APPENDIX **1**

Introduction to Java 2 Micro Edition and KVM

1.1 Introduction to Java 2 Platform Micro Edition

Recognizing that one size doesn't fit all, Sun has recently regrouped its Java™ technologies into three editions: Java 2 Platform Enterprise Edition (J2EE), Java 2 Platform Standard Edition (J2SE), and Java 2 Platform Micro Edition (J2ME). Each of these editions is aimed at a specific market segment:

- *Java 2 Enterprise Edition*. For enterprises needing to serve their customers, suppliers, and employees with solid, complete, and scalable Internet business server solutions.
- *Java 2 Standard Edition*. For the well-established desktop market.
- *Java 2 Micro Edition*. For consumer and embedded device manufacturers who build a diversity of information devices, service providers who wish to deliver content to their customers over those devices, and content creators who want to make compelling content for small, resource-constrained devices.

Each edition defines a set of tools and supplies that can be used with a particular product:

- Java virtual machines that fit inside a wide range of computing devices,
- libraries and APIs that are specialized for each kind of computing device,
- tools for deployment and device configuration.

FIGURE 1-1 illustrates the target markets of each Java edition.

Java 2 Micro Edition (J2ME or Java 2 ME) specifically addresses the large, rapidly growing consumer space, which covers the range of tiny commodities such as pagers all the way up to the set-top box, an appliance almost as powerful as a desktop computer. Like the larger Java editions, Java 2 Micro Edition aims at maintaining the qualities that Java technology has become known for:

- built-in consistency across products in terms of running anywhere, any time, on any device,
- the power of a high-level object-oriented programming language with a large developer base,
- portability of code,
- safe network delivery,
- upward scalability with J2SE and J2EE.

With J2ME, Sun aims to provide a complete end-to-end solution for creating dynamically extensible, networked products and applications for the consumer and embedded market. J2ME enables device manufacturers, service providers, and content creators to gain a competitive advantage and capitalize on new revenue streams by developing and deploying compelling new applications and services to their customers worldwide.

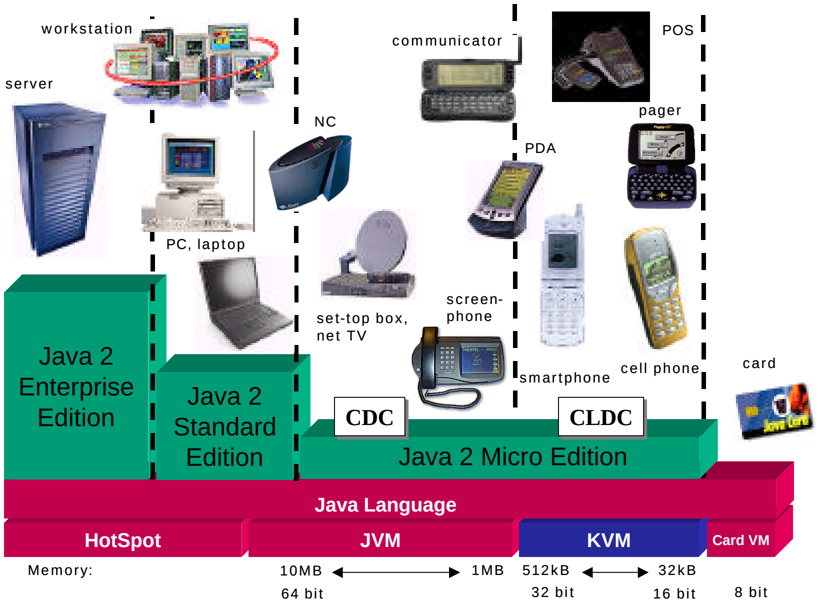

FIGURE 1-1 Java 2 editions and their target markets

At the high level, J2ME is targeted at two distinct groups of products (see FIGURE 1-1):

- *Personal, mobile, connected information devices.* Cell phones, pagers and organizers are the best examples of devices in this class. These devices have very simple user interfaces compared to desktop computer systems, total memory budget in the range of 128 to 512 kilobytes, and low bandwidth, intermittent connection to a network. Network communications in this group of products are not necessarily based on the TCP/IP protocol suite.
- *Shared, fixed, connected information devices*. Typical examples of devices in this category include set-top boxes, Internet TVs, Internet-enabled screenphones, high-end communicators and car entertainment/navigation systems. These devices have a larger range of user interface capabilities, total memory budget in the range of 2 to 16 megabytes, and persistent, high bandwidth, TCP/IP connection to a network.

It must be emphasized that the line between these two product groups is fluid. As a result of the ongoing technological convergence in the computer, telecommunication, consumer electronics and entertainment industries, the distinction between more general-purpose computers, personal communication devices, consumer electronics devices and entertainment devices is becoming blurry. Also, it can be anticipated that in the future devices will increasingly use wireless connectivity instead of traditional fixed networks. In practice, the line between the two groups is defined more by the total memory budget and the physical screen size of the device, rather than by specific functionality or by a certain type of connectivity.

1.2 J2ME Configurations and Profiles

While connected consumer devices such as cell phones, pagers, personal organizers and set-top boxes have many things in common, they are also diverse in form, function and features. Information appliances tend to be special-purpose, limited-function devices. To address this diversity, an essential requirement for J2ME is not only small size but also modularity and customizability.

The Java 2 ME architecture is modular and scalable so that it can support the kind of flexible deployment demanded by the consumer and embedded markets. To enable this, Java 2 ME provides a range of virtual machine technologies, each optimized for the different processor types and memory footprints commonly found in the consumer and embedded marketplace.

For low-end, resource-limited products, Java 2 ME supports minimal configurations of the Java virtual machine and Java APIs that capture just the essential capabilities of each kind of device. As device manufacturers develop new features in their devices, or service providers develop new and exciting applications, these minimal configurations can be expanded with additional APIs or with a richer complement of Java virtual machine features. To support this kind of customizability and extensibility, two essential concepts are defined:

- *Configuration*. A J2ME configuration defines a minimum platform for a "horizontal" class or family of devices, each with similar requirements on total memory budget and processing capabilities. A configuration defines the Java language and virtual machine features and minimum libraries that a device manufacturer or a content provider can expect to be available on all devices.
- *Profile*. A J2ME profile addresses the demands of a certain "vertical" market segment or device category. The main goal for a profile is to guarantee interoperability in a certain vertical device category or domain by defining a standard Java platform for that market. Profiles are implemented on top of a configuration.

Configurations and profiles are defined through the Java Community Process (JCP). Both configurations and profiles are discussed in more detail below.

1.2.1 Configurations

J2ME can be deployed in a number of *configurations*. A configuration defines a Java platform for a "horizontal" class or family of devices with similar requirements on total memory budget and other hardware capabilities. More specifically, a configuration:

- specifies the Java programming language features supported,
- specifies the Java virtual machine features supported,
- specifies the Java libraries and APIs supported.

Each configuration specifies the Java virtual machine features and a set of APIs that the users and content providers can safely assume to be present on all devices when shipped from the factory. Application developers and content providers must design their code to stay within the bounds of the Java virtual machine features and APIs specified by that configuration.

To avoid fragmentation, there will be a very limited number of J2ME configurations. Currently, the goal is to define two standard J2ME configurations (see FIGURE 1-1 on page 2):

- **Connected, Limited Device Configuration (CLDC)**. The market consisting of personal, mobile, connected information devices is served by the CLDC. This configuration includes some new classes, not drawn from the J2SE APIs, designed specifically to fit the needs of small-footprint devices.

- **Connected Device Configuration (CDC)**. The market consisting of shared, fixed, connected information devices is served by the Connected Device Configuration (CDC). To ensure upward compatibility between configurations, the CDC shall be a superset of the CLDC.

This specification focuses on the Connected, Limited Device Configuration (CLDC).

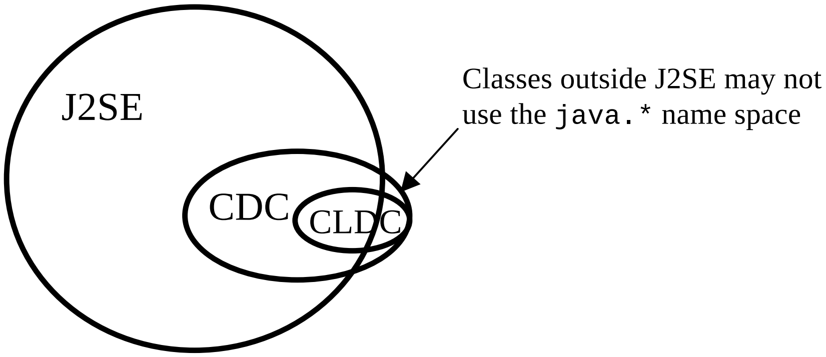

FIGURE 0-1 Relationship between J2ME configurations and J2SE

FIGURE 0-1 illustrates the relationship between CLDC, CDC and Java 2 Standard Edition (J2SE). As shown in the figure, the majority of functionality in CLDC and CDC has been inherited from J2SE. Each class inherited from J2SE must be precisely the same or a subset of the corresponding class in Java 2 Standard Edition. In addition, CLDC and CDC may introduce a number of features, not drawn from the J2SE, designed specifically to fit the needs of small-footprint devices. For further details, refer to *Configurations and Profiles Architecture Specification, Java™ 2 Platform Micro Edition (J2ME),* Sun Microsystems, Inc.

1.2.2 Profiles

Application portability is a key benefit of Java technology in the desktop and enterprise server markets. Portability is also a critical element of the J2ME value proposition for consumer devices. However, application portability requirements in the consumer space are generally quite different from portability requirements demanded by the desktop and server markets. In most cases consumer devices have substantial differences in memory size, networking, and user interface capabilities, making it very difficult to support all devices with just one solution.

In general, the consumer device market is not so homogeneous that end users expect or require universal application portability. Rather, in the consumer space, applications should ideally be fully portable between devices of the same kind. In addition, some kinds of applications, such as payment and banking applications, are expected to be freely portable between many kinds of devices.

In order to make it possible to define Java platforms for specific vertical markets, J2ME framework provides the concept of a *profile*. A profile defines a Java platform for a specific vertical market segment or device category. Profiles can serve two distinct portability requirements.

- To provide a complete toolkit for implementing applications for a particular kind of device such as cell phone, pager, or set-top box.
- Profiles may also be created to support significant, coherent groups of applications (for example, personal information management), which might be hosted on several kinds of devices.

At the implementation level, a profile is defined simply as a collection of Java APIs and class libraries that reside on top of a configuration to provide domain-specific capabilities for devices in a specific market segment. Profiles and the specific rules for defining J2ME profiles are described in more detail in separate specifications.

1.3 Introduction to KVM

At the present time, the CLDC runs only on top of Sun's new KVM. However, this specification allows for the possibility of running on other virtual machines. KVM is a compact, portable Java virtual machine specifically designed from ground up for small, resource-constrained devices. The high-level design goal for KVM was to create the smallest possible “complete” Java virtual machine that would maintain all the central aspects of the Java programming language, but would run in a resource-constrained device with only a few hundred kilobytes total memory budget.

More specifically, KVM was designed to be:

- small, with a static memory footprint in the range 30 kilobytes to 80 kilobytes,
- clean and highly portable,
- modular and customizable,
- as “complete” and “fast” as possible without sacrificing the other design goals.

The K in KVM stands for “kilo.” It was so named because its size is measured in tens of kilobytes. KVM is suitable for 16/32-bit RISC/CISC microprocessors with a total memory budget of no more than a few hundred kilobytes (and sometimes less than 128 kilobytes). This typically applies to digital cellular phones, pagers, personal organizers, and small retail payment terminals.

The minimum total memory budget required by a KVM implementation is about 128 kB, including the virtual machine, minimal Java libraries and some heap space for running Java applications. A typical implementation requires a total memory budget of 256 kB, of which half is used as heap space for applications, 30 to 80 kB is needed for the virtual machine itself, and the rest is reserved for class libraries. The ratio between RAM and ROM in the total memory budget varies considerably depending on the implementation. A simple KVM implementation without system class prelinking support needs more RAM than a more advanced KVM implementation with system classes preloaded in ROM.

The actual role of KVM in the target devices can vary significantly. In some implementations, KVM is used on top of an existing software stack to give the device the ability to download and run dynamic, interactive, secure Java content on the device. In other implementations, KVM is used at a lower level to implement the system software and applications of the device in the Java programming language. Several alternative usage models are possible.

For further information on KVM, refer to the official KVM web site at `http://java.sun.com/products/kvm`. KVM is derived from a research system called *Spotless* developed originally at Sun Microsystems Laboratories. More information on Spotless is available in the Sun Labs technical report “The Spotless System: Implementing a Java system for the Palm Connected Organizer” (Sun Labs Technical Report SMLI TR-99-73).