



## Pervasive Java, Part II

Sumi Helal, University of Florida

In the first issue, I covered Java 2, Micro Edition (J2ME) technology, focusing on its Connection Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP) (see the “Part I” sidebar). Here, I take a closer look at J2ME’s business and commercial side, comparing several development toolkits and addressing challenges the growing community of J2ME developers faces.

### J2ME PLATFORMS

Java-enabled phones and PDAs are a fast-growing market. Over three million Java handsets have been sold since April 2001, and companies are racing and teaming up with each other to capture a piece of this new market. The main players are handset and PDA manufacturers, wireless providers, Sun Micro-

systems, and companies specializing in software development tools.

With new devices continually entering the marketplace, it would be of little value to survey all available devices. Instead, I take a quick sampling of Java-enabled phones and PDAs. For complete, up-to-date information from over 12 different manufacturers, see JavaMobiles ([www.javamobiles.com](http://www.javamobiles.com)), the MicroJava Network ([www.microjava.com/devices](http://www.microjava.com/devices)), and the Sun Microsystems Web site (<http://wireless.java.sun.com/device>).

### Java-enabled phones

I surveyed three markets: Japan, North America, and Europe. Starting with Japan, its leading wireless provider, NTT DoCoMo, was the first to adopt J2ME technology. Its Java-enabled i-mode phones and services have attracted over 30 million subscribers since February 1999. In fact, i-mode has been a success story for DoCoMo, turning around its performance and revenues in the past couple of years. Currently, i-mode phones

don’t use the MIDP profile (instead, they use DoJa or DoCoMo Java), but the company plans to migrate the phones to the next MIDP generation (MIDP 2.0) by the end of the year.

Japan’s third-largest wireless carrier, J-Phone Communications, has outlined its Java service for mobile phones and plans to offer geographic information and multimedia services (with JPEG and PNG graphics synchronized with sounds). The J-Phone group offers several handsets based on J2ME technology, including a built-in digital camera, color display, and 3D graphics engine, which can display images with total freedom of perspective. The service’s 3D features are offered in the form of J-Phone’s original equipment manufacturer (OEM) proprietary Java library. Mitsubishi, Matsushita, NEC, Sony, Fujitsu, Toshiba, and Sharp are among the main handset manufacturers in the Japanese market (see Figure 1).

In the US, Motorola introduced the world’s first MIDP-compliant Java phone, the Motorola i85s. Motorola was

### GLOSSARY

<b>3G</b>	Third-Generation Wireless Networks
<b>CLDC</b>	Connected, Limited Device Configuration
<b>IDE</b>	Integrated development environment
<b>J2ME</b>	Java 2 Platform, Micro Edition
<b>JCP</b>	Java Community Process
<b>JSP</b>	Java Specification Request
<b>MIDP</b>	Mobile Information Device Profile
<b>OEM</b>	Original equipment manufacturer
<b>PDA</b>	Personal Digital Assistant
<b>PDAP</b>	PDA Profile

### PART I

As I discussed in the January–March issue, J2ME specifications are divided into configurations that are specific to different device categories. Each configuration is further divided into profiles, which are specifications of particular types of devices within a device category. For instance, the Connected Limited Device Configuration is targeted toward devices with limited network connection. The Connected Device Configuration is a more powerful configuration targeted towards devices such as set top boxes. See the last issue’s column for further details.

## STANDARDS, TOOLS, &amp; BEST PRACTICES



Figure 1. A small sample of commercially available Java-enabled phones.

also the first handset manufacturer to introduce Java phones to the North American market. The J2ME phones, serviced mainly by Nextel Communications, are based on the iDEN wireless technology and are currently directed at business users. The iDEN Java phones provide OEM proprietary support for TCP/IP and mobile networking, which

makes these handsets unique in the North American marketplace. Phones can be serviced with routable, static IPs (a form of mobile IP), which lets the phones serve as both mobile application clients and as mobile servers. Motorola plans to move other handsets (for example, PCS and GSM) to the J2ME platform.

In January 2002, Sprint Communica-

tions announced its plans to use third-generation phones from Hitachi when it launches its 3G service in the US later this year. Sprint's next-generation 3G network (known as 3G1X) will provide up to 144-kbit-per-second wireless packet data streams.

In one of the most ambitious efforts yet to use Java in cell phones, mobile phone maker Nokia has announced that it will ship 50 million Java-enabled mobile phones by the end of 2002 and 100 million by 2003. The Finnish company wooed its biggest competitors to join a mobile phone standard plan that will put Java, among other features, in next-generation phones. If Nokia goes through with its plans, the company could effectively increase the number of existing Java phones by over 30 percent.

For space limitation, I list specifications for only one phone—the Motorola i95cl (see the “Specifications” sidebar). It is the first color-enabled Java phone to enter the US market, and it has roughly twice the performance and features as the Motorola i85s. (It is worth mentioning that this doubling in performance occurred over 14 months.)

### Java-enabled PDAs

So far, the only profile supported in commercially available devices is the MIDP. Its capabilities are intentionally limited and cannot replace native applications on PDAs such as the Palm, PocketPC, or Nokia 9210 communicator. Nevertheless, J2ME is supported on some of today's PDA platforms, including the Palm OS, RIM BlackBerry, and Windows CE (see Figure 2).

The Java Community Process (JCP) is developing a new CLDC profile specification geared toward PDAs. This PDA Profile is supposed to offer greater sophistication than MIDP, including a richer user interface. The PDAP aims to provide a standard set of Java APIs for small, resource-limited handheld devices characterized as having

- No less than 512 Kbytes total memory (ROM plus RAM) available for

## SPECIFICATIONS

The specifications of the i95s, the first color-enabled J2ME mobile phone introduced in North America, are

**Airlink Interface:** iDEN/TDMA

**Carrier:** Nextel Communications

**Platform/configuration/profiles:** KVM/CLDC 1.0/MIDP 1.0

**Display:** 8-bit color display with PNG and JPEG support

**Memory:** The i95s has data memory, program memory, and heap memory. The phones are supplied with approximately 1.5 Mbytes of data memory, approximately 1.5 Mbytes of program memory, and 640 Kbytes of heap memory. Data memory is used as an initial download storage of MIDlets, persistent storage for MIDlet programs available through APIs, and voice recordings handled by the native phone OS. Program memory stores a program in an executable image format after it has been loaded from the data memory, expanded, and verified. Heap memory is used for runtime execution of programs.

**Network protocols:** UDP and TCP/IP (wireless packet data, including server sockets) and routable IP addressing (HTTP as well as SSL and HTTPS)

**Serial interface:** Java Serial Interface API

- Java runtime and libraries, and no more than 16 Mbytes total memory
- Limited power, typically battery operated
- UIs of varying degrees of sophistication with displays that have a total resolution of at least 20,000 pixels, a pointing device, and character input



Figure 2. Sample Java-enabled PDAs.

Palm is actively driving the formation of the PDAP (Java Specification Request 75, <http://jcp.org>) through the JCP. Unfortunately, the PDAP has been so long in the making that it might be too limited for where PDAs are headed and might not do justice to the capabilities of a high-end device such as the Compaq iPAQ. We will have to wait and see, but PDAP might be too little, too late.

### J2ME DEVELOPMENT TOOLKITS

To make business sense out of this emerging technology, we must create a profitable community of J2ME developers. Online communities such as the Motorola iDEN Developer Community ([www.idendev.com](http://www.idendev.com)) and the Sprint PCS Application Developer Program ([www.developer.sprintpcs.com](http://www.developer.sprintpcs.com)) offer shared resources and communication forums among their registered developers. In addition, handset manufacturers, carriers, and specialized software productivity companies are offering several J2ME development toolkits that aim to accelerate the process of introducing new applications.

J2ME developers must consider sev-

eral factors when deciding which toolkit best suits their needs. Developers also must often deal with multiple toolkits, so knowing how they differ helps. Of utmost importance are quality device emulators that can reliably test a J2ME application on prospective devices. J2ME applications also must pass through a *preverification* process, which lets the desktop compiler verify that the compiled code can run with J2ME's K Virtual Machine (J2ME toolkits include preverification tools that handle this process). In addition, several other features such as packaging and debugging also play a role in deciding the toolkit. Finally, most quality toolkits come complete with J2ME-specific documentation and sample applications.

There are several different configurations available for the software development kits. Some software packages presented here are full-fledged J2ME software development kits with integrated development environments (IDE), some are only development front ends,

and yet others are J2ME plug-ins to Java development environments.

The following comparison of the main tools and software development kits should help clarify the key aspects a developer must examine when selecting a development environment (also see Table 1).

### CodeWarrior 6.0

CodeWarrior from Metrowerks combines an IDE with the original MotoSDK development environment from Motorola ([www.metrowerks.com/desktop/java](http://www.metrowerks.com/desktop/java)). Its MIDP support evolved from a set of tools called the J2ME SDK Components Developer Edition, formerly maintained and distributed by Motorola. Developers can develop, debug, and test MIDlets using the familiar CodeWarrior IDE but with the convenience of a graphical user interface and menu-driven commands (the original MotoSDK was command-line only). In addition, CodeWarrior 6 has more bug fixes and enhancements, and developers can use

TABLE 1  
Comparison of J2ME tools and development toolkits.

Tool	Supported platforms	IDE	Device editor	Device emulator	Source editor	Object browser	MIDlet packager	Debugger	Preverifier
CodeWarrior for Java	MacOS, Win32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Zucotto Whiteboard	Win32	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Forte for Java	Linux, Solaris, Win32	Yes	No	No	Yes	Yes	No	Yes	No
Jbuilder	Linux, MacOS, Solaris, Win32	Yes	No	No	Yes	Yes	Yes	Yes	No
VisualAge Micro Ed.	Linux, Win32	Yes	No	No	Yes	Yes	No	Yes	No
Sun Wireless	Linux, Solaris, Win32	No	No	Yes	No	No	Yes	No	Yes
Nokia J2ME	Win32	No	No	Yes	No	No	Yes	No	Yes
Siemens J2ME	Win32	No	No	Yes	No	No	Yes	Yes	Yes
RIM BlackBerry JDE	Unknown	Yes	?	Yes	Yes	?	?	Yes	Yes

Motorola phone emulators to test MIDlets. The SKD comes with a JAD and JAR packager and a handy debugger and preverifier.

### Zucotto Wireless Whiteboard SDK

The Zucotto Wireless Whiteboard SDK is a free IDE and emulation environment for developers to create and deploy J2ME applications ([www.zucotto.com/whiteboard/product\\_downloads.html](http://www.zucotto.com/whiteboard/product_downloads.html)). The basic version (as well as support) is free, but if you want the Bluetooth version, it will cost you some money. The Zucotto package is by far the best overall package. It even comes with a PNG editor for creating and editing the PNG graphics files to be displayed in your MIDlet.

This IDE is derived from the NetBeans package, so the interface looks similar to Sun's Forte interface (also derived from NetBeans). The software development kit is the first to extend Bluetooth functionality to wireless Java application development. This will come in handy for emerging phones with integrated Bluetooth. Some of the distinctive features include seamless creation of jad/jar files, source-level debugging with multilevel tracing, a PNG Painter to create colorful MIDP-compliant images, excellent documentation to simplify development, and easy-to-reuse templates and code samples.

### Sun Forte for Java

Forte for Java is a complete Java development environment available from Sun ([www.sun.com/forte](http://www.sun.com/forte)). The Community Edition is free, but commercial versions are available that have more powerful functionality. Forte for Java supports Java development in general with a configuration section for specific libraries and compilers. You can set it up for Java IDE to use different J2ME plug-ins, but you must set up Forte before installing the plug-in. Forte 3.0 requires Java 2 1.3.1. It also requires the Netscape browser version 4.7 or above.

### Borland's JBuilder 5.0 Personal and Mobileset

Combining Borland's JBuilder 5.0 Personal and its Mobileset add-on package ([www.borland.com/jbuilder/mobileset](http://www.borland.com/jbuilder/mobileset)) lets developers create MIDP applications and test them using one of three toolkits: Sun's Wireless Toolkit (any phone), Nokia's Developer's Suite for J2ME (Nokia phones), or the Siemens Mobility Toolkit (Siemens phones). JBuilder contains perhaps the nicest GUI features, such as "drag and drop" creation of MIDlets and their resources. Similar to CodeWarrior and Whiteboard, Jbuilder and Mobileset include prime features such as a packager, debugger, and device editor. They also offer clear online documentation. A free personal edition is available, but

**The Zucotto package is by far the best overall package. It even comes with a PNG editor for creating and editing the PNG graphics files to be displayed in your MIDlet.**

it doesn't package the jad and jar files for you.

### IBM VisualAge Micro Edition

VisualAge Micro Edition is an IDE with a comprehensive set of tools and runtime components for developing and deploying Java applications on connected embedded devices ([www.embedded.oti.com](http://www.embedded.oti.com)). At its core is the J9 virtual machine, which is a single-sourced, multiple processor and platform combination, a feature that increases virtual-machine quality and provides a portability layer. The latter gives developers the flexibility they need to port applications to new processors and platforms quickly and easily. The new J2ME Java Powered compatible configurations (MIDP/CLDC) let developers

create devices that can host several applications written to the specifications. The combination of virtual machine and J2ME Java Powered configurations and profiles delivered across multiple target platforms, along with integrated tooling, is an industry first.

However, IBM's IDE differs from the others covered in this column. It is built from a previous version that lets embedded Java developers create their applications targeting the IBM J9 JVM. This new version also allows the creation of J2ME applications, although there is no device emulator available for it yet. This is a professional package, but it is not integrated well for developers who want to get set up and running quickly. It appears to be more suited for embedded Java development.

### Sun's wireless toolkit

Sun's toolkit is most appealing because it comes from the Java and J2ME creator (<http://java.sun.com/products/j2mewtoolkit>). There is a version for Solaris and Linux, but Sun does not fully support it. I tested one of the Windows packages (Windows 98/NT/2K supported).

The toolkit requires the Java 2 Standard Edition 1.3.0 or higher. It does not come with an editor, but does come with a special GUI for compiling, preverifying, jad/jar file packaging, and device emulation. The default development environment that the toolkit supports is known as Ktoolbar, which lets developers edit project settings using a GUI and select from a range of emulators to test the application. The Sun toolkit installs nicely, and the user guide that comes with it gets you started quickly. It explains how to run an example MIDlet and create your own using one of many different templates.

Although KToolbar is a great first step for J2ME beginners, the toolkit can also integrate with Sun's Forte development environment to provide a full-featured J2ME solution with sophisticated IDE. At the University of Florida, my students

## next issue

**I will look at standards for service discovery and delivery and will examine their use in pervasive computing environments.**

and I took one of our own MIDlets we wrote and used the Forte IDE to create a new MIDlet suite and to compile and emulate it. Everything worked smoothly. The emulator includes several default “skins” such as a color- and grayscale-screen mobile phone and a two-way pager.

The latest version of the toolkit (1.0.4 Beta) became freely available just before this issue went to press. It seems to include powerful features such as support for profiling MIDlet’s performance (memory, networking, methods, and so forth) as well as support for obfuscators. The new version also provides much better options for defining parameters of the target MIDP device’s virtual machine. This is a great feature for the developer, which lets him or her match the emulator’s performance with that of the target device.

### Nokia toolkit (beta)

The Nokia toolkit, which is available for free, comes with a GUI interface to compile, package the jad/jar files, and emulate Nokia devices (<http://americas.forum.nokia.com/java/default.asp>). This package can be integrated into Sun’s Forte for Java or Borland’s JBuilder to provide GUI project management functions. The only device emulator provided is Nokia’s emulator.

### Siemens Mobility Toolkit (beta)

The Siemens Mobility Toolkit is a set of tools based around Siemens’s SL45i and 6688i mobile phones ([www.siemens-mobile.com](http://www.siemens-mobile.com)). The SDK has two versions, a SL45i version for European markets and a 6688i version (both available for free), which includes emulator support for Chinese characters. The SDK includes an emulator, the CLDC and MIDP classes, Siemens-specific classes, and documentation. It comes with its own emulator as well as command-line tools for compiling and packaging the jad/jar combination. We can integrate this package, like Sun’s WTK and Nokia’s Toolkit, into Sun’s Forte for Java or Borland’s JBuilder to provide a

GUI project management functions. The only device emulator provided is a Siemens emulator. Several other features, such as a device editor, are not present in this toolkit.

**S**everal challenges and uncertainties face application developers that could limit pervasive Java’s progress. However, challenges often bring new opportunities for creative research and better practices.

For practical reasons, developers extensively use a device emulator (of the type and “skin” of a target device) throughout most of an application’s development and testing cycles. This unavoidable emulation could be a major source of discrepancy between the behavior of a MIDlet running on a given device emulator and that running on an actual device, which raises a serious uncertainty problem for the J2ME developer.

Another unexpected problem facing developers is the potential difficulty in porting MIDlets across different MIDP devices. Each device manufacturer can add its OEM’s “exotic features” to MIDP in the form of an API. This has quickly led to market fragmentation, which is contrary to the portability spirit of Java. Developers will have to weigh the benefits of specializing their applications for a specific device against the ease of porting their MIDlets across multiple MIDP devices.

Developers could face another serious problem if they are unable to freely download applications to MIDP devices. Consider the wireless telecom industry. Currently, carriers attempt to stick to their business territories when they deal with pervasive Java. One legitimate goal is to increase revenue

streams by selling minutes (or packets) to subscribers and businesses. Another goal is to safeguard against losing any business due to the loaded applications (imagine voice over IP over Bluetooth on the phone). Toward these goals, carriers are actively pursuing control of application distribution to their subscribers’ Java phones. Developers must follow whatever procedures the different carriers set forth to be allowed to download MIDlets to the phones. This could range from a registration requirement to signing legal contracts.

Finally, developers risk lack of support for network-side developments (especially Web services) in their development processes. Most development toolkits focus on the MIDlet client, which is certainly important. However, networked applications are bound to use a “proxy” between the MIDlets and the network resources and Web services. There is currently no support available for developers to create MIDlet proxies. Many functionalities and responsibilities of a MIDlet proxy are common or at least similar, leading to code redundancy. For example, protocol and graphics conversion are two examples of simple adaptation tasks that any MIDlet proxy will have to perform. Integrating MIDlet proxy support into J2ME Toolkits will speed up development of the full application. Of late, the JSP has proposed a Java Specification Request to address this need (JSR 172, accessible from <http://jcp.org>). ■

---

**Sumi Helal** is an associate professor in the Computer and Information Science and Engineering Department at the University of Florida. Contact him at [helal@cise.ufl.edu](mailto:helal@cise.ufl.edu); [www.cise.ufl.edu/~helal](http://www.cise.ufl.edu/~helal).