“Programando con J2ME”

Rogelio Ferreira Escutia

MIDlet mínimo

Primer MIDlet

- **Todos los MIDlets deben tener una clase principal que herede de la clase “javax.microedition.midlet", contenida en el API MIDP estándar.**

- **Esta clase define varios métodos, de los cuales los mínimos son los siguientes:**

  - **startApp() - Lanza el MIDlet**
  - **pauseApp() - Para el MIDlet**
  - **DestroyApp() - Destruye el MIDlet**

MIDlet mínimo

- **La estuctura mínima para un programa J2ME queda de la siguiente manera:**

```
import javax.microedition.midlet.*;

public class hola extends MIDlet {

    public void startApp() { }

    public void pauseApp() { }

    public void destroyApp(boolean unconditional) { }

}
```

Desplegar Información

Objeto Display (1)

- **Para hacer el primer programa que imprima un mensaje en pantalla requerimos varias cosas mas.**

- **Primero deberemos agregar la clase**

  “import javax.microedition.lcdui”

- **El cual contiene los métodos para el control de la pantalla del dispositivo y los elementos que permiten al usuario interactuar con la aplicación.**

- **Nunca se debe crear un objeto Display, obtendremos una referencia al objeto Display que nos proporciona el propio entorno.**

Objeto Display (2)

- **El manejo de la pantalla se hace dentro del constructor del MIDlet y se usa para crear la interfaz de usuario en el método startApp().**

- **Para lo anterior se requiere de una instancia del Display por cada MIDlet que se esté ejecutando en el dispositivo en cada momento**

```
Display.getDisplay(this).setCurrent(pantalla);
```

Clase Screen

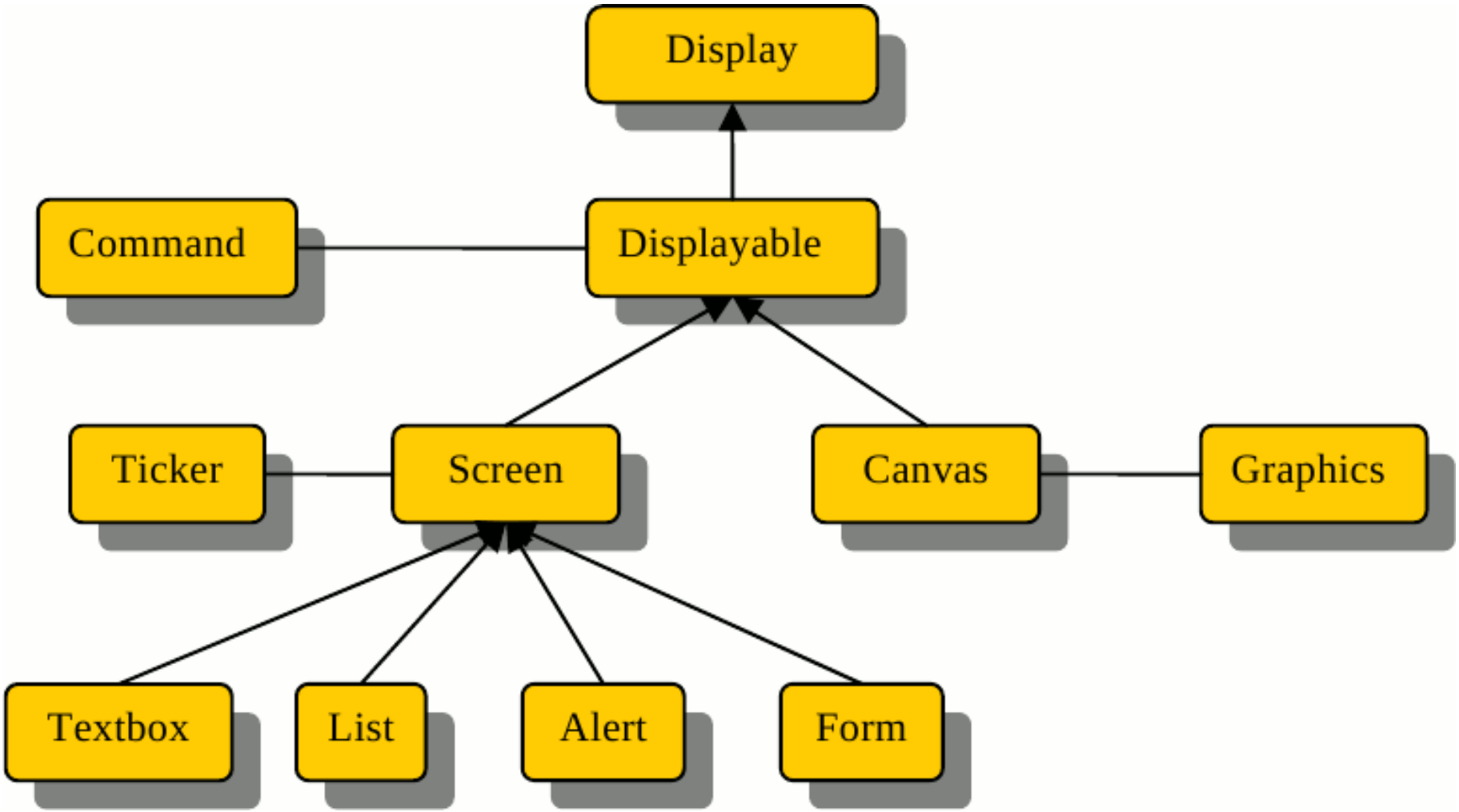
- **Mientras que del objeto “Display” sólo hay una instancia, del objeto “Screen” pueden existir varias.**
- **El objeto “Screen” es un componente de la GUI (Graphical User Interface) genérico que sirve como base para otros componentes.**
- **Estos objetos representan una pantalla entera de información. Sólo se puede mostrar una pantalla a la vez.**
- **La mayoría de los MIDlets usan subclases de la clase Screen, como Form, TextBox o List.**

Clase Canvas

- **Una clase similar a Screen es la clase Canvas, que pertenece al mismo paquete.**

- **Los objetos Canvas son usados para realizar operaciones gráficas directamete sobre la pantalla, como puede ser pintar líneas o imágenes.**

- **No se puede mostrar un objeto Canvas y un objeto Screen a la vez, pero se pueden alternar en la misma aplicación.**


Display
Command
Displayable
Ticker
Screen
Canvas
Graphics
Textbox
List
Alert
Form

Form (1)

- **Continuando con nuestro programa que imprime un mensaje en pantalla, utilizaremos un objeto que contendrá nuestra pantalla (que en nuestro caso se llama "pantalla"), por lo cual declaramos el objeto de tipo "Form" el cual contendrá el mensaje que deseamos imprimir:**

```
private Form pantalla;
```

Form (2)

- **Ahora dentro del constructor de nuestra clase principal inicializamos el objeto “pantalla” que previamente habiamos declarado y le agregamos un texto que se convertirá en el título de nuestra ventana:**

```
pantalla = new Form("Titulo - Hola Mundo");
```

Form (3)

- **Por medio del método “append” le agregamos un mensaje a nuestra pantalla utilizando el objeto de tipo Form que se llama “pantalla” y que previamente declaramos e inicializamos:**

```
pantalla.append(new StringItem(null, "Hola Mundo desde un celular"));
```

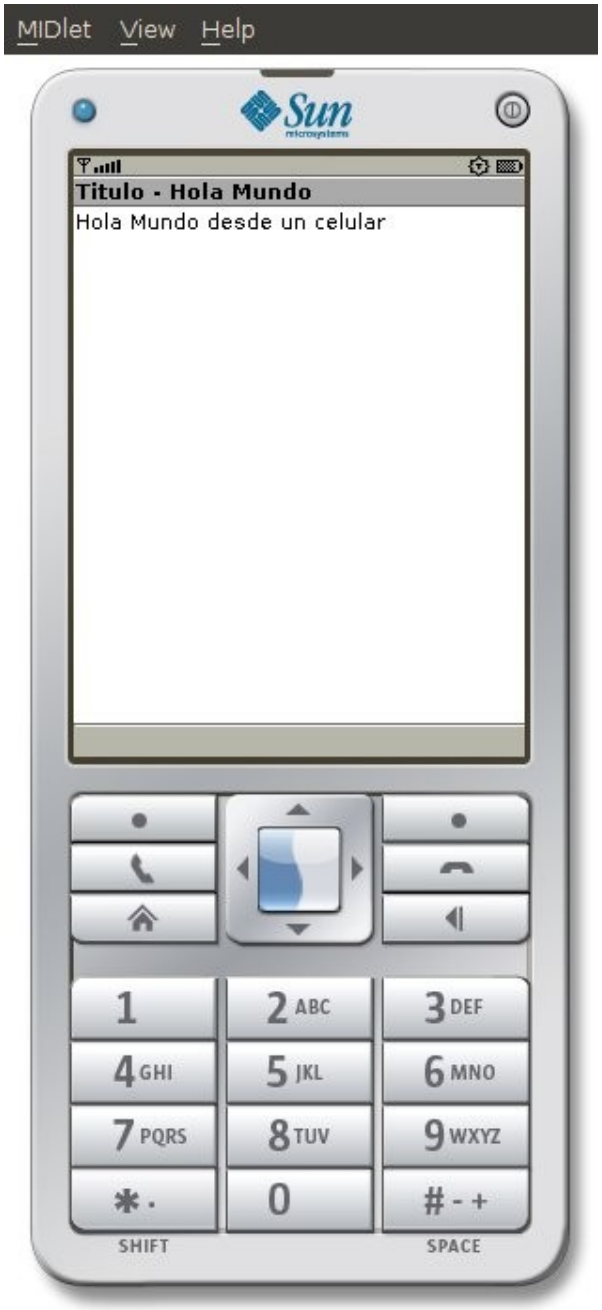
startApp()

- **Dentro de nuestro método “starApp” asignamos a nuestro objeto “Display” el objeto que nos definirá nuesrta primera pantalla de arranque:**

```
Display.getDisplay(this).setCurrent(pantalla);
```

- **El código completo queda de la siguiente manera:**

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

public class hola extends MIDlet {

    private Form pantalla;

    public hola() {
        pantalla = new Form("Titulo - Hola Mundo");
        pantalla.append(new StringItem(null, "Hola Mundo desde un celular"));
    }

    public void startApp() {
        Display.getDisplay(this).setCurrent(pantalla);
    }

    public void pauseApp() { }

    public void destroyApp(boolean unconditional) { }

}
```



- **El programa corriendo en el simulador que trae el Java Wireless Toolkit.**

Manejo de Eventos

CommandListener

- **El programa anterior no tiene interacción con el usuario, es decir, no detecta ninguna señal de entrada por parte del usuario hacia el dispositivo, ya que sólo imprime un mensaje.**

- **Ni siquiera tiene un botón de “salir”, por lo que lo implementaremos en el siguiente programa. Primero, dentro de nuestra clase principal implementaremos el “CommandListener” que nos permitirá “escuchar” los eventos que se generen.**

public class hola extends MIDlet implements CommandListener

commandAction

- **Ahora requerimos implementar el método "commandAction" que se encargará de recibir los eventos que se generen y procesar las acciones necesarias.**

public void commandAction(Command c, Displayable s)

notifyDestroy()

- **Dentro del “commandAction” se invoca a “notifyDestroy” que se encargará de enviar un mensaje al sistema de que se ha terminado el programa y que proceda con el borrado de los objetos utilizados (recordar que no existe el “Garbagge Collector” como en el J2SE):**

```
notifyDestroyed();
```

Botones

- **Agregamos a nuestra pantalla un botón “Salir” que se encargará de generar la acción de “salir” de la aplicación:**

```
pantalla.addCommand(new Command("Salir", Command.EXIT, 0));
```

Asignar el manejador del evento

- **Por último, falta asignar quién será el encargado de monitorear los eventos que se generen y para ello seleccionamos la pantalla que utilizamos y le asignamos con “setCommandListener” el manejo del evento:**

```
pantalla.setCommandListener(this);
```

- **El código completo queda de la siguiente manera:**

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

public class hola extends MIDlet implements CommandListener {
	private Form pantalla;
	public hola() {
		pantalla = new Form("Titulo - Hola Mundo");
		pantalla.append(new StringItem(null, "Hola Mundo desde un celular"));
		pantalla.addCommand(new Command("Salir", Command.EXIT, 0));
		pantalla.setCommandListener(this);
	}
	public void startApp() {
		Display.getDisplay(this).setCurrent(pantalla);
	}
	public void pauseApp() { }
	public void destroyApp(boolean unconditional) { }
	public void commandAction(Command c, Displayable s) {
		notifyDestroyed();
	}
}
```

- **El programa corriendo en el simulador que trae el Java Wireless Toolkit.**

Entrada de Datos

TextField

- **Por medio de “TextField” podemos capturar información que introduce el usuario por medio del teclado.**

Ejemplo

- **Por medio del siguiente ejemplo podemos observar el funcionamiento de TextField.**

- **Se utilizarán 2 pantallas, en la primera se preguntará el nombre de una persona, y por medio del botón “continuar” pasará a la segunda pantalla en donde se imprimirá un mensaje y el nombre que se capturó en la primer pantalla. También en esta segunda pantalla agregaremos un botón de “salir” para terminar con el programa.**

Declaración de objetos

- **Primero declaramos los objetos que vamos a utilizar:**

  Un objeto de tipo Display (pantalla).
  2 pantallas (Form) para desplegar información.
  Un campo TextField para entrada de datos.
  2 botones.

```
private Display pantalla;
private Form pantalla_preguntar, pantalla_imprimir;
TextField nombre;
private Command continuar, salir;
```

Objeto Display

- **En el constructor inicializamos el objeto Display:**

```
pantalla = Display.getDisplay(this);
```

Primer pantalla

- **En el constructor inicializamos la primer pantalla, preguntamos el nombre, inicializamos el botón “continuar” y ambos objetos los agregamos a nuestra pantalla, y por último asignamos el manejo de eventos:**

```
pantalla_preguntar = new Form("Pantalla 1 > Preguntar Nombre");
nombre = new TextField("Escribe tu nombre:","",20,TextField.ANY);
pantalla_preguntar.append(nombre);
continuar = new Command ("Continuar",Command.OK,1);
pantalla_preguntar.addCommand(continuar);
pantalla_preguntar.setCommandListener(this);
```

Segunda pantalla

- **Iguamente en el constructor inicializamos la segunda pantalla, inicializamos el botón “salir” y lo agregamos a nuestra pantalla, y por último asignamos el manejo de eventos:**

```
pantalla_imprimir = new Form("Pantalla 2 > Imprimir mensaje");
        salir = new Command("Salir",Command.EXIT,1);
        pantalla_imprimir.addCommand(salir);
        pantalla_imprimir.setCommandListener(this);
```

Manejo de Eventos

- **En el commandAction verificamos cual botón se oprimió (continuar ó salir).**

- **En caso de oprimir “continuar” generamos un texto que agregamos a nuestra segunda pantalla y nos cambiamos a esta pantalla. En caso de oprimir “salir” mandamos una notificación de terminación del programa.**

```
public void commandAction(Command c, Displayable s) {
    if (c == continuar) {
        pantalla_imprimir.append(new StringItem(null, "hola
"+nombre.getString()+"!!!"));
        pantalla.setCurrent(pantalla_imprimir);
    }
    if (c == salir) {
        destroyApp(false);
        notifyDestroyed();
    }
```

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class nombre extends MIDlet implements CommandListener {
	private Display pantalla;
	private Form pantalla_preguntar, pantalla_imprimir;
	TextField nombre;
	private Command continuar, salir;
	public nombre() {
		pantalla = Display.getDisplay(this);
		pantalla_preguntar = new Form("Pantalla 1 > Preguntar Nombre");
		nombre = new TextField("Escribe tu nombre:","",20,TextField.ANY);
		pantalla_preguntar.append(nombre);
		continuar = new Command ("Continuar",Command.OK,1);
		pantalla_preguntar.addCommand(continuar);
		pantalla_preguntar.setCommandListener(this);

		pantalla_imprimir = new Form("Pantalla 2 > Imprimir mensaje");
		salir = new Command("Salir",Command.EXIT,1);
		pantalla_imprimir.addCommand(salir);
		pantalla_imprimir.setCommandListener(this);
	}
```

```
	public void startApp() {
		pantalla.setCurrent(pantalla_preguntar);
	}

	public void pauseApp() { }

	public void destroyApp(boolean unconditional) { }

	public void commandAction(Command c, Displayable s) {
		if (c == continuar) {
			pantalla_imprimir.append(new StringItem(null, "hola
"+nombre.getString()+"!!!"));
			pantalla.setCurrent(pantalla_imprimir);
		}
		if (c == salir) {
			destroyApp(false);
			notifyDestroyed();
		}
	}

}
```

Estructuras de Control

Rogelio Ferreira Escutia

Instituto Tecnológico de Morelia
Departamento de Sistemas y Computación

Correo: ***rogeplus@gmail.com***
rferreir@itmorelia.edu.mx
Página Web: ***http://antares.itmorelia.edu.mx/~kaos/***
http://www.xumarhu.net/
Twitter: ***http://twitter.com/rogeplus***
Facebook: ***http://www.facebook.com/group.php?gid=155613741139728***