

## Tema 2

# Manejo de ficheros. Lectura y escritura

En este tema examinaremos el paquete opcional o librería JSR-75 [15], utilizado para acceder al sistema de ficheros del dispositivo móvil con el objetivo de poder leer y escribir ficheros. Del mismo modo veremos el paquete `javax.microedition.rms`, el cual contiene las clases necesarias para llevar a cabo lo que se conoce como almacenamiento persistente.

Sin embargo, será preciso comenzar el tema hablando del modelo de threads en J2ME. De lo contrario no podremos realizar ninguna operación de entrada/salida.

### 2.1. Modelo de Threads en J2ME

Dada la naturaleza intrínseca de los dispositivos a los que está orientado J2ME (como hemos comentado varias veces, dispositivos con capacidades de cómputo, entre otras, limitadas), se dice que es recomendable que cada operación que vayamos a realizar, y que suponga un consumo elevado de recursos (entrada/salida, gráficos, comunicaciones, etc.), se realice en un *thread* [28] o hilo diferente de aquel en el que se ejecuta la aplicación principal.

Pues bien, nosotros no tomaremos esa recomendación como tal, sino como una obligación para todos los Midlets que veamos en este curso, ya que de esta manera conseguiremos solventar muchos problemas, a la vez que mejoramos la calidad de nuestras aplicaciones.

Para aplicar este modelo a nuestros Midlets lo primero que tenemos que hacer es que la clase principal de nuestro Midlet (aquella que hereda de la clase `MIDlet`), implemente la interfaz `Runnable`<sup>1</sup>, y por tanto implemente el método `public void run()`.

Desde ese método `run()` es desde el cual se debe llamar al método `commandAction()` de la interfaz `CommandListener`. Sin embargo NetBeans es capaz de hacer este trabajo por nosotros. Para ello basta con seleccionar la imagen titulada *Mobile Device*, dentro de la vista *Flow Design* y, en el panel de *Propiedades* (abajo a la derecha), seleccionar la casilla *Generate Threaded Command Listeners*.

El código 2.1 muestra el código generado automáticamente por NetBeans. El flujo de ejecución inicial sería el siguiente: el método `startApp()` (línea 69) llama al método `initialize()` (línea 63). El código de la línea 65 hace que se llame al método `run()` (línea 11), el cual se ejecutará indefinidamente hasta que finalice el Midlet debido al bucle `for` de la línea 16. Pero cada vez que se ejecuta, se detiene en la línea 23, esperando a que se haga un `notify()` sobre el array de objetos `_asyncParameters`. Dicho `notify()` se realiza en la línea 47, cada vez que se invoca al método `commandAction()` (línea 42). Entonces el método `run()` se desbloquea y sigue ejecutándose hasta llamar, en la línea 34, al método `_commandAction()` (línea 55), que es el que realmente implementa las acciones que se llevarán a cabo al ejecutar un comando.

<sup>1</sup><http://ants.dif.um.es/~felixgm/docencia/j2me/javadoc/jsr118/java/lang/Runnable.html>