Plataforma Java para móviles

Florina Almenárez Mendoza
Celeste Campo
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
florina@it.uc3m.es, celeste@it.uc3m.es

Parte de este material se basa en transparencias de Natividad Martínez Madrid (nati@it.uc3m.es)

UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Introducción a la programación de dispositivos limitados

Contexto

- **Objetivos**
  - Conocer las distintas arquitecturas de desarrollo de aplicaciones para entornos y dispositivos móviles portables
  - Identificar las características y restricciones de los dispositivos móviles portables
  - Conocer el soporte de red que nos proporcionan para el desarrollo de aplicaciones

- **Bibliografía**
  - ***Mobile and Wireless Design Essentials***. Mallick, Martyn. Wiley 2003. L/D 621.396.4 MAL. Capítulo 1, 2 y 3.
  - ***Pervasive computing handbook***. Hansmann, Uwe. Springer 2001. L/D 621.39:004 PER.

Índice

- Introducción
- **Sistemas operativos**
- Interfaces de usuario
- Interfaces de comunicaciones
- Nuevas funcionalidades

Introducción

- **Diversidad de dispositivos móviles** portables asociados al usuario: agendas electrónicas, teléfonos móviles, buscas, etc.
- Multitud de **nuevos dispositivos con capacidad de computación**: electrodomésticos, electrónica de consumo,…
- Capacidad de **comunicación** ⇨ nuevos protocolos inalámbricos
  - Bluetooth, WLAN, UMTS, WUSB, WiMax…
- Se puede acceder a **servicios** tradicionales y a nuevos servicios
  - *mBusiness/m-Commerce*, *mLearning*, *m-Marketing*, *m-Health*, etc.
- Desarrollo de las aplicaciones para estos nuevos dispositivos presenta nuevos retos:
  - **Limitaciones** en capacidad de proceso y memoria
  - Diferentes interfaces con el usuario
  - Conectividad **intermitente**, **bajo** ancho de banda, varias interfaces

Sistemas Operativos

- Windows CE (**Windows Phone**)
- **Symbian** (Symbian OS 9.5, **Symbian^3**)
- **Palm webOS** (anterior Garnet OS)
- **Embedded/Mobile Linux** (Familiar, Maemo, Moblin, OpenZaurus, MobiLinux, LiMo platform)
- **Android** (Google)
- MAC **OS X** (iPhone)
- Otros: **RIM** Blackberry **OS, Hiptop**, propietarios (bada…)

Sistemas Operativos Windows CE

- Sistema operativo ⇨ 32 bits, modular, de tiempo real
- La primera versión se distribuyó en Noviembre 1996
- Plataforma reducida de Windows
  - fundamentalmente PDAs y teléfonos móviles ("*smartphones*")
- Utiliza los mismos lenguajes y entornos de desarrollo que se emplean con Windows para PC
  - Código nativo: C/C++ (Visual C++)
  - Código manejado ("*managed code*"): Visual Basic .NET, C#
  - J2ME, Python, …
- A partir de la versión 4.2 ⇨ Windows Mobile 2003 (6.5)
- Windows Phone 7 series ⇨ Photon (WM7)

Windows phone

Sistemas Operativos Symbian

- Empresa fundada por Nokia, Motorola, Ericsson y Psion
  - versión 6 de EPOC, 1998
  - Psion ⇨ en 1989 comenzó a desarrollar EPOC (para PDAs)
- **Objetivo**: crear un sistema operativo para dispositivos inalámbricos, especialmente teléfonos móviles.
- Sistema operativo de 32 bits con características de tiempo real y multitarea.
- Desarrollo de aplicaciones:
  - Código nativo: C/C++
  - OPL (< v8), **Python**, Visual Basic, Simkin, Perl, J2ME

symbian

Sistemas Operativos
Palm OS

- Jeff Hawkins desarrolló la primera versión, 1996
  - pensado exclusivamente para PDAs
- Características (Garnet OS)
  - mono-tarea, sistema de ficheros ⇨ Utiliza base de datos para representar archivos ejecutables y datos
  - Necesidades asequibles de potencia (16-33 Mhz)
- Desarrollo de aplicaciones
  - Código nativo: C/C++
  - Visual Basic, J2ME, Python, …
  - aplicaciones ejecutables ⇨ archivos con extensión PRC
- Palm ALP (Access Linux Platform), 2006/2007
- Palm WebOS (Enero, 2009) ⇨ basado en Linux

Sistemas Operativos
Embedded Linux

- Mismo software que en el PC o servidor pero en un dispositivo limitado
  - Linux empotrado puede ocupar aproximadamente 2 MB
- Ventajas de ser software libre: disponibilidad de fuentes, modificación y adaptación del sistema operativo (a medida)
- Qtopia, Maemo, OpenMoko, MobiLinux, …
- Se comercializan PDAs y móviles con Linux y también existen distribuciones para instalar sobre otros sistemas
  - Motorola presentó su primer teléfono basado en Linux en 2003
- Librerías compactas de glibc and gcc

Sistemas Operativos
Android

- Plataforma de software que incluye un SO basado en Linux y desarrollado por Google y Open Handset Alliance.
- Características
  - núcleo monolítico
  - pantalla táctil
  - teclado QWERTY
  - OpenGL, SQLite, OpenSSL, …
- Desarrollo oficial de aplicaciones
  - actualmente no soporta ejecución de código nativo
  - únicamente Java ⇨ APIs propietarios y VM **Dalvik**
- Primer dispositivo (2008) ⇨ **T-Mobile G1/HTC Dream**

Sistemas Operativos
Otros

- **Mac OS X** optimizado para procesadores ARM
  - 4 capas de abstracción ⇨ núcleo del SO, servicios principales, media y cocoa *touch*
  - Desarrollo de aplicaciones
    - C, Pascal, Objective-C (orientado a objetos), Java
    - iPhone SDK
- **RIM** (Research In Motion) **OS** para Blackberry
  - Blackberry OS 4.3/5.0
  - Desarrollo de aplicaciones
    - C, (Visual) C++, Java (JME)
  - Arquitectura orientada a eventos

Sistemas Operativos Dispositivos

Windows CE

Symbian

Palm OS

Sistemas Operativos Dispositivos (II)

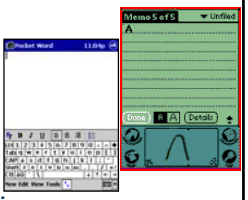
Embedded Linux

iPhone

Otros

Android

Interfaces con el usuario
Entrada



- Pantallas sensibles (*touch screen*):
  - Lápiz especial.
  - Reconocimiento de escritura o teclado simulado.
- Teclado:
  - Más seguro y rápido.
  - Teclado reducido o externo (plegable, de goma!).

- Keypad:
  - Datos numéricos y caracteres asignados a números.
  - Sistema T9.

- Reconocimiento de voz
- Tracking balls, botones, etc.

Interfaces con el usuario
Salida

- Pantalla:
  - Pequeña, pocas líneas.
  - Poca disponibilidad de gráficos, tipos de letra, etc.

- Leds:
  - Actividad de red, estado de la batería.
- Audio
- Vibrador

Interfaces de comunicaciones

- **WAN** (*Wide Area Network*)
  - GSM, GPRS, UMTS
- **LAN** (Local Area Network)
  - Wi-Fi
- **PAN** (Personal Area Network)
  - Bluetooth, IrDA, WUSB
- Incluso con soporte de **WiMax**
- **NFC** (Near-field Communication)
- Algunos de estos interfaces vienen integrados en el propio dispositivos y otros se incluyen a través de tarjetas de expansión (CF/SD WiFi – CF/SD Bluetooth).

Funcionalidades añadidas

- Cámara
- Reproductores de mp3
- Grabadores de voz
- GPS
- Sensores ⇨ acelerómetros
- **Dispositivos multi-función:**
  - Convergencia de dispositivos:
    - PDAs convergen a ser teléfonos móviles
    - Teléfonos móviles convergen a ser PDAs
  - Sustituyen a
    - cámaras digitales, reproductores de audio, sistemas GPS, …

Introducción a Java Micro Edition (Java ME)

Contexto

- **Objetivo**
  - Conocer la plataforma Java para desarrollar aplicaciones en dispositivos limitados
  - Aprender a desarrollar aplicaciones multi-plataforma para dispositivos móviles portables

- **Bibliografía**
  - ***Wireless Java Programming with Java 2 Micro Edition***. Feng, Yu and Zhu, Jun. SAMS 2001 . L/D 004.438 JAVA FEN. Capítulo 2 y 3.
  - http://java.sun.com/j2me
  - **Programming wireless devices with the Java 2 platform, micro edition:** J2ME Connected Limited Device Configuration (CLDC), Mobile Information Device Profile (MIDP). R. Riggs. Addison-Wesley, 2003.

Índice

- **Introducción**
- **Generalidades**
- **Arquitectura**
  - Máquinas Virtuales
  - Configuraciones
  - Perfiles

Introducción

- Historia de Java
  - Oak (Proyecto Green) (1990)
    - Software para dispositivos electrónicos de consumo
    - Cambia el nombre a Java ⇨ propiedad intelectual
  - Java 1 ⇨ 1.0 (96), 1.1 (97)
  - Java 2 ⇨ 1.2 (98), 1.3 (2000), 1.4 (2002), 1.5 (2004), 1.6 (2006)
- Sun ha estructurado la tecnología Java 2 dirigiéndose a sectores distintos (1999):
  - Java 2 Enterprise Edition (**J2EE**):
    - Soluciones de empresa: e-commerce, e-business.
  - Java 2 Standard Edition (**J2SE**):
    - Soluciones de PCs de sobremesa: applets, aplicaciones de usuario.
  - **Java 2 Micro Edition (J2ME):**
    - **Dispositivos móviles**
    - **Dispositivos de consumo y embebidos**

Introducción (II)

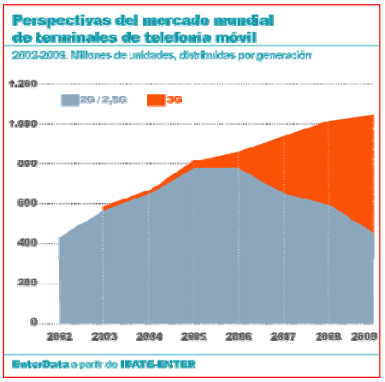
- También Java Card (1996)
  - Tarjetas inteligentes (“smart cards”)
  - CPU: 8-16 bits; 1-5Mhz.
  - Memoria: 1.2K RAM, 32K memoria no volátil.
- Historia de J2ME
  - **PersonalJava** (1997)
    - Dispositivos conectados con interfaces de usuario (set-top boxes, etc).
    - Basado en el jdk 1.1.8
    - Incorporado en el Personal Profile de J2ME.
  - **EmbeddedJava** (1998)
    - Dispositivos embebidos con funcionalidad dedicada y restricciones de memoria (control automóvil)
    - Incorporado en un perfil CDC.

Arquitectura de la plataforma Java 2

Diferentes API’s y VMs, pero siempre el lenguaje de programación es Java

¿Java en dispositivos móviles?

- Éxito de la telefonía móvil
- Java proporciona :
  - Una plataforma estándar para el desarrollo de aplicaciones
  - Capacidades gráficas para diseñar interfaces de interacción con el usuario
  - Gran número de programadores Java: facilidad y rapidez en el desarrollo de aplicaciones
  - Portabilidad de las aplicaciones entre diferentes dispositivos y distintos fabricantes

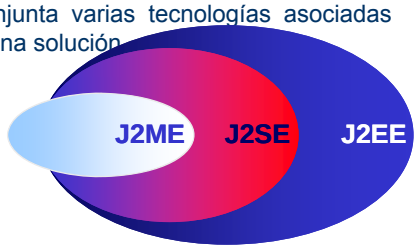


Generalidades de J2ME

- Nueva plataforma para la programación de aplicaciones Java en dispositivos limitados
- Abarca un gran tipo de dispositivos limitados no sólo teléfonos móviles
  - PDAs, buscas, electrodomésticos inteligentes, etc.
- En el mundo de los sistemas móviles:
  - J2ME es complementaria, NO es una alternativa a:
    - WAP, iMode, …
  - J2ME añade:
    - Mayor riqueza de contenidos
    - Descarga de software en dispositivos móviles:
      - Personalización de servicios proporcionados por terceras partes

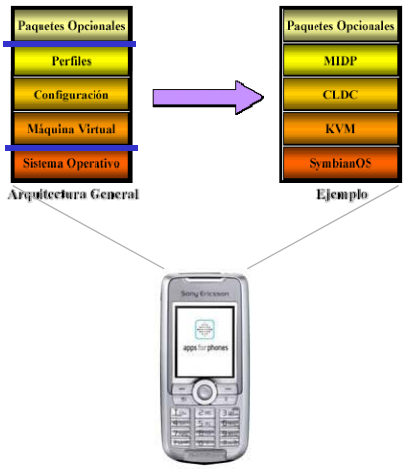
Generalidades J2ME

- Versión muy simplificada de J2SE
- Estandarizado bajo el Java Community Process (JCP)
  - JSR 68: J2ME Platform Specification
    - Arquitectura de la plataforma
    - Actividades de estandarización
  - JSR 185: Java Technology for Wireless Industry (JTWI)
    - Específico para teléfonos móviles de siguiente generación
    - Cómo trabajan de forma conjunta varias tecnologías asociadas con MIDP para proporcionar una solución para la industria de servicios inalámbricos
- *Java Specification Reports* (JSRs) separados para los diferentes APIs de J2ME



Arquitectura

- Para conseguir flexibilidad y adaptación, J2ME se estructura en tres capas:
  - Máquina virtual
  - Configuración
    - Mínimo conjunto de clases disponibles.
    - Engloba un segmento horizontal de mercado.
  - Perfiles
    - Clases adicionales para un segmento vertical de mercado.
- Un dispositivo puede soportar múltiples perfiles



Máquinas virtuales

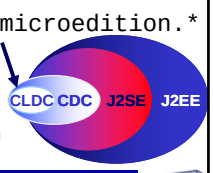
- Una JVM
  - Interpreta código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma
  - Efectúa las llamadas pertinentes al sistema operativo
  - Observa las reglas de seguridad
- Ligadas a una configuración
- Existen dos VM en la actualidad:
  - **CVM**: Compact Virtual Machine, C Virtual Machine
  - **KVM**: “Kilo” Virtual Machine , K Virtual Machine
- **CVM**:
  - Orientada a dispositivos embebidos y electrónica de consumo (set-top box, TV digital, electrodomésticos,...)

Máquinas virtuales (II)

- **CVM**:
  - Misma funcionalidad que JVM con:
    - Mejor uso de la memoria (≈ 2MB).
    - Procesadores de 32 bits.
  - Ligada a la configuración **CDC**
- **KVM**:
  - Antecedentes: Spotless (VM para PalmOS)
  - Dispositivos con poca memoria, capacidad de proceso limitada y con conexión a red intermitente:
    - Memoria mínima 128 KB.
    - Procesadores de 16 ó 32 bits RISC o CISC.
  - Acepta el mismo conjunto de bytecode (con algunas excepciones) y formato de ficheros de clase que la JVM.
  - Ocupa entre 40 y 80 KB.
  - Ligada a la configuración **CLDC** ⇨ más pequeña

Configuraciones

- ¿Qué es una configuración?
  - Mínimo conjunto de clases disponibles para un grupo de dispositivos. Los grupos se establecen según requisitos similares de memoria y procesamiento.
- ¿Qué define?
  - Características soportadas del lenguaje de programación Java.
  - Características soportadas por la Máquina Virtual Java.
  - Bibliotecas básicas de Java y APIs soportadas.
- Las configuraciones se especifican vía la iniciativa JCP que genera los correspondientes JSR
- Existen dos configuraciones actualmente:
  - **Connected Device Configuration (CDC)**
  - **Connected, Limited Device Configuration (CLDC)**



CDC
Connected Device Configuration

- Orientado a dispositivos con:
  - 512 KB de ROM.
  - 256 KB de RAM.
  - Conexión a red (fija).
  - Soporte completo a la especificación de JVM.
  - Interfaz de usuario relativamente limitado.
  - Basado en J2SE v1.3
- Especificado en JSR 36 (CDC 1.0) y JSR 218 (CDC 1.1)
- Ejemplos: Internet screen phones, DTV set-top boxes y sistemas telemáticos de automóviles.
- Iniciativas anteriores: PersonalJava, JavaTV, JavaPhone.

CDC
Librerías incluidas

Nombre de Paquete CDC	Descripción
`java.io`	Clases e interfaces estándar de E/S
`java.lang`	Clases básicas del lenguaje
`java.math`	Paquete de matemáticas
`java.net`	Clases e interfaces de red
`java.security`	Clases e interfaces de seguridad
`java.security.cert`	Clases de certificados de seguridad
`java.text`	Paquete de texto
`java.util`	Clases de utilidades estándar
`javax.microedition.io`	Clases e interfaces para conexión genérica CDC

CLDC
Connected Limited Device Configuration

- Orientado a dispositivos con:
  - 160 KB a 512 KB de memoria disponible para Java.
  - Procesador de 16 o 32 bits, velocidad 8-32 MHz.
  - Limitaciones de consumo (baterías).
  - Conectividad a red (inalámbrica).
  - Restricciones importantes en el interfaz de usuario.
- Especificado en JSR 30 (CLDC 1.0) y JSR 139 (CLDC 1.1)
- Especificación CLDC 1.0/1.1 disponible:
  - Sun proporciona una implementación de referencia de CLDC sobre KVM, para Linux, Windows y Solaris.
  - Principales fabricantes de móviles la implementan en la mayoría de sus modelos (Nokia, Siemens, Samsung,...)

CLDC
Librerías incluidas

Nombre de paquete CLDC	Descripción
`java.io`	Clases y paquetes estándar de E/S. Subconjunto de J2SE
`java.lang`	Clases e interfaces de la VM. Subconjunto de J2SE
`java.util`	Clases, interfaces y utilidades estándar. Subconjunto de J2SSE
`javax.microedition.io`	Clases e interfaces de conexión genérica CLDC

Perfiles

- Conjunto de clases Java que complementan una configuración para un conjunto específico de dispositivos (“segmento vertical”).
- ¿Qué definen?
  - APIs que controlan el ciclo de vida de la aplicación,
  - Interfaz de usuario, etc.
- Los perfiles permiten la portabilidad de aplicaciones J2ME entre diferentes dispositivos.
- Las perfiles se especifican vía la iniciativa JCP que genera los correspondientes JSR.

Perfiles sobre CDC

- **Foundation Profile** (JSR 46, JSR 219):
  - Perfil básico para dispositivos sin interfaz gráfico.
- **Personal Basis Specification** (JSR 129):
  - Perfil gráfico para dispositivos con interfaz gráfico básico.
- **Personal Profile** (JSR 62, JSR 216):
  - Perfil gráfico basado en AWT (dispositivos con interfaz gráfico).
  - Evolución de Personal Java.

Perfiles sobre CLDC

- **Mobile Information Device Profile** (JSR 37, JSR 118):
  - Perfil para dispositivos inalámbricos: móviles, PDAs,…
- **Information Module Profile** (JSR 195):
  - Perfil para dispositivos con interfaz gráfica limitada: parquímetros, alarmas,…

Paquetes opcionales

- Conjunto de APIs adicionales que pueden ser añadidos de forma flexible sobre diferentes perfiles
  - Extiende un perfil
- Son utilizadas en una multitud de dispositivos y familias de dispositivos
- Un paquete opcional contiene una funcionalidad que es independiente del segmento vertical
  - Bluetooth, gestión de contenido multimedia, localización, …
- Un dispositivo puede soportar múltiples paquetes opcionales

Paquetes opcionales sobre CDC

- **JSR 66: RMI Optional Package**
  - Subconjunto de J2SE RMI.
- **JSR 169: JDBC Optional Package**
  - Soporte JDBC en dispositivos CDC.
- **JSR 209: Advanced Graphics and User Interface Optional Package**
  - Facilidades de migración para interfaces de usuario y gráficos avanzados de J2SE a J2ME

Paquetes opcionales sobre CLDC

- **JSR 75: PDA Optional Packages**
  - Acceso a ficheros y datos personales
- **JSR 82: Bluetooth API**
  - Desarrollo de aplicaciones que usan Bluetooth.
- **JSR 120, JSR 205 (2.0): Wireless Messaging API**
  - Acceso a sistemas de envío de mensajes (SMS, CBS)

  CBS: Cell Broadcast Service
- **JSR 135: Mobile Media API (MMAPI)**
  - Acceso y reproducción de recursos multimedia (audio, video).
  - JSR 234: Funcionalidades multimedia avanzadas

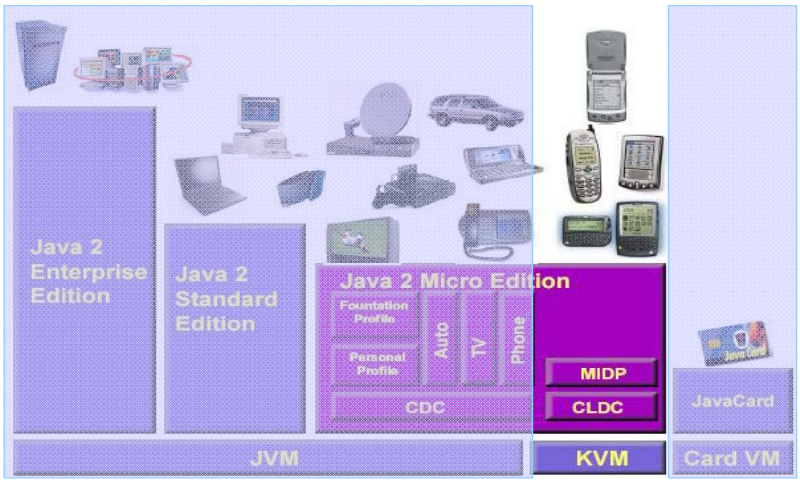
Paquetes opcionales sobre CLDC (II)

- **JSR 172: Web Services APIs**
  - Desarrollo de clientes Web en dispositivos móviles
- **JSR 177: Security and Trust Services**
  - Mejora la seguridad añadiendo almacenamiento seguro, APIs criptográficas, firmas digitales, gestión de credenciales.
- **JSR 179: API de Localización**
  - Acceso a la información de localización física
  - JSR 293: API de localización 2.0
- **JSR 180: Session Initiation Protocol (SIP)**
  - Desarrollar clientes SIP

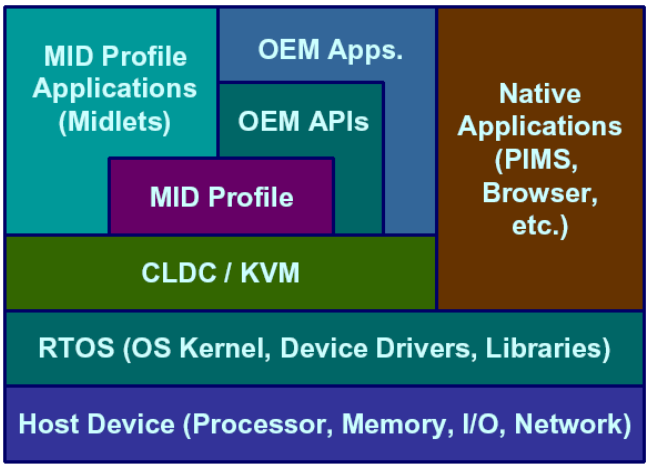
Paquetes opcionales para CDC y CLDC

- **JSR 80: USB API** ⇨ Comunicación con dispositivos conectados por USB
- **JSR 229: Payment API** ⇨ Cliente de transacciones de pago móvil
- **JSR 230: Data Sync API** ⇨ Sincronización de información
- **JSR 256: Mobile Sensor API** ⇨ Gestión y acceso a la información de sensores conectados a dispositivos móviles
- **JSR 257: Contactless Communication API**
- **JSR 281** (**IMS** Services API)
- …

MIDP/CLDC/KVM



Arquitectura MIDP/CLDC/KVM



CLDC/KVM Ámbito

- CLDC/KVM **cubre**:
  - Máquina virtual y soporte al lenguaje Java.
  - Modelo de seguridad.
  - Entrada/Salida.
  - Soporte a conexiones de red.
  - Internacionalización.
- CLDC/KVM **no cubre**:
  - Instalación y gestión del ciclo de vida de las aplicaciones Java.
  - Interfaz de usuario.
  - Gestión de eventos.
  - Modelo de aplicación a alto nivel.
  - Soporte a almacenamiento persistente.

CLDC 1.0/KVM
Diferencias con JVM

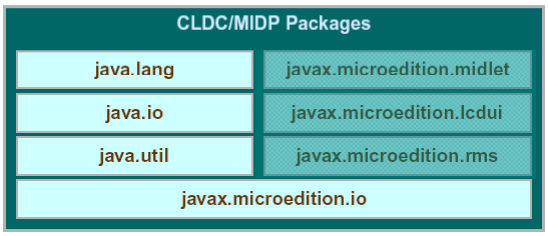
- No soporta tipos en punto flotante (*float* y *double*).
- No soporta finalización de instancias de clase
- Limitaciones en el manejo de errores.
- No soporta Java Native Interface (JNI).
- No soporta reflexión (*reflection*).
- No soporta cargadores de clase definidos por el usuario.
- No soporta grupos de hilos ni demonios (*thread groups, daemon groups*).
- Verificación de código en dos fases: preverificación y comprobación de clases más ligera.

CLDC 1.1/KVM
Diferencias con CLDC1.0

- Soporta punto flotante (`Float` y `Double`).
- Soporta referencia débil
- `Calendar`, `Date`, `TimeZone` rediseñadas
- Requisitos para gestión de errores han sido aclarados
- Memoria mínima incrementada de 160 a 192 KB

CLDC/KVM
Librerías

- Clases heredadas de J2SE:
  - **`java.lang.*`**
  - **`java.io.*`**
  - **`java.util.*`**
- Clases específicas introducidas por CLDC:
  - **`javax.microedition.io.*`**



CLDC/KVM
Internacionalización

- Todos los dispositivos CLDC soportan por defecto **`ISO-LATIN1`** (**`microedition.encoding`** con valor "**`ISO8859_1`**").
- Los fabricantes pueden proporcionar códigos adicionales:
  - Por ejemplo, NTT DoCoMo requiere que los teléfonos i-mode soporten la codificación japonesa ShiftJIS.
- No se soportan soluciones relacionadas con el formato de fechas, tiempo, o moneda.

CLDC/KVM
Propiedades

- Las propiedades del sistema se obtienen vía **`java.lang.System`**
  - No incluye **`java.util.Properties`**
- La llamada a **`System.getProperty(String key)`** devuelve el valor de la propiedad como un **`String`**
- CLDC deben proporcionar al menos las siguientes propiedades:
  - **`microedition.platform`**
  - **`microedition.encoding`**
  - **`microedition.configuration`**
  - **`microedition.profile`**
- Un perfil CLDC puede requerir ciertos valores de las propiedades:
  - MIDP 1.0 requiere **`microedition.profile`** contener al menos "MIDP-1.0"

CLDC/KVM
Entrada y salida a sistemas de almacenamiento y red

- Nuevo soporte porque el de J2SE presenta los siguientes problemas:
  - Gran tamaño: más 100 clases (200 kB).
  - No estaba pensado para pequeños dispositivos:
    - Se suponía TCP/IP siempre disponible.
    - No es fácil de extender a nuevos protocolos no TCP/IP tipo Bluetooth o IrDA.
- CLDC introduce **`Generic Connection Framework`**:
  - Soporte a diferentes tipos de protocolos de red.
  - Permite definir y usar nuevos protocolos de forma sencilla.
  - Compatibilidad con Java estándar, mediante mapeo.

CLDC/KVM
Generic Connection Framework

- CLDC especifica un mecanismo general de conexión:
  - `Connector.open("<protocol>:<address>;<parameters>");`
  - Por ejemplo:
    - Ficheros:
      - `Connector.open("file://midp.txt");`
    - HTTP:
      - `Connector.open("http://www.sun.com");`
    - Sockets:
      - `Connector.open("socket://129.144.111.222:9000");`
    - Puerto serie:
      - `Connector.open("comm:0;baudrate=9600");`
- CLDC no implementa ningún protocolo, son los perfiles los que deben definir qué conector(es) debe(n) implementarse.

CLDC/KVM
Seguridad

- No soporta el modelo completo de J2SE.
- Modelo de seguridad de CLDC:
  - Seguridad a nivel máquina virtual: verificador de clases.
  - Seguridad a nivel de aplicación: modelo “sandbox”.
- Verificador de clases en dos pasos:
  - Pre-verificador externo.
  - Verificador en el dispositivo.
- Modelo “sandbox”:
  - No se pueden sobreescribir clases del sistema.
  - No se pueden acceder a clases nativas.
  - Restringido al API proporcionada por el CLDC y el perfil sobre el que desarrolla.