CONCEPTES AVANÇATS DE SISTEMES OPERATIUS
Departament d'Arquitectura de Computadors

J2ME

(Java to Micro Edition)

(Seminaris de CASO)

Autors

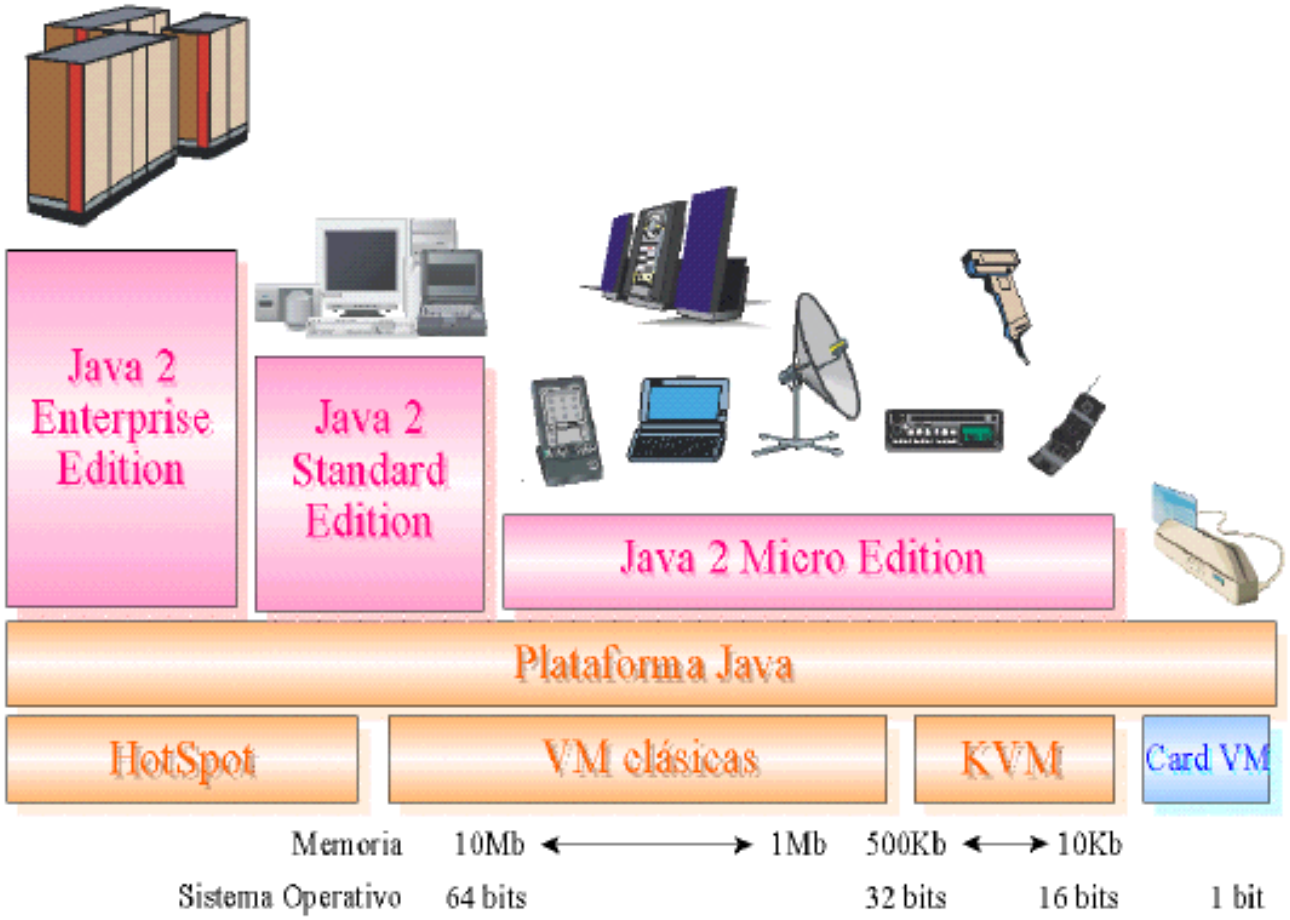
José Antonio Carmona Gallardo

Valentí Moncunill González

Introducción

- J2ME es la versión del lenguaje java orientada al desarrollo de aplicaciones para dispositivos con capacidades limitadas en el apartado gráfico, procesamiento y memoria *(PDA's, móviles, etc.)*.

La plataforma Java



Componentes

J2ME está constituido por:

- Una Máquina virtual (***K**ilo **V**irtual **M**achine*) que se encarga de ejecutar el "*bytecode*" de las clases java.
- Conjuntos de clases básicas o *Configuraciones,* orientadas a conformar el corazón de las implementaciones para dispositivos con características específicas.
- Librerías Java o *Perfiles*, destinadas a implementar funcionalidades de más alto nivel

La máquina virtual KVM

- Es la más pequeña de las desarrolladas por Sun hasta el momento.

- Requiere entre 50 y 80 KB de memoria para una configuración estándar y en principio, un máximo de 180 KB (contando la memoria dinámica en ejecución).

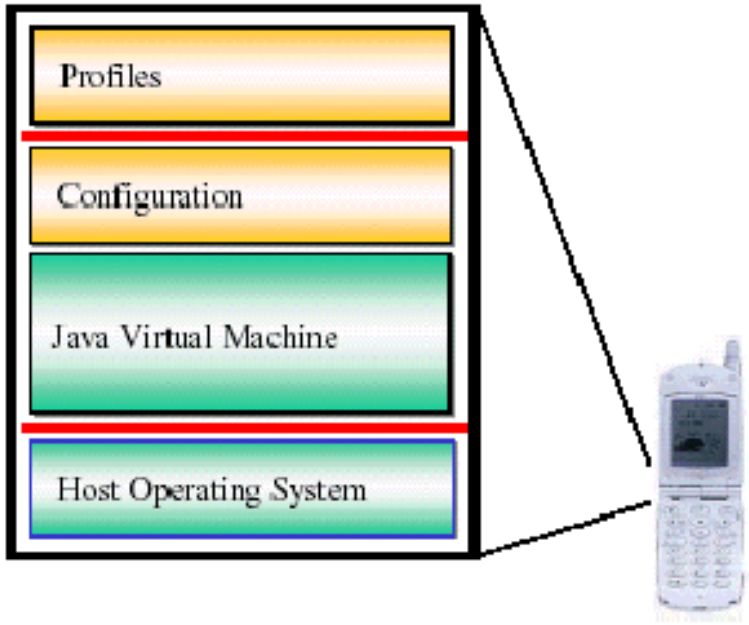
Configuraciones

- ***C**onnected **L**imited **D**evice **C**onfiguration* (CLDC):
  - Destinada a cubrir las necesidades de pequeños aparatos con limitaciones.
  - Utiliza la KVM, preparada para microprocesadores de 16/32 bits y pocos Kb de memoria (problema: no se puede utilizar los formatos *double* y *float* y su emulación es muy costosa).

- ***C**onnected **D**evice **C**onfiguration* (CDC):
  - La CDC está orientada a dispositivos con cierta capacidad computacional y de memoria (necesita 2Mb de memoria).
  - CDC usa una Máquina Virtual Java similar en sus características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo.

Perfiles (*Profiles*)

- Un perfil añade las clases específicas para cada una de las configuraciones de J2ME.



El más extendido es el MIDP (*Mobile Information Device Profile*), diseñado para funcionar con CLDC

Mobile Information Device Profile (MIDP)

- Las clases que contiene este perfil son:
  - javax.microedition.midlet: se ocupa del ciclo de vida de la aplicación
  - javax.microedition.lcdui: interfaz de usuario
  - javax.microedition.rms: sistema de mantenimiento de registros (Record Management System) usado para guardar información
  - javax.microedition.io: clases para usar redes
  - java.lang: clases de lenguaje
  - java.util: clases de utilidades

El ciclo de vida del MIDP

- Está bien definido y ayuda al MIDlet (aplicación) a coexistir con otros programas en el MIDP.
- Sus fases son:
  - **Recuperación**: se consigue la aplicación desde la fuente (IRDA, Bluetooth, INET, etc.).
  - **Instalación**: la aplicación se instala en el MID. La implementación de MIDP verifica que no viola la seguridad del MID.
  - **Lanzamiento**: ejecución de el MIDlet en la KVM.
  - **Gestión de la versión**: permite el control de versiones de los MIDlets, así como su actualización.
  - **Borrado**: el MIDlet es eliminado del dispositivo.

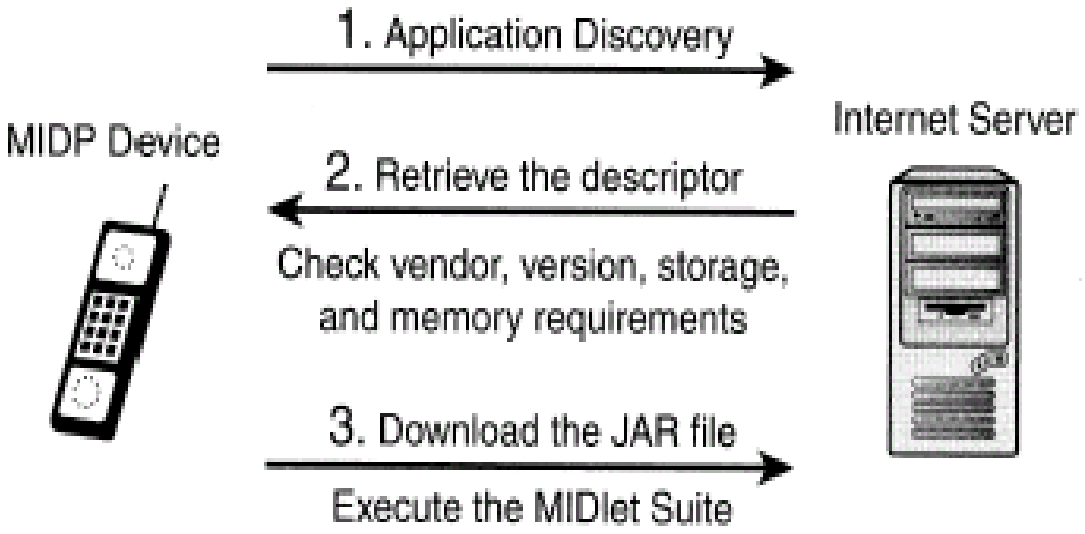
Aplicaciones MIDP: MIDlets

- Un MIDlet es una aplicación escrita especialmente para el perfil MIDP de J2ME
- Se organizan en ficheros .jar, que contienen la aplicación en si, el resto de ficheros necesarios para que ésta funcione (imágenes, sonidos, etc.) el descriptor y el manifiesto:
  - **Manifesto**: contiene una descripción del contenido del fichero (nombre, versión, etc) y una entrada por cada MIDlet que compone la aplicación.
  - **Descriptor:** proporciona la información requerida por el Application Management Software (programa que gestiona las descargas de aplicaciones) para comprobar si la aplicación se puede ejecutar en el dispositivo.

Aplicaciones MIDP: MIDlets (2)

- El MIDlet es una clase que extiende la clase del MID: javax.microedition.MIDlet.
- Un MIDlet permanente debe ser descargado y escrito en el almacenamiento persistente del MID (ROM o EEPROM).
- Un usuario puede ejecutar el MIDlet repetidamente sin necesidad de volverlo a descargar.
- Una aplicación para MID tiene las siguientes propiedades:
  - Está concebida para un dispositivo con unas características específicas).
  - Se instala, interactua y se borra, por ejemplo decargándolo via redes inalambricas desde un servidor web.

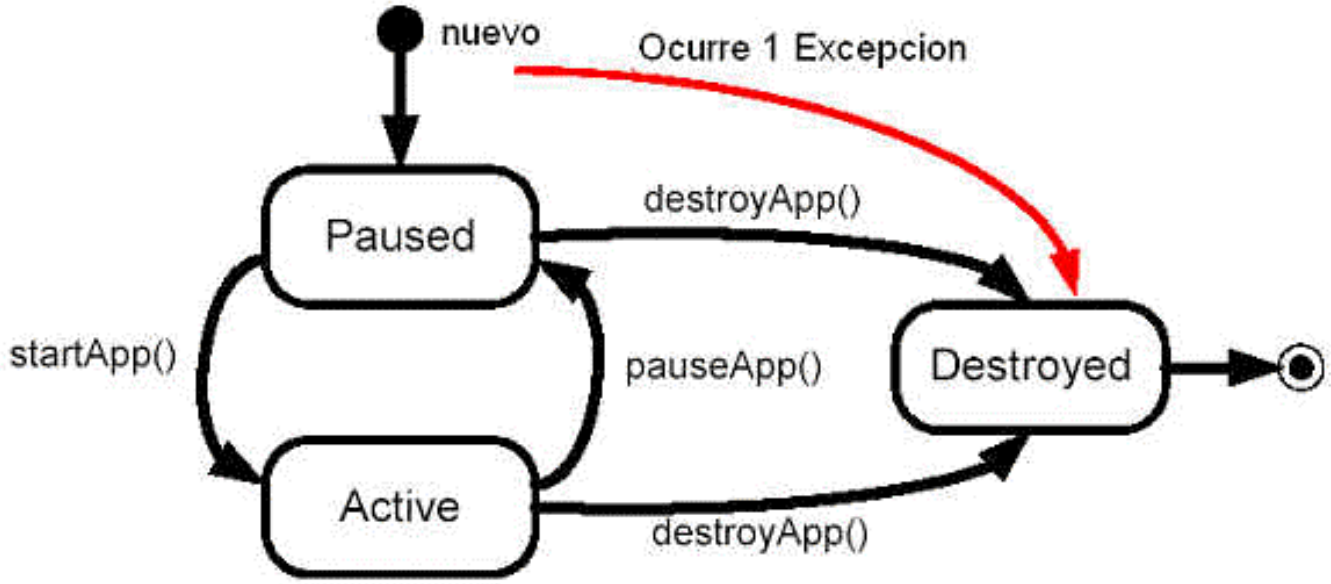
Aplicaciones MIDP:



Aplicaciones MIDP: Métodos básicos

- **startApp():** activa la aplicación. El método setCurrent() debería ser llamado aquí si no fue llamado antes. setCurrent() define que display será visible al usuario, ya que solo un display puede ser visible al mismo tiempo.
- **pauseApp():** pausa la aplicación, cuando una aplicación es reactivada, puede aparecer con otro display usando el método setCurrent().
- **destroyApp():** libera y destruye todos los recursos usados por la aplicaciones, incluidos los componentes del interfaz de usuario.

El ciclo de vida del MIDlet



NOTA: Este ciclo de vida exige la preverificación o emulación del programa antes de su ejecución en el dispositivo.

Comunicación y Redes

- MIDP presenta una implementación del protocolo HTTP, que puede ser usada para realizar conexiones a Internet y redes usando TCP/IP y otros protocolos que no usan IP, como WAP o I-mode usando un gateway que facilita su acceso a servidores http.

- Gracias al CLDC Generic Connection Framework, MIDP soporta la tecnología cliente/servidor y el uso de datagramas.

- MIDP 1.0 solo implementa el protocolo HTTP 1.1. Aunque el teléfono pueda usar sockets, MIDP 1.0 no tiene esta opción, así que habría que desarrollar nuestra propia clase. La versión MIDP 2.0 contiene una implementación de sockets y datagramas.

Comunicación y Redes

- **Generic Connection Framework**

  Generic Connecition Framework hace uso de la clase Connector para abrir o crear una conexión a Internet:

  - HTTP:

    Connection hc = Connector.open("http://www.fib.upc.es")

  - Socket:

    Connection sc = Connector.open("socket://localhost:9000")

  - Datagram

    Connection dc = Connector.open("datagram://http:// www.fib.upc.es:9000")

  - Puerto Serie:

    Connection cc = Connector.open("comm:0;baudrate=9000")

  - Fichero I/O:

    Connection fc = Connector.open("file:/archivo.dat")

Ejemplo: ConsultarHora.java

- Ejemplo de un MIDlet que establece una conexión utilizando sockets con el servidor “time.nist.gov” para consultar la hora.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;

public class ConsultarHora extends MIDlet implements CommandListener {
private Command exitCommand;
private Display display;

public ConsultarHora() {
display = Display.getDisplay(this);
exitCommand = new Command("Exit", Command.SCREEN, 1);
}
```

Ejemplo: ConsultarHora.java (2)

```
public void startApp() {
    StringBuffer stringHora = new StringBuffer();
    SocketConnection client;
     try {
        client = (SocketConnection) Connector.open("socket://time.nist.gov:13");
        InputStream is = client.openInputStream();
        int c = 0;
        while((c = is.read()) != -1) {
        stringHora.append((char) c);
         }
        is.close();
        client.close();
     }
     catch(IOException e) {}
```

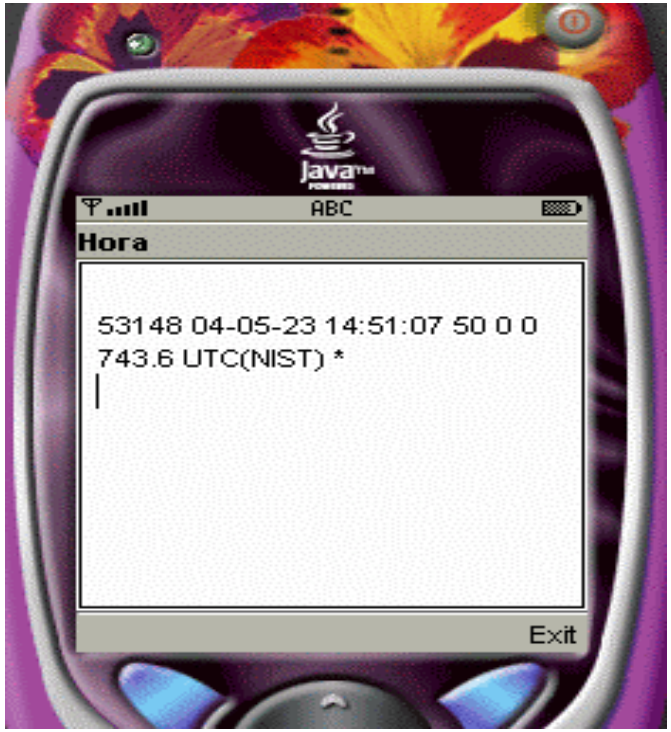
Ejemplo: ConsultarHora.java (3)

```
    TextBox t = new TextBox("Hora", stringHora.toString(), 256, 0);
    t.addCommand(exitCommand);
    t.setCommandListener(this);
    display.setCurrent(t);
}

public void pauseApp() { }
public void destroyApp(boolean unconditional) { }
public void commandAction(Command c, Displayable s) {
if (c == exitCommand) {
   destroyApp(false);
   notifyDestroyed();
  }
 }
}
```

Ejemplo: ConsultarHora.java ()

- Resultado de la ejecución:



J2ME: pros y contras

- **Pros:**
  - Admite las infraestructuras ya existentes (compatible con la tecnología WAP) y es compatible con los protocolos estándar de transmisión de datos TCP/IP, UDP, IMAP (mail).
  - Flexibilidad: una vez programada, la aplicación puede funcionar sobre una amplia variedad de dispositivos.
  - Interfaz: las aplicaciones ofrecen mejoras en el aspecto gráfico (colores, animaciones, menús desplegables).
- **Contras**:
  - Exige la preverificación o emulación del código para prever las excepciones que se puedan producir antes de la ejecución.
  - Es misión del programador liberar la memoria, ya que no se dispone de un Garbage Collector

Bibliografia

- Página de SUN

  http://java.sun.com/j2me/index.jsp

- Java a tope: J2ME (http://www.lcc.uma.es/~galvez/J2ME.html)

  Dpto. de Lenguajes y Ciencias de la Computación:

  E.T.S. de Ingeniería Informática: Universidad de Málaga

- Otras páginas de referencia:

  http://psoler.8m.net/J2ME/indexJ2ME.html

  http://www.todosymbian.com

  http://www.microjava.com

  http://www-106.ibm.com/developerworks/java/library/j-j2me/