Telefónica

Telefónica Móviles
España, S.A.

Java 2 Micro Edition

1. Introducción.

La empresa Sun Microsystems lanzó a mediados de los años 90 el lenguaje de programación Java, orientado en un principio a la creación de componentes interactivos embebidos en páginas HTML, y que presentaba las siguientes características:

- Inspirado inicialmente en C++, pero con adiciones de alto nivel, como soporte nativo de *strings* y *garbage collection*.
- Código independiente de plataforma, precompilado a *bytecode* intermedio y ejecutado en el cliente por una JVM (*Java Virtual Machine*).
- Modelo de seguridad tipo *sandbox* proporcionada por la JVM.
- Abstracción del sistema operativo subyacente mediante un juego completo de APIs de programación.

Con el curso de los años, Java ha progresado enormemente en dos dimensiones:

- Nuevos ámbitos de ejecución: Programas aislados, servicios HTTP (*servlets*, JSP), servidores de aplicaciones (EJB).
- Mayores y más diversas funcionalidades: APIs para acceso a bases de datos (JDBC), telefonía (JTAPI), multimedia (JMF), mensajería (JMS), transacciones (JTS).

De resultas de esta explosión tecnológica, en la actualidad es poco realista concebir Java como un simple lenguaje de programación; en su lugar, resulta más conveniente considerarlo como un conjunto de tecnologías diseminadas en todos los ámbitos de computación con dos elementos comunes:

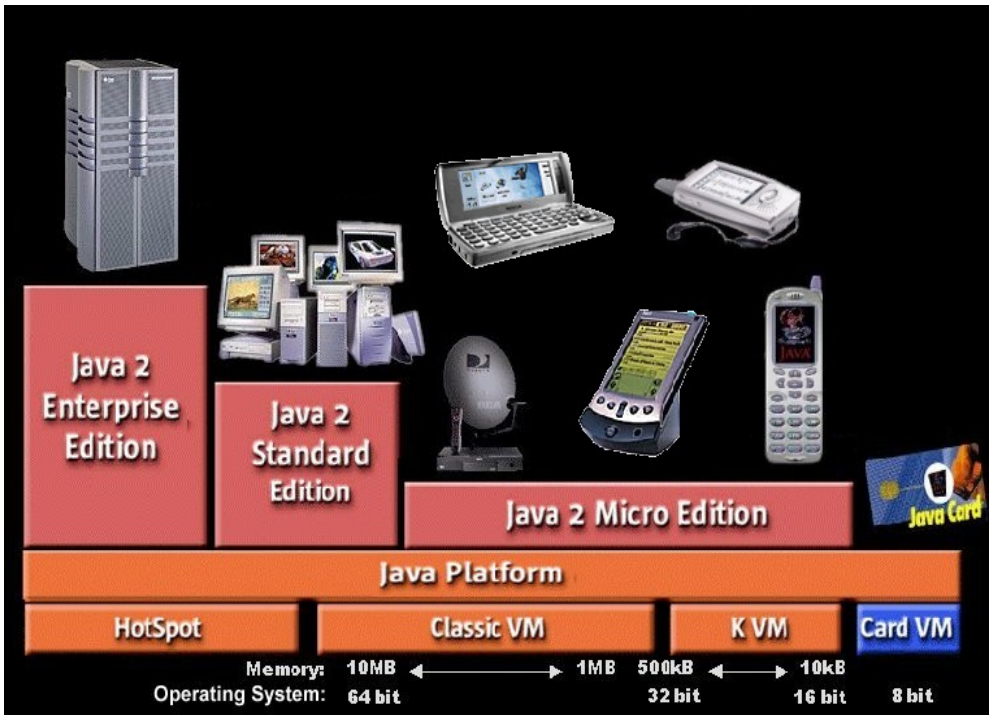
- El código fuente en lenguaje Java es compilado a código intermedio interpretado por una JVM.
- Todas las tecnologías comparten un conjunto más o menos amplio de APIs básicas del lenguaje, agrupadas principalmente en los paquete java.lang y java.io.

En el mundo de los dispositivos electrónicos, como teléfonos móviles, PDAs o dispositivos embebidos, la situación es aún más caótica, debido a que las plataformas en consideración son fuertemente disimilares en cuanto a capacidad y requisitos. Han surgido en el pasado numerosas iniciativas, con frecuencia incompatibles entre sí, para introducir Java en el desarrollo de aplicaciones para estos dispositivos.

En los últimos tiempos, Sun ha reconocido esta diversidad de ámbitos tecnológicos en los que Java se desenvuelve, y los ha agrupado en tres grandes entornos de aplicación, o *ediciones*, por seguir la terminología de la empresa:

- J2EE (Java 2 Platform, Enterprise Edition), orientada al desarrollo de servicios web y aplicaciones distribuidas mediante EJBs (*Enterprise Java Beans*) y tecnologías relacionadas, con APIs para la gestión de transacciones, persistencia de objetos, servicios de nombres, XML, autentificación, etc.
- J2SE (Java 2 Platform, Standard Edition), que abarca las APIs orientadas a la programación de aplicaciones de usuario final: interfaz gráfica de usuario, multimedia, *networking*. Esta edición es la que en cierta forma recoge la iniciativa original del lenguaje Java.
- J2ME (Java 2 Platform, Micro Edition), enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes.

Figura 1: Relaciones entre ediciones de Java.



Sólo de forma muy simplista puede considerarse que J2ME y J2EE son versiones reducida y ampliada, respectivamente, de J2SE: en realidad, cada una de las ediciones enfoca ámbitos de aplicación radicalmente distintos e independientes entre sí. Después de todos, las necesidades computacionales y APIs de programación requeridas por, por ejemplo, un juego gráfico ejecutándose sobre una PDA no tienen mucho que ver con los de un servidor distribuido de aplicaciones basado en EJB.

En lo que sigue estudiaremos con más detalle los componentes que integran la plataforma J2ME, la taxonomía propuesta por Sun de entornos de ejecución basados en las capacidades gráficas y computacionales de los dispositivos y la funcionalidad para la que están diseñados, la integración en J2ME de otras tecnologías Java relacionadas y la relevancia de la iniciativa J2ME para el desarrollo de aplicaciones y servicios embebidos en dispositivos móviles 2,5 y 3G.

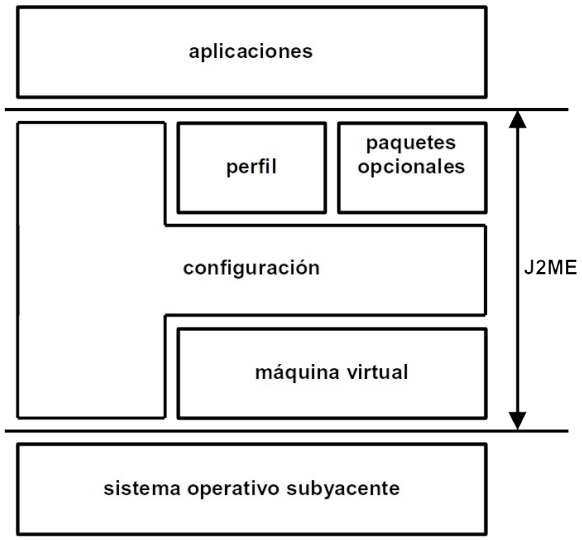
2. Componentes de J2ME

Al contrario que en otras tecnologías orientadas a PCs y ordenadores convencionales, en J2ME el espectro de dispositivos considerados varía enormemente en cuanto a capacidad computacional, memoria y capacidades gráficas. Ante la imposibilidad de establecer una arquitectura común que se adecue a esta variedad de entornos hardware, J2ME define una serie de componentes (building blocks) a partir de los cuales se construye una implantación concreta para un dispositivo determinado. Estos componentes se agrupan en los siguientes tipos:

- Máquina virtual
- Configuración
- Perfil
- Paquetes opcionales.

Un entorno de ejecución determinado de J2ME se compone entonces de una selección de máquina virtual, configuración y perfil, y posiblemente otros paquetes opcionales. En la figura se muestran las relaciones entre los distintos componentes de un entorno de ejecución de J2ME.

Figura 2: Relaciones entre componentes J2ME.



2.1. Máquina virtual

Una máquina virtual de Java (JVM, *Java Virtual Machine*) es un programa encargado de traducir código intermedio (*bytecode*) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente.

Las implementaciones tradicionales de JVMs son en general muy pesadas en cuanto a memoria ocupada (*footprint*) y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos, que en algunos casos se apartan de la especificación estándar relajando algunas exigencias con el fin de obtener implementaciones más ligeras.

2.2. Configuración

Una configuración es un conjunto de APIs básicas de Java que definen un entorno generalizado de ejecución. Una configuración no está orientada a ningún campo específico de aplicación, sino que más bien trata de agrupar los dispositivos en cuanto a sus capacidades computacionales y entorno genérico de operación (por ejemplo, si tienen conectividad o no). En particular, las librerías de interfaz gráfica nunca pertenecen a la definición de una configuración J2ME.

Usualmente, una configuración especifica una JVM de referencia sobre la que debe correr; en algunos casos, además, esta JVM se aparta del estándar, relajando algunas características. Esta JVM de referencia no es una exigencia de la configuración, sino que más bien establece la funcionalidad mínima que debe pedirse a las implementaciones de la JVM utilizadas. Por supuesto, una implementación inteligente de una configuración J2ME dada se ceñirá en la medida de lo posible a las especificaciones mínimas requeridas.

2.3. Perfil

Un perfil es un conjunto de APIs orientadas a un ámbito de aplicación determinado, para una configuración J2ME dada. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proveen (por ejemplo, teléfonos móviles, electrodomésticos) y el tipo de aplicaciones que típicamente correrán sobre ellos.

La librería de interfaz gráfica es generalmente una componente muy importante de la definición de un perfil. Las interfaces gráficas difieren mucho entre distintos perfiles (displays textuales en teléfonos móviles, interfaces basadas en pen para PDAs, etc.), y en algunos perfiles ni siquiera existen (sistemas embebidos, por ejemplo).

Un perfil siempre se define a partir de una configuración dada. Así, cabe pensar en un perfil como un conjunto de APIs que dotan a una configuración J2ME de funcionalidad específica para una familia de dispositivos concreta y para un tipo de aplicaciones determinado.

3. Estandarización de componentes J2ME

La intencionada indefinición en cuanto a especificación y selección de componentes de la arquitectura J2ME podría dar lugar a la proliferación de numerosos entornos de ejecución incompatibles y pseudopropietarios de no mediar alguna clase de esfuerzo coordinado de estandarización en el seno de la industria de los dispositivos electrónicos.

Con el fin de involucrar a la industria software y hardware en la evolución y el soporte de la tecnología Java, Sun ha creado el programa Java Community Process (JCP), un foro de estandarización con aproximadamente 300 participantes entre particulares, instituciones académicas y empresas de software, electrónica, telefonía móvil, comunicaciones, etc. Notablemente, ni Microsoft ni Intel pertenecen al programa JCP. El trabajo del programa JCP se articula en torno a la producción y aprobación de documentos de estándar conocidos como JSRs (Java Specification Requests), similares en forma e intención a las RFCs de la IETF.

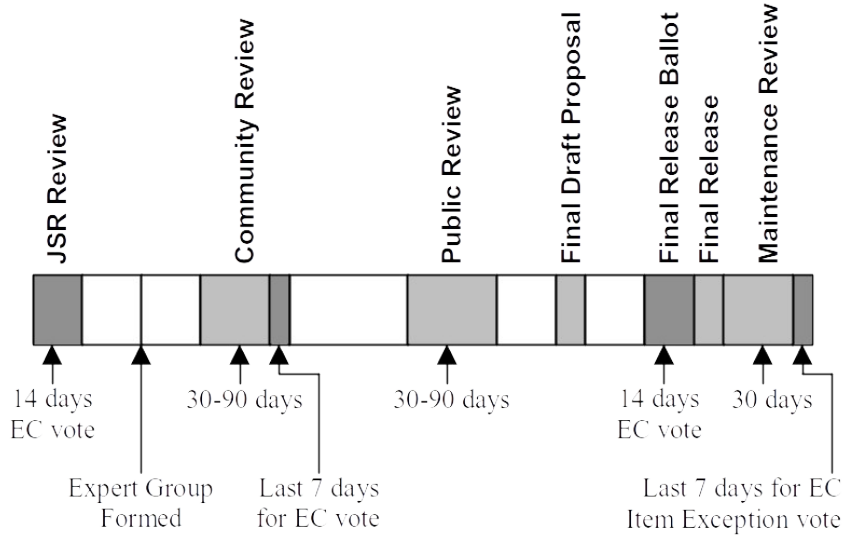
Uno de los campos de actividad del programa JCP es la especificación de componentes de J2ME, con el objetivo de llegar a un consenso amplio en torno a un número razonablemente pequeño de entornos de ejecución de J2ME agrupados por mercados de aplicación bien definidos, como el de las PDAs o el de la telefonía móvil. De los esfuerzos del programa JCP han surgido ya componentes estandarizados de J2ME, con diferentes grados de aceptación e implantación en la industria.

3.1. Ciclo de vida de una JSR

En la Figura 3 se definen los sucesivos estadios por los que una Java Specification Request (JSR) atraviesa hasta que se convierte en un estándar aprobado (Final Release). Estos estadios van desde la aprobación del comienzo de la JSR y la formación de un grupo de trabajo hasta la publicación del Final Draft y la aprobación final del mismo.

En la figura se muestra el ciclo de vida de una JSR junto con las estimaciones temporales de algunos procesos críticos en la evolución del proyecto. Esta información puede ser útil a la hora de evaluar el tiempo que resta para que un estándar en progreso sea completado por el JCP.

Figura 3: Ciclo de vida de una JSR.



EC: Executive Committee

3.2. Compatibilidad entre componentes

Ante la multiplicidad de entornos de ejecución que presenta J2ME, el antiguo lema “Write Once, Run Anywhere” asociado al lenguaje Java no parece muy aplicable en el mundo de los dispositivos electrónicos. Esto difícilmente puede verse como una debilidad de la plataforma J2ME, teniendo en cuenta la enorme variedad de dispositivos y ámbitos de aplicación que esta tecnología abarca.

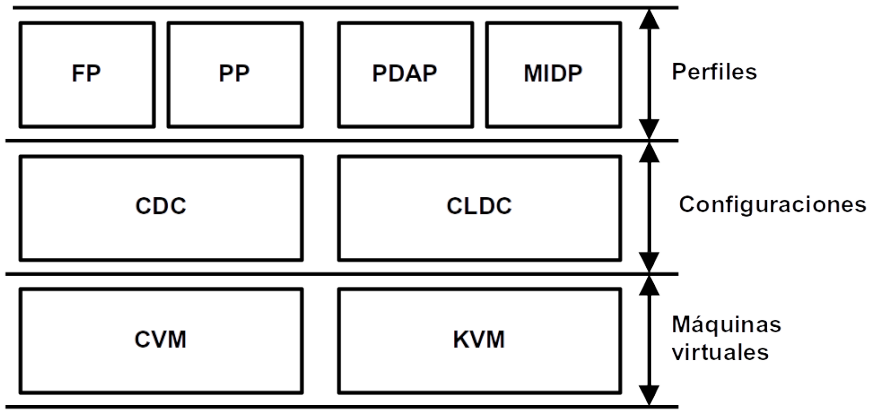
En general, los desarrolladores de aplicaciones para J2ME pueden esperar y deberían promover la compatibilidad entre plataformas dentro del ámbito de un mismo perfil. De esta forma, las aplicaciones para teléfonos móviles 2,5G, por ejemplo, deberían poder correr en todos aquellos dispositivos que soporten el perfil MIDP.

Otras compatibilidades cruzadas, como entre perfiles más limitados y menos limitados, en general no están garantizadas, y lo máximo que puede esperarse en este campo es que al menos grandes porciones de código puedan utilizarse en proyectos basados en distintos entornos J2ME.

4. Arquitectura J2ME

Como se ha explicado anteriormente, el programa de estandarización JCP ha definido diversos entornos de ejecución de J2ME identificados por una selección particular de máquina virtual, configuración y perfil J2ME. Es de suponer que esta arquitectura evolucionará en dos sentidos: Refinamiento de los entornos de ejecución definidos (nuevas versiones de los componentes), definición de nuevos entornos de ejecución.

Figura 3: Arquitectura de J2ME.



En la figura se muestra el estado actual de la arquitectura J2ME. Solamente se han incluido en el diagrama los componentes de J2ME que están estandarizados o muy próximos a ser estandarizados y que tienen especial relevancia para el mercado de los dispositivos para comunicaciones móviles.

Así como en el campo de los perfiles la arquitectura de J2ME es aún dispersa y está sujeta a cambios, parece que se ha alcanzado un consenso bastante estable en cuanto a las dos configuraciones disponibles:

- Connected Device Configuration (CDC), para dispositivos dotados de conectividad y con (relativamente) alta capacidad computacional,
- Connected Limited Device Configuration (CLDC), para dispositivos con capacidades (proceso y memoria) limitadas dotados de conectividad.

En la Figura 4 se relacionan unos cuantos tipos de dispositivos con la configuración más apropiada para los mismos.

Figura 4: Relación de configuraciones con tipos de dispositivos

4.1. Máquinas virtuales

4.1.1. C Virtual Machine

La C Virtual Machine (CVM) es una implementación de referencia escrita en lenguaje C de la máquina virtual de Java (JVM) estándar, orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2 MB o más de memoria RAM. Según Sun, la CVM está diseñada de forma que su portado a distintos sistemas operativos y/o procesadores sea relativamente fácil, lo cual debe ayudar a extender las configuraciones J2ME basadas en una JVM estándar (no reducida). La ocupación en memoria (*footprint*) de la CVM está en torno a los 256 KB.

En la actualidad, CVM tan sólo se encuentra disponible para el sistema operativo Linux sobre procesadores Intel x86, pero Sun apoya el portado de la CVM a otras plataformas.

La liberación de la CVM obedece a una política de *open source* de Sun Microsystems encaminada a promover la ubicuidad de Java, favoreciendo de esta forma la migración de la tecnología Java a muchas más plataformas de las que en principio una sola empresa podría abarcar.

La CVM ha sido tomada como JVM de referencia en la definición de la CDC

4.1.2. K Virtual Machine

La K Virtual Machine ó KVM (la *K* viene de *kilobyte*, haciendo referencia a la baja ocupación en memoria) se desarrolló en parte en el programa de investigación *Spotless* de Sun Microsystems dedicado a la implementación de un sistema Java para el Organizer de Palm. Se trata de una implementación de referencia para una JVM reducida especialmente orientada a dispositivos con bajas capacidades computacionales y memoria limitada. En concreto, la KVM presenta las siguientes limitaciones (entre otras) con respecto a la especificación estándar de la JVM:

- No hay soporte para tipos de coma flotante (float y double).
- No hay soporte para métodos nativos (Java Native Interface, JNI).
- No se soporta la reflexión de clases.
- No se soporte la finalización de objetos (método Object.finalize()).
- Tratamiento limitado de excepciones.

Además, el sistema de seguridad de la JVM ha sido simplificado y modificado para ajustarse a limitaciones computacionales y de memoria más estrictas.
La KVM está escrita en lenguaje C y, según Sun, en mayo de 2000 había sido portada al menos a 25 plataformas distintas, entre las que se encuentran:

- Palm OS,
- Win32,
- Solaris.

A fecha actual, los autores no conocen ninguna versión para Windows CE de la KVM, aunque los dispositivos típicos sobre los que corre Windows CE tienen potencia suficiente para ejecutar una JVM estándar.

La ocupación en memoria (*footprint*) de la KVM oscila entre los 40 y los 80 KB.

La KVM ha sido tomada como JVM de referencia en la definición de la CLDC

Las primeras versiones de la KVM se incluyeron así mismo en las distribuciones de KJava.

4.2. Configuraciones

4.2.1. Connected Device Configuration

La configuración J2ME conocida como Connected Device Configuration (CDC) está siendo elaborada dentro de la JSR 36 del JCP; la versión 0.2 alcanzó el estatus de *Final Draft* el 21 de febrero de 2001.

La CDC está orientada a dispositivos electrónicos con las siguientes capacidades:

- Procesador de 32 bits.
- 512 KB mínimo de memoria ROM.
- 256 KB mínimo de memoria RAM.
- Conectividad a algún tipo de red.
- Soporte total de la JVM v.2

Como ya se ha explicado, la CDC no se hace ninguna asunción acerca de las capacidades gráficas de los dispositivos. Dispositivos típicos a los que la CDC se orienta son:

- Descodificadores de TV,
- teléfonos con navegación de Internet (*communicators*),
- sistemas de navegación para coches.

Como se ha dicho, la CDC corre sobre una JVM estándar; no obstante, las implementaciones de referencia, desarrolladas para los sistemas Linux y VxWorks, utilizan la CVM.

La CDC está basada en J2SE v1.3, e incluye varios paquetes Java de la edición estándar. Las peculiaridades de la CDC están principalmente contenidas en el paquete javax.microedition.io, que incluye soporte genérico para comunicaciones HTTP y comunicaciones basadas en datagramas.

Un aspecto importante de la CDC es que engloba totalmente las APIs de la CLDC.

4.2.2. Connected Limited Device Configuration

La configuración J2ME conocida como Connected Limited Device Configuration (CLDC) está siendo elaborada dentro de la JSR 30 del JCP; la versión 1.0 alcanzó el estatus de Final Release el 30 de mayo de 2000.

La CLDC está orientada a dispositivos dotados de conectividad con reducidas capacidades computacionales y memoria limitada, de acuerdo a los siguientes parámetros:

- Procesador de bajas prestaciones, posiblemente de 16 bits.
- De 128 a 512 KB de memoria total, incluyendo ROM/Flash y RAM.
- Potencia limitada, frecuentemente alimentados por baterías.
- Conectividad reducida (9600 bps o menos), como en el caso de GSM.

Entre los dispositivos a los que la CLDC se orienta tenemos:

- PDAs de bajas prestaciones (tipo Palm),
- teléfonos móviles GSM y 2,5G (GPRS).
- buscas (pagers),
- terminales de transacciones electrónicas.

Con objeto de adaptarse a las restrictivas capacidades de este tipo de dispositivos, la CLDC no opera sobre una JVM estándar, sino que define un subconjunto de ésta conocido como KVM.
En cuanto a las APIs incluidas, parte de ellas provienen de la edición estándar de Java (J2SE), y además se incluyen unas APIs genéricas de conectividad en el paquete javax.microedition.io. Todas las APIs de la CLDC están también contenidas en la CDC.

CLDC para Palm

Posiblemente, la implementación más popular actualmente de CLDC/KVM es la del Palm Connected Organizer Error: no se encontró el origen de la referencia, que actualmente ya está siendo usada para desarrollo de pequeñas aplicaciones en este dispositivo.
Una característica interesante de la distribución que hace Sun de CLDC/KVM para Palm es que incorpora una pequeña librería no estándar denominada Palm API para gestión de la interfaz gráfica de la Palm. De esta forma es posible desarrollar aplicaciones gráficas para Palm directamente sobre CLDC/KVM sin instalar ningún perfil J2ME (los cuales, según los filosofía de la arquitectura, son los encargados de especificar las interfaces gráficas). La Palm API está contenida en el paquete com.sun.kjava, proveniente de la tecnología pre-J2ME KJava. Sun efectúa esta distribución no estándar de CLDC/KVM con el objeto de proporcionar soporte y continuidad a los desarrolladores de KJava.

4.3. Perfiles

4.3.1. Foundation Profile

El perfil J2ME conocido como Foundation Profile (FP) está siendo elaborado dentro de la JSR 46 del JCP; la versión 0.2 alcanzó el estatus de *Final Draft* el 21 de febrero de 2001 **Error: no se encontró el origen de la referencia**.
El FP define una serie de APIs sobre la CDC para dispositivos *sin interfaz gráfica* dentro del rango de aplicabilidad de la CDC, esto es, algún tipo de conectividad y potencia y memoria suficiente (1024 KB ROM, 512 KB RAM).

El mismo nombre del FP sugiere que este perfil incorpora muy poca funcionalidad específica, y en la mayor parte de los casos se espera que esté complementado por paquetes opcionales, o que sea extendido a otro perfil. Se añaden al API de la CDC tres nuevos paquetes y 130 clases adicionales, con funcionalidades variadas.

4.3.2. Personal Profile

El Personal Profile (PP) de J2ME está siendo elaborado dentro de la JSR 62 del JCP; actualmente, está en el estado JSR Approved, esto es, en fase de definición. El PP se definirá a partir de FP/CDC y añade una interfaz gráfica completa, con capacidades web y soporte de applets Java.

En la actualidad no hay mucha información disponible sobre el PP. La principal motivación para crear este perfil es la de ofrecer una vía de migración a J2ME de la tecnología PersonalJava

PDA Profile

El perfil para PDAs (PDAP) de J2ME está siendo elaborado dentro del marco de la JSR 75 del JCP, y en la actualidad está en el estado JSR Approved, esto es, en fase de definición. PDAP se basa en la CLDC y pretende abarcar PDAs de gama baja, tipo Palm, con una pantalla y algún tipo de puntero (ratón o pen) y una resolución de al menos 20.000 pixels (esto es, de al menos 200×100 pixels para un factor de forma típico 2:1).

Actualmente, no es posible dar mucha más información sobre este perfil en vías de definición. La principal motivación para crear este perfil es la de ofrecer una vía de migración a J2ME de la tecnología KJava.

4.3.3. Mobile Information Device Profile

El *Mobile Information Device Profile* de J2ME (MIDP) ha sido desarrollado dentro de la JSR 37 del JCP, que aprobó la Final Release de la versión 1.0 el 19 de septiembre de 2000.
El MIDP se basa en CLDC/KVM y se orienta a dispositivos con las siguientes características:

- Reducida potencia computacional y memoria.
- Conectividad limitada (como la provista por las redes celulares actuales, en torno a 9600 bps).
- Capacidades gráficas muy reducidas (mínimo un display de 96×54 pixels monocromo).
- Entrada de datos alfanuméricos reducida (p.ej., como la provista por los teléfonos móviles).

Los tipos de dispositivos que se adaptan entonces al MIDP son: Teléfonos móviles del rango de los actuales teléfonos WAP, buscas bidireccionales (two-way pagers) y PDAs de gama baja con conectividad celular.

El MIDP proporciona APIs que soportan la siguiente funcionalidad:

- Una forma limitada de almacenamiento de datos.
- Conectividad celular basada en HTTP 1.1.
- Timers.
- Presentación de imágenes de baja resolución.
- Entrada de datos de usuario.
- Un modelo rudimentario de ciclo de vida de las aplicaciones (llamadas midlets, por simpatía con “applet”).

Estas APIs están contenidas en los paquetes específicos javax.microedition.rms (persistencia), javax.microedition.midlet (ciclo de vida), y javax.microedition.lcdui (interfaz de usuario).

Sun proporciona una implementación de referencia basada en un emulador de MIDP para Windows y un entorno completo de desarrollo para este mismo sistema operativo.

Figura: Ejemplos de aplicaciones MIDP (midlets) provistos por Sun: Selector, Sokoban,

La tecnología MIDP parece solaparse en cuanto al ámbito de aplicación con otra iniciativa de Sun conocida como JavaPhone. La postura oficial de Sun al respecto es que JavaPhone está orientada a dispositivos de mayores prestaciones que los que soportarán MIDP. En cualquier caso, la comparación de requisitos para ambas tecnologías es complicada, puesto que JavaPhone no está incluida en la arquitectura J2ME.

Se está desarrollando una nueva versión de MIDP, la llamada MIDP NG, y se prevee que estará lista para principios del 2003.