Introducción a Java para MIDs - Ejercicios

Índice

1. Primeros pasos con Wireless Toolkit

Abrir la herramienta `Wireless Toolkit 2.5`. Abrir desde ella alguna de las aplicaciones de ejemplo que tenemos disponibles (`UIDemo`, `demos` o `games`).

a) Probar a compilar estas aplicaciones y ejecutarlas en distintos emuladores (creando el paquete en caso necesario). Observar las diferencias que encontramos de unos emuladores a otros.

b) Consultar los ficheros JAD y `MANIFEST.MF` de las aplicaciones probadas. Identificar los elementos encontrados en estos ficheros.

c) Cargar el monitor de red y el monitor de memoria para monitorizar la utilización de recursos que realizan las aplicaciones que estamos probando.

2. Ejecución de aplicaciones vía OTA

Cargar alguna de las aplicaciones de ejemplo, pero esta vez vía OTA. Seguir los pasos necesarios para descargar, instalar y ejecutar la aplicación en el emulador.

3. Crear un nuevo proyecto

Crear un nuevo proyecto con Eclipse ME. El proyecto se llamará `PruebaAplicacion` y el MIDlet principal será `es.ua.jtech.ajdm.prueba.MIDletPrueba`.

4. Hola Mundo!

Vamos a hacer nuestra primera aplicación *"Hola mundo"* en J2ME. Para ello debemos:

a) Abrir el proyecto `PruebaAplicacion` creado en el ejercicio anterior. Si no lo tuviesemos creado, crear el proyecto, cuyo MIDlet principal será `es.ua.jtech.ajdm.prueba.MIDletPrueba`.

b) Una vez abierto introduciremos en la clase del MIDlet principal el siguiente código:

```
package es.ua.jtech.ajdm.prueba;

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

public class MIDletPrueba extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
        Form f = new Form("Hola mundo");
        Display d = Display.getDisplay(this);
```

```
        d.setCurrent(f);
    }

    protected void pauseApp() {
    }

    protected void destroyApp(boolean incondicional)
        throws MIDletStateChangeException    {
    }
}
```

c) Guardar el fichero y desde Eclipse ME compilar y ejecutar la aplicación en emuladores, comprobando que funciona correctamente.

d) Modificar el ejemplo para hacerlo parametrizable. Ahora en lugar de mostrar siempre el mensaje *"Hola mundo"*, tomaremos el mensaje a mostrar del parámetro `msg.bienvenida`. Crear este parámetro dentro del fichero JAD, y leerlo dentro del MIDlet para mostrarlo como título del formulario.

5. Recursos

Vamos a añadir recursos a nuestra aplicación. Mostraremos una imagen en la pantalla, introduciendo el siguiente código en el método `startApp` de nuestro MIDlet:

```
protected void startApp() throws MIDletStateChangeException {
    Form f = new Form("Hola mundo");
    try {
        f.append(Image.createImage("/logo.png"));
    } catch(Exception e) {}

    Display d = Display.getDisplay(this);
    d.setCurrent(f);
}
```

Para poder mostrar esta imagen deberemos añadirla como recurso a la *suite*. Añadir una imagen con nombre `logo.png` al directorio de recursos. Puedes encontrar esta imagen en el directorio `PruebaAplicacion/res` de las plantillas de la sesión.

Compilar y ejecutar la aplicación para comprobar que la imagen se muestra correctamente.

6. Librerías opcionales

Ahora añadiremos sonido a la aplicación. Para ello deberemos utilizar la API multimedia que es una API adicional. Deberemos:

a) Incorporar la librería MMAPI a nuestro proyecto en Eclipse (con el plug-in Eclipse ME ya no es necesario este paso).

b) Una vez hecho esto podremos utilizar esta API multimedia en el editor de Eclipse sin que nos muestre errores en el código. Modificaremos el código del MIDlet de la siguiente forma:

```
package es.ua.jtech.jdm.sesion10.prueba;

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

public class MIDletPrueba extends MIDlet {

    protected void startApp() throws MIDletStateChangeException {
        Form f = new Form("Hola mundo");
        try {
            f.append(Image.createImage("/logo.png"));
        } catch(Exception e) {}

        try {
            Manager.playTone(80, 1000, 100);
        } catch(MediaException e) {}

        Display d = Display.getDisplay(this);
        d.setCurrent(f);
    }

    protected void pauseApp() {
    }

    protected void destroyApp(boolean incondicional)
        throws MIDletStateChangeException {
    }
}
```

c) Guardar y comprobar que la aplicación funciona correctamente.

7. Temporizadores

Vamos a incorporar un temporizador a una aplicación. Lo único que haremos será mostrar un mensaje de texto en la consola cuando se dispare el temporizador, por lo que no será una aplicación útil para visualizar en el móvil.

a) En el directorio `Temporizador` de las plantillas de la sesión se encuentra implementado este temporizador. Compilarlo y ejecutarlo.

b) Modificar este temporizador para que en lugar de dispararse pasado cierto intervalo, se dispare a una hora fija.

8. Serialización

Vamos a ver ahora como leer datos codificados de forma binaria. Tendremos entre los

recursos de la aplicación un fichero de datos donde éstos se encuentran almacenados de forma binaria con un formato dado. Vamos a ver cómo leer este formato.

a) En el directorio `Serializacion` de las plantillas de la sesión tenemos una aplicación que lee un fichero binario y muestra los datos leídos en la consola. El fichero del que lee (`curso.dat`) almacena la información sobre un curso codificada de la siguiente forma:

```
<UTF>Nombre
<UTF>Departamento
<short>Número de horas
```

En la aplicación esta información se deserializa en el método `deserialize` de la clase `Curso`. Este método nos devolverá un objeto `Curso` con los datos leídos del fichero.

Compilar y probar la aplicación con WTK. Nos dará una salida como la siguiente:

```
Nombre: Programacion de Dispositivos Moviles
Departamento: DCCIA
Horas: 24
```

b) Vamos a hacer lo mismo, pero en este caso para leer datos de un alumno. En el fichero `alumno.dat` tenemos estos datos almacenados de la siguiente forma:

```
<UTF>DNI
<UTF>Nombre
<UTF>Apellidos
<char>Sexo
<short>Edad
<UTF>Teléfono
<boolean>Casado
```

Implementar un método `deserialize` para la clase `Alumno`, que deserialice los datos almacenados en este formato y obtenga un objeto `Alumno` que los contenga.

Deserializar este fichero desde nuestro MIDlet e imprimir el objeto `Alumno` resultante. En este caso deberemos obtener una salida como la siguiente:

```
DNI: 48123456-A
Nombre: Pedro
Apellidos: Lopez
Sexo: V
Edad: 24
Telefono: 965123456
Soltero
```