Programación de dispositivos móviles con Java (J2ME)

1. Dispositivos móviles
Características de los dispositivos

Dispositivos conectados

- **Dispositivos con pequeños ordenadores embebidos**
- **Tienen la capacidad de conectarse a la red**
  - Dispositivos móviles de información
    - MIDs: Mobile Information Devices
    - Teléfonos móviles, PDAs, etc
  - Descodificadores de TV (set top boxes)
  - Electrodomésticos
  - Impresoras de red
  - Routers
  - etc

sin interfaz

Limitaciones de los dispositivos

- **Escasa memoria**
  - Normalmente 128-512Kb de RAM
- **CPU lenta**
  - 1-10 MIPS (Pentium 4 3.0GHz, ~10000 MIPS)
- **Pequeña pantalla**
  - 96x65 – 178x201 píxeles, monocromo – 65536 colores
- **Dispositivos de entrada restringidos**
  - Teclado 0-9, #, *
- **Fuentes de texto limitadas**
  - Normalmente sólo una fuente

Características de los MIDs

96x65
Monocromo
164kb

101x64
Monocromo
150kb

178x201
4096 colores
1,4mb

128x128
4096 colores
200kb

640x200
4096 colores
8mb

240x320
65536 colores
64mb

Redes de telefonía celular

- **1G: Red analógica**
  - Sólo voz
- **2G: Red digital**
  - Voz y datos
  - GSM (Global System for Mobile communications) en Europa
  - Red no IP
    - Protocolos WAP
    - Un gateway conecta la red móvil (WAP) a la red Internet (TCP/IP)
  - Conmutación de circuitos (Circuit Switched Data, CSD)
    - 9’6kbps
    - Se ocupa un canal de comunicación de forma permanente
    - Se cobra por tiempo de conexión

Redes de telefonía celular (2)

- **2,5G: GPRS (General Packet Radio Service)**
  - Transmisión de paquetes
    - No ocupa un canal de forma permanente
    - Hasta 144kbps teóricamente (40kbps en la práctica)
    - Cobra por volumen de información transmitida
  - Se implementa sobre la misma red GSM
- **3G: Banda ancha**
  - Red UMTS (Universal Mobile Telephony System)
  - Entre 384kbps y 2Mbps
  - Servicios multimedia: Videoconferencia, TV, música, etc
  - Transmisión de paquetes
  - Requiere nueva infraestructura

Conectividad de los MIDs

- **Los dispositivos deben conectarse para descargar las aplicaciones**
  - Over The Air (OTA)
    - Conexión a Internet por la red móvil (GSM, GPRS, UMTS)
  - Cable serie o USB
    - Conexión física
  - Infrarrojos
    - Los dispositivos deben verse entre si
  - Bluetooth
    - Ondas de radio (10 metros de alcance)
    - Alta velocidad (723kbit/s)

WAP (Wireless Application Protocol)

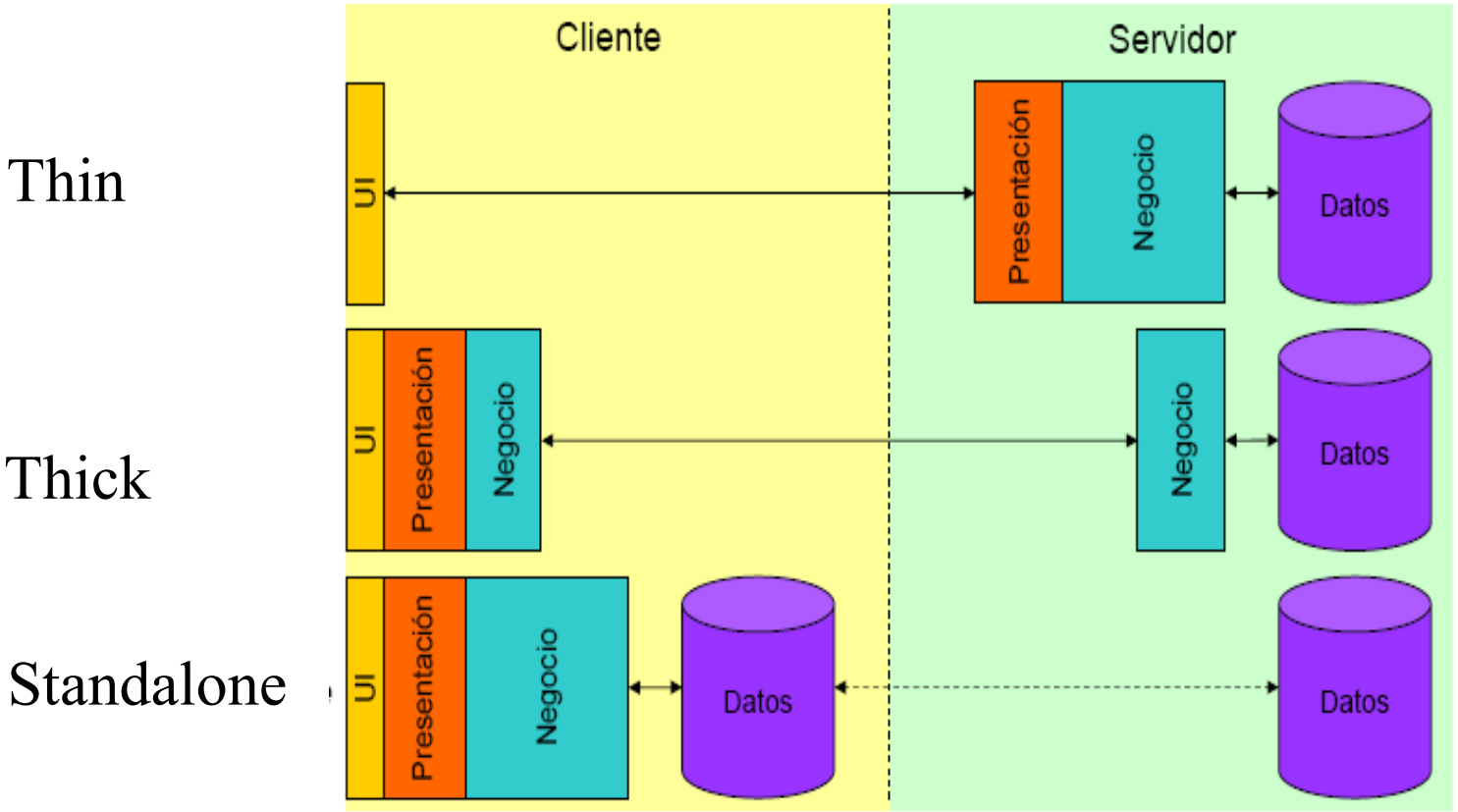
- **Estándar para aplicaciones de Internet en móviles**
  - Define un conjunto de protocolos
- **Objetivo**
  - Permitir acceder a aplicaciones de Internet desde móviles
  - Poder realizar aplicaciones independientes del fabricante, operador, y tipo de red de los dispositivos
- **Documentos en lenguaje WML**
  - Lenguaje basado en XML orientado a móviles
- **WAP NO ES J2ME!!!**

2. Clientes para móviles

Tipos de clientes para móviles

Tipos de cliente

- **Según el reparto de la carga entre cliente/servidor**



Clientes thin

- **Todo el procesamiento se realiza en el servidor**
  - El cliente sólo se ocupa de la interfaz de usuario (UI)
- **El cliente normalmente se compone de:**
  - Navegador
  - Documento web (p.ej. HTML)
- **Descarga documentos y los muestra en un navegador**
- **Los documentos HTML no son adecuados para móviles**
  - Se definen nuevos tipos de documento (WML, XHTML-MP...)
  - Se muestran en un navegador especial (Microbrowser)

Tecnologías para clientes thick

- **Sistema operativo**
  - Symbian OS, Palm OS, Windows Pocket PC, etc
  - Poco portable
  - Requiere aprender nuevas APIs
  - Problemas de seguridad
- **Runtime Environments**
  - **J2ME**
    - Soportado por gran cantidad de dispositivos
    - Existe una gran comunidad de desarrolladores Java
  - BREW
    - Soportado por pocos dispositivos
    - Requiere aprender una nueva API
  - NET Compact Framework (Pocket PC, Windows CE y Windows Mobile)

Clientes thick

- **La aplicación se descarga e instala en el cliente**
  - Se ejecuta de forma local
  - Trabaja de forma coordinada con el servidor
  - Realiza en el cliente todo el procesamiento posible
- **Aplicaciones dedicadas**
  - Para una tarea concreta
- **Sólo necesita intercambiar información, no presentación**
  - Los documentos web (p.e. HTML) no sirven
- **Se puede comunicar mediante diferentes protocolos**
  - Servicios Web, RPC ...

¿Thick o Thin?

- **Thin**
  - Requiere conectar a la red para descargar cada documento
  - Velocidad de descarga lenta en móviles
  - Limitado a las posibilidades del navegador
  - Fácil de mantener
- **Thick**
  - Mantenimiento costoso
  - Interfaz de usuario (UI) más flexible
  - Minimiza el tráfico en la red
  - Puede funcionar sin conexión
  - Mayor rendimiento multimedia

3. Introducción a J2ME
Arquitectura de la plataforma J2ME

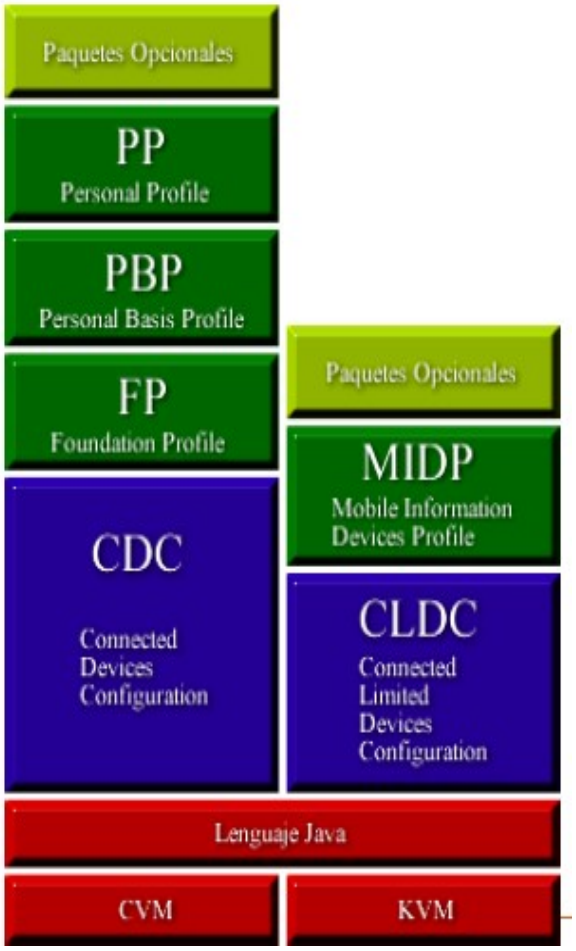
Java 2 Micro Edition

- **Edición de la plataforma Java 2 para dispositivos móviles**
- **Independiente de la plataforma**
  - Adecuado para programar dispositivos heterogéneos
- **Gran comunidad de desarrolladores Java**
  - Los programadores Java podrán desarrollar aplicaciones para móviles de forma sencilla
  - No hace falta que aprendan un nuevo lenguaje
- **Consiste en un conjunto de APIs**
  - Una sola API es insuficiente para la variedad de tipos de dispositivos existente
  - Cada API se dedica a una distinta familia de dispositivos

Capas de J2ME

- **Configuraciones (CDC, CLDC)**
  - API común para todo un gran conjunto de dispositivos
  - Elementos básicos del lenguaje
- **Perfiles (MIDP, DoJa)**
  - API que cubre las características propias de una familia de dispositivos concreta
  - P.ej, para acceder a la pantalla de los teléfonos móviles
- **Paquetes opcionales**
  - APIs para características especiales de ciertos dispositivos
  - P.ej, para acceder a la cámara de algunos teléfonos móviles

APIs de J2ME



Configuraciones

- **CDC: Dispositivos conectados**
  - Sobre JVM
- **CLDC: Dispositivos conectados limitados**
  - Sobre KVM (limitada)
  - Paquetes:
    - java.lang
    - java.io
    - java.util
    - javax.microedition.io

CLDC

- **Dispositivos con memoria del orden de los KB**
  - Puede funcionar con sólo 128KB
  - Teléfonos móviles y PDAs de gama baja
- **Se ejecuta sobre KVM (Kilobyte Virtual Machine)**
- **Muy limitada, para poder funcionar con escasos recursos**
  - P.ej, no soporta reales (tipos float y double)
- **Perfil MIDP**
  - Dispositivos móviles de información (MIDs)
  - Paquetes:
    - javax.microedition.lcdui
    - javax.microedition.midlet
    - javax.microedition.rms
- **Perfil DoJa (DoCoMo Java)**
  - Dispositivos iMode

Paquetes opcionales

- **Wireless Messaging API (WMA)**
  - Envío y recepción de mensajes cortos (SMS)
- **Mobile Media API (MMAPI)**
  - Multimedia, reproducción y captura de video y audio
- **Bluetooth API**
  - Permite establecer conexiones vía Bluetooth
- **J2ME Web Services**
  - Invocación de servicios web desde dispositivos móviles
- **Mobile 3D Graphics**
  - Permite incorporar gráficos 3D

4. Introducción a J2ME

Arquitectura de la plataforma J2ME

MIDlets

- **Las aplicaciones para dispositivos MIDP se denominan MIDlets**
- **Estas aplicaciones se distribuyen como una suite de MIDlets, que se compone de:**
  - Fichero JAD
    - Fichero ASCII
    - Descripción de la aplicación
  - Fichero JAR
    - Aplicación empaquetada (clases y recursos)
    - Contiene uno o más MIDlets
    - Contiene un fichero MANIFEST.MF con información sobre la aplicación

Fichero JAD

- **Ejemplo de fichero JAD:**

```
MIDlet-Name: HelloWorld
MIDlet-Version: 1.0.0
MIDlet-Vendor: Escola del Treball
MIDlet-Description: Aplicación de ejemplo para móviles
MIDlet-Jar-Size: 16342
MIDlet-Jar-URL: helloworld.jar
```

- **En un dispositivo real es importante que** MIDlet-Jar-Size **contenga el tamaño real del fichero JAR**
- **Si publicamos la aplicación en Internet,** MIDlet-Jar-URL **deberá apuntar a la URL de Internet donde se encuentra publicado el fichero JAR.**

Fichero MANIFEST.MF

- **Ejemplo de fichero MANIFEST.MF:**

MIDlet-Name: HelloWorld

MIDlet-Version: 1.0.0

MIDlet-Vendor: Escola del Treball

MIDlet-Description: Aplicaciones ejemplo para móviles

MicroEdition-Configuration: CLDC-1.0

MicroEdition-Profile: MIDP-1.0

MIDlet-1: HelloWorld1, /icons/HelloWorld1.png, es.et.helloWorld.helloWorld1

MIDlet-2: HelloWorld2, /icons/ts.png, es.et.helloWorld.helloWorld2

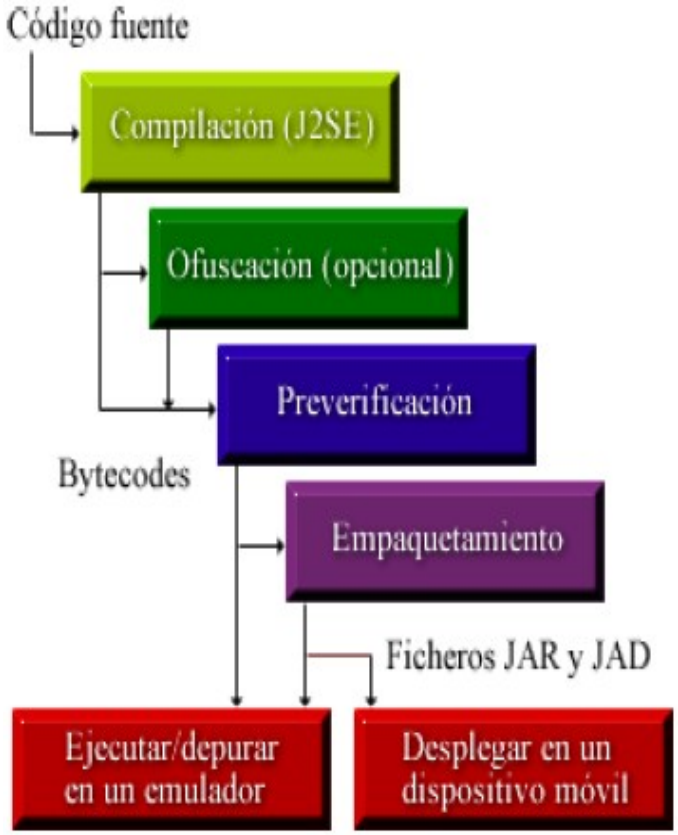
MIDlet-3: HelloWorld3, /icons/panj.png, es.et.helloWorld.helloWorld3

- **Si el dispositivo real no soporta la configuración o el perfil indicados, se producirá un error en la instalación**

Software gestor de aplicaciones

- **Los dispositivos móviles con soporte para Java tienen instalado un software gestor de aplicaciones**
  - AMS: Application Management Software
- **Gestiona las aplicaciones Java:**
  - Descarga
    - Descarga primero el fichero JAD y muestra los datos de la aplicación
    - Si la aplicación es compatible y el usuario acepta, descarga el JAR
- **Instalación**
- **Actualización**
- **Desinstalación**
- **Ejecución**
  - Es el contenedor que da soporte a los MIDlets
    - Contiene la KVM sobre la que se ejecutarán las aplicaciones
    - Soporta la API de MIDP
  - Controla el ciclo de vida de los MIDlets que ejecuta

Pasos del proceso



- **Compilar**
  - Utilizar como clases del núcleo la API de MIDP
- **Ofuscar (optativo)**
  - Reducir tamaño de los ficheros
  - Evitar descompilación
- **Preverificar**
  - Reorganizar el código para facilitar la verificación a la KVM
  - Comprobar que no se usan características no soportadas por KVM
- **Empaquetar**
  - Crear ficheros JAR y JAD
- **Probar**
  - En emuladores o dispositivos reales

Sun Wireless Toolkit (WTK)

- **Incluye las APIs necesarias**
  - MIDP y APIs adicionales
- **Incluye herramientas que no están en Java 2 SDK**
  - Preverificador
- **Incluye emuladores para probar las aplicaciones**
  - Se puede integrar con emuladores proporcionados por terceros (Nokia, Ericsson, etc).
- **Facilita el proceso de construcción de aplicaciones**
  - Entorno de creación de aplicaciones
- **Es necesario contar con Java 2 SDK para compilar y empaquetar**

5. MIDlets
Ciclo de vida y propiedades

Ciclo de vida

- **La clase principal de la aplicación debe heredar de MIDlet**
- **Componente que se ejecuta en un contenedor**
  - AMS = Software Gestor de Aplicaciones
- **El AMS controla su ciclo de vida**

Esqueleto de un MIDlet

```
import javax.microedition.midlet.*;
    public class MiMIDlet extends MIDlet {
      protected void startApp ( ) throws
    MIDletStateChangeException{
            // Estado activo -> comenzar
      }
      protected void pauseApp ( ) {
            // Estado pausa -> detener hilos
      }
      protected void destroyApp (boolean incondicional)
                        throws MIDletStateChangeException {
            // Estado destruido -> liberar recursos
      }
    }
```

Propiedades

- **Leer propiedades de configuración (JAD)**

```
String valor = getAppProperty(String key);
```

- **Salir de la aplicación**

```
public void salir() {
  try {
        destroyApp(false);
        notifyDestroyed();
  } catch(MIDletStateChangeException e) { }
}
```

6. Interfaz gráfica
Display y pantallas

Display

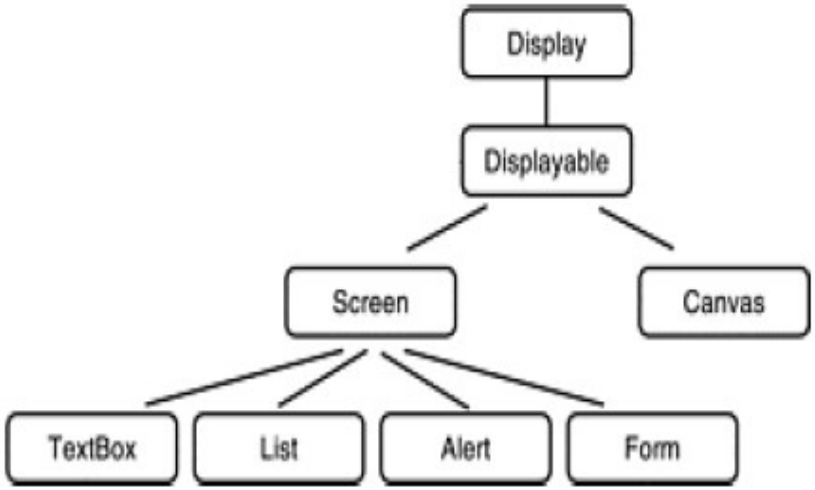
- **La interfaz gráfica se realizará con la API LCDUI**
  - LCDUI = Limited Connected Devices User Interface
  - Se encuentra en el paquete javax.microedition.lcdui
- **El display representa el visor del móvil**
  - Nos permite acceder a la pantalla
  - Nos permite acceder al teclado
- **Cada MIDlet tiene asociado uno y sólo un display**

```
Display display = Display.getDisplay(midlet);
```

- **El display sólo mostrará su contenido en la pantalla y leerá la entrada del teclado cuando el MIDlet esté en primer plano**

Componentes displayables

- **Son los elementos que pueden mostrarse en el display**
- **El display sólo puede mostrar un displayable simultáneamente**



- **Establecemos el displayable a mostrar con**

```
Display.setCurrent ( displayable);
```

Alto nivel vs Bajo nivel

- **Podemos distinguir dos APIs:**
  - Alto nivel
    - Componentes predefinidos: listas, formularios, campos de texto
    - Se implementan de forma nativa
    - Aplicaciones portables
    - Adecuados para front-ends de aplicaciones corporativas
  - Bajo nivel
    - Componentes personalizables: canvas
    - Debemos especificar en el código cómo dibujar su contenido
    - Tenemos control sobre los eventos de entrada del teclado
    - Se reduce la portabilidad
    - Adecuado para juegos

Campos de texto

```
TextBox tb = new TextBox("Contraseña",
             "", 8, TextField.ANY |
             TextField.PASSWORD);
Display d = Display.getDisplay(this);
d.setCurrent(tb);
```

Listas

```
List l = new List("Menu",
                  Choice.IMPLICIT);
l.append("Nuevo juego", null);
l.append("Continuar", null);
l.append("Instrucciones", null);
l.append("Hi-score", null);
l.append("Salir", null);
Display d =
  Display.getDisplay(this);
d.setCurrent(l);
```

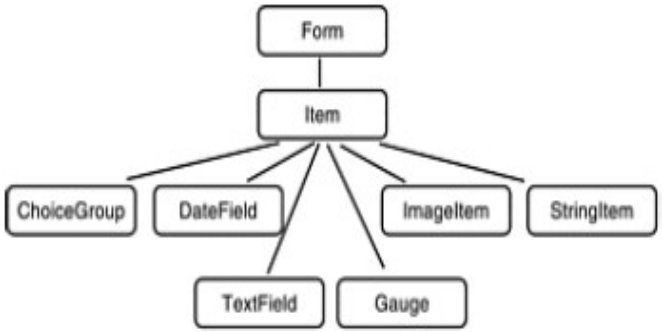
Implícita

Múltiple

Exclusiva

Formularios

```
Form f = new Form("Formulario");
Item itemEtiqueta = new StringItem(
        "Etiqueta:",
        "Texto de la etiqueta");
Item itemTexto = new TextField(
        "Telefono:", "", 8,
        TextField.PHONENUMBER);
Item itemFecha = new DateField(
        "Fecha",
        DateField.DATE_TIME);
Item itemBarra = new Gauge("Volumen",
        true, 10, 8);
f.append(itemEtiqueta);
f.append(itemTexto);
f.append(itemFecha);
f.append(itemBarra);
Display d = Display.getDisplay(this);
d.setCurrent(f);
```



Alertas

```
Alert a = new Alert("Error",
        "No hay ninguna nota
seleccionada",
        null, AlertType.ERROR);
Display d = Display.getDisplay(midlet);
d.setCurrent(a, d.getCurrent());
```

Imágenes en MIDP

- En muchos componentes podemos incluir imágenes
- Las imágenes se encapsulan en la clase Image
- El único formato reconocido por MIDP es PNG
- Las imagenes de puden obtener a partir de un ficheros contenido en el JAR

```
Image img = Image.createImage("/logo.png");
```

6. Comandos
Modelo de eventos e hilos

Comandos de entrada

- **La entrada de usuario se realiza mediante comandos**

Creación de comandos

- **Podemos crear comandos y añadirlos a un displayable**

```
TextBox tb = new TextBox("Login", "", 8, TextField.ANY);
Command cmdOK = new Command("OK", Command.OK, 1);
Command cmdAyuda = new Command("Ayuda", Command.HELP, 1);
Command cmdSalir = new Command("Salir", Command.EXIT, 1);
Command cmdBorrar = new Command("Borrar", Command.SCREEN, 1);
Command cmdCancelar = new Command("Cancelar", Command.CANCEL,
1);
tb.addCommand(cmdOK);
tb.addCommand(cmdAyuda);
tb.addCommand(cmdSalir);
tb.addCommand(cmdBorrar);
tb.addCommand(cmdCancelar);
Display d = Display.getDisplay(this);
d.setCurrent(tb);
```

Modelo de eventos

- **En Java para tratar los eventos se utilizan listeners**
- **Listener**
  - Componente que “escucha” un determinado evento
  - Cuando el evento sucede, se ejecuta el código que hayamos escrito en el listener
- **Para crear un listener**
  - Creamos una clase que implemente la interfaz correspondiente al tipo de listener deseado
    - Por ejemplo, CommandListener para “escuchar” la ejecución de comandos
  - Implementamos los métodos de la interfaz
    - Por ejemplo, commandAction
  - Registramos el listener en el componente del cuál queremos recibir eventos
- **Una vez registrado, permanece a la “escucha” de eventos**
  - Cada vez que se produzca un evento, se ejecutará el método correspondiente
    - Por ejemplo, si se pulsa un comando se ejecuta commandAction

Listener de comandos

- **Debemos crear un listener para dar respuesta a los comandos**

```
class ListenerLogin implements CommandListener {
  public void commandAction(Command c, Displayable d) {
        if(c == cmdOK) {
                // Aceptar
        } else if(c == cmdCancelar) {
                // Cancelar
        } else if(c == cmdSalir) {
                // Salir
        } else if(c == cmdAyuda) {
                // Ayuda
        } else if(c == cmdBorrar) {
                // Borra r
        }
  }
}
```

- **Registrar el listener en el displayable**

```
tb.setCommandListener(new ListenerLogin());
```