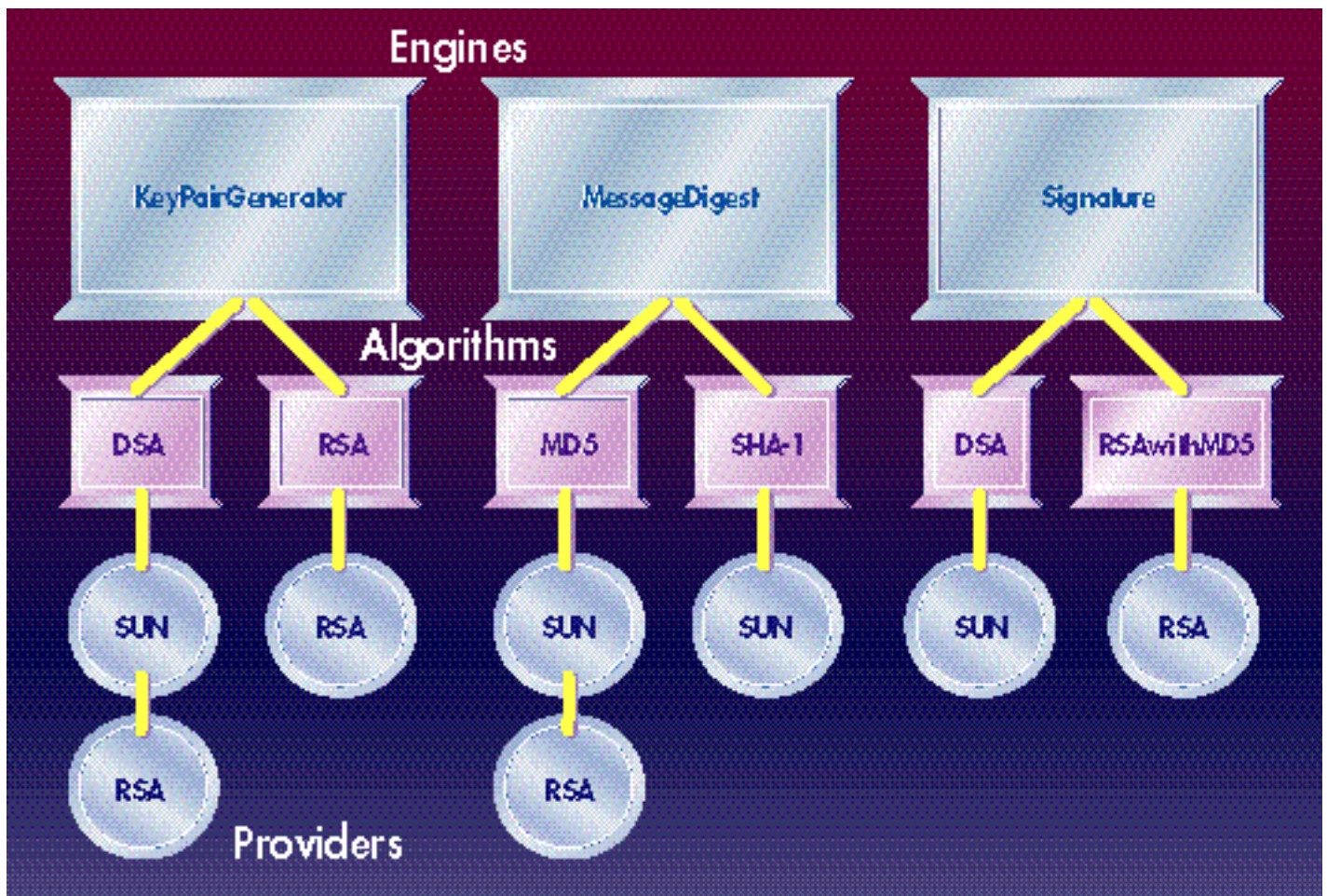


Figure 1: Basic hierarchy of the JCA

Step 1: Create a signer identity and generate key pair.

Using Javakey, you would create a signer and trusted identity. In Listing 1, with the command [1] we create a signer identity called OpenHorizon. After creating a signer identity, you need to generate a public/private key pair for that identity. The algorithm you choose for the signing and verifying signatures and length of the key are important parameters contributing to the overall security of your environment. In Listing 1, with command [2] we choose the Digital Signature Algorithm (DSA) with a key length of 1024 bits. The public and private keys are stored in two files called oh.pub and oh.priv respectively. Note that it is extremely important to protect the private key file.

Step 2: Generate a certificate for the signer

Every signer must have its public key certified. In a real Public Key Infrastructure (PKI) environment, a Certification Authority (CA) issues a certificate which binds a signer's identity to the signer's public key. The CA itself may have a certificate issued to it by another, higher authority CA. This is called chain-of-trust and is not supported by JDK 1.1. In Listing 1, with command [4]

we create a self-issued certificate: a certificate issued by the identity OpenHorizon for itself. Note that certificates are issued according to a certificate directive file. The certificate directive is shown in Listing 1, with command [3]. It specifies, among other information, a 1 year validity period, a serial number (1100) and the name of the file where the certificate will be stored (oh.cert).

Step 3: Create and sign the archive

You would follow the procedures in steps 1 and 2 to prepare your system for signing applets. Follow the procedures in steps 3 and 4 every time you need to create and deploy a signed applet. In Listing 1, with command [5] we first create a Jar file of all classes that comprise a sample of a software called Ambrosia. We call the target Jar file AmbrosiaSamples.Jar. Using command [7], we sign the Jar file with the private key of an identity whose name appears in a signature directive file (listed with command [6]). The signature file name that appears at the end of the signature directive file is used to create the signature as an individual file (OHSig.DSA). The signature file is included in the Jar file.

Step 4: Deploy signed archive on the Web server.

Javakey creates a file with the same name (AmbrosiaSamples.Jar) and the extension .sig. We rename Ambrosia.Jar.sig to Ambrosia.Jar, which is a more suitable name for deploying the signed classes on the Web server. In Listing 1, with command [8] we rename the Jar file. Listing 2 shows the HTML file for one applet (oh.pub.class) embedded in the signed Jar file.

### Configuring the Client

The client must be configured to recognize and validate the signer of the applet. Listing 3 shows the Javakey commands you can use to configure the client. You may have to use a different set of commands depending on your browser. The client must obtain the certificate file (oh.cert in this example) before loading it to the identity database (Listing 3, command [9]).

### The Java Cryptographic Architecture

In this section of the article we focus on the Java Cryptographic Architecture (JCA). Figure 1 shows the basic hierarchy of the JCA. There are three classes that form the foundation of the JCA: KeyPairGenerator,



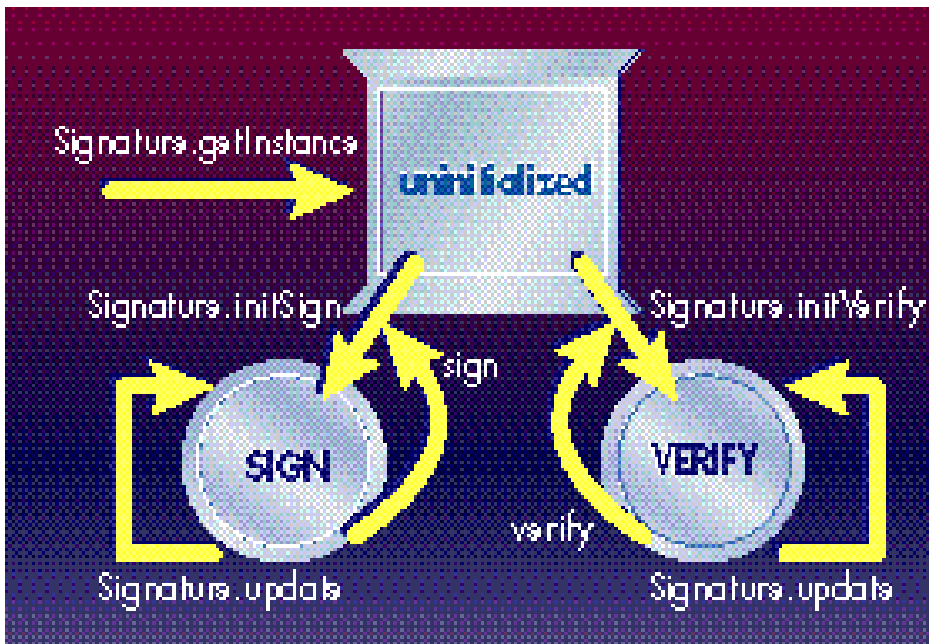


Figure 2: States of the Signature class

MessageDigest and Signature. You may notice that the JCA does not include any classes for encryption. The reason is that including encryption classes or interfaces would make the JDK non-exportable. Therefore, JavaSoft has elected to provide an adjunct package called the Java Cryptographic Extensions (JCE), which is only available to customers in the United States and Canada.

The JCA is based on the notion of Cryptographic Package Providers (CPP). As such, the JDK itself simply provides engine classes and interfaces. Engine classes and interfaces may be implemented by one or more CPP's. JDK 1.1 comes with a default implementation for the DSA, MD5 and SHA algorithms. Other CPPs can provide implementations for the same or new algorithms. CPPs may be installed statically at the time of JDK 1.1 installation. You can also use methods of the `java.security.Security` class to dynamically manage providers.

### Getting an Instance of an Engine Class

All three engine classes have a factory method that allows you to instantiate an object with a specified algorithm and a specified provider. For example, in Listing 4 line 9 we obtain an instance of the DSA signature algorithm provided by the default CPP (in this case, SUN). If desired, one can use another version of the `getInstance` method to name a specific provider.

### Signing Data and Verifying Signatures

The Signature class can be used to both sign data and verify the digital signature of the data. Consequently, when you instantiate a Signature object you must initialize it to be either in a SIGN state or a VERIFY state. Figure 2 illustrates the states of the Signature class. Once in a given state, you can supply the Signature class with as much data as you wish and with as many update method calls as you wish. After you

have supplied all the data, you would invoke either the `sign` or the `verify` method to perform the corresponding operation. After the data is signed or verified, the Signature object returns to an uninitialized state.

### Digitally Signing Data

Listing 4 shows a partial code segment for digitally signing data. Lines 1 and 2 import the security packages needed for signing data. Lines 3 and 4 define two byte arrays: one is the input data to be signed, and the other is the digital signature. At line 9 we instantiate a Signature object which implements the DSA algorithm by the default provider (i.e., SUN). Line 10 places the object in the SIGN state. Line 11 feeds the data to the object and Line 12 produces the digital signature. Note that if we were receiving the data in chunks, say from a user, we could call the update method as many times as required until there was no more data. Also note that the private key that is used to sign the data must be loaded (not shown in the code) prior to calling the `initSig` method. Use methods of `KeyPair` and `Key` interfaces to set the private key.

### Verifying the Digital Signature

Listing 5 shows a partial code segment for verifying the digital signature of some arbitrary data. Lines 1 and 2 import the security packages needed for verifying the digital signature on the data. Lines 3 and 4 define two byte arrays: one is the input data whose digital signature is to be verified and one is the actual digital signature. Line 6 defines a boolean that will be set by the `verify` method to true or false depending on whether the signature is valid or not. At Line 10 we instantiate a Signature object which implements the DSA algorithm by the default provider (i.e. SUN). Line 11 places the object in the VERIFY state. Line 12 feeds the data to the object and Line 13 verifies the digital signature. Again, note that if we were receiving the data in chunks,

#### Listing 1: Creating a signed applet.

```
ambrosia[1] -> javakey -cs OpenHorizon true
created identity [Signer]OpenHorizon[uninitialized][trusted]

ambrosia[2]-> javakey -gk OpenHorizon DSA 1024 oh.pub oh.priv
Generated DSA keys for OpenHorizon (strength: 1024).
Saved public key to oh.pub.
Saved private key to oh.priv.

ambrosia[3]-> cat oh.certDir
issuer.name=OpenHorizon
issuer.cert=1
subject.name=OpenHorizon
subject.real.name=OpenHorizon, Inc.
```

```
subject.org=Software Development
subject.org=OpenHorizon
subject.country=US
start.date=10 April 1997
end.date= 10 April 1998
serial.number=1100
out.file=oh.cert
ambrosia[4]-> javakey -gc oh.certDir
Generated certificate from directive file oh.certDir

ambrosia[5]-> jar cf AmbrosiaSamples.jar *.class

ambrosia[6]-> cat oh.signDir
signer=OpenHorizon
cert=1
```



# A Primer on Digital Signatures

Digital signatures are a cornerstone in JDK 1.1 security and applet signing. On the sending side, the originator of the document (in this case the entire applet is treated as the data in the document) prepares a digest of the document. Digests are one way functions. That is, knowing the digest, you can not reproduce the original document. Furthermore, if you use a strong digest algorithm, the chances of two documents producing the same digest are very low, particularly if the documents themselves are very similar. For example, if I have two texts as follows:

```
It is time for all good men to come to the aid of the party
and
It is time for all good men to come to the aid of their party
```

These two sentences will produce a different digest.

After you prepare the digest, you encrypt it with your private key. Then you send the original document along with its encrypted digest. Anyone can view your document. Anyone can alter the document too. Anyone can produce a digest for the altered document as well. But no one can reproduce the encrypted digest. That is because the digest is encrypted with your private key which is known only to you.

On the receiving side, the recipient takes the document you have sent and produces its own digest. The recipient uses your public key to decrypt the digest you have sent. If the two digests (the one you sent and the one computed by the recipient) are the same, then no one has altered the document. If the two digests are different, then the document is corrupted and the recipient rejects it.

The most popular digest algorithms are Message Digest 5 (MD5) and the Secure Hash Algorithm (SHA). With both of these algorithms, any message of any size digests to a few hundred bits. Hence, the performance cost of signing a digest is the same regardless of the size of the message. Obviously the larger the document, the longer it takes to compute the digest.

we could call the update method as many times as required until there was no more data. Also, note that the public key that is used to verify the digital signature must be loaded (not shown in the code) prior to calling the initVerify method. Use methods of KeyPair and Key interfaces to set the public key.

## Conclusion

JDK 1.1 provides an excellent starting point for building a secure and trustworthy infrastructure. Java security is much more than code safety and sandboxing. With JavaSoft's plan to provide more security classes and interfaces, either as part of the JDK or as adjunct packages, Java will solidify its position as the language of choice for developing and deploying Internet-based applications. ☘

### About the Author

Jahan Moreh is the chief security architect at Open Horizon, Inc. ([www.openhorizon.com](http://www.openhorizon.com)). He is a frequent speaker on the topic of Java security at various conferences. Additionally, he is a senior member of the teaching staff at UCLA's department of Information Science where he teaches classes in distributed system security and CORBA. You can reach him at [jmoreh@openhorizon.com](mailto:jmoreh@openhorizon.com)



[jmoreh@openhorizon.com](mailto:jmoreh@openhorizon.com)

```
chai n=0
signature.file=OHSig

ambrosia[7]-> javakey -gs oh.sigDir AmbrosiaSamples.jar
Adding entry: META-INF/MANIFEST.MF
Creating entry: META-INF/OHSIG.SF
Creating entry: META-INF/OHSIG.DSA
Adding entry: ohsub.class
Adding entry: ohpub.class
Signed JAR file AmbrosiaSamples.jar using directive file
oh.sigDir

ambrosia[8]-> mv AmbrosiaSamples.jar.sig AmbrosiaSamples.jar
```

### Listing 2: HTML file for deploying on the Web server.

```
<HTML>
<HEAD>
<TITLE> Open Horizon Publishing Application</TITLE>
</HEAD><BODY>
<P>
<HR>
<APPLET code = ohpub.class
archi ve = AmbrosiaSamples.jar
width = 200
height = 300>
</APPLET>
</HR>
</BODY>
</HTML>
```

### Listing 3: Setting up the client to verify a signed applet.

```
ambrosia[9]-> javakey -c OpenHorizon true
Created identity OpenHorizon[uninitialized][trusted]
```

```
ambrosia[10]-> javakey -ic OpenHorizon oh.cert
Imported certificate from oh.cert for OpenHorizon
```

### Listing 4: Producing a Digital Signature.

```
1 import java.security
2 import java.security.interfaces
3 byte toBeSigned [];
4 byte signedData [];
5 . . . . .
6 // priv must be initialized with a copy of the
7 // private key. Not shown here.
8 PrivateKey priv;
9 Signature sig = Signature.getInstance("DSA");
10 sig.initSign(priv);
11 sig.update(toBeSigned);
12 signedData = sig.sign();
```

### Listing 5: Verifying the Digital Signature.

```
1 import java.security
2 import java.security.interfaces
3 byte toBeVerified [];
4 byte theSignature [];
5 boolean isValid;
6 . . . . .
7 // pub must be initialized with a copy of the
8 // public key. Not shown here.
9 PublicKey pub;
10 Signature sig = Signature.getInstance("DSA");
11 sig.initVerify(pub);
12 sig.update(toBeVerified);
13 isValid = sig.verify(theSignature);
```





# Wrangling Big Iron

## Using Java to move mainframes into the future

by Eric Lehrfeld

Web-based distribution of applications is a proven IT winner. Since the ascendance of the browser, hundreds of success stories have emerged surrounding the deployment of Internet/Intranet applications. Many of these, in one way or another, involve the opening up of legacy applications and data through distributed clients. According to commonly cited figures, around 70 percent of the world's data and associated applications reside on mainframe platforms. It is, therefore, not surprising that numerous products and frameworks have emerged over the past year designed to integrate the mainframe into the emerging world of Java and distributed network technology.

Russ Bartels, Director of Middleware Consulting Services for Hitachi Data Systems (HDS) regards the extension of the mainframe into the object world as an important IT trend. Hitachi Ltd., the parent company and a major global mainframe vendor, is developing object-oriented products to enable businesses to link electronic commerce applications to existing applications and data residing on legacy computer systems.

"Most of our clients that are developing electronic commerce applications require sophisticated, object-oriented solutions. As the coordination of multiple applications with legacy code and data becomes the norm, the new technology will supercede previous approaches, such as simple screen scraping," said Bartels.

Over the past year, a number of companies have begun to offer products that provide a starting point for bringing the mainframe into the Java universe (or vice versa if you're of the mainframe world). Products have appeared that allow you to distribute a robust GUI interface over the Internet

based upon your mainframe application. Most importantly, they limit writing off the huge investment you've made in your mainframe as much as possible, evolving your existing systems rather than replacing them.

The current offerings of mainframe/Java technologies can be divided into two general classes:

- accessing mainframe applications through the network
- accessing mainframe data and CICS transactions as services from external applications

We will look at a few vendors offering products in each space.

The ability to access mainframe applications over a network and through a GUI OS has been around for some time through technologies such as terminal emulation and screen-scraping. It should come as no surprise then that they were among the first applications of the mainframe world to move to Java.

Over the past year, products have emerged that fit the traditional "screen-scraping" model; products that allow you to build thin Java clients by cutting and pasting fields from existing character-based host screens, and by mixing in GUI attributes such as buttons, pull-downs, color, etc. These technologies provide the low-cost reach of a browser-based distributed client and the benefits of GUI look and feel.

A product called Jacada from Client/Server Technology ([www.cst.com](http://www.cst.com)) automates the generation of thin Java clients from host screens using a rules-based technology called KnowledgeBase. The rules engine is designed to recognize patterns in the original host screen and

translate them into corresponding graphical objects.

"Java-based graphical clients deliver on the promise of intuitive, easy user access, without the complexity and cost of traditional client/server architectures," said David Holmes, Vice President of Marketing for CST, Inc. "The future is bright for organizations to reap huge returns from their mainstay application investments."

A recent offering from Advanced Transition Technologies ([www.att-inc.com](http://www.att-inc.com)) called ResQ!Net uses a patented technology to build thin clients directly from the mainframe datastream. Using the ResQ!Net authoring tool, "ResQ!Net instantly enhances the look and feel of host legacy applications and is such a natural fit for the NC that I won't be surprised to see it running on all IBM's Network Station computers, as well as NCs from other vendors," Todres Yampel, President of AT2 commented.

The next generation of mainframe/Java technology has already begun to appear. In general, they are built around the External Presentation Interface (EPI) and External Call Interface (ECI) methods of accessing CICS transactions.

The ECI allows a non-CICS application to call a CICS program in a CICS server. These calls can be either synchronous or asynchronous, meaning that the application has the option of maintaining control while waiting for a return from the called CICS program. ECI also allows for simultaneous connection to multiple CICS servers from a single application.

The EPI allows you to develop GUI front ends for existing CICS transactions without needing to modify the CICS. Applications can use the EPI to communicate with a CICS transaction and can exploit the presentation facilities of the client system to communicate with the end user. For example, if an application receives input from an external device such as a bar code reader, the application can use the EPI to convert the input into a 3270 data stream to start a CICS transaction and pass the data to it. Output from the CICS transaction is passed back





and converted into the normal data type used by the application.

One product from IBM that takes advantage of these interface methods is the CICS Gateway for Java (<http://www.hursley.ibm.com/cics/internet/cicsgw4j/index.html>). The Gateway sits on the same processor as the Web Server and routes ECI and EPI calls made from Java through a CICS Client to the desired CICS server applications; manages the many communication links to the connected browser or network computers; and controls asynchronous conversations to the CICS server systems.

Another product called Interspace from PlanetWorks (<http://www.planetw.com>) supports calls to CICS through ECI and EPI. Interspace can also serve as a generic bridge between Java applications and various mainframe messaging middleware, including Distributed CICS, Encina, MQSeries, TOP END or TUXEDO.

"Interspace allows customers to integrate the best of the old with the best of the new and the best of the future. This means that customers are not locked into a dead-end solution, as they would be with a screen scraper approach," said John Santoro, VP of Development for Planetworks.

Blue Lobster Software (<http://www.bluelobster.com>) Mako Server allows you to create CORBA interfaces to ECI, meaning

The ability to access mainframe applications over a network and through a GUI OS has been around for some time..."

that Java calls to CICS can be made transparently through an ORB. By using the Mako server, applications can take advantage of all the benefits of Java to CORBA communication.

"Our customers want access to their mainframe data without having to change existing applications," said Andrew Wilson, Chief Architect at Blue Lobster Software. "Our approach is to look at the Mainframe as just another server. Using Java technologies, we can ensure cross-platform compatibility. Using CORBA, we can take advantage of its distributed object capabilities and its inherent security benefits."

Bartels's group at HDS is developing products and services focused on security,

wrapping of legacy applications and tools that will enhance object request broker products. Working with some of the vendors mentioned, and developing new tools as they go along, HDS seeks to make all this meaningful to their clients. As an example, he noted that their experience with home banking for a major bank involved interfacing with 15 different legacy applications or data bases. To build a home banking capability rapidly, banks must be able to analyze the existing code and develop quickly the wrappers to link to the coordinating, object oriented application. And home banking, like many other industries, demands applications that are secure and fast.

Over the last twenty years, no platform has a better track record for speed and security than the mainframe (UNIX lovers, please address angry responses to deny-it-if-you-can.com). Now, Java, middleware technologies and distributed architectures promise to extend that record into a new millenium. ●

#### About the Author

Eric Lehrfeld is Director of Business Development for Random Walk Computing, Inc. (<http://www.randomwalk.com>). You can reach Eric at [lehrfeld@randomwalk.com](mailto:lehrfeld@randomwalk.com)



[lehrfeld@randomwalk.com](mailto:lehrfeld@randomwalk.com)

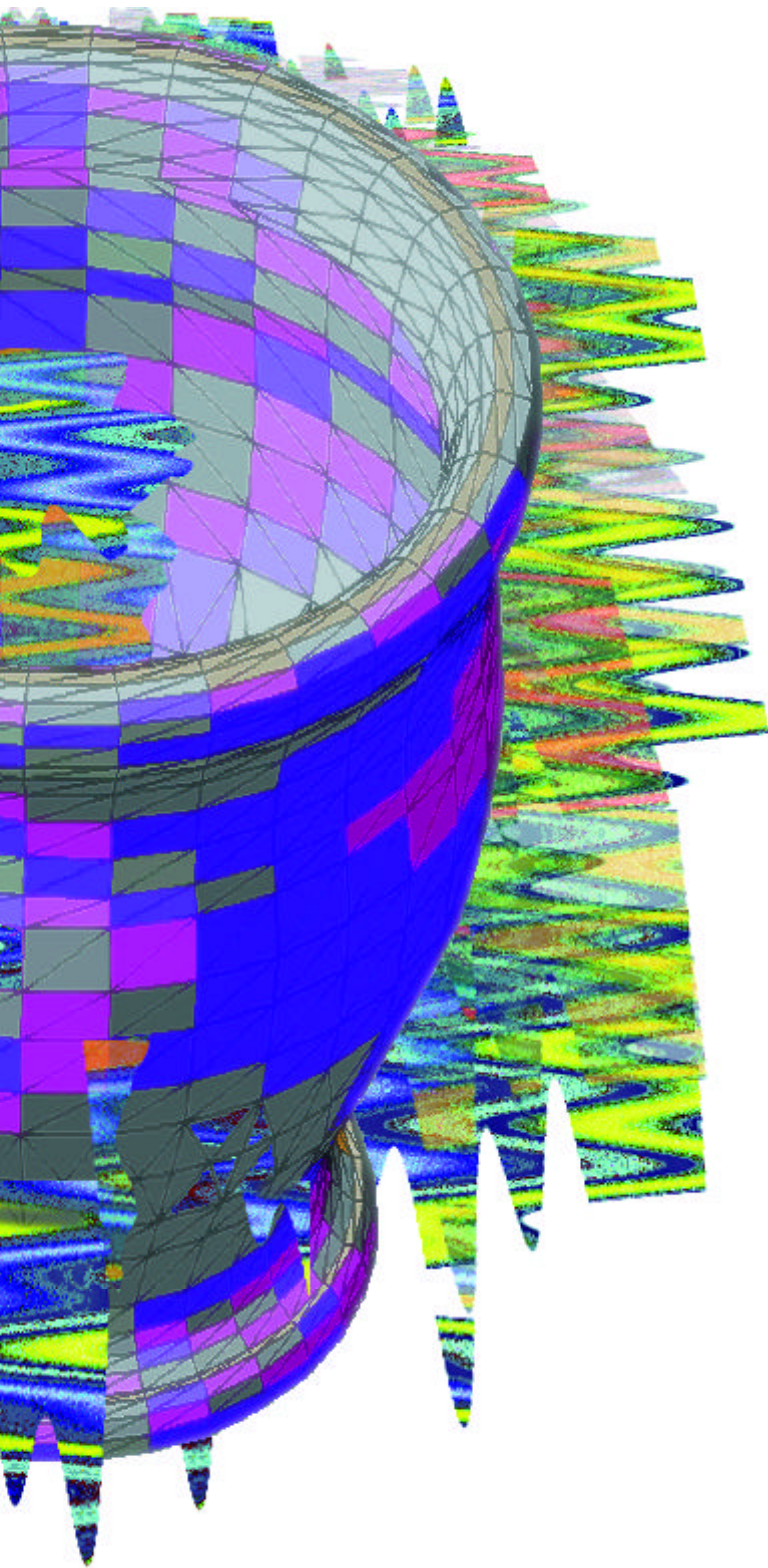
# 1/2 Ad



# Writing Unix Filters in Java

by Kenneth J. Kranz

Using Java in place of Perl, ksh and bash



Over the past year, it's been difficult to pick up a technical journal and not see an article extolling Java's usefulness with regard to creating applets.

more recently, due in part to Sun's "Java Everywhere" campaign, we are beginning to see applications featuring server-side Java (servlets) and imbedded Java devices (phones, light switches, etc.), as well as large-scale standalone applications (Sun Java Server). What continues to be a seldom discussed subject is small-scale standalone Java applications.

While there have been several major standalone Java applications released, many are geared toward the Internet. Some are even more narrowly focused as Java development tools. What many of these applications have in common is that they tend to be graphical in nature and are invoked via a wrapper function. These wrapper functions are typically written in shell or batch script. They define the proper Java runtime environment and then invoke the application.

What is missing today is the ability for non-Internet developers to easily use Java in place of C or Perl to write standalone programs. This article will describe an automated method whereby the necessary Java runtime environment can be set up prior to invoking a Java application. With this environment, you can begin to use Java in place of other languages (Korn shell, bash, Perl, etc.) for your normal daily systems administration needs. These small utilities can then be deployed into your system bin directory for all to use.

For the purposes of this article, I assume that you have a good understanding of client/server technologies, object-oriented programming and C/C++ as well as a basic understanding of the Unix operating system and shell programming.

#### Clarification

It should be noted that several third-party Java compilers now support "native" code generation. Namely, the compile process generates an executable image that will run only on the target platform, much like a C compiler generates a binary executable file. To accomplish this, most compiler vendors are imbedding the Java Virtual Machine (JVM) in the binary object. While this solves the problem of a standalone application's runtime environment, these compilers are mostly limited to 95/NT platforms. This will be a suitable solution once these compilers are available on all platforms. Until then we must make do with supplying our own runtime environment.

#### The Goal

Our objective is to create a facility whereby we can write "Unix type" filters in Java. For this discussion, we will be focusing on the filter framework and not the filter applications themselves.

In general, a Unix filter is a process that reads standard input (stdin), or a file, and writes to standard output (stdout), or a file, all the while applying a filter algorithm to the input data. A filter can be something simple like transforming all upper case letters to lower or it can consist of complex mathematical formulations.

A good filter should support the following features:

1. Unix-style optional switches: a dash "-" followed by either a letter or word
2. switches that can have other optional arguments: "-filename foo.txt"
3. Redirection and pipes, optionally
4. systematic error checking on all switch settings.
5. Be cross-platform compatible
6. Once installed in the system bin directory, the ability to be invoked directly by entering the name of the filter at the command prompt.

To achieve these goals we must develop two distinct entities:

1. A standardized Java front-end for processing filter switch settings

2. A standardized script front-end for creating the necessary run-time environment for a given filter

Additionally, a method whereby the wrapper script is married to the application filter in an automated process would help to simplify the make process.

## Java Filter Front-End

For our discussions, the front-end filter process will be designed to read switch settings, detect user input errors and display on-line help. More importantly, it will also invoke the core program logic. Listing 1, Java Filter Front-End, depicts just such a program. This program consists of the following sections:

1. Main function declaration – Necessary for all standalone Java programs.
2. Switch parsing section – Two switches are defined by default: verbose and help
3. On-line help – If the help option is specified, a detailed usage message will be displayed.
4. Error handler – If an illegal or malformed switch is set, the error handler will display a message and terminate.
5. Application code – Once all switches are successfully processed, normal/core execution can be resumed.

With this simple template file, we now have the structure necessary to support most of the desired functionality – namely, five of the six features of a standalone filter. By modifying this structure we can easily support additional switch settings, input and output arguments.

Unfortunately, this template program is not sufficient to solve the sixth requirement: “direct invocation of the target program from the command line.” To achieve this last requirement, we need to resort to a wrapper script.

## Invoking a Standalone Java Program

All Java programs must be invoked via the Java Virtual Machine. Java programs cannot be invoked directly – as is the case for C/C++ programs, batch files, or script files. There is no equivalent of a magic cookie (e.g., #!/usr/java/bin) for a Java program. Trying to execute a Java program by typing the class name will not work, nor will making the class file executable.

To run your program, you must make sure that the Java Virtual Machine can find your class file. This can be done in several ways: setting the CLASSPATH variable, starting the JVM from the same directory as your .class file, etc.

Example 1:

```
> export CLASSPATH =  
/usr/local/Javabin: SCLASSPATH  
> Java /usr/local/Javabin/MyApp
```

Example 2:

```
> cd /usr/local/Javabin  
Java MyApp
```

Either of the above methods will ensure that the Java Virtual Machine can find the target class file.

If your Java program relies on other class definitions then you must specify the path to the other class files too. Again assuming

the current CLASSPATH definition does not contain the necessary path information, you must specify the proper path spec during invocation. For example:

```
> Java -classpath  
/usr/local/Javabin:  
~/Javabin: SCLASSPATH  
MyApp
```

Since our goal is to make your Java application easy for others to use, we must wrap all of the aforementioned constraints into an easy to use package. The end user should not have to type anything more than “MyApp” to invoke your application. The remainder of this article will address just how this can be accomplished.

## Wrapper Script

The wrapper script (Listing 2) depicts a shell script designed to invoke your Java applications for you. The purpose of the wrapper script is to simplify the process of invoking your Java Applications to the point where you only need enter the program name along with any application specific parameters.

Let’s review exactly what this script does:

- Line 1: Magic cookie: Ensures that the script’s contents can be interpreted no matter what the current shell environment is set to.

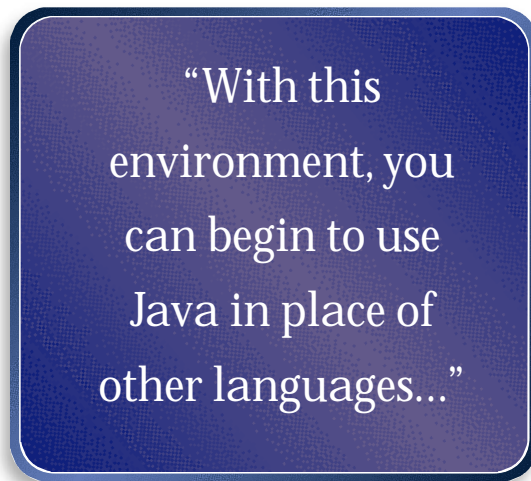
- Line 2: Isolates the target program name from the invocation path. This allows the script to be called relatively, absolutely or via the PATH environment variable. Example:

```
... /myBin/MyApp  
/home/kkranz/myBin/MyApp  
MyApp
```

- Line 3: Separates the invocation path (e.g. ../myBin/) from the full path to the script (/home/kkranz/myBin)
- Line 4: Assumes that the .class file and the script file are named the same: MyApp.ksh, vs. MyApp.Java, vs. and MyApp.class
- Line 6: Computes the absolute path to the script (and presumably to the .class file – assuming that the script and class files are in the same directory)
- Line 8: Switches to the script’s bin directory
- Lines 10-13: Looks for a .class file named after the script file in the class directory. If there is no .class file the script aborts.
- Line 15: Adds the script’s bin directory to the CLASSPATH environment variable. An acceptable alternative would have been to pass this information via the -classpath option to the Java Virtual Machine. Either method is acceptable.
- Line 17: Invokes the Java Virtual Machine with the base name of the script as the first argument and any remaining options as the remainder of the line.

With the use of the wrapper script in Listing 3, you can make it very easy to invoke your Java applications, regardless of what the installation directory is. The only implementation criteria that must be met to use this script are:

1. You must use the same name for the script file as you do for the .class file.
2. The script file and the .class file must reside in the same bin





directory, although you could easily modify the script to allow the script files to reside in one directory and the .class files to reside in a separate class bin directory.

3. The script file must reside in a directory defined by the PATH variable.
4. Other .class files that are accessed at run-time must be in a directory defined by the CLASSPATH environment variable or in a script directory.

## Generating the Wrapper Script

So far, we have described a template Java application program and wrapper script that can be used to write standalone Java programs. The final step is to write a program that can generate a wrapper script for a given standalone Java program.

The Java Compiler script (Jcc, Listing 3) performs the following basic functions:

1. Invokes the Java compiler to create the .class file from your source
2. Invokes Javadoc, to create html based documentation files
3. Creates a wrapper script tailored toward invoking your Java application
4. Creates a manual page for your application

Using the Java Compiler script is not mandatory. Its purpose is to simplify the process whereby your Java program and its associated files are generated. Listing 4, Example Makefile, demonstrates how the Jcc would be used in a Unix-style makefile.

## Conclusion

If you are a systems administrator and are otherwise not involved in the Internet, you do not have to feel left out of the Java learning curve. By using this wrapper script you can begin to use Java in places where you would have traditionally used Perl, C or a shell language. Given Java's rapid growth, it is very likely that it will become a standard feature on most future operating systems. By learning and using Java now, you will help prepare yourself for that time. ☘

### About the Author

Ken Kranz is the Director of Internet Services for Interaxis Corporation, a developer of database-driven Web sites. He can be reached at [kkranz@ebbtide.com](mailto:kkranz@ebbtide.com). This and the expanded source code used in this article can be found at [www.ebbtide.com/UnixFilters](http://www.ebbtide.com/UnixFilters).



[kkranz@ebbtide.com](mailto:kkranz@ebbtide.com)

### Listing 1: Java Filter Front-End.

```
import Java.util.*;

class MyApp {

    public static void main(String args[] ) {

        int verboseLevel = 0 ;
        boolean helpFlag = false;
        String usage = "-v[erbose] -h[elp]";
        int ArgSpec= 1; // minimum number of expected arguments
        int switchCount= 0; // number of switches found
        boolean CLIErrror = false; // user invocation error

        for (int i=0; i < args.length; i++) {
            if (args[i].equals("-verbose") || args[i].equals("-v")) {
                verboseLevel++;
                switchCount++;
            } else if (args[i].equals("-help") || args[i].equals("-h")) {
                helpFlag = true;
                switchCount++;
            } else if (args[i].startsWith("-", 0) ) {
                System.err.println("Illegal switch: " + args[i]);
                CLIErrror = true;
                switchCount++;
            }
        }

        if (helpFlag) {
            System.out.println(" +
            -v[erbose] Increase verbosity level\n" +
            -h[elp] Display this on-line help message\n");
            System.err.println("Usage: " + usage);
            System.exit(0);
        }

        int totalArgs = args.length - switchCount;
        int firstArgOff= switchCount;
        if (ArgSpec != totalArgs) {
            System.err.println(totalArgs + " arguments found " +
            ArgSpec + " expected\n");
        }
    }
}
```

```
System.err.println("Usage: " + usage);
System.exit(1);
} else if (CLIErrror) {
    System.err.println("Invalid/illegal switches encountered");
    System.err.println("Usage: " + usage);
    System.exit(2);
}

/*****
* Place your application code here**/
*****/
}
```

### Listing 2: Wrapper Script

```
1 #!/sbin/ksh # magic cookie
2 BASENAME=$(basename $0) # eg: MyApp
3 ME=$(whence $0 ) # eg: /usr/local/bin/MyApp
4 CLASS=${BASENAME}.class # eg: MyApp.class
5
6 PATH2ME=$(dirname $ME ) # eg: /usr/local/bin
7
8 cd $PATH2ME # go to the bin directory
9
10 if [[ ! -a $CLASS ]]; then
11 echo "SBASENAME: unable to locate target: $CLASS" >&2
12 exit 1
13 fi
14
15 export CLASSPATH=$PATH2ME:$CLASSPATH
16
17 Java $BASENAME $@ # invoke the application
```

### Listing 3: Java Compiler (JCC).

```
#!/usr/bin/ksh

# initialize shell overhead variables
BASENAME=$(basename $0)
USAGE="USAGE: $BASENAME [-c] [-d] [-s] [-o string] [-p CLASSPATH]
[-m] \
[-v [-v]] [-h] filename[.Java]"
(( ARG_COUNT = 1 )) # This number must reflect true argument count
(( OPT_FLAG = 0 )) # Command line mistake flag
(( OPT_COUNT = 0 )) # Number of options on the command line
```





```

(( HELP_FLAG = 0 ))# Default: no help required
(( WARNING = 0 ))# General purpose no fail warning flag
(( JAVAC = 1 )) # call the compiler
(( JAVADOC = 1 ))# call Javadoc
(( SCRIPT= 1 ))# generate the script
(( PATH_OPT= 0 ))# extend the CLASSPATH switch
JAVA_OPTS="" # default Java compile options
PATH_OPT_VAL="" # default extended CLASSPATH

if [[ $VERBOSE_FLAG = "" ]]; then
  (( VERBOSE_FLAG= 0 )) # Default: no verbose
fi

# Parse command line options
while getopts :p:cdso:hv arguments
do
  case $arguments in
    p)(( PATH_OPT = 1 )) # extend the CLASSPATH
      PATH_OPT_VAL=$OPTARG;;
    s)(( SCRIPT= 0 ));; # generate script (& man page)
    c)(( JAVAC = 0 ));; # compile the Java program
    d)(( JAVADOC = 0 ));; # create the Java Documenta-
    tion
    o)JAVA_OPTS=$OPTARG;; # read compile switches
    v)(( VERBOSE_FLAG = VERBOSE_FLAG + 1 ))
      if (( VERBOSE_FLAG > 1 )); then
        vbs=" -verbose "
      fi
      ;;
    h)(( HELP_FLAG = 1 ));; # display help
    \?) echo "Illegal switch: $OPTARG" # flag illegal switch
      (( OPT_FLAG = 1 ));;
    esac
  done
  (( OPT_COUNT = OPTIND - 1 ))
  shift $OPT_COUNT

# check for help
if (( $HELP_FLAG == 1 )); then
  echo " Usage: $USAGE"
  echo " -c Do not compile the target file"
  echo " -d No not create Java Documents"
  echo " -h Display this help message"
  echo " -o stringJavac compiler options"
  echo " -p stringExtend the CLASSPATH"
  echo " -s No not create the front-end script"
  echo " -v Display status information during execution"
  exit 0
fi

# check for illegal switches
if (( $OPT_FLAG == 1 )); then
  echo "$BASENAME: Illegal or invalid switche(s) encountered"
  echo "Usage: $USAGE"
  exit -1
elif (( $# != $ARG_COUNT )); then
  echo "$BASENAME: $# arguments found, at least $ARG_COUNT expect-
  ed."
  echo "Usage: $USAGE"
  exit -1
fi

# reset umask to all files are rw-able
umask 000

dir=$(echo $1 | dirname )
target_basename=$(basename $1)
target=$(echo $target_basename | cut -d\. -f1)
ext=$(echo $target_basename | cut -d\. -f2)

if [[ $ext = $target ]]; then
  # assume .Java
  ext="Java"
elif [[ $ext != "Java" ]]; then
  echo "$BASENAME: invalid extension"
  exit 1
fi

source=${target}.Java # a.k.a MyApp.Java
class=${target}.class # a.k.a MyApp.class
html=${target}.html # a.k.a MyApp.html
script=${target} # a.k.a MyApp
manpage=${target}.1 # a.k.a MyApp.1

# compile program to create target.class from target.Java
if (( JAVAC )); then
  if (( VERBOSE_FLAG )); then
    echo "$BASENAME: Javac $JAVA_OPTS $vbs $source"
  fi
  Javac $JAVA_OPTS $vbs $source
  status=$?
  if (( status != 0 )); then
    echo "$BASENAME: Java compiler error: $status"
    exit $status
  fi
fi

# create the target.html file
if (( JAVADOC )); then
  if (( VERBOSE_FLAG )); then
    echo "$BASENAME: Javadoc $vbs $source"
  fi
  Javadoc $vbs $source
fi

# if the script section is not selected then terminate
if (( ! SCRIPT )); then
  exit 0
fi

# otherwise create the shell script that will invoke the .class file

if (( VERBOSE_FLAG )); then
  echo "$BASENAME: creating $script"
fi

# remove the old "executable" version and man page
rm -f $script $manpage

# insert magic cookie and set debug flags (if required)
ShellPath=$(which ksh)
jar=$?
if (( jar != 0 )); then
  echo $COOKIE
  echo "$BASENAME: Error: Unable to determine magic cookie, abort-
  ing..."
  exit 1
fi

# create the execut script
COOKIE="$ShellPath $StartFlag"
echo "#! $COOKIE ${SHDEBUG}" > $script
echo "BASENAME=${basename $0} " > >
$script
echo "ME=${whence $0} " > >
$script
echo "TARGET=${BASENAME}.class " > >
$script

```



```

echo ""          >> $script
echo "PATH2ME=\$(dirname \"$SME\" ) "          >>
$script
echo ""          >> $script
echo "cd \"$PATH2ME\" "
>> $script
echo ""          >> $script
echo "if [[ ! -a \"$TARGET\" ]]; then "          >>
$script
echo " echo \"\$BASENAME: unable to locate target: \"$TARGET\" " >&2
" >> $script
echo " exit 1 "          >> $script
echo "fi "          >> $script
echo ""          >> $script
echo "export CLASSPATH=\"$PATH2ME:\$CLASSPATH\" " >>
$script
if (( PATH_OPT )); then
    echo "export CLASSPATH=$PATH_OPT_VAL:\$CLASSPATH " >>
$script
fi
echo ""          >> $script
echo ""          >> $script
echo "Java \"$BASENAME\" \"$@" >>
$script

# create a dummy man page file
if (( VERBOSE_FLAG )); then
    echo "$BASENAME: creating $manpage"
fi
echo "" >> $manpage
echo "${target} is a front-end to the Java application

```

```

${target}.class" \
>> $manpage
echo "please see ${target}.html for more details." \
>> $manpage
echo "" >> $manpage
echo "For details regarding Java script wrappers please see $BASE-
NAME" \
>> $manpage
echo "" >> $manpage

# make the script executable
chmod a+rx $script

```

#### Listing 4: Example Makefile.

```

BIN=/usr/local/Javabin
MANBIN=/usr/share/man/cat1/
HTMLBIN=/usr/people/kkranz/www/htdocs/ManPages

```

```

MyApp.class: MyApp.java
    javac -v MyApp

```

```

install:
    cp -p MyApp $(BIN)
    cp -p MyApp.class $(BIN)
    cp -p MyApp.1 $(MANBIN)
    cp -p MyApp.html $(HTMLBIN)

```

# 1/2 Ad



# JavaBean Component Reuse

With industry momentum behind the development of powerful tools and diverse components, the JavaBean component market is growing rapidly. It is important to promote commercial quality JavaBeans components and tools; namely, components that can be used and reused by different users in different tools, can interoperate with other components from other vendors and are robust and functionally complete.

A JavaBean component, or simply a Bean, is a platform-neutral component architecture that extends Java's "Write Once, Run Anywhere"™ capability. With tools like Sun Microsystems' Java Studio, developers can easily create reusable JavaBeans components that can be reassembled to create a wide variety of applications from spreadsheets to chat programs.

Following are some guidelines for Java Beans component developers to help ensure that the maximum level of component reuse is achieved.

## New Beans from Old Beans

JavaBean components are intended to be reused. An end-user uses a Bean within a JavaBeans-compliant tool. Some developers may want to create new JavaBeans components by using an already existing Bean. The JavaBean component architecture provides some mechanisms for this purpose that need to be taken into consideration:

- A new JavaBeans component can be created by customizing a Bean through the manipulation of its properties, or through a Customizer, and then serializing the Bean.
- The functionality of a Bean can be exposed in new ways by providing a new Customizer targeted to a different audience.
- A new Bean can be created by subclassing an existing Bean.

## Keeping the Next User in Mind

When creating Beans, consider how BeanInfo, Bean properties and Customizers will help the next user understand the Bean and how it functions.

## BeanInfo

Each Bean may have a BeanInfo class

which is defined by the creator of the Bean. The BeanInfo class lets the application construction tools uncover information that the creator specifies about the bean, making it possible to hook up bean components through a visual programming paradigm. The creator of the Bean has complete control over how the Bean is presented to programmers inside these application builders.

## Tips for creating BeanInfo:

- Decide whether each feature (method, event or property) should be exposed at all to Bean users, and if so whether it should be marked as hidden or expert. Not everything inherited in implementing a Bean should necessarily be exposed.
- Provide a display name for the feature. Note that, for localization purposes, this name should be extracted from a resource bundle
- Provide quality iconic representations. This makes the Bean easier to use and more recognizable within the builder tool.

## Properties

For some important Bean properties it may be useful for the Bean developer to define events that are fired when that property changes in some specific way. A special example of this is a bound property, but other events may be fired as well. When the value of one Bean's property should be extended to a property on another Bean, that is a bound property. For example, on a slider that is moving between two values, treat the current value as a bound property because turning a slide movement on a slider Bean into a setFoo on a target Bean is a convenient user model.

Not all properties should be made available as a bound property but consider whether or not some useful properties of the Bean are to be made bound. For example, a button might be made bound for its font size and type and background and foreground colors.

Another example of useful bound property for a JavaBeans component is its internationalization locale: Changing the locale is likely to be of interest to other components

that are interacting with this component.

The JavaBean component architecture provides support for both constrained and bound properties. This support is orthogonal so that it is possible to have a property that is constrained but not bound. In practice, Beans that export a property as constrained should also export it as a bound property; then listeners would register for both notifications. This allows a simple two-phase protocol for reacting to changes in one such property. When a property is about to change the listener is invoked through VetoableChangeListener and given an opportunity to veto the proposed change. When the change has actually happened, the (same) listener is invoked through PropertyChangeListener and the listener can react to the change.

## Customizers

A Bean may need to be represented in a specific manner to users or may need to be configured in a certain sequence. Builder tools have no advance knowledge of this. By providing a Customizer the developer can address these situations. The Customizer has full access to the Bean and is normally packaged with the Bean. The Customizer can be a full-fledged JavaBeans component itself. A Customizer can set a private state of the Bean to which the property sheet and methods and events do not have access. Customizers do not override property editors. By using BeanInfo it is possible to associate property editors with Customizers.

A Customizer can encapsulate deep expertise on the use of a component, and a Customizer can be delivered independently of the Bean itself. Some JavaBeans components may even want to include different Customizers to address the needs of different customers' backgrounds, markets and level of sophistication.

Through careful consideration of these Bean characteristics, a JavaBean component developer secures a future in the emerging network software market without losing customers who use proprietary systems. By maximizing component reuse, a developer can also quickly attack new market opportunities and new ways to sell smaller packages of software. ●

## About the Authors

Laurence P.G. Cable and Eduardo Pelegrillo work for Sun Microsystems, Inc.





# InstallShield Express 2

by InstallShield Software Corp.

An amazingly easy and fun way to get your application installed!

by Jason Cohen



ou spend weeks, months, maybe years developing your application. Your testing phase is going well and you're almost ready to begin thinking out your deployment phase. Just when you think it's safe to breathe again, you're faced with the daunting task of deploying your application on the desktops of thousands of users. If your application is for the commercial market, then you have no idea how many users you will need to deploy it to. You need an easy, fail-safe way to get your application installed. Have no fear, InstallShield Express is here!

## What is InstallShield Express?

InstallShield Express 2 is the first completely visual installation development system. Built on the technology of InstallShield, the world's most reliable software for Windows installation, it provides automatic support for more visual development environments than any other program, so developers can quickly and easily deploy the components their applications require. With InstallShield Express 2, developers can create Windows 95/NT logo-compliant installations in less than 10 minutes. No more frustrating days of grappling with Install Builder.

## Setup Checklist

InstallShield Express' main screen is the setup checklist. It is from this screen that you will create every piece of your installation. Each step is logically organized in nine easy steps. I will briefly explain each step.

## Set the Visual Design

This is where you define application information such as your application's name and version number. You can also define certain features of your installation

routine such as your target platform (InstallShield Express supports both 16 and 32 bit platforms).

## Specify Components and Files in Planning your Installation

Before you actually begin using InstallShield Express, it's a good idea to organize all the files you will need on paper and organize them into groups, components and setup types. Groups, components and setup types provide the framework for copying your files. Groups are a collection of files such as readme, help and tutorial files. Groups can then be assigned to components in which the readme, help and tutorial files group would be assigned to the Help Docs component. Setup types are usually typical, compact and custom. If you are offering multiple setup types, you will specify which components are included with each setup type. If you are not offering multiple setup types, you don't even need

InstallShield 2 Express  
 InstallShield Software Corporation  
 900 National Pky., Ste: 125  
 Schaumburg, IL 60173-5108  
 Phone: 800 374-4353 Fax: 847 240-9120  
 Web: [www.installshield.com](http://www.installshield.com)  
 Email: [sales@installshield.com](mailto:sales@installshield.com)  
 Requirements: Windows 95 or NT or higher;  
 10M disk space, VGA monitor or better.  
 Price: \$245.

to worry about components.

As you can see, creating your installation takes careful planning. Believe me, a well thought out plan will save you a lot of time when creating your installation. This portion took the most time for me. I did not prepare ahead of time and spent more time than necessary on this step. Note: You can open Windows Explorer from this section and drag and drop files from explorer to your groups.

## Select User Interface Components

This is the fun part. Here is where you design the GUI portion of your installation. You have 13 different dialog boxes to

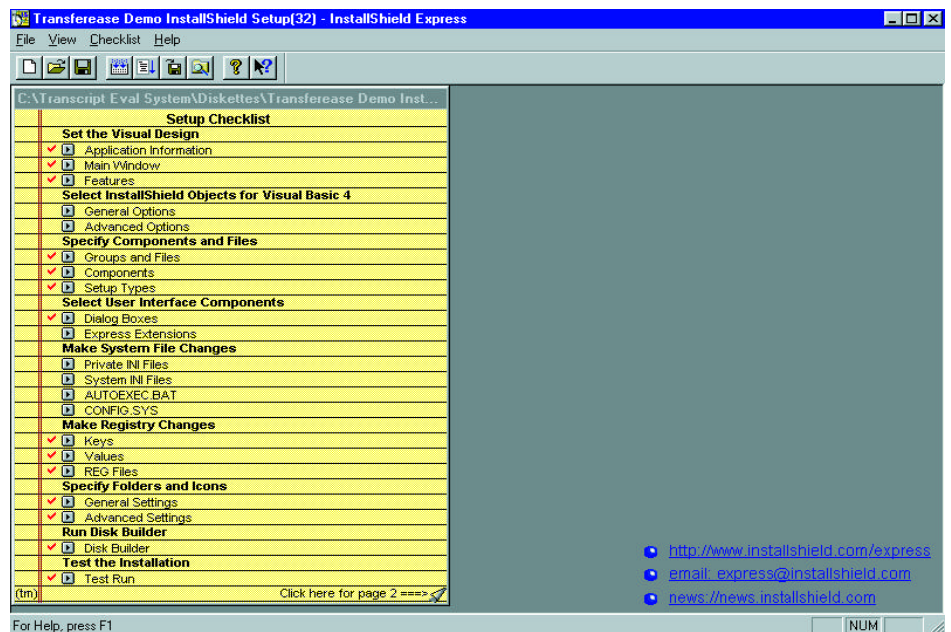


Figure 1: Setup Checklist Screen





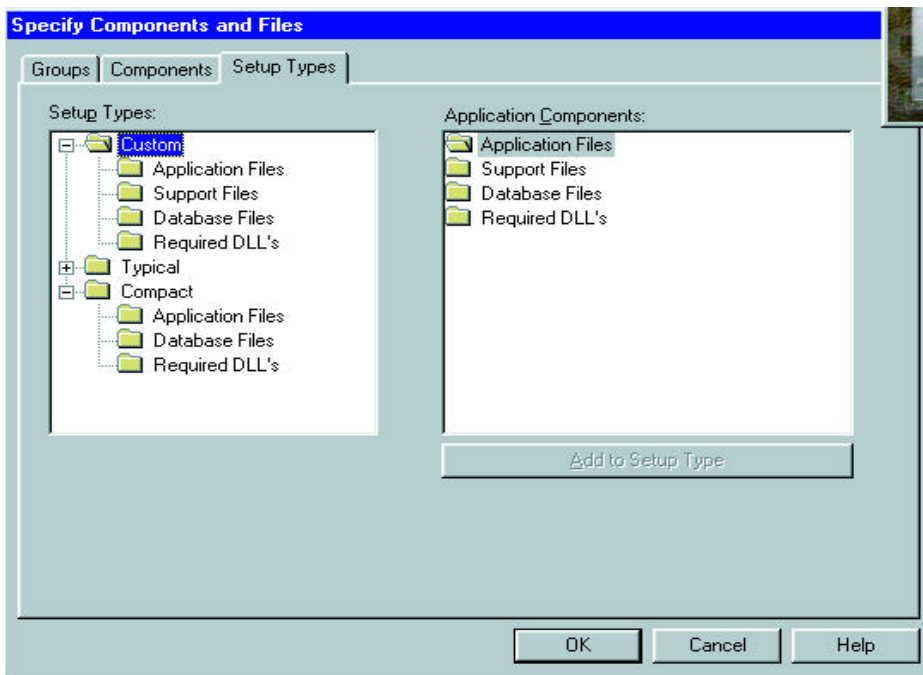


Figure 2: Specify Components and Files

choose from. Some options are a welcome message, software license agreement and a progress indicator. Before there was InstallShield Express, I spent hours with InstallShield 3, which comes with its own scripting language and is the only way to create an installation. With InstallShield Express, all I have to do is point and click on the dialog boxes I want and I'm done. This feature alone is well worth the cost of the product.

### Make System File Changes

If you're building an installation for a 16-bit application, you will need to use this section. Here is where you will define any .INI file settings that need to be modified or updated. You can also make changes or add to your autoexec.bat and config.sys files as well.

### Make Registry Changes

If you're building an installation for a 32-bit application you will need to use this section. Here is where you will define any registry settings that need to be inserted or updated. InstallShield Express automatically makes the necessary registry entries to enable uninstallation.

### Specify Folders and Icons

This dialog allows you to specify the icons you want to place in your application's folder and define the initial size of your application's window. You can also select an alternate working directory, choose an image from a separate resource, specify a shortcut key or place an icon in a specified folder.

### Run Disk Builder

All the hard work has been done and you're ready to create the disk images. With one click of a button InstallShield Express will verify that all of your settings are correct, compress your files and create diskette images. You have many options for disk size such as: 1.44mb, CD-ROM (includes an option for autorun) and 120MB. I have to say the compression routine is excellent. InstallShield Express created one less diskette than my original Install Builder installation.

### Test the Installation

It is a good idea to test your installation routine before giving it to your users or customers. By clicking on this option, InstallShield Express will take you through a dry

run of your installation using all the settings you previously defined. Be careful here; it will actually make your .INI or registry modifications that you specified.

### Create Distribution Media

Once you are satisfied with your installation, you can copy the diskette images to floppy. You have the choice of copying all diskettes or only the ones you choose.

### Summary

I can't say enough about this amazingly easy and fun product. I am now able to sleep at night knowing my customers are not going to be calling me with problems installing my application. There was one bug in creating a 16-bit target platform installation: At the end of the installation it would GPF. This was fixed with InstallShield Express 2.01. I was able to download the upgrade over the Internet. You can download a full working demo of InstallShield Express at <http://www.installshield.com/express> [www.installshield.com/express](http://www.installshield.com/express). I was very pleased with their technical support as well. They were very responsive to my e-mail questions, and their news server on the Internet was also very helpful. I highly recommend this product to anyone who needs to create an installation routine for their application. ☺

#### About the Author

Jason Cohen is Director of Technology at WEBS-PEDITE, Inc, a consulting and development company specializing in Object-Oriented Analysis and Design and developing N-Tier systems using PowerBuilder. Jason is currently assisting an insurance firm re-architect their current two-tier client/server systems into an object based, N-Tier environment. He also teaches PowerBuilder at the University of South Florida.

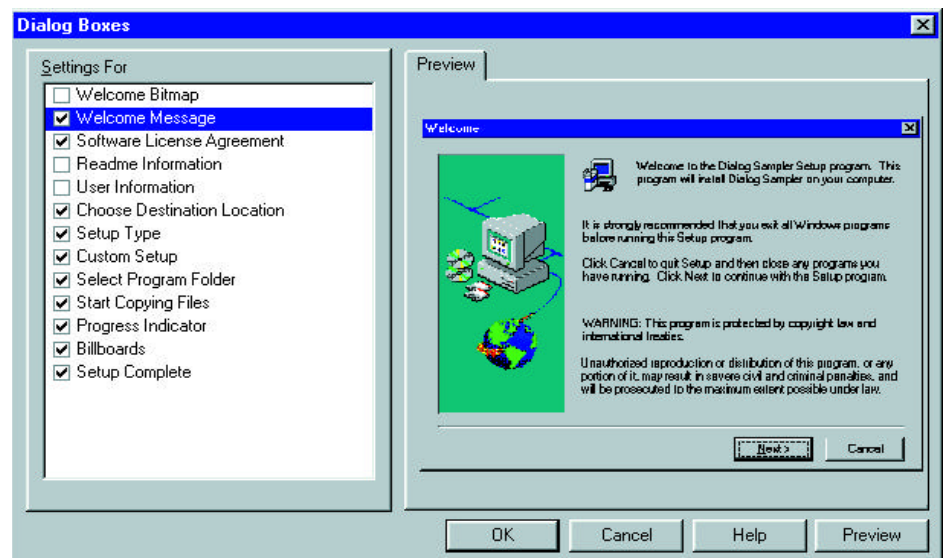


Figure 3: Select User Interface Components



# Implementing Assertions in Java

## Providing greater control and “self-documentation”

by John Hunt & Fred Long

Any software system, whether object-oriented or not, relies on the state of the system being “correct” at certain stages of its execution. To take a very simple example, when a numerical division operation is performed, the divisor must be non-zero. If this is not the case, the system may crash in an unpredictable and uncontrolled manner.

One way of indicating such requirements is to state that the system must be in some state either before or after an operation. Such a statement about the state of a software system is called an assertion.

Assertions often form the basis for software specification. In some systems, the assertions are embedded in the software as annotations or formal comments. However, it can be useful to make the assertions executable so that the correctness of the system is checked at runtime.

A simple technique is to make the code do its own self-testing using a method call and a programming languages exception mechanism. The code checks that the statement passed to it is true, thus allowing it to test that the code is behaving as expected. If the statement is false, the method uses an exception to stop processing, thereby enabling debugging.

This article looks at how assertions can be implemented within the Java programming language [1]. In the remainder of this article, we introduce the concept of assertions, how they can be used and how they can be implemented. In particular, we introduce two types of assertions: one which generates a runtime exception if it fails and one which generates a checked exception (which must be handled by the programmer) if it fails. We then provide a brief analysis of the facilities provided by this approach.

### Assertions and Their Uses

An assertion is a logical statement about

the state of a software system. In formal methods, such as VDM [10], software is specified by making assertions about the pre- and post-conditions of functions and operations. These assertions may be used to formally prove properties of the system, such as consistency. During the development process, the abstract specification is refined into code. The problem with this approach is that the assertions used in the specification are often lost during the refinement process and indeed, it may not be possible to express the assertions in the final implementation language.

Programming languages have been developed which enable assertions to be embedded into the code itself, for example Gypsy [5] or SPARK [3, 2]. Tools available with these language systems enable the assertions to be checked and, in some cases, proved.

Another approach is to introduce assertions into existing programming languages by using formal comments, called annotations, or by pre-processing. The C programming language has been extended with annotations[7] and Ada has been similarly extended into Anna [11]. An annotation pre-processor for C is described in Rosenblum's “A Practical Approach to Programming with Assertions” [13]. In these systems, tool support enables the assertions to be checked against the program code. However, extra-language syntax is required and, in the case of annotations, the assertions are lost when the program is compiled.

It may be better to have assertions as part of the code so that they provide a permanent defensive programming mechanism for detecting faults at runtime [12]. The exception mechanism of a programming language can be used for this [8]. In the following sections we show how this can be done for Java.

### Using Assertions in Java

Assertions can be implemented in Java as part of an assertion package (available at <http://www.aber.ac.uk/~jjh/Java/assertion>). This package can provide assertion classes which take an assertion and throw an exception if the assertion is false. The package could also define its own exceptions, thereby allowing a programmer to catch assertion exceptions. The assertion package we have defined provides two separate assertion classes and two different types of exception:

- Assertion – The class Assertion allows programmers to use assertions within their code.
- CheckedAssertion – The class CheckedAssertion allows programmers to use assertions within their code but forces them to explicitly catch a CheckedAssertionException.
- AssertionError – The AssertionError class extends the RuntimeException class so that programmers do not have to handle the exception raised.
- CheckedAssertionException – The CheckedAssertionException class extends the Exception class so that programmers are forced to handle the exception raised.

The difference between the checked and the non-checked assertions for the developer is that they must explicitly handle any exceptions raised by the CheckedAssertion class, whereas they can choose to ignore the fact that an exception may be raised by the Assertion class.

In Java, a package is an associated group of classes and interfaces [1]. We have defined a new package assertion which holds the four classes listed above. Other classes can then use these four by importing the assertion package. An example of how these classes are used is presented in Listing 1.

This example creates a simple class Example which uses the Assertion class to handle the result of checking the relationship between a and b. The result of running this application is presented in Listing 2.

We could have caught the AssertionError within a try{} catch{} block. This construct allows a piece of code to be wrapped up within a try block. The try block indicates the code which is to be mon-

more for the exceptions listed in the catch expressions. The catch expressions can then be used to indicate what to do when certain classes of exception occur; for example, "resolve the problem" or "generate a warning message", etc. If no catch expressions are included the throws clause must be handled by the method definition. For example:

```
try {
    e.test(1, 2);
    e.test(2, 1);
}
catch (AssertionException e) {
    System.out.println("First parameter larger
than second");
}
```

This would have allowed us to respond in an appropriate manner. Below we examine the implementation of the Assertion and AssertionException classes and their checked counterparts.

### Exceptions in Java

In Java, (almost) everything is an object; therefore, exceptions are objects as well. All exceptions must extend the class Throwable or one of its subclasses [4].

The class Throwable has two subclasses, Error and Exception. Errors are unchecked exceptions. These are exceptions which your methods are not expected to deal with. The compiler therefore does not check that the methods can deal with them. In contrast, most exceptions (but not RuntimeException and its subclasses) below the class Exception are checked exceptions. These exceptions must be dealt with by your methods. The compiler thus checks to see that methods throw only exceptions which they can deal with (or are passed up to other methods to handle). For example, if we wished to raise an ArithmeticException, for a divide by zero error then we would write:

```
throw new ArithmeticException("Division By
Zero");
```

In Java, you are not limited to system-provided exceptions; it is also possible to provide user-defined exceptions. This can be very useful as such an approach can allow the developer to have more control over what happens in particular circumstances. To do so, a developer must subclass the Exception class or one of its subclasses.

### Assertions in Java

#### The Assertion class

The Assertion class (a direct subclass of Object) makes two methods publicly available for handling boolean expressions. These are:

- assert(boolean bool) – Determines the

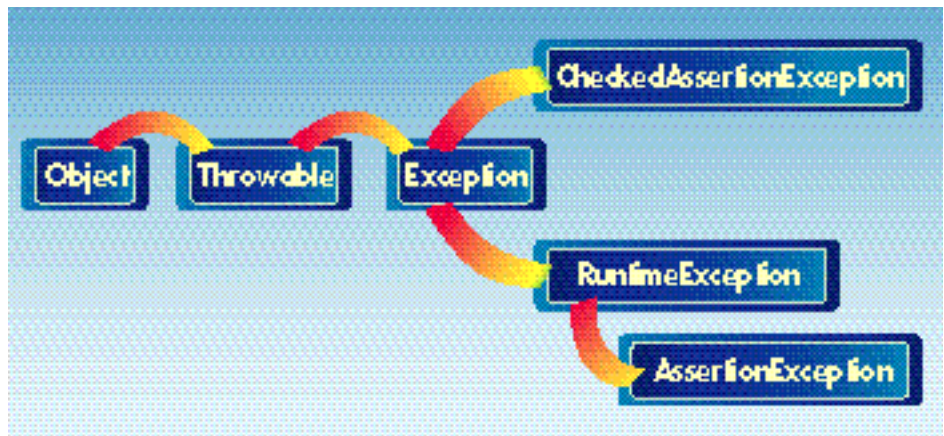


Figure 1 : The assertion exception hierarchy

effect of the boolean result passed to it.

- assert(boolean bools[]) – Determines the effect of the array of expressions passed to it.

The second method is really a convenience method which repeatedly calls the first method on each element in the array.

The implementation of the assert(boolean bool) method is relatively straightforward. The method checks whether the boolean value is false or not. If it is false an exception is raised. The assert(boolean bools []) method simply calls the previous method iteratively (see Listing 3).

#### Switching Assertion Checking Off

In the released version of the system, the assertion checks may waste processing time. They could therefore be replaced by a null version of the method:

```
public static void assert (boolean bool)
throws AssertionException {
    return true;
}
```

As Java dynamically loads the appropriate .class file at runtime, we could distribute the final version of the system with an Assertion.class file based on the above implementation of assert. We would therefore not even need to recompile the system being released.

The overhead of the extra method dispatch is relatively small; of course, if it is still significant we could write a utility to find the senders of the assert method and to comment out the assertion statements.

#### The AssertionException class

The AssertionException class is a subclass of the RuntimeException class as illustrated in Figure 1. The definition of the AssertionException class is provided in Listing 4. Note that it provides two constructors which are used to initialize the string displayed to the user.

CheckedAssertion and CheckedAsse -

#### RuntimeException

These classes are essentially the same as the Assertion and AssertionException classes except that the CheckedAssertion class throws the CheckedAssertionException and the CheckedAssertionException extends the Exception class (as shown in Figure 1). For the developer, the difference between the two types of assertion checking is that any exceptions raised by the CheckedAssertion class must be handled explicitly; they can not simply be propagated out of the program.

The advantage is that the developer using the assertions can decide how much control to give to the user of a component. If they wish, they can force that user to handle the exception and thereby allow them to take charge of any problems which occur. This can result in code which is more resilient to erroneous situations.

For completeness, the two classes are presented in Listings 5 and 6.

### Analysis

#### Design methods Using Assertions

Assertions are particularly useful for testing pre-conditions, post-conditions and class invariants. The Syntropy design method makes extensive use of such assertions [6]. It would therefore be possible to use the assertions identified during the design as the basis of the assertions to place in the Java code.

#### JavaBeans™

The use of assertions could be extremely useful when working with JavaBeans (the Java component model). In such situations developers produce reusable components which will be used by developers in many different ways. By using assertions, developers can ensure that the state of the Bean at any time is correct. If the state is not correct then they can take appropriate action. The assertions can therefore be used as guards against inappropriate states, parameters, etc.

# ADVERTISER INDEX

Advertiser	Page
3D Graphics www.threedgraphics.com	31
Activeverse www.activeverse.com	17
Borland www.Borland.com	43
Bristol Technology www.bristol.com	4
Coriolis www.coriolis.com	47
DCI www.DCIexpo.com/Internet	88
Greenbrier & Russel www.gr.com/java	19
IBM www.software.ibm.com/ad/vaja	69
Installshield www.installshield.com	21
Intuitive www.intuitivesystems.com	75
Iona www.iona.com	23
JavaWorld www.javaworld.com	83
KL Group Inc. www.klg.com	84
Marimba www.marimba.com/download	63
Mecklermedia www.iworld.com	81
Microsoft Corporation www.microsoft.com/VisualTools	91
MindQ www.mindq.com	44
Net Guru www.ngt.com	14
Net Guru www.ngt.com	56
Object Matter www.objectmatter.com	56
OMG www.omg.org	79
O'Reilly www.software.oreilly.com	57

Advertiser	Page
Petronio Technology Group www.petronio.com	59
PreEmptive Solutions, Inc. www.preemptive.com	33
Progress Software www.progress.com	27
ProtoView www.protoview.com	15
RogueWave Software, Inc www.roguewave.com	11
Sales Vision www.salesvision.com	13
Schlumberger, Ltd. www.cyberflex.austin.et.slb.com	25
SofTech Computer Systems www.softech.com	20
Stingray Software Inc. www.stingsoft.com	2
Sun Microsystems www.sun.com	51
SunTest www.suntest.com	67
SuperCede www.asymetrix.com	6
Sybex Books www.sybex.com	35
Symantec cafe.symantec.com	3
SYS-CON Publications www.sys.con.com	77
SYS-CON Publications www.sys.con.com	84
Thought, Inc. www.thought.com	29
Zero G. Software www.zerog.com	34
5 More ads www.zerog.com	34
5 More ads www.zerog.com	34
5 More ads www.zerog.com	34
5 More ads www.zerog.com	34
5 More ads www.zerog.com	34

## Shortcomings

The simple approach to inserting assertions into Java code described here cannot provide the following, more sophisticated types of assertions:

- Assertions that relate output values of methods to their input values: For example, suppose one defined a method which swapped the values of its two parameters.

```
void swap (SomeType x, SomeType y) {
    SomeType z;
    z = x;
    x = y;
    y = z;
}
```

One might want to assert, just before the method returns, that the new value of x is equal to the old value of y and the new value of y is equal to the old value of x. This is not possible with our assertion mechanism. Assertion mechanisms for which this is possible have additional syntax which allows the "before and after" values of variables to be distinguished.

- Assertions which involve quantified expressions: Formal methods, and the annotations of Anna and SPARK, allow expressions to be quantified by "for all", "exists", or "not exists". So, the fact that a natural number N was prime could be asserted by saying that there did not exist numbers P and Q with  $1 < P, Q < N$  and  $P*Q = N$ . Again, this is not possible with our assertion mechanism. Mechanisms which can handle the full power of the predicate logic must have additional syntax and be provided with sophisticated theorem proving tools.

## Conclusions

Recent commentators have suggested that assertions provide a valuable defensive programming technique. One of the nine ways advocated for making code more reli-

able is to "use assertions liberally" [9].

We have demonstrated a simple mechanism for introducing assertions into Java programs. Two classes have been defined: Assertion and CheckedAssertion. The former allows assertions to be used very simply. Any exceptions thrown as a result of the assertions failing may be caught or propagated out of the program. The CheckedAssertion class requires the exceptions to be caught or explicitly listed in a throw clause for the methods involved. This provides greater control and "self documentation" of the assertion mechanism.

Our assertion mechanism is particularly valuable when Java is used in larger, critical applications, where developers are implementing classes for general use and for Java Beans components.

## References

1. K. Arnold and J. Gosling, *The Java Programming Language*, Addison Wesley, ISBN 0-201-63455-4, 1996.
2. J.G.P. Barnes, *High Integrity Ada: The SPARK Approach*, Addison-Wesley, ISBN 0-201-17517-7, 1997.
3. B.A. CarrÉ, J.R. Garnsworthy and D.W.R Marsh, "SPARK: A Safety-Related Ada Subset", in W.J. Taylor (ed.), *Ada in Transition*, IOS Press, Amsterdam, 1992, pp. 31-45.
4. P. Chan and R. Lee, *The Java Class Libraries: An Annotated Reference*, Addison-Wesley, 0-201-69581-2, 1996.
5. R.M. Cohen, *Proving Gypsy Programs*, Ph.D. Thesis, The University of Texas at Austin, 1986.
6. S. Cook and J. Daniels, *Designing Object Oriented Systems: Object-oriented Modeling with Syntropy*, New York: Prentice Hall, 0-13-203860-9, 1994.
7. D.W. Flater, Y. Yesha and E.K. Park, *Extensions to the C Programming Language for Enhanced Fault Detection*, Soft-

ware - practice and experience, vol. 25, no. 6, pp. 617-628, June, 1993.

8. P. Gauthon, *An Assertion Mechanism Based on Exceptions*, in Proc. 4th C++ Tech. Conf., USENIX Association, pp. 245-262, August, 1992.
9. A. Joch, *How Software Doesn't Work*, Byte, pp. 49-60, December, 1995.
10. C.B. Jones, *Systematic Software Development using VDM*, Prentice Hall, ISBN 0-13-880733-7, 1990.
11. D.C. Luckham, F.W. von Henke, B. Krieg-Brückner, and O. Owe, *Anna - A Language for Annotating Ada Programs*, Lecture Notes in Computer Science, vol. 260, Springer-Verlag, 1987.
12. B. Meyer, *Applying Design by Contract*, IEEE Computing, vol. 25, pp. 40-51, October, 1992.
13. D.S. Rosenblum, *A Practical Approach to Programming with Assertions*, IEEE Transactions on Software Engineering, vol. 21, no. 1, pp. 19-31, 1995. ●

## About the Authors

Dr. John Hunt has been working in the object-orientation field since the mid-1980s in both industry and academia. He became interested in Java early in 1995 and has worked with a number of companies (including Sun) with Java. He is particularly interested in developing real world applications using Java and in issues associated with Java performance, integrity and architecture. John can be reached at [jjh@aber.ac.uk](mailto:jjh@aber.ac.uk)

Dr. Fred Long lectures on software engineering and formal methods. He is a visiting scientist at the Software Engineering Institute in Pittsburgh. Recently, he has been researching Java's suitability for the development of critical applications. Fred can be reached at [fwl@aber.ac.uk](mailto:fwl@aber.ac.uk)



### Listing 1.

```
import assertion.*;

public class Example {
    public static void main (String args []) {
        Example e = new Example();
        e.test(1, 2);
        e.test(2, 1);
    }

    public void test(int a, int b) {
        System.out.println("In test with " + a + " and " + b);
        Assertion.assert(a < b);
    }
}
```

### Listing 2.

```
>java Example
In test with 1 and 2
In test with 2 and 1
assertion.AssertionException: Failed assertion
    at assertion.Assertion.assert(Assertion.java:41)
    at Example.test(Example.java:11)
    at Example.main(Example.java:7)
```

### Listing 3.

```
public class Assertion {
    /* Don't make assertion instances */
    Assertion () {};

    public static void assert (boolean bool) throws AssertionException
    {
```

```

    if (!bool)
        throw new AssertionError("Failed assertion");
}

public static void assert (boolean bools []) throws AssertionEx-
ception {
    for (int i = 0; i < bools.length; i++) {
        assert(bools[i]);
    }
}
}

```

#### Listing 4.

```

public class AssertionError extends RuntimeException {
    AssertionError () {
        this("Assertion exception");
    }
    AssertionError (String information) {
        super(information);
    }
}

```

#### Listing 5: CheckedAssertion class.

```

public class CheckedAssertion {
    /* Don't make checked assertion instances */
    CheckedAssertion () {};

    public static void assert (boolean bool)

```

```

throws CheckedAssertionException {
    if (!bool)
        throw new CheckedAssertionException("Failed asser-
tion");
}

public static void assert (boolean bools [])
throws CheckedAssertionException {
    for (int i = 0; i < bools.length; i++) {
        assert(bools[i]);
    }
}
}

```

#### Listing 6: CheckedAssertionException class.

```

public class CheckedAssertionException extends Exception {
    CheckedAssertionException () {
        this("CheckedAssertion exception");
    }
    CheckedAssertionException (String information) {
        super(information);
    }
}

```

1/2 Ad







# CodeBase 6

## by Sequiter Software, Inc.

High performance xBase data access across multiple development environments with a small footprint

by David Jung



Without data from a database, business applications really don't do much. To take your Java experience to the next level, you should have base connectivity. There are still a lot of custom built DOS-based applications that are using xBase database technology, like dBASE, Fox Pro and Clipper. You're trying to convince your company to wake up and smell client/server, that Java can work as a database client and you can keep your data in their existing database format. The solution is using Sequiter Software's CodeBase database management library.

Your manager has decided that your department is going to start using Java as your standard development language. It sounds simple enough, until you start to wonder how you're going to connect Java to your xBase databases to use.

CodeBase is a management program for high-performance data access within a very small memory footprint. It uses Java and others such as C/C++, Visual Basic and Delphi. This means that no matter what your development environment is, you only have to learn one database object model. Its library is designed for single-user, multi-user and client/server environments. Whether you are developing for one user, one department or one enterprise, you are covered with CodeBase. For developers who have a tendency to rely on custom controls, CodeBase includes 16-bit VBX and 32-bit OCX data-aware controls.

### CodeBase Versus JDBC

With the big push to marry Java to databases, the question of JDBC (Java Database Connectivity) comes up. Why should you use CodeBase instead of JDBC? JDBC can be used instead of CodeBase provided that

you are 1) using JDK1.1 or greater and 2) have a JDBC-ODBC driver that can connect to an xBase database. Using CodeBase as your method to connect to your database, it doesn't matter what version of the JDK you're using because CodeBase doesn't need any particular version of Java. You also won't need to invest in a JDBC-ODBC driver because CodeBase is your driver.

In addition to the multiple language support, you also get all the source code for the C/C++ and Java libraries. This protects your investment in CodeBase because it allows you to rebuild the library with any changes you might need for your organization. It also means that you don't have to wait for Sequiter to come up with a new release of the CodeBase engine. You can get the code fixes from Sequiter, apply it to your source code and rebuild it. You won't lose any of your library customization because you're in control of the source code. Lastly, you get a report writer and code utility.

### CodeReporter

CodeBase comes with a report writer and a database administration tool. The report writer, CodeReporter, is a report design tool writer with all the standard features of most report writers, like designing a report by painting the layout on a form, Print to Screen, subtotal and total and so on. The database administration tool, Code Util, allows you to do analysis and maintenance on your database. The two significant features CodeUtil has are the Backup and Restore features. CodeBase keeps track of all changes to your database in a log file. If the database becomes damaged, Code Util, using the log file, can restore all the data. It also has a database analysis tool that creates reports on your database activity.

With all the development language support and utilities, you probably think that your shelf will be laced with manuals.

CodeBase 6 with  
Java Documents  
Sequiter Software, Inc  
PO Box 783 Greenland, NH 03840  
Phone: 403-437-2410  
Fax: 403-436-2999  
Web: [www.sequiter.com](http://www.sequiter.com)  
Email: [info@sequiter.com](mailto:info@sequiter.com)  
Requirements: Windows 95 or NT for CodeBase  
server; JDK or Java compiler  
Price: \$395US

Sequiter does provide you with all the manuals, but they thought ahead and they are in the Adobe Portable Document Format (PDF). This makes finding the information you need very easy because CodeBase installs only PDF files for the languages you are using. If you want to see the manuals, they are all available on the CD-ROM.

### Library

The power of CodeBase is in its library. For Java developers, there are four class libraries available. They are Code 4, Data 4, Field 4 and Error 4. There are also some minor classes that support the main four. The Code 4 class is used to connect your Java application to the database server and lock the data files when needed. The Data 4 class contains the methods you will use to retrieve and store information to your data files as well as database index maintenance. The Field 4 class is used to access and store field values within a row of the database. Finally, the Error 4 class provides you with database exception handling. It is also an abstract class based on the Java Exception class.

So what does this all mean and how does this help you? The classes have been broken down into logical units of work. In order to connect your Java applet to your xBase database, you use the Code 4 class. Once a successful connection is established, you use the Data 4 class to actually work with the database. You can open the database, close it, create an index and so

on. The methods in the Field4 class are used to handle the field manipulation of a given row. To provide exception handling, the methods of the Error4 class are used.

### CodeBase Implementation

CodeBase's Java implementation is based on a simple client-to-server model. CodeBase comes with a server component that will run on your Intel-based non-Unix Web server. The server supports both TCP/IP and IPX/SPX transport protocols, so if you're running a Novell networks you don't have to invest in any TCP/IP Winsock. Your Java applet, the client, will run on your user's Java virtual machine and talk to the CodeBase server. Note: Sequiter will

soon have a server component for Unix-based Web servers; it should be ready in the first half of 1998.

**Editor's Note:** Sequiter Software has informed us that the CodeBase server now supports only TCP/IP. Also, their development plans have changed and they do not know if they will be developing a server component for Unix.

Listing 1 is an example of how you can use CodeBase to access information. It will retrieve the age and date of birth an employee and calculate their age based on their date of birth and compare it with their age in the database.

You will find that CodeBase 6 is a great database management library to help you

migrate your xBase systems to a broader platform. It's fast, doesn't require a lot of resources on either the client or server side, and with royalty-free distribution and source code included in the price of the product, you are getting a lot for your money. ☛

---

#### About the Author

David Jung is a systems engineer specializing in client/server and distributed database development using VB, Access, SQLServer, Oracle and Internet technology. He has also co-authored several Visual Basic books including "Visual Basic 5 Client/Server How-To" (Waite Group Press). David can be reached at Davidj@vb2java.com



Davidj@vb2java.com

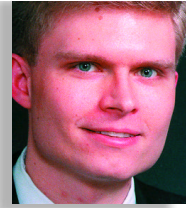
#### Listing 1.

```
import codebase.*;
import java.util.Date;
import java.io.IOException;
import java.net.UnknownHostException;

class CodeBaseEG
{
    public static void main(String args[])
    throws Error4, IOException, UnknownHostException
    {
        Code4client = new Code4();
        client.connect(" ", -1, "user 1", " ");
        Data4db = new Data4(client);
        db.open("PERSONAL");
```

```
Field4double age = new Field4double(db, "AGE");
Field4date bdate = new Field4date(db, "BIRTH_DATE");
db.top();
Date now = new Date();
long milliAge;
int dayAge;
milliAge = now.getTime() - bdate.contents.getTime();
dayAge = (int)((milliAge/1000)/3600)/24;
System.out.println(bdate.contents + " " + now);
System.out.println("Age in days based on birthdate:" + dayAge);
System.out.println("Age based on age field" + age.contents);
db.close();
}
```

# 1/2 Ad



# CorbaBeans

## Adding the power of Beans to CORBA

by Jeff Nelson

Component software has swept through the software industry. Millions of developers now drag and drop components on a form rather than writing source code. JavaBeans™ has provided an elegant component model for the Java development environment and is widely accepted in many development environments. However, JavaBeans lacks any support for distributed computing. This article explores how CORBA and JavaBeans could work together to provide an excellent distributed component model with the added benefit of the cross language interoperability that is a trademark of CORBA. A prototype of such "CorbaBeans" is demonstrated.

### Introduction

The past few years have seen enormous changes in the software development process. Not so long ago, implementing a client/server application meant writing raw data back and forth to a TCP/IP socket. Worse, the toolkits that were available to simplify the task of developing client/server systems were all incompatible with each other. Today, CORBA has turned this around; objects can communicate with other objects directly even if they are located in different software applications, on different hardware or in different languages.

### JavaBeans

Just as revolutionary is the move to component-based software. For the last thirty years, nearly all software has been written by typing long lists of text commands into a computer screen. Only recently has this advanced with the focus on graphical component-based development environments. Reusing components within such a graphical development environment is a simple matter of drag-and-drop. This simple and quick reuse has been quantified

in some studies with the conclusion that component re-use can speed development by as much as a factor of 30.

Components are incredibly important for software development and re-use. It's safe to say that while a few of us like to write code, roughly 75 percent of the development community just drags and drops icons onto a form. Notably, even Java developers like the speed with which a complete application can be crafted using visual components. It's actually sort of a testament to the strength of Java that it has survived this long without a strong offering of component development environments. JavaBeans is what will ensure that Java can compete in that marketplace.

JavaBeans introduced component software development to Java just over a year ago. Today, JavaBeans is widely accepted in many different development environments. Most notably, Borland and Symantec have restructured their entire development environments around JavaBeans. Borland's JBuilder includes more than 150 JavaBeans which can be used and distributed royalty-free. Symantec's latest Visual Café 2.0 editions include more than 100 JavaBeans with similar royalty-free licensing.

JavaBeans has even made it into the browser market. Marc Andreessen recently announced that the Netscape browsers would be rewritten with reusable JavaBeans. For example, one Bean might represent an HTML page while another would represent a newsreader. Developers could use these components to write their own software by simply importing them into their favorite software development environment and then using drag-and-drop to include the JavaBean's functionality into custom applications. In addition, Sun has recently released a JavaBean for its HotJava browser and other Web browser. JavaBeans is available from small companies.

The architecture for JavaBeans is very

Lately, the trade press has been full of the word "component." Ever since one of the leading magazines in the computing industry published an article saying that "components would replace objects," the concept has been hot. But what are components? And do they replace objects, or are objects a mechanism to implement components? This month, Jeff Nelson of DiaLogos returns to the column to explore what is meant by the JavaBeans component composition model and how that relates to the CORBA distributed systems integration platform.

Richard Soley  
Editor, CORBACORNER  
President and Technical Director of the  
Object Management Group, Inc.

elegant. The design of JavaBeans is based on two key architectural insights. First, any Java class can be treated as a trivial JavaBean. Second, a technique called Reflection can be used to examine the capabilities of a JavaBean.

The first insight guarantees that creating JavaBeans is basically just as easy as making any other Java object. In fact, many Java Beans can be used just like any other Java object. The JavaBeans architecture is object-oriented from the ground up. This will be a gigantic relief to people struggling with the ugly APIs defined for other component models.

The second insight has to do with the use of a technology which is new to many of us: Reflection. The powerful capability of Reflection is that it permits objects to examine each other at runtime to discover the capabilities of objects. Java remains type-safe, but Reflection provides many of the same benefits of weak typing in allowing objects to adapt to each other at runtime rather than at compile time.

Components usually have an exposed set of properties, methods and events that interact with other components. Reflection

can be used to examine what a JavaBean can do in the form of these properties, methods and events without requiring the original developer to spend hours writing code to configure the component framework, a tedious task required with less elegant component models. The tool that facilitates all of this in JavaBeans is called, appropriately, the Introspector. It provides excellent default behavior for exposing the capabilities of any component, while allowing the component developer to configure the information which is published to other components if required. However, most developers find this unnecessary for nearly all components.

One limitation that JavaBeans has traditionally had is that it is restricted to a single application. Developers can easily "wire up" some JavaBeans into an application but not into a set of distributed applications. Let's consider one concrete example: Suppose a consulting company wants to track how many hours are spent by each employee on different projects. This would be a perfect application of distributed computing, because you would want potentially hundreds or thousands of desktops to send back schedule summaries to the accounting department. JavaBeans would help you quickly and easily build the front end for each consultant and the back end for the accounting department, but then you are stuck. The JavaBeans developer would need to bring in some other technology to solve this problem. Enter: CORBA.

### CORBA

CORBA has done a wonderful job over the past few years of allowing objects in different programs to communicate with each other. In fact, a year ago I showed one of my managers a distributed application that I had written over the course of several years. The networking code came out to about 5,000 lines to handle several different failure conditions. However, the application still wasn't very robust. The same application, rewritten using CORBA, was just 400 lines and made use of several different fault-tolerant features built into the ORB. Furthermore, the CORBA port opened the door for migrating the front end away from C++ into Java. At the same time, while CORBA is entirely object-oriented, it currently lacks any form of component model. As we stated earlier, the trend in the software industry seems to be away from source code and in the direction of components.

So to summarize, JavaBeans has an excellent component model but no support for distributed computing. CORBA lacks a good component model but is the tool of

choice for distributed computing. These differences in the two technologies beg the question: How do we bring these two wonderful technologies together to make each a more powerful tool?

### CORBABeans

The first goal of bringing the two technologies together would be to make it simple to "wire up" not just a single application with JavaBeans, but open up the possibility of creating whole systems of interacting distributed software. This goal could be realized if the wiring between JavaBeans could extend between applications.

The properties, methods and events of a JavaBean can easily be wrapped into a CORBA server by paralleling many of the key architectural designs of JavaBeans in CORBA. For example, the methods of a JavaBean can be exposed as a member method of a CORBA interface. The properties of a JavaBean can be exposed as a CORBA attribute. The events of a JavaBean can be exposed using either the CORBA Event Service or by following the same naming conventions used in Java to distribute EventObjects.

Since JavaBeans can act as both CORBA servers and CORBA clients, this design provides a way to make JavaBeans itself into a distributed component architecture, perhaps called "DBeans". Each JavaBean, even Beans which have already been written by component developers, could be used in a distributed application and integrate naturally with the network. This is a powerful capability that opens the door to writing powerful distributed applications with a simple drag and drop.

### Prototype


Rather than just discuss it, let's jump into a working example of how CorbaBeans could be implemented. By the way, the source code for this example is available on-line, so you can grab a copy and try it out for yourself! The prototype is composed of five components with each component demonstrating a separate point.

### CORBA Server as JavaBeans

HelloImpl is a JavaBean which is also a complete CORBA server. This Bean demonstrates that existing CORBA servers could be wrapped inside of JavaBeans to provide powerful functionality within existing JavaBean compatible development environments. This Bean can be imported and used in any JavaBeans compatible development tool. Once the Bean is instantiated, it will

list itself in the visigenic Smart Agent and begin accepting remote invocations. Since it is a JavaBean, its methods, properties, and events are also exposed through the normal mechanisms provided by the JavaBeans development environment.

A Bean like this would allow JavaBeans developers to implement server applications with only a couple of drag and drop applications. Suppose you are developing a client/server ordering tracking system for use by departments in a large company. Unfortunately, as always happens, each department might have a slightly different idea about what the backend should do. One group wants it to save data to a relational database. Another department always uses object databases. Yet another group would like all transactions to be approved by a human attendant before processing them. If the server was implemented as a CorbaBean, new functionality could be plugged into the server quickly. The server could be cus-



"JavaBeans is what will ensure that Java can compete in that marketplace."

tomized with only a drag and drop operation within any JavaBeans development environment.

### CORBA Clients as JavaBeans

The second part of the prototype is VHello, a JavaBean which implements a CORBA client. Many software developers might see JavaBeans clients as more useful than the above JavaBeans server. When a JavaBean implements a CORBA client, new client applications can be built with drag and drop operations. Clients often require such customization, since often the clients are the front ends of a distributed system. Front ends, user interfaces in particular, often experience a great deal of change as customers provide feedback about the human factors of an application.

Suppose you are developing a set of applications for the infrastructure of a large company, including order tracking,

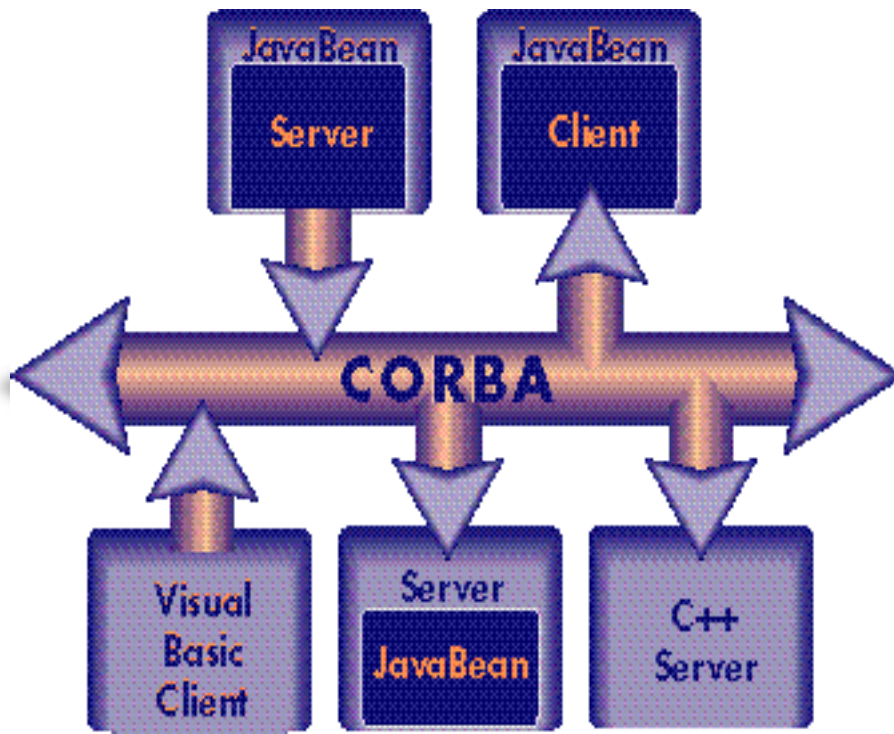


Figure 1: JavaBeans in distributed systems

customer service and accounting systems. All of these individually are very successful client/server applications. However, they could work even better if they could arbitrarily be plugged together to meet the needs of different departments within the organization. Implementing the CORBA clients as JavaBeans means that making a new application into a client of one or more of the order, customer, or accounting systems is a simple matter of drag and drop.

### Building CORBA With JavaBeans

The third component of the prototype is a JavaBean affectionately called CorbaBean. This name is used because this JavaBean automatically converts other JavaBeans into CORBA servers. This JavaBean uses the Introspector on other JavaBeans to discover what properties, methods, and events they support. It then generates the CORBA files required to implement a CORBA server using this JavaBean as the implementation. This would permit component developers to take their existing Beans and create servers for them automatically.

For example, suppose you are running a savvy software development organization. Your group already has written hundreds of JavaBeans to perform all of your mission critical business activities. The problem is that you want your order tracking Bean in one application to work with your account auditing Bean in another applica-

tion. CorbaBean, the third portion of this prototype, demonstrates that you could automatically compile your Beans into CORBA server, and facilitate this kind of application to application communication instantly.

### Component Migration

The fourth component of this prototype is not a JavaBean. Rather, this component is a C++ CORBA server written with VisiBroker for C++. This component interoperates seamlessly with the VHello component, demonstrating the following key point. One of the strengths of CORBA is that it permits different programming languages to work naturally with each other by providing a middleware for their objects to invoke each other. The CorbaBeans

architecture could also make use of this benefit.

When a CORBA server is compiled from a JavaBean, the resulting CorbaBean defines a particular CORBA interface. This interface is not bound to any particular language; in fact, the implementation of this interface in Java is completely hidden by the interface. For all the user of the CorbaBean knows, the Bean may have been implemented in C++, Visual Basic or Perl.

This observation opens the door to the design of a cross-language distributed component environment. While the architectural principles of JavaBeans are simple and elegant, the same architecture can be adopted and implemented in many different languages.

Any CORBA interface is a trivial CorbaBean. However, a CORBA interface can be written in any language, even COBOL. Once the CORBA interface is published and compiled into a CorbaBean, the implementation of the Bean in COBOL is not important. The CorbaBean is still compatible with any JavaBeans development environment.

Conversely, the design principles behind the JavaBeans architecture could be used to implement a component model in other languages, such as C++. For example, the methods, attributes and events of a CORBA interface could be mapped into the methods, properties, and events of a component in C++. This is actually rather straightforward since the CORBA Interface Repository works in much the same way as Reflection in Java.

Due to our use of CORBA as the glue for this component architecture, CorbaBeans are inherently compatible with many different development environments. In particular, tools are already available to work with CORBA interfaces in Visual Basic, Delphi, PowerBuilder and several other development environments. Once a JavaBean is compiled into a CorbaBean, that Bean

## OMG Component RFP

The OMG recognized the need to add a component model to CORBA. To address this need, a Request For Proposals was issued by a consortium of OMG members including Oracle, Sun, IBM and Netscape, for interested parties to submit designs for a component model for CORBA. The initial submissions were not available at the time of this writing, but indications are that submissions will be based on many of the architectural insights of JavaBeans.

could be re-used not just in Java development environments but in nearly any development environment through CORBA's powerful interoperability.

The reverse is also true. CORBA tools can convert COM interfaces into CORBA interface. If CORBA interface could be converted into CorbaBeans, existing Visual Basic OCXes could be re-used within JavaBeans development environments. This sort of interoperability provides an excellent path for component migration to JavaBeans.

### Scripting

Scripting is a common programming technique of writing small, useful programs in only a few lines of code, called scripts. Netscape's JavaScript is a good example of a scripting language. Using JavaScript, simple tasks, such as intelligently animating an image, can be performed by embedding a script directly in an HTML page.

One of the key requirements of a good scripting language is a simple, easy to understand API. Without this, too much time must be spent on the overhead of learning how to write the scripts, destroying the value of writing short scripts to perform useful tasks.

Component developers strive to make components which have a simple, easy to use interface within a visual development

environment. One of the side effects of this is that the API usually makes sense in other contexts as well. For example, when writing an e-mail component, the component would support properties such as to, from and subject, methods to send e-mail and events to indicate when new e-mail has arrived. This simple API is also exactly what a scripting developer would want to have access to when writing scripts for e-mail.

This natural marriage of scripting with components provides benefits to CorbaBeans as well. Some scripting languages allow small scripts to be written with JavaBeans. This article has suggested that CorbaBeans are just JavaBeans with special distributed computing capabilities, so scripting languages which support JavaBeans could also be used with CorbaBeans.

### Summary


JavaBeans and CORBA are both powerful development tools individually. However, the whole is greater than the sum of the parts. CorbaBeans allows JavaBeans to implement and use network services within the same JavaBeans development environments that are currently installed on your computer. CorbaBeans makes writing CORBA application a matter of drag and

drop using those same tools. The cross language capabilities of CORBA mean that CorbaBeans written in one language work with components written in widely different languages. Scripting languages for JavaBeans would naturally work spectacularly well with CorbaBeans, as well.

### Where to Go From Here

The prototype described in this article can be downloaded from <http://www.DistributedObjects.com>

Information on CORBA standards can be found at <http://www.omg.org>

More information on JavaBeans can be found at <http://www.javasoft.com/beans/> 

---

#### About the Author

Jeff Nelson is a consulting architect with Dialogos Incorporated, experts in CORBA and Java Technologies (<http://dialogosweb.com>) and active participants in the Object Management Group. He has 8 years of experience in distributed computing and object technology. He holds a Master's Degree in Applied Mathematics and industry certifications from Sun, Microsoft, and IBM. He can be found on the web at <http://www.distributedobjects.com/> and reached through e-mail at [jnelson@distributedobjects.com](mailto:jnelson@distributedobjects.com).



[jnelson@distributedobjects.com](mailto:jnelson@distributedobjects.com)

1/2 Ad



# POP Goes the Server

## Building a class that can be used to interrogate a POP3 mailbox

by Alan Williamson

In our last column we addressed one of the most commonly asked questions regarding the sending of e-mail from within a Java applet or application. This was achieved using the SMTP protocol, and by the end of the article a fully functional SMTP class was constructed. Before we continue the development of our column project, Informer, I thought it would be a good idea to complete the e-mail service by presenting the other half of the equation: picking up mail from a mailbox. This article will concentrate on building a class that can be used to interrogate a POP3 mailbox.

If you read last month's column, you may have noticed how easy it was to communicate with the SMTP server. We opened up a socket connection to the server, and then passed commands back and forth using ASCII text strings, terminated in lines.

Well, the good news is that the majority of TCP servers operate in exactly the same way; that is, using ASCII text strings to pass commands. This includes, mail servers, news servers and FTP servers. On the face of it, this doesn't seem to be the most efficient way to communicate with servers, and it isn't. But you have to look at the global picture to understand why it is implemented the way it is.

What's the one greatest strength of the Internet? And, what is the one biggest headache for software developers? The same answer can be applied to both questions: variety of platforms. The Internet is a collection of networks all communicating with one other without having to rely on a single operating system. TCP/IP is a way for a UNIX box to talk to an NT box without having to worry about its operating system. However, when asking a developer to write an application for both platforms, questions about byte and word size will be

asked and invariably the answers will be different. One good thing about the computing world is that of all the different standards floating around, nearly every single machine knows of ASCII text. This makes it the ideal data transport for communicating with servers.

### POP3 Server

A POP (Post Office Protocol) server is a piece of software that collects and holds mail until a client comes and requests it. Taking the analogy presented last month one step further, a POP server can be thought of as your mailbox at the bottom of your garden. When someone mails you a letter, it is placed into the post office network where it is sorted and delivered, usually by a mail carrier, to your mailbox. It is not the responsibility of the post office to try to tell you how to read it or open it. It is merely a pigeonhole in which incoming mail can be placed and removed. A POP server is no different. It performs the exact same function: providing a holding area until you pick up your mail, using your e-mail package. How the mail actually gets to your mailbox, is of no real concern to you. All you need to worry about is how to pick up the mail once it has arrived.

As stated earlier the POP server sits and listens for connections on port 110 and, once connected, uses a series of commands to communicate with the client. The complete POP version 3 (sometimes referred to as POP3) protocol can be found in the RFC document 1225. You can read this document on the Web site <http://www.academy97.com>. If you do read this specification, you may be surprised by how small it actually is (only 16 pages) with the majority of them showing examples.

### Connection

Before we can begin to send commands, we must first open a connection to the server, and this can be achieved using the Socket class, as shown in Listing 1.

Just like the SMTP class we developed last month, we create two streams that will make communicating with the server much easier. For example the `BufferedReader` class gives us the method `readLine()`, which can be used to receive data back from the server without having to worry about carriage returns, etc.

If a connection is successful, the first line we will receive back is a welcome message which may read something like:

```
+OK mail.n-ary.com POP server ready; Sun, 21
Sep 1997 12: 42: 00 GMT
```

Whereas the SMTP host used status codes to indicate success or failure, the POP server uses +OK and -ERR to flag status values. This makes the receiving function at the client much easier to code.

The next thing we have to do is log on. We pass the username and password of the mailbox addressee and, if it is successful, +OK will be sent back as shown in Listing 2.

We have coded two special functions, `sendMessage()` and `rxdLine(...)`, that perform some additional error checking. The source for both of these can be found in Listing 3.

### Mail Headers

The POP3 server only supports a very small set of commands. We have commands to retrieve just the mail headers, the number of mail messages waiting, the complete mail message and commands to delete mail messages from the server.

Messages are queued at the server one after another, generally in the order in which they arrived. The POP server gives each message an ID starting at 1. Note, that this ID is only unique for that session. If you delete message 3, for instance, and then log off and log back on again, then the message that was ID 4, now has the ID of 3. So be very careful when coding e-mail clients.

One of the most basic things that is very useful, is the ability to see what mail is waiting for us without having to download the



complete message. This is especially useful for clients on a dial-up connection, where they can choose whether to accept or not accept delivery of attachments. To achieve this we must first know the number of e-mails waiting for us, so that we can then count forward to that number. We do this by sending the command STAT to the server. The server will then return the result:

+OK 4 3210

The first part of the response is the status flag, and the second is the number of messages waiting in collection. The last value is the total size of all the messages, expressed in octets. This is particularly useful, as one can calculate a very accurate progress monitor for the download status.

Having received the number of messages, we then use the TOP command to download just the message headers. The TOP accepts two parameters: the message ID and a block number. For example, we can return the header for message 2, by issuing the command 'TOP 2 1' to the server. The server will then return with a mail header for that message, terminated with a single ".".

As you can see, if you refer to the getMailHdr() function in the Listing 3, the header consists of many different fields, of which Subject, Date and From are of partic-

ular interest to us. We package this information up into a class WMail and return.

Using this functionality, our e-mail client can display a list of all incoming mail without having to download each one.

### Mail Transfer

The next step, is to actually download the complete e-mail. This is performed using the RETR command with the mail ID. For example, to receive the e-mail, complete with header and body, the following command would be issued to the server: 'RETR 2.'

Again, if you refer to Listing 3, and look at the getMail(...) function, you will see this occurring. Notice how we determine the difference between the main body and the mail header. The mail header is transmitted first, with a blank line indicating the end of the header and the beginning of the main body. For ease of use, we place the body into a Vector, with each line of text as a new entry.

### Mail Deletion

Deleting the message on the server, as you may have already guessed, is a simple matter of issuing the DELE command, complete with the message ID. The server will then return with a success or failure depending on whether or not the message exists.

### Summary

That's it! That is the basic functionality required to implement a POP3 client. Listing 3 shows a couple of extra commands, but the core has been demonstrated in the main article body.

To see this class in action, I converted it into a Java Servlet, which allows you to check, read and delete your e-mail from a Web browser. Visit [http://www.academy97.com/email\\_login.html](http://www.academy97.com/email_login.html).

So far, we have implemented a very basic front-end to Informer, complete with database access to a list of contacts. We also have the functionality to send e-mails and, with this article, the ability to receive e-mails as well.

Next month, we will be adding more features to Informer ☺

### About the Author

Alan Williamson is on the Board of Directors at N-ARY Limited, a UK based Java software company, specializing in Java/JDBC/Servlets. He has recently completed his second book, which focuses purely on Java Servlets, while his first book looks at using Java/JDBC/Servlets to provide a very efficient database solution. He can be reached at [alan@n-ary.com](mailto:alan@n-ary.com) (<http://www.n-ary.com>) and he welcomes all suggestions and comments.



[alan@n-ary.com](mailto:alan@n-ary.com)

#### Listing 1.

```
BufferedReader    In;
DataOutputStream  Out;
Socket            OutPort;
OutPort = new Socket( "mail.n-ary.com", 110 );
Out        = new DataOutputStream( OutPort.getOutputStream() );
In         = new BufferedReader( new InputStreamReader(OutPort.getInputStream()) );
```

#### Listing 2.

```
sendMessage( "USER " + userName );
if ( rxdLine("+OK") == null ) throw new Exception( "Invalid username" );
sendMessage( "PASS " + passWord );
if ( rxdLine("+OK") == null ) throw new Exception( "Invalid password" );
```

#### Listing 3.

```
import java.io.*;
import java.util.*;
import java.net.*;
import java.text.*;

public class popMail extends Object
{
    BufferedReader    In;
    DataOutputStream  Out;
    Socket            OutPort;
    String host;
    String userName;
    String passWord;

    public popMail(String _Host, String _userName, String _passWord)
    {
        host        = _Host;
        userName    = _userName;
```

```
        passWord    = _passWord;
    }

    public boolean open() throws Exception
    {
        return open( 110 );
    }

    public boolean open( int _port ) throws Exception
    {
        try{
            OutPort = new Socket( host, _port );
            Out      = new DataOutputStream( OutPort.getOutputStream() );
            In       = new BufferedReader( new InputStreamReader(OutPort.getInputStream()) );

            if ( rxdLine("+OK") == null ) throw new Exception( "Server did not say hello." );

            /*- Send Username
            sendMessage( "USER " + userName );
            if ( rxdLine("+OK") == null ) throw new Exception( "Invalid username" );

            sendMessage( "PASS " + passWord );
            if ( rxdLine("+OK") == null ) throw new Exception( "Invalid password" );

            return true;
        }catch( IOException E ){
            throw new Exception( "Server not Ready" );
        }

        public void close()
        {
            sendMessage( "QUIT" );
            try{
                OutPort.close();
            }catch( IOException E){
            }

            private void sendMessage( String _M )
```

```

{
    try{
        Out.writeBytes( _M + "\r\n" );
    }catch(IOException E){}
}
private String rxdLine( String _command )
{
    try{
        String i = In.readLine();
        if ( i.indexOf(_command)==0 )
            return i;
    }catch(IOException E){}
    return null;
}

private String rxdLine()
{
    try{
        return In.readLine();
    }catch(IOException E){}
    return null;
}

public int mailNo()
{
    String LineIn;
    sendMessage( "STAT" );
    if ((LineIn=rxdLine("+OK"))==null ) return 0;
    int x = LineIn.indexOf(" ");
    LineIn = LineIn.substring(x+1, LineIn.indexOf(" ", x+1));
    try{
        return Integer.parseInt( LineIn );
    }catch (Exception E){}

    return 0;
}

public int[] getMailIDs()
{
    String LineIn;
    sendMessage( "LIST" );
    if ((LineIn=rxdLine("+OK"))==null ) return null;

    int x = LineIn.indexOf(" ");
    LineIn = LineIn.substring(x+1, LineIn.indexOf(" ", x+1));
    int No=0;
    try{
        No = Integer.parseInt( LineIn );
    }catch(Exception E){
        No = 0;
    }

    if (No == 0 )
        return null;

    int list[] = new int[No];
    int a=0;
    while ( (LineIn=rxdLine())!=null )
    {
        if ( LineIn.indexOf(".")!=-1) break;
        list[a++] = Integer.parseInt(
LineIn.substring(0, LineIn.indexOf(" ")));
    }
    return list;
}

public WMail getMailHdr( int _id )
{
    WMail M = new WMail();
    M.ID = _id;
    String LineIn;
    sendMessage( "TOP " + _id + " 1" );
    while ( (LineIn=rxdLine())!=null )
    {
        if ( LineIn.indexOf(".")==0)
            break;
        else if ( LineIn.indexOf("Date:") != -1 )
            M.Date = new Date(LineIn.substring(LineIn.index-
Of(":")+2, LineIn.length()).getTime());
        else if ( LineIn.indexOf("Subject:") != -1 )
            M.Subject = LineIn.substring( LineIn.index-
Of(":")+2, LineIn.length() );
        else if ( LineIn.indexOf("From:") != -1 )
            M.From = LineIn.substring( LineIn.indexOf(":")+2,
LineIn.length() );
        else if ( LineIn.indexOf("+OK") != -1 )
            {
                try{
                    int q = LineIn.indexOf("(")+1;
                    M.Size = Integer.parseInt( LineIn.substring(
q, LineIn.indexOf(" ", q+1) ) );
                }catch(Exception E){}
            }
    }
    return M;
}

public WMail getMail( int _id )
{
    WMail M = new WMail();
    M.ID = _id;
    String LineIn;
    sendMessage( "RETR " + _id );
    boolean bHdr = true;
    while ( (LineIn=rxdLine())!=null )
    {
        if ( LineIn.indexOf(".")==0 && LineIn.length()==1)
            break;
        else if ( LineIn.indexOf("Date:") != -1 )
            M.Date = new Date(LineIn.substring(LineIn.index-
Of(":")+2, LineIn.length()).getTime());
        else if ( LineIn.indexOf("Subject:") != -1 )
            M.Subject = LineIn.substring( LineIn.index-
Of(":")+2, LineIn.length() );
        else if ( LineIn.indexOf("From:") != -1 )
            M.From = LineIn.substring( LineIn.indexOf(":")+2,
LineIn.length() );
        else if ( LineIn.indexOf("+OK") != -1 )
            {
                try{
                    int q = LineIn.indexOf("(")+1;
                    M.Size = Integer.parseInt( LineIn.substring(
q, LineIn.indexOf(" ", q+1) ) );
                }catch(Exception E){}
            }
        else if ( LineIn.length() == 0 && bHdr)
            {
                bHdr = false;
                M.vBody = new Vector(10, 5);
            }
        else if ( !bHdr )
            {
                if ( LineIn.indexOf(".") == 0 )
                    LineIn = LineIn.substring( 1,
LineIn.length() );
                M.vBody.addElement( LineIn );
            }
    }
    return M;
}

public void deleteMail( int _id )
{
    sendMessage( "DELE " + _id );
    rxdLine("");
}
}

class WMail extends Object
{
    public int ID=0;
    public int Size=0;
    public String Subject="";
    public String From="";
    public long Date=0;
    public Vector vBody=null;
}

```

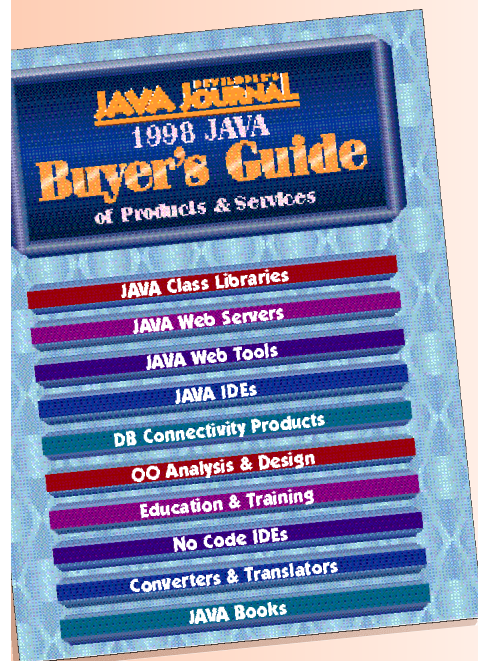


ere's a Special  
review of what  
is to come!

sampling of the  
1998 Java Buyer's  
Guide to Products  
& Services

rought to you  
courtesy of

DEVELOPER'S  
JAVA JOURNAL



Coming  
Soon!

# 1998 JAVA Buyer's Guide to Products & Services

## JAVA CLASS LIBRARIES:

### scsGrid Control

scsGrid Control is a grid control that provides many of the common features found in grids for displaying and entering data. It is designed for usability and fast uploads and execution speeds on the client machine.

SoFTech Computer Systems, Inc.  
814 696-3715  
[www.scscompany.com](http://www.scscompany.com)

### ProtoView WinJ

The WinJ Component Library is a rich collection of ProtoView JavaBeans™. While geared towards the professional developer, these feature-rich controls are simple enough for even the most basic Java programmer.

ProtoView Development Corporation  
Russell Frith  
609 655-5000 FAX: 609 655-5353  
[www.protoview.com](http://www.protoview.com)

## DATABASE CONNECTIVITY:

### STOR/QM Data Storage

STOR/QM, an innovative Report Mining solution goes beyond traditional COLD systems, offering record level access to reports and transaction information in them, even vast information archives. STOR/QM can export results with databases and spreadsheets.

People Net, Inc.  
Bart Huitema  
818 783-0606 FAX: 818 783-4999  
[storqm.peoplenet.net](http://storqm.peoplenet.net)

### SouthWare Innovation

Financial and management information software with two dozen applications. The SouthWare Excellence Series™ runs on more than six hundred hardware/operating system combinations including DOS, UNIX, Windows and most popular networks.

SouthWare Innovations, Inc.  
Rick Hulsey  
334 821-1108 FAX: 334 821-1146  
[www.southware.com](http://www.southware.com)

### SOLID Server

SOLID Server is an embeddable database server for mission critical applications. Designed for robust unattended operation, it is ideally suited for wide deployment in large numbers. SOLID Server is powerful, scalable and standards-compliant. SOLID Server powers Web sites, Internet appliances and e-commerce solutions. As one of the pioneers in the industry, SOLID released its 100% Pure Java™ JDBC driver which is key to delivering dynamic content on the Web and in Java applications.

Solid Information Technology Ltd.

Marten Mickos  
FAX: +358 9 477 473 90  
[www.solidtech.com](http://www.solidtech.com)

### UIM/Orbix™

UIM/Orbix™ is a graphical C++ development solution providing a CORBA™ standard framework for distributed objects. It is built upon the industry leading UIM/X®, Orbix® and Orb/Enable™ tools. Key features include: built-in C++ interpreter, graphical editors for constructing IIOP-enabled clients and servers, graphical/non-graphical object support, integration of UI and distributed object code including combined event loops, IDL development and CORBA conformant capabilities and application deployment.

Black & White Software, Inc.  
Susan Karas  
408 369-7400 FAX: 408 369-7406  
<http://www.blackwhite.com>

## JAVA EDUCATION & TRAINING:

### Java™ Training

Become a Java™ expert with one of our Java™ Training courses. Topics include: The Java Language, Visual J++, Apps and Applets, use with HTML, use with ActiveX/COM and ActiveX Controls, AWT, Threads, JDBC, J/Direct, AFC, Security, Java Beans™.

Mastermind  
Ken Ramirez  
717 688-9927 FAX: 717 688-9568  
<http://www.mastermind.com>



submit your product for inclusion in the 1998 JAVA BUYER'S GUIDE

Click Here



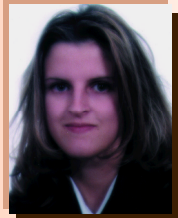
at [JavaBuyersGuide.com](http://JavaBuyersGuide.com)

Each year Java Developer's Journal publishes our annual Java Buyer's Guide Products & Services, the most comprehensive sourcebook and database of companies providing Java-related products and services available. Over 100,000 purchasers of Java products and services will see your company's listing in the 1998 Java Buyer's Guide.

To have your company's product or service included in the 1998 Java Buyer's Guide to Products & Services, simply fill out our online listing submission form.

Point your Web browser to [www.javabuyersguide.com](http://www.javabuyersguide.com) to have your product or service listed today!

## Contact



Christy Wrightington  
at 914-735-7300  
or [Christyw@sys-con.com](mailto:Christyw@sys-con.com).

### Java™ and Object Design

Java™ and Object Design is a five day course targeted at software professionals moving to Java. This course enables participants to master both the use of Java, and how to think and design in terms of objects.

Below are a few of the objectives students will achieve:

- Use container classes in the Java Developer Kit
- Design using polymorphism, inheritance and encapsulation
- Use and customize the Java development tools

ObjectSpace, Inc.  
Rita Lundeen  
972 726-4100 FAX: 972 726-4200  
[www.objectspace.com](http://www.objectspace.com)

### JAVA WEB TOOLS:

#### Hybrid Shopping Cart

Hybrid Shopping Cart is a Java™ applet that provides a complete user interface package for internet shopping Web sites.

A "hybrid" is defined as an offspring of two varieties - an item produced by the blending of two diverse traditions. That's exactly what the Hybrid Shopping Cart offers; a blending of the best features from our CGI and Java shopping products.

Eastland Data Systems  
Ed Zarrella  
301 831-9681  
[www.eastland.com/hybrid.html](http://www.eastland.com/hybrid.html)

#### CUChat

CUChat is a Java™ Visual Chat client/server platform. Clients upon arriving at a Web site enabled with CUChat will be able to communicate/chat visually (represented as Avatars) with one another. There is no need to install a plug-in since CUChat is in 100% Java.

NetDIVE Corporation  
Sue Berna  
415 474-3756 FAX: 415 474-3756  
[www.netdive.com](http://www.netdive.com)

#### PeopleNet HelpDesk

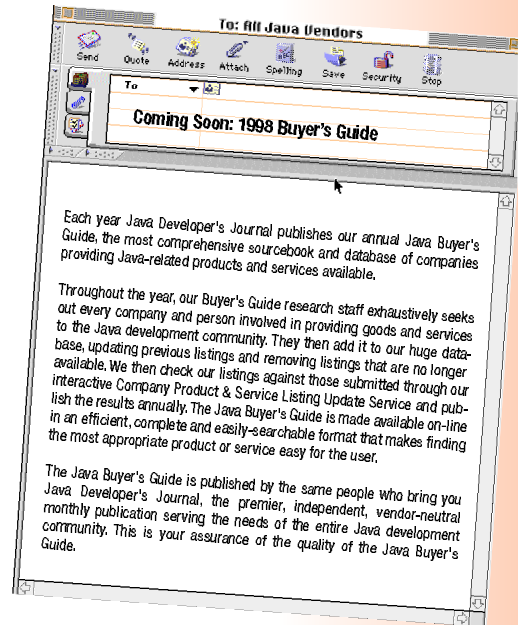
Help Desk Manager, an integrated multi-user Windows package, provides complete front-line support and management. Our WWW interface gives any browser complete access to Help Desk Manager's impressive features.

PeopleNet, Inc.  
Bart Huitema  
818 783-0606 FAX: 818 783-4999  
[helpdesk.peoplenet.net](http://helpdesk.peoplenet.net)

### CyberCAT

CyberCAT Java Catalog is a Java applet for deploying Internet catalog and product information easily and inexpensively without programming. Simply add your product information, images, advertisements and user interface graphics. CyberCAT Java Catalog takes care of the rest.

Betacorp  
John J. Bekto  
905 564-2424 FAX: 905 564-6655  
[www.betacorp.com](http://www.betacorp.com)



### Interrogate

The Interrogate search agent integrates with Web browsers, to provide a simple user interface for searching files -- either within Web sites or on disk or CD-ROM. It operates entirely on the local computer, removing any necessity for server-side scripting when searching pages on the World Wide Web.

Bigfoot Partners, L.P.  
Chris Lacey  
[cslacey@bigfoot.com](mailto:cslacey@bigfoot.com)

### JAVA DEVELOPMENT TOOLS:

#### Visual Enabler 2.0

Visual Enabler 2.0 is a version control toolset which delivers automated build support for Visual Basic, Visual C++ and Visual J++ projects (including .dsp and .dsw). Visual Enabler 2.0 has tight integration to IDEs, automatic team notification and the ability to register interest at a file level.

Softlab Enabling Technologies  
Nita Holcombe  
770 668-8811 FAX: 770 668-8712  
[www.softlabna.com](http://www.softlabna.com)





# VisiBroker 3.0

## by Visigenic Software, Inc.

CORBA 2.0 compliant ORB; a good choice for your distributed object applications

by Khanderao Kand



After making the important decision to embrace the Common Object Request Broker (CORBA), the next important decision that you will have to make is choosing an Object

Request Broker (ORB) vendor suitable for your needs. You have many options to select from: Digital's ObjectBroker, Expert-Soft's PowerBroker, HP's Orb Plus, IBM's SOM, Iona's Orbix, Sun's Joe, VisiBroker from Visigenic, etc.

Six months back, when I first worked on VisiBroker2.5, I found it a good ORB but without many supporting services. In July 1997, Visigenic announced version 3.0 of VisiBroker which made it more promising than before.

### Instant CORBA

In 1991, Object Management Group (OMG) specified CORBA 1.0, an Object Management Architecture (OMA), for cross-platform distributed computing. Using ORB, a client object can transparently invoke a method on a server object sitting at any location on a network and developed in any language. CORBA enables this functionality with the help of its Interface Definition language (IDL), ORB bus, and services and facilities extending it. The key feature of CORBA is its IDL. All that the client object knows about the remote object, is its interface. CORBA provides location, language and platform transparency. The OMG left the implementation details of the specifications up to the vendors. Different vendors implemented the architecture differently. Subsequently, in December 1994 OMG specified CORBA 2.0 with Internet-Inter ORB protocol (IIOP) for the interoperability across the vendors.

### Installation

I could easily install Visibroker 3.0 for Java (VBJ) on both Windows 95 and Solaris 2.5. Visigenic separately ships other CORBA components like Naming service, Event Service and VisiBroker Manager. So I repeated the installation procedure for them too. Configuration was also easy. VisiBroker is a completely dynamic system and easy to manage. VisiBroker 3.0 comes with good documentation, including an installation guide, programmer's guide and reference manual.

### Development

VBJ comes with a bunch of examples covering almost all important features of the product. It is easy to start with them. You can go through the following steps for developing CORBA based application:

1. Identify remote objects required for the application. Write an interface for the object using IDL.
2. Then compile the interface using the IDL compiler, namely idl2java. The compiler creates an interface class, stub class, skeleton classes, and supporting classes per interface. The stub, `_st_<interface name>.java`, is a proxy of the remote object for the client. The proxy implementation invokes operations on the remote objects. It does this by forming a method, marshaling the data, and sending the request over ORB. The skeleton object provides the server side functionality and connects the server object to ORB. VBJ 3.0 generates skeleton classes named `<interface name>ImplBase.java`. It also generates two more classes, namely `<interface name>Helper.java` and `<interface name>Holder.java`. The helper class defines utility functions related to binding, etc. The holder class provides a holder for passing parameters.
3. The next step is to implement the client

and server interfaces. The implementation generally involves initialization of ORB, binding and registering implementation.

4. Now, compile all the generated code and application. Start the VBJ specific daemon called OSAgent. You can run the server and client using a program called vbj. The vbj program actually invokes Java Virtual Machine (JVM) and runs your code. It also enables it to configure the threads and connection information. You can even plug an object debugger into it. This debugger is a GUI based interceptor of the IIOP messaging. The debugger is a very useful tool for development.

### Important Components and Their Functions

VBJ supports a full language mapping and complete implementation of mandatory CORBA features, like pseudo-interfaces, ORB, BOA, and IR. Visigenic ORB fully implements the IIOP protocol. VBJ 3.0 comes with naming and event services fully implemented in Java.

VBJ offers idl2java compiler to automatically generate Java code from IDL. The

### VisiBroker 3.0 for Java

Visigenic Software, Inc.

951 Mariner's Island Boulevard

San Mateo, CA 04404 USA

Phone: 800-632-2864 or 415-312-7197

Fax: 415-312-7195

Web: [www.visigenic.com](http://www.visigenic.com)

Email: [info@visigenic.com](mailto:info@visigenic.com)

Requirements: 10-14MB HDD, JDK1.1 Solaris 2.5,

HP-UX, IBM AIX 4.1, SGI, Digital Windows NT

3.51/4.0/5.0 or Windows 95

Price: \$1995 per Developer on Windows

\$2995 per Developer on UNIX

(Excluding Naming and Event Services. Call for Deployment)





idlj2java provides IDL2.0 compliant IDL to Java language binding. VBJ also contains a fully implemented Interface Repository(IR). The IR is an on-line database of meta information (object interface) about ORB interfaces. It is essential for Dynamic Invocation Interface(DII). In such cases, the clients learn about an unknown object's interface at runtime. You need to create IR using VBJ's irep program. VBJ also provides API for IR. You can populate the IR by running idl2ir on the IDL file. VBJ comes with a proprietary, distributed object location and a directory service called an OSAgent (Smart Agent). Multiple OSAgents running in the same network locate each other and automatically partition the name spaces. Thus, it offers replication and load balancing. When client application invokes the bind method on an object or when object implementations register their objects, it sends a UDP broadcast and the first OSagent to respond is used. Once the OSAgent has been located, point to point UDP connection is established. If one OSAgent is failed, the other OSAgent takes over the client and object tasks. Thus, VBJ offers a fault tolerance. Smart Agents can use VBJ's Object Activation Daemon(OAD) to launch instances of a server process on demand. You can register or unregister the implementations using OAD's API or command-line utilities. VBJ also provides an interface to OSagent called a Location Service. It is an extension of the CORBA specification to provide general purpose facilities for locating object instances. The location services query the smart agents about the instances. This information can be useful for load-balancing. This service is available via the Agent API. The API has two parts: one for query related and another for registering and de-registering the triggers. These triggers are, in fact, notifications by which clients of the Location Service can be notified of the changes of the availability of instances. Typically you can use it as a callback mechanism.

### Putting CORBA clients on Internet

The major market of Java is Internet applications. There are two hurdles for making CORBA applets available on the Web. One is the security restrictions from the browser. Browsers restrict applets to connect only to the domain from which they have been downloaded. Another problem comes from firewall restrictions. Firewalls deny the services, like IIOp, that are not explicitly allowed. VBJ resolves these problems with an approach called HTTP tunneling. In this mechanism, IIOp calls are wrapped in an HTTP envelope for passing through the firewalls. Then it sends all the requests to a daemon (Gatekeeper). The

daemon then forwards the request to the host nominated in the object reference. VBJ's Gatekeeper runs on a Web server and enables the client program to make calls to objects that do not reside on the Web server and to receive callbacks. One of the advantages of this approach is that you can still use the same firewall that you are currently using.

In CORBA, the objects are identified by Interoperable Object References (IOR). IORs are generally clumsy. VBJ provides URL naming Service-Interface which allows you to associate a URL with an object's IOR (Interoperable Object Reference). Unlike the gatekeeper, URL naming enables you to associate with a transient object's IOR and connect to it. It also skips smart agents for locating the objects. Thus, you can locate the object by URL like: `http://myhost:15000/URLNaming/myObject.ior`

### Advanced Features

1. Security: VBJ supports IIOp over Secure Socket Layer (SSL) protocol. The VBJ's SSL pack provides privacy (through encryption), integrity (through checksum), and client-server authentication.
2. Scalability and Performance: VisiBroker supports scalability and performance through multi-threading. You can select either thread per client session model or thread pool model. Controlled Connection Management multiplexes and recycles all requests from the same client.
3. Improved ORB API: The interceptor between, requests and enables developers to monitor or modify the request. You can use smart stub API to improve the performance by caching frequently used and non-volatile results and values.
4. VisiBroker Manager: This is a GUI product that is available separately but is very useful. It provides Interface Repository Browser, Location Service Browser, and Implementation Repository Browser. It also provides a performance monitor!

### Comments on Compliance, Performance, Interoperability and Evaluation of CORBA

I did some testing related to performance and fault tolerance but I was far from calling it a benchmark. In fact I could not find any proper bench marking suites for the same. I came across some work in that direction done by some research scholars[1]. OMG does not certify the compliance on its own. The Open Group(TOG)[2] has announced VSorb, test suites, to certify the functionality conformance with CORBA specifications. DSTC's[3] project, CORBAnet, proves interoperability between different ORBs including Visigenic's one. Patricia Carando[4] gave some

tips for the selecting and evaluating the ORBs. Visigenic satisfies almost all of them.

### Some Expectations

Now that I like the core product, when I want to make something meaningful out of it, I need more and more services. Currently, Visigenic provides the naming and event services. Visigenic does not provide services like Integrated Transaction Support, connectivity to ODBMS/RDBMS, messaging services or Trader services. Fortunately, Visigenic is working on these and they have released a plan for making them available. I would also like to see more support for tools for application development from Visigenic, either directly or through third parties. Currently Visigenic supports Java and C++ language binding. Considering the legacy systems, it would be great if Visigenic also could provide bindings for languages like smalltalk, Cobol, and Visual Basic. Probably some third parties can take up the work. Now, last but not least! The main competitor to CORBA is a COM model proposed by Microsoft. Visigenic currently supports client side support to OLE/ActiveX. In the future, we may also need a server side bridge to OLE.


### Conclusion

VisiBroker 3.0 is a major breakthrough for Visigenic. The new functionality galvanized the ORB. The scalable, fault tolerant architecture of ORB, along with its ease in using Java development support and GUI based management tool, makes it a good choice. The existing service, along with the implementation plan of various other services, makes the picture more promising. Visigenic's technology has been licensed by Netscape, Oracle, Novell, Sybase, Borland and others. This makes the technology more suitable from a business perspective.

**Editor's Note: Visibroker 3.0 also includes a java2IIOp compiler and a java2Idle compiler.**

Smalltalk is available now; the server-side bridge to OLE is in development.

### References

1. <http://www.cs.wustl.edu/~schmidt/new.html#corba>
2. <http://www.rdg.opengroup.org/public/vsorb/rndatash.htm>
3. <http://www.corba.net>
4. <http://members.aol.com/carando/OOP-SLAWorkshop.html> 

---

#### About the Author

Khanderao Kand is a Principal Consultant at TekEdge Corp., CA. As a consultant he has been working with companies like Informix and Cisco. He can be reached at [kand@nestors-world.com](mailto:kand@nestors-world.com).



[kkand@cisco.com](mailto:kkand@cisco.com)





# What's All The Fuss About?

## The Hype, The Use, The Fiction, The Facts

by Clive Boustred

What's all the fuss about Java? Like coffee addicts, people are running about with Java in their veins, hyped up in this new euphoria known generically as 'Java'. Unfortunately, many, if not most, don't really know or understand what it is all about, but they are enjoying the high anyway. The great majority seem to have garnered some idea that Java represents the uprising underdog, here to fight the battle against the great evil empire of the all-present Microsoft juggernaut. So what, after all, is this Java phenomena?

Having its origins at Sun Microsystems in a language called Oak, Java has come to represent an entire environment. Java technology now covers many domains, from a well structured third generation language (3GL) to an operating system environment and distributed object Middleware infrastructure.

### The Java Phenomena

'Java' encompasses a number of fundamental technologies:

- Java Language – The Object-oriented third generation Java language (Java 3GL).
- Java Applets – 'Half-compiled' architecture-neutral bytecode objects one creates with the Java language
- Java Scripting – A scripting environment similar to Visual Basic Script
- Java Virtual Machine - Provides the interpreter for the Bytecode Java Applets and a protected operating environment for Java applications
- JDBC and JSQL – Java Database Connectivity, an ODBC lookalike which provides database connectivity for Java applications and embedded SQL capabilities
- JNI – Java Native Interface, which provides an 'Interface Definition Language' allowing other non-Java languages to expose Java-based interfaces
- Java RMI – The Java Distributed Object

Framework called Java Remote Method Invocation (RMI)

- JavaBeans™ – The Java Object Assembly environment called JavaBeans.

But what do these components really deliver?

### The Java Language: What, Another 3GL?

Surely, in a day and age where any product that needs to be successful requires the word 'visual' imbedded somewhere in it, we don't need another third generation language! One of the primary problems with software development is the necessary evil of programming. The concept of forcing a

“Object-orientation  
allows us  
to separate  
the interface  
from the  
implementation.”

domain expert (someone who really knows what they are doing and want to accomplish) to get on their knees and bow down to some programmer guru to try to get the message across to them of what exactly they want the computer to do is a little backward. In other words, the problem with 3GL environments is that the programmer is not the domain expert. This causes all types of problems. Ever try relaying a simple message from individual to individual in

a party? It is amazing how easily the message gets jumbled up somewhere along the way. The same happens in software development.

3GLs mandate a syntactically rich, or should we infer poor, environment. That is, you need to write a highly structured document (program) where everything must be just right. If you don't put the brackets in the right place and get all your verbs, synonyms, language, etc. absolutely in order, it won't work.

The human brain is not naturally a syntactical beast, we don't naturally think in terms of 'structured language'. The human mind thinks primarily in terms of pictures; we are generally visual animals, living in a visual world. So the marketing folk at most of the computer companies have grasped this basic concept and consequently, any new product must have the word 'visual' in it! It took some time to drum into the heads of many computer people that a Graphical User Interface (GUI) with Windows Icons Mice and Pointers (WIMPs) was important. In fact, you still manage to find a few individuals in the mainframe world and programming community who still believe that a command prompt is all you need. So, getting back to our Java issue, what's the fuss; who needs another 3GL?

Java is, quite simply, a better 3GL than the other 3GLs like C, C++, SmallTalk, Ada, Pascal and the like. Java gets rid of the imbecilic concept of pointers and pointer arithmetic which would leave you scratching your head as you tried to decipher what someone was trying to do with a program. Java also incorporated the novel concept of garbage collection which, like other concepts embedded in the Java programming model, is wisely borrowed from other languages. Garbage collection removes the problems of things like memory leaks. Memory leaks? Sounds like a problem we all have when trying to remember someone's name. Well memory leaks are a bit like that; users don't always identify them on PCs since they typically turn the PC off or hit the all encompassing Ctl+Alt+Del which effectively blows out all the memory and lets you start afresh. Memory leaks typically crop up on servers and workstations where the system is left running days and



weeks, if not years, on end. It all comes from those blasted pointers they gave us in languages like C and C++, where programmers were responsible for allocating and de-allocating memory segments. Programmers, like any normal individuals, tend to forget to de-allocate the memory. The net result is a system whose memory is filled with stale data, effectively reducing available memory living space. By the way, if you are still mired in the world of C and C++ programming, there are great tools, such as Purify, which will examine your code and tell you when you have a leaky program.

This brings us to another one of the great advantages of the Java 3GL, one of security. The old pointer problem again comes up as a key instigator of problems. Smart hackers can use pointers to capture information off specific segments of memory in other applications, using, providing a default path to secure data away the ability to use pointers and you get rid of many of these problems.

Pointers can, however, be useful to the wizard who wants to play tricks and get optimal performance out of some piece of code. A bit like the guy who is really nifty with his abacus or slide rule.

Other elements Java dumps that have caused problems in the C++ world are things like no operator overloading (Java does have method overloading) and no multiple inheritance (only single inheritance). Also, Java has no templates, extensive automatic coercions, etc. Java provides true arrays instead of having to use pointers. It also enforces things such as static typing, which is enforced by the compiler, producing clean code. While the dynamic loading feature of Java eliminates any explicit link phase, method lookup occurs on-demand at run-time, which enables applications to dynamically exploit the latest revisions or replace objects on the fly during runtime.

The other key element of Java, which helps address the limitations of 3GLs and brings us into the 'visual' world, is JavaBeans, which we will get to shortly. There is, however, one more aspect to the Java lan-

guage we must examine, that of bytecode.

### Bytecode?

Sounds a bit like a sandwich. Well, bytecode is also not new; it has also been around for some time in other languages. Bytecode provides a halfway architecture-neutral state to fully-compiled code. The key feature to bytecode is that it does not force unique features of a specific hardware or operating system on a platform. Java is an interpreted environment but the interpreter does not have to start from scratch in order to interpret the program. It interprets the precompiled architecture-neutral Java bytecode. This allows you to run the exact same Java program on our PC, on all of your current Unix systems and on your mainframe or superputer.

The Java bytecode environment is faster than other interpreted languages like SmallTalk, but slower than C or C++. You can, however, fully compile Java, making the platform independent getting closer to C and C++ performance.

### Internet is Good

Another advantage of the Java language environment is that it comes with an extensive library of routines for TCP/IP protocols like HTTP and FTP, which makes it handy for Internet applications.

### So, What's an Object?

Isn't my coffee mug an object? Well yes, and herein lies the problem of explaining the real advantages of object-orientation: everything is an object. But in the computer or programming world, the honest explanation of an object lies in the following statement: 'Object-orientation allows us to separate the interface from the implementation'.

So what does this mean? The separation of the interface means that we take out the interface part of the program, the instructions like start, stop, print, etc. and only expose these to the outside world. The actual implementation details of how we internally structure and create this piece of code is hidden. You need not know any details of how the object is written; you only need to know what interfaces are

made available and now you can use them. It's a bit like your car. You do not need to know the details and workings of the combustible engine to drive a car; all you need to know is how to operate the interfaces: the steering wheel, the ignition key, the brakes, etc. Other programmers can then take your object and exploit its interfaces without having to know anything about the internals of the program. Thus, objects provide the fundamental building block technology for assembling comprehensive applications from object building blocks.

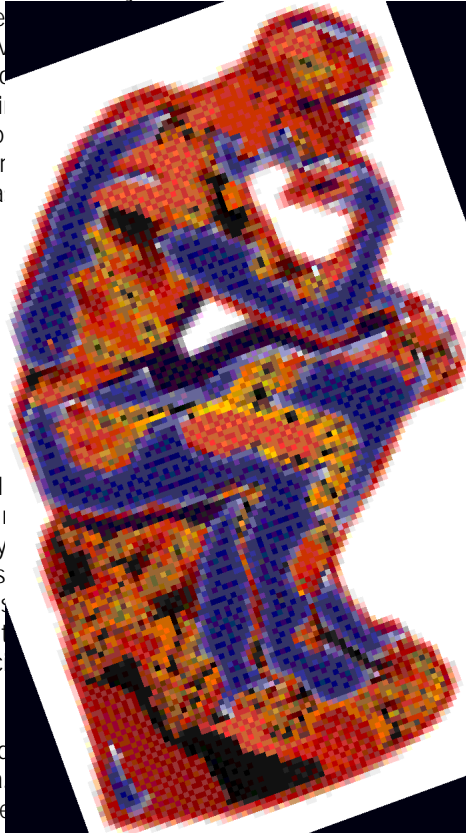
There are other aspects to object-orientation that are important, such as inheritance, polymorphism, encapsulation, classes, etc. But we will not get into the details here. The important thing to recognize is that Java provides us with an object oriented environment which allows us to write pieces of code, expose only the interfaces to the code and let other people use the object (program) without having to know the gory details of the code itself. In the Java world, we call these objects Java Applets.

### JavaScript

The other element to the Java environment is an object-based scripting language that you can embed in HTML, the Hyper Text Markup Language we use to describe Web pages. This scripting language allows us to make calls to initiate Java Applets and provides us some other features to make the HTML environment a little richer. JavaScript is based on the Java language and is used to call Java applets from HTML and 'glue' the Java applets to each other. You use it for scripting of events and actions such as startups, exits and user mouse clicks. It also provides constructs for database connectivity. Of course, there are many other scripting alternatives as well, such as Visual Basic Script from Microsoft and Dynamic HTML, etc.

### The Java Virtual Machine

This is, again, an old concept borrowed from the mainframe world. Mainframers have been providing hosted Virtual Machine environments for years. On mainframes, we would host other 'operating systems' which provided 'protected worlds' in which programs could operate without affecting the whole system. So if you had the concept of Ctrl+Alt+Del on the mainframe, it would not bring the whole system to a grinding halt, but would only kill the offending segment. It's a little like the kill command in Unix, or the new popup you get on NT or Windows 95 which allows you to get rid of badly behaved programs without killing anything else. Except, like the mainframe concept, the Java Virtual





machine provides a completely separate world in which programs can execute – a virtual ‘sandbox’ in which programs can play without causing problems in the rest of the house.

One of the primary responsibilities of the Java VM is to interpret incoming bytecode into the native machine language. This allows the exact same Java Applet to be shared across many different hardware implementations without recompiling the application. It also imposes a natural performance disadvantage when compared to fully compiled environments where the object is already compiled into the binary code of the specific platform. The first time you import an object into the Java VM, it gets interpreted and is slow; however, future calls made to that object while in memory can be nearly as fast as C or C++ calls.

The Java Virtual Machine does take away some key elements that are typically considered fundamental to most computer operating environments, particularly that of an integral and permanent file system. The Java VM has no persistent storage system; only a cache is provided in which objects are temporarily stored and then cleared out when they are not referenced regularly. Applications in the Java VM are restricted from making local OS file system I/O calls amongst other operating system specific calls. While this provides some security advantages, it is probably the most fundamental problem with the Java VM in that it requires you to reload all applications into the Java VM each time you start up. Imagine installing your word processor, spreadsheet and database each time you turn on your computer!

However, the advantage of the Java VM “sandbox effect” is that it allows you to run applications in a protected environment. This is particularly important if you are downloading programs off the Internet and you are not necessarily sure if they contain viruses or if they will behave properly.

## JDBC

Java DataBase Connectivity, an ODBC lookalike, provides database connectivity for Java applications. JDBC itself is a low-level interface which incorporates SQL commands into the Java language. An API is provided for database drivers which makes the actual connection and translation to the specific database. A JDBC to ODBC bridge allows the exploitation of standard ODBC drivers.

ODBC is really the market leader in this segment by such a long margin that JDBC does not really pose any threat to ODBC itself. Microsoft, however, is interested in moving ODBC to their new OLE DB environ-

ment. OLE DB provides a common data access method to any OLE-compliant object, whether it is a database, a word processor, or a spreadsheet document.

## JSQL

JSQL, like JDBC, provides database connectivity to the Java language. JDBC, however, provides a more concise environment than JSQL while it allows more efficient static analysis and type checking.

## Java Native Interface (JNI)

JNI provides a standard native programming interface that allows the integration of applications and libraries that are not written in Java. A key aspect of JNI is that it provides binary compatibility across all Java VM implementations on a specific platform.

Java provides us with an object-oriented environment which allows us to write pieces of code, expose only the interfaces to the code and let other people use the object (program) without having to know the gory details of the code itself.

JNI is independent of the implementation of the underlying Java VM, enabling programmers to write applications or libraries in different languages that should run on any standard-compliant Java VM on that platform.

JNI, of course, competes with Microsoft’s MIDL (Microsoft Interface Definition Language) and CORBA IDL (Common Object Request Broker Architecture Interface Definition language). Both MIDL and CORBA IDL provide language bindings, which allow method invocations (instructions) to be passed from one object to another object regardless of what programming language the object is written in.

## Java RMI (Remote Method Invocation)

The Java Object Request Broker (ORB), called Java RMI, provides a distributed object framework for Java applications. That is, it allows developers to create applications that run transparently over multiple different computers, harnessing the horsepower of multiple parallel computers and exploiting the distributed demographics of networked systems.

Java RMI came out of Sun Microsystems at a time when the rest of the industry, including Sun but excepting Microsoft, had provided full endorsed support for the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA). Java RMI is not CORBA and is not CORBA-compliant. Sun, an advocate of Open Systems, essentially gave the rest of the Open Systems industry an ‘I’ll do it my way’ and produced Java RMI, even though Sun does have a CORBA-compliant ORB called NEO (by the way, JOE is Sun’s NEO implementation that provides Java language mapping). Sun’s defense might be to raise the question as to which successful products in our market have been designed by a committee. To fend off the understandable fury of the CORBA camp, Sun has added the CORBA Internet Inter ORB Protocol (IIOP) support as a low level means (not Java’s standard) of communication from Java RMI to other IIOP ORB implementations. Java RMI’s standard Inter ORB protocol is the Java Remote Method Protocol (JRMP).

Java RMI’s limitation is that it basically supports only the Java language. It is, however, relatively simple for a Java developer to implement distributed Java applications using Java RMI. It is much easier than implementing distributed applications using a CORBA ORB or Microsoft DCOM. For this reason alone, Java RMI might offer a long-term solution to distributed computing.

We will not go into the details of how Java RMI works here, but recommend that any person who is interested should definitely learn more about ORBs as they will form the foundation of future computing. The three key ORB standards are OMG’s CORBA, Microsoft’s DCOM and Sun’s Java RMI.

## JavaBeans

The Java object assembly environment, JavaBeans, embodies the heart and soul of future development environments. This building block approach will enable domain experts to develop advanced applications with naturally cognizant development tools. JavaBeans provides a standard way for implementing development environments where components (beans/objects/Applets) can be visually





integrated.

There is already an impressive array of JavaBeans-compliant tools on the market, such as Sybase's PowerJ™, Borland's JBuilder™, IBM's Visual Age™ for Java, SunSoft's Java Workshop™ and Symantec's Visual Café, amongst others.

If we were to rate any of the Java elements in order of importance, JavaBeans comes out on top. Providing a standard Computer Aided Software Engineering (CASE) environment with standard underlying communications infrastructure, the Java RMI ORB. It is what the industry desperately needs to enter into the next generation of computer technologies, where domain experts will at last be able to create the software they need out of standard building blocks. An interesting aspect to consider is that the next CORBA interface standard, CORBA 3.0, may well embrace the JavaBeans model.

### The Java Coup

Now that we know a little of what Java is all about, we can start to understand the excitement. But what are the underlying currents? Why, for example, is Sun Microsystems trying to sue Microsoft Corporation? What are the 'war room' strategies that are driving the computer systems juggernauts?

No one denies the fact that Microsoft has captured the heart of the industry, seriously threatening companies like IBM and Sun. Can the Java phenomenon undermine the Microsoft phenomenon?

Windows controls the desktop market and NT is rapidly making inroads into controlling the server market. How can any vendor stop this massive landslide?

Browsers were the first to recently raise a challenge to Microsoft's desktop dominance. They even had Microsoft worried for a while. The Internet explosion caught most of the vendors unaware, including Microsoft. When Marc Anderson's Mosaic brought a new user-friendly 'desktop' interface to the largest network in the world, the Internet and browsers took off. The browsers were definitely not from Microsoft, providing a significant new alternative desktop interface to Microsoft Windows.

Recognizing the serious threat browsers posed to their control of the desktop, on December 7th 1995, Microsoft announced a massive reorganization and a host of new products, all focused entirely around the Internet. Microsoft spearheaded these efforts with a free Microsoft-based browser, the Internet Explorer. They subsequently came up with an even smarter strategy to combat the browser threat – by simply incorporating browser technology into the heart of Windows itself, Microsoft will make

the need for a separate browser mute.

But Java goes beyond the browser. Java provides its own Virtual Machine (VM) environment, an operating system in itself. The Java VM can run independent of Microsoft's Windows, providing an alternative 'platform' to Windows. It is this alternative that Microsoft's competitors are trying so desperately to protect and propagate. And this is where Microsoft is so eager to come up with a strategy to divert any massive migration to the Java VM.

The Java VM, however, was designed primarily around the concept of providing a protected environment in which you can run potentially suspect code on your primary operating system without affecting the whole system. This is why the Java VM

The Java VM can run independent of Microsoft's Windows, providing an alternative 'platform' to Windows.

limits access to file system I/O and is also the very reason why the Java VM in its current state will probably never manage to usurp Microsoft Windows. It is necessary to download an application over the network each time you restart. As we mentioned before, imagine having to re-install your word processor and spreadsheet each time you turned on your computer. The other problem is that when compared to standard Windows applications, bytecode is inherently slower than compiled code, handicapping true Java applications. You can, however, fully compile Java code to a specific machine's binary environment and obtain nearly the same performance as C or C++. However, this eliminates Java's machine independence.

Microsoft's current strategy to dilute any Java VM threat is to expose Java Applets as full COM (Component Object Model) objects. Turning Java objects into COM objects (the same thing as ActiveX objects), makes Java objects the same as any other language's objects in the Windows environment. Microsoft is thus 'opening up'

the Java VM to be an integral part of the Windows Operating System environment. Developers are consequently likely to exploit Microsoft Windows-specific features even when developing Java applications, particularly by exploiting the ability to utilize local persistent storage on Windows and thus effectively locking the application to the Windows platform. The Java VM then becomes a mute point, since it is inherently incorporated in the Windows operating system.

Microsoft is obviously not playing along with Sun's Java strategy and 'licensing model' and this is why Sun is jumping up and down frantically because Microsoft won't play properly. To be perfectly honest, why should Microsoft have to play along with Sun? Are vendors not free to innovate as they choose? If Sun wants to compete with Microsoft, they have to do it in the 'Open'. In the Open Systems world, the vendor with the most cost-effective product (and the best marketing) wins.

So far, Microsoft has managed to dilute any attempts to dethrone their dominance on the desktop. Microsoft's strategy reflects that of a wise commander who has been through, and won, many battles. What the other vendors seem to be unaware of, in their frantic attempt to unseat Microsoft's dominance of the desktop, is that while they are deploying all their forces against the desktop, probably futilely, Microsoft has already entered the back door of the ivory tower and is taking over their server market.

However, the real battle lies in the adaptation of vendors' distributed object Interface Definition Languages and underlying ORB infrastructure DCOM vs. CORBA vs. Java RMI. In particular, the features and functionality of CASE tools that allow applications to be built around any of the distributed environments will determine the industry direction. Whoever manages to control the hearts and minds of developers and manages to get developers to build applications around their distributed object model wins! ●

#### About the Author

Clive Boustred is the Chief Technical Strategist for Advanced Technical Strategy, Inc. (www.strategize.com). He has held senior positions and consulting engagements with numerous corporations such as Microsoft, Sun Microsystems, General Electric and Teknekron, where he was involved in the development of Corporate Computerization Strategies. Clive specializes in Distributed Systems, CASE and Networking and is responsible for providing the vision for many highly advanced systems and some of the largest Distributed Object implementations to date. He can be reached at [clive@strategize.com](mailto:clive@strategize.com)



[clive@strategize.com](mailto:clive@strategize.com)

# Java World

New

Ads

house

bread



# Welcome to the JDJ FORUM

www.JavaDevelopersJournal.com

by Ashok Ramachandran

The Web is full of resources for Java. There are applets, code samples and FAQs everywhere and several free tutorials. But when it comes to getting a simple question answered, there are not many options:

- Usenet: You can always post your questions on the Usenet news group, which now boasts 13 subgroups under the comp.lang.java main group. However, the 'free-for-all' kind of Usenet culture can overwhelm some people, especially if you are a newbie.
- Mailing Lists: There are also some mailing lists to which you can subscribe and get the messages in the discussion sent to you in the form of e-mail. This is great, I guess, if you have the ability to correlate the questions and answers into some kind of a thread!
- Java Developer Connection (JDC): Sun Microsystems Inc. provides a variety of resources for developers at JDC, which includes a Developer Forum. But this is a scheduled chat type of forum, rather than a threaded discussion forum.
- Mentor: You can pick up the phone and call your good friend!

Forum, where all you need is a standard browser to get access to many dream features in a discussion forum:

- Read the entire thread in one page (not

one message at a time)

- Search the topics and message bodies
- Spell check your messages before posting.
- ...and many more

The URL of the JDJ Forum is: <http://www.sys-con.com/java/board.htm>

Access is free; however, we do ask you to register initially. You are also welcome to come in as a guest, without registra-

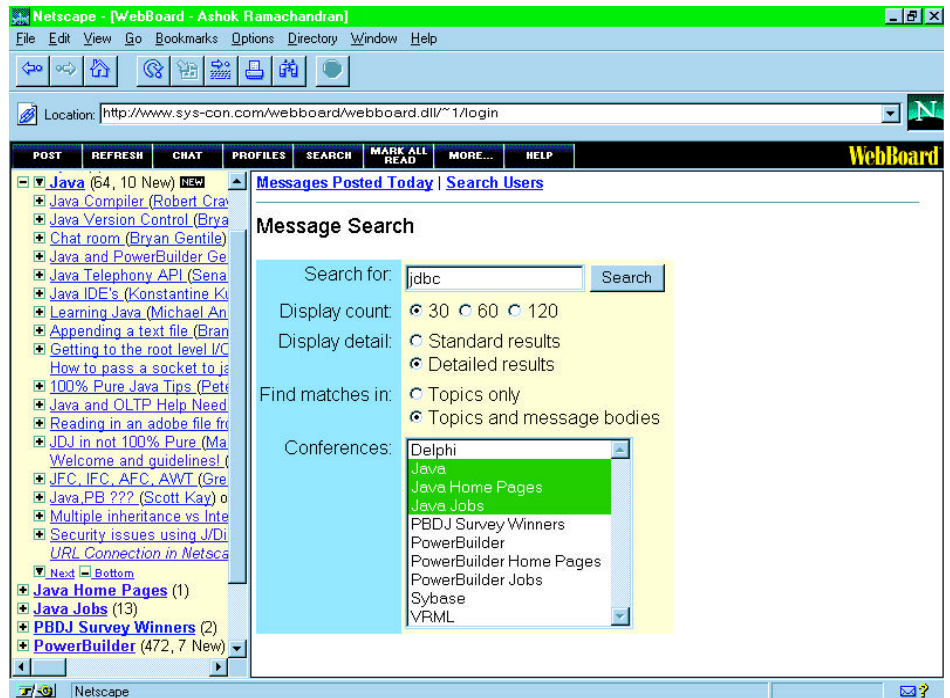


Figure 1: On the JDJ Forum, you can search the topics as well as message bodies.

Now, SYS-CON Interactive offers the JDJ

**Message Search Results for "rmi"**

Found 3 messages.

Conference	Subject	Date
Java	<a href="#">Using RMI over multiple routers</a>	8/22/97
Java	<a href="#">Security issues using J/Directory</a>	8/21/97
Java	<a href="#">Java and OLTP Help Needed</a>	8/12/97

will try: 1) Depending upon user permissions, just switch the visibility of

Figure 2: The search returns a clickable list of links

tion, to view the forum. And we have plans for a Team JDJ to ensure that you get a speedy response to all of your questions on Java. ☺

#### About the Author

Ashok Ramachandran is a Consulting Manager with Noblestar Systems Corporation, Falls Church, VA. Ashok is the SYSOP (Moderator) of the Forums at SYS-CON Interactive. He spends his free time(!) maintaining his Java/PB Newbie home page (<http://ashok.pair.com/>).

 <http://ashok.pair.com/>



**Java Developer's Journal Signs with Curtis Circulation Company** (Pearl River, NY) - SYS-CON Publications, Inc., publisher of Java Developer's Journal, has signed a distribution agreement with Curtis Circulation Company for the worldwide distribution of their best selling monthly title, Java Developer's Journal.

Besides Java Developer's Journal, SYS-CON publishes the well-known PowerBuilder Developer's Journal. In addition, SYS-CON will be launching VRML Developer's Journal in February '98 at the VRML '98 Conference in Monterey, CA, SilverStream Developer's Journal in March '98 and a new consumer-oriented title, CyberGamer Magazine, devoted to online Internet games, in April '98.

Java Developer's Journal has held the number one spot in newsstand sales since its premiere issue almost three years ago, according to International Periodical Distributors, JDJ's largest U.S. and Canadian distributor.

Curtis Circulation Company, a division of Hachette Filipacchi Group, is the largest distributor of magazines in the world. For more information on this agreement, please contact lhoffer@sys-con.com

## Cloudscape Announces First Embeddable Java™ Database

(Oakland, CA) - Cloudscape™, Inc. has announced that JBMS™, the first lightweight Java-based relational data management system, has been delivered to beta customers within the US. JBMS enables "smart applications" that are highly productive yet small enough to be deployed via the Internet and corporate intranet/extranet to run on platforms from laptops to the slimmest of clients, such as personal digital assistants (PDAs) or the network computer and other skeletal computing platforms.

Serving as the smart embedded data management engine for distributed applications such as e-commerce, personal information managers and Push technology, JBMS offers an embeddable Java data manager to developers building Java applications. Because it is based on Java, it can be extended with Java class libraries to handle new kinds of data within a SQL framework.

JBMS is expected to be generally available by the first quarter of 1998. Pricing is \$195 per development or deployment seat.

Cloudscape expects to provide a free downloadable evaluation kit at that time. For more information, visit their Web site at [www.cloudscape.com](http://www.cloudscape.com), e-mail [info@cloudscape.com](mailto:info@cloudscape.com) or call 510 873-0900.

## Perspective JavaChart™ Makes Java Charting Easy, Powerful

(Los Angeles, CA) - Three D Graphics has introduced a charting program that offers Java developers a 100% Pure J Class Library, a JavaBean™ and fully pre-configured Java 1.1 applet for creating professional charts directly in a Web page. It has a full set of properties, methods and user interface tools for developers who want to create driven graphics, and works in Java-compatible development environment, browser or operating system.

The program is a complete Java 1.1 applet. This product also is designed to support the latest browser technology, including Netscape™ 4.0 and Internet Explorer™ 4.0.

Perspective JavaChart has a licensing fee of \$995 and the source code is provided for an additional \$995 per developer. This price includes maintenance and updates for a period of six months. For more information visit Three D Graphics' Web site at [www.threedgraphics.com](http://www.threedgraphics.com) or you can contact them by phone at 800 913-0008 or by fax at 310 788-8975.

## Sun Service Solutions for Java in the Enterprise

(Palo Alto, CA) - Sun Microsystems Inc. has announced new services to help enterprises create Java™-powered solutions to realize business objectives. Sun's Java service solutions allow companies to investigate, evaluate, architect, pilot and implement/manage Java computing in safe, distributed and user-friendly environments around the world.

Sun Java Design Centers provide technical consulting services to help MIS managers and developers, Independent Software Vendors and system integrators develop Java expertise for creating a competitive business advantage. SunClient™ Support delivers cost-effective services for JavaStation™ network computers.

For more information on Sun's services, see their Web site at [www.sun.com/service](http://www.sun.com/service).

Inquiries about pricing and availability should be directed to local Sun locations.

## Visigenic will Resell Visual Edge's COM/CORBA

(Cupertino, Calif.) - Visigenic Software, Inc. and Visual Edge Software, Ltd. have announced that Visigenic will resell the COM/CORBA Interworking Product, which allows Visigenic's CORBA ORB to transparently and bi-directionally communicate with ActiveX/COM objects.

For more information about this product, visit the Visual Edge Web site at: [www.visualedge.com](http://www.visualedge.com) or else call 408 973-7823 or fax 408 973-7250.



## Black & White Software® Introduces OrbixBuilder™

(Campbell, Calif) - Black & White Software has introduced its OrbixBuilder family of graphical CORBA development products. The software incorporates visual drag-and-drop techniques and automatic generation of application code that is CORBA IIOP-enabled.

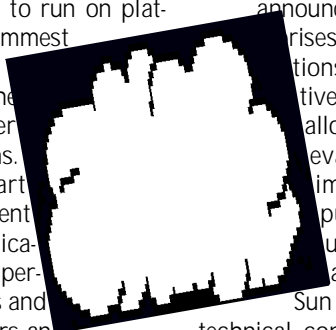
For Java™-based development on Windows® platforms, OrbixBuilder installs itself as an integrated extension to Symantec Visual Cafe™, thus extending graphical Java development capabilities to CORBA-based distributed products. The result is an infrastructure for building or migrating applications that function across the Internet and corporate intranets.

Quantity-one pricing is \$2000 and more information is available at the Black & White Web site: [www.blackwhite.com](http://www.blackwhite.com), or you may call 408 369-7400 or fax 408 369-7406. Their e-mail address is: [info@blackwhite.com](mailto:info@blackwhite.com)

## Progress Software Corporation Introduces Apptivity 2.0

(Scottsdale, Ariz.) - Progress Software Corp. has released Apptivity 2.0, a tool for building business-critical database applications in Java™. It incorporates a number of new features including a suite of new wizards, an enhanced visual user interface, a unique distributed debugger and CORBA integration.

Apptivity Developer is priced at \$1995 and includes a 5-user Apptivity Application Server for deploying pilot projects. Apptivity Server pricing begins at \$5000 for a 20-user server. Apptivity Server supports any Java server platform, and Apptivity applications can be deployed to any 1.02 or 1.1 JDK platform, including Netscape Navigator 3.x/4.x





and Microsoft Internet Explorer 3.x. Apptivity is compliant with any JDBC- or ODBC-compliant data source.

For more information, visit Progress Software's Web site at [www.progress.com](http://www.progress.com), phone 617 280-4000 or fax 617 280-4095. ●

## NetFactory's NetCharts Selected by Oracle Corporation

(Silver Spring, Maryland) - NetFactory Inc. has announced that Oracle Corporation will be incorporating NetCharts technology into its new Oracle Enterprise Manager V2 Web-enabled products for Oracle performance monitoring and capacity planning. NetCharts is an HTML-configurable, 100% Pure Java™-based, business charting package available for any Java-enabled platform including PCs, Unix and network computers.

For more information visit NetFactory's Web site at: [www.netcharts.com](http://www.netcharts.com), e-mail: [info@netcharts.com](mailto:info@netcharts.com) or call 301 625-5600. ●

## InstallShield® Java™ Edition Deploys Cross-Platform Apps

(Schaumburg, IL) - InstallShield Software Corp. has announced the availability of InstallShield Java Edition installation development version 1.0. It includes new features that enable developers to create true cross-platform installations, providing a consistent experience for users of all supported operating systems and Java Virtual Machines (VM) version 1.0.2 and higher.

By building a single Java package file, InstallShield Java Edition allows developers to create and maintain only one installation that will run on all supported VMs.

InstallShield Java Edition is available at an MSRP of \$495 and can be ordered directly from the InstallShield Web page at: [www.installshield.com/isorder](http://www.installshield.com/isorder). For more information about this product, call 800 374-4353, fax 847 240-9120 or e-mail: [info@installshield.com](mailto:info@installshield.com). ●

## Passport Corporation Incorporates TIBCO's Push Technology

(Paramus, NJ) - Passport Corporation has announced that the company's application development environment, Passport Intrprise™, now offers support for TIBCO's suite of publish/subscribe and multicast technologies.

Passport Intrprise is a comprehensive application development environment for building thin client applications in Internet and enterprise environments. It combines a heterogeneous architecture with open mid-

dleware and database access, application fault tolerance and scalability. Passport Intrprise is available for Windows 3.1, Windows NT, Windows 95, Open/VMS and UNIX. Additionally, Passport Intrprise applications can be deployed on all Java-supported platforms, including network computers. Pricing starts at \$8995.

For more information on Passport and its products, check the company's Web site at: [www.passport.com](http://www.passport.com) or call 1 800 926-6736. ●

## ObjectSpace Partners to Accelerate Agent Research

(Austin, Texas) - ObjectSpace, Inc. has been awarded the \$2.5M NIST Award to develop applications of "agent software technology" for semiconductor manufacturing. The resulting systems will be based on the ObjectSpace Voyager™ Core Technology (Voyager), an advanced 100% Java™ object request broker (ORB), and will be validated in AMD's latest generation microprocessor factory.

The ObjectSpace/AMD project will explore ways to develop and deploy a variety of goal-directed software agents that mimic and improve the functioning of real-world agents, such as factory workers, material, equipment and processes.

Voyager is a Java-centric distributed computing platform that includes seamless support for mobile objects and autonomous objects. Voyager contains an easy-to-use set of features found in other ORBs and agent platforms, including CORBA, RMI, Aglets and Odyssey. Voyager's design is based on the Java object model.

Additional information on Voyager can be found at ObjectSpace's Web site, [www.object-space.com](http://www.object-space.com), or call 972 726-4100. ●

## SuperCede, Inc. Announces Complete Support for JDK 1.1

(Washington, D.C.) - SuperCede, Inc., a leading developer of interactive software development tools for Java™ applications, has announced SuperCede 2.0, the latest version of the highly-acclaimed SuperCede visual development environment. This second generation of the SuperCede® product family offers many new features, including complete support for JDK 1.1 and JavaBeans.

For additional information visit

SuperCede's Web site at [www.supercede.com](http://www.supercede.com), call 425 462-7242 or fax 425 637-5886. ●

## GEO's Emblaze WebCharger™ Creates Rapid Web Site Graphics

(Woodland Hills, CA) - GEO Publishing, an Internet software publisher, has released Emblaze WebCharger™, an Internet image compression software. It solves the challenge of creating attractive Web sites without subjecting an audience to the delays associated with downloading and viewing image-intensive Web pages.

Emblaze WebCharger™ gives Web developers the ability to compress high-quality 16 or 24-bit full-color graphic images up to 400% more than JPEG, resulting in rapid loading Web site graphics since the small files can be quickly transmitted and viewed. Images converted with the software are completely JPEG-compatible.

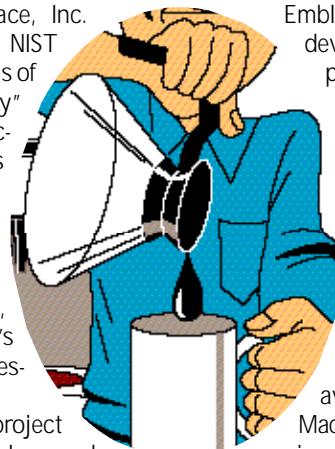
Emblaze WebCharger is available for Windows 95 and Macintosh 7.1+ and is available at major software retailers. The street price is expected to be \$99.95. Users may also order directly from GEO Publishing at 800 576-7751 or at their Web site at [www.emblaze.com](http://www.emblaze.com). ●

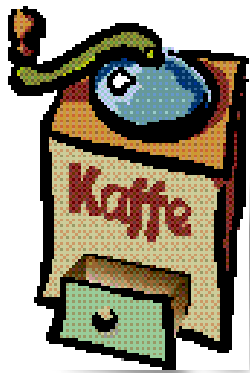
## Global Stock Games Using Open Market's Technology

(Cambridge, MA) - Open Market, a leading provider of Internet commerce software, has licensed its software, Transact™, to a leading Internet entertainment company, Global Stock Games. The new service is designed to educate as well as entertain customers as they use their skills to predict the rise and fall of selected shares on several worldwide stock markets. It can be found at [www.stockgames.com](http://www.stockgames.com).

Swiss-based Global Stock Games and its parent company, GMG (Int'l.) Ltd. will rely on Open Market's Transact to manage players' payments and offer credits for correctly predicting share movements. Customers will be able to participate in various online games based on developments selecting stock in 12 different markets, using Transact to manage the entire order management process.

For more information on Open Market, please call 617 949-7000 or see their Web site at [www.openmarket.com](http://www.openmarket.com). ●





# "The Brass Ring"

## THE GRIND

by Joe S. Valley

"Well, that reminds me of THE most important factor in hitting the Big One at a startup: blind luck."

Joe S. Valley is a scarred veteran of the Silicon Valley wars. It was either writing this column or heading back into therapy. His company can't afford mental health care coverage anymore, so writing is the only option. There are a million stories in the Valley and Joe knows lots of them. Got a good story? E-mail him at [Joe@sys-con.com](mailto:Joe@sys-con.com)



[Joe@sys-con.com](mailto:Joe@sys-con.com)

"Look, if I drink any more Espresso, my head will explode!"

I am back at Peet's Coffee discussing job search strategies with Ying and Yang, the nicknames I have given to my two former summer interns. They are out in the real world now, and have decided to split from their current jobs at big, boring companies. Both are interviewing at Internet startups in Silicon Valley. For some strange reason, they are meeting me again for additional job search advice.

"So why do you guys drink this? It tastes like motor oil!"

Yang fires back, "How would you know, Joe? That cheap gin you drink killed your taste buds years ago." Actually, it wasn't gin, but I didn't want to talk about my years of going to Grateful Dead concerts.

Ying chimes in, "So Joe, we both have gone pretty far down the interview road at these startups. No illusions - will be 70 hour weeks for a while. We took your advice about stock options, and we are negotiating some protection if they get taken out. We want to get some of our stock vested!"

"Good move!" I shout. I bet no one else at those companies has thought of what happens to their stock options if the company gets bought out. They could be in for a rude shock.

"So, who are you interviewing with?" I ask in my 'leading question' voice.

"The engineering managers, who else?" replies Yang. "Sounds like a lead in to another 'Wisdom of Joe' lecture."

"No, just curious...but how can you tell if these companies are going to make it by talking just to the engineering managers?" I ask.

"So far they have only had us interview the engineering guys, so what's the big deal?" snaps Yang.

I start in on another monologue that even John Galt would be proud of. "You are going to be working your butts off for the next two years. Make it count. If the job looks good, talk to several other departments, especially in marketing and sales channels."

"Why?" Yang asks. "What do I ask a marketing guy? Why would he or she want to interview me?"

"Look, marketing is simple in theory, extremely tough in practice; sort of like using Java in the real world. If the marketing and sales guys know what they are doing, then all your 70 hour weeks will give you a shot at that 'big win'. If they are clowns, then the tightest code you can write will be for nothing. Look what happened to Apple toward the end - the inability to market their products against Win-Tel negated the efforts of a lot of good engineers. So ask a couple of questions about marketing strategies. Lots of marketing guys in Internet startups came from big computer

companies where they really didn't do anything except show up in a suit and regurgitate the company line."

"So how did those guys end up at the hot startups?" asks Yang, who has come down from his espresso high long enough to look worried.

"They had good resumes and references that made them sound great. The fact that the references themselves were not that great was never checked out. Look, I can ask you in an interview to hack up some Java code and know in a few minutes if you are good. Marketing guys can scam their way for quite a while, sometimes for a whole career. So, how do you know if a marketing guy knows what he is talking about? Meet the Director of Marketing or the senior product guy associated with your product. Ask him some basic stuff. Who is the competition now and who do they think it will be in two years? Why is his product better than the other guy's? What keeps him awake at night worrying? And, the most important question, the elevator speech. If he had only 30 seconds, like in an elevator, explain what his product is, what it does and why it is better than the competition. Any marketing guy that will make a startup happen has already developed an elevator speech. They need it when they have a chance encounter with a potential customer, investor or reporter. If the marketing guy doesn't answer those questions in a way that you are comfortable with, then there are problems ahead."

"So," says Ying, "You are asking us to interview them as hard as they interview us."

"You know, most young programmers in Silicon Valley know more about the San Francisco 49ers that they watch for 3 hours a week, than they know about the startup they will be putting in 12 hours a day for. And one last tip, meet the president of the company. Ask him or her about what they see the company doing. Is it an IPO candidate or will it be taken out by Microsoft or Cisco? Ask the toughest questions you can think of, like what was their worst business failure and what did they learn from it? If they can't handle an engineer asking those questions, then there are bigger problems. Move on."

"So Joe, you seem like a smart guy, but you aren't at a startup that will make you rich!" fires Yang in his normal sarcastic tone.

"Well, that reminds me of THE most important factor in hitting the Big One at a startup: blind luck. Even with the best team, it's a roll of the dice."

"Great," says Ying. "Maybe I should just stay at this boring company I'm at and do the 9 to 5 thing."

"You won't," I reply. "The brass ring is out there, go for it. And if the dice roll your way, at least give me a ride in the Ferrari!"

# Microsoft Ad

# KL Group Full Page Ad