J2ME - Programando un cronómetro para nuestro celular

A lo largo de este apunte veremos como desarrollar una aplicación que sea capaz de correr en un teléfono celular (MIDlet). Para ello haremos uso del entorno de desarrollo que Sun Microsystem dispone para programar en dispositivos de bajos recursos: J2ME, hoy llamado JME y el IDE NetBeans 6.5.

¿Qué es J2ME?

J2ME (Java 2 Micro Edition) es un subconjunto del popular lenguaje Java, más precisamente de J2SE (Java 2 Standar Edition). Este subconjunto fue adaptado y contiene nuevas funcionalidades para permitir el desarrollo de aplicaciones destinadas a dispositivos electrónicos con escasos recursos de memoria, proceso y display.

Las características mas sobresalientes de J2ME son:

- Inspirado en C++.
- Es completamente orientado a objetos.
- Es multiplataforma, genera bytecodes que serán interpretados independientemente del celular por una JVM (Java Virtual Machine).
- Es software libre.

Arquitectura del entorno de ejecución de J2ME.

Se suele asociar a J2ME con la programación de celulares, sin embargo J2ME apunta a todos aquellos dispositivos que no cumplan con los requisitos necesarios para correr la máquina virtual de J2SE.

J2ME dispone de dos configuraciones. Cada fabricante de dispositivos que desea soportar J2ME debe decidir que configuración implementará, dependiendo de las características de su producto. Las dos configuraciones de J2ME son CDC y CLDC.

Una configuración es el conjunto mínimo de APIs que permiten desarrollar aplicaciones para un grupo de dispositivos. Por ejemplo la configuración CLDC (Connected Limited Device Configuration) está enfocada a dispositivos con limitaciones en procesamiento, memoria y pantalla como ser celulares, pagers, PDAs o agendas electrónicas; mientras que la configuración CDC se enfoca a dispositivos con mayores recursos que poseen procesadores de 32 bits, y más de 2MB de memoria, en esta configuración trabajan los decodificadores de TV digital, palms, televisores con internet y electrodomésticos inteligentes.

Como verán, la configuración que implementan los celulares es CLDC. Cada configuración tiene su propia máquina virtual encargada de correr las aplicaciones. La máquina virtual de la configuración CLDC es la KVM (Kilo Virtual Machine), cuyas características son las siguientes:

La KVM es la máquina virtual mas pequeña desarrollada por Sun Microsytem.

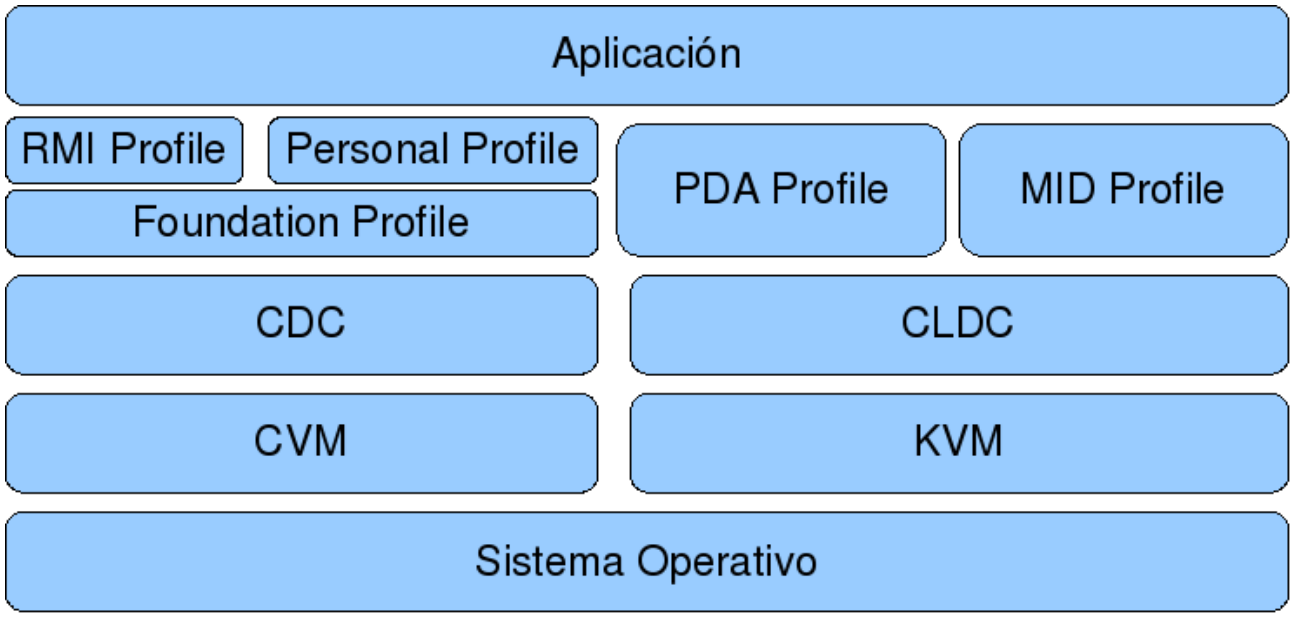
- Ocupa entre 40Kb y 80Kb.
- No soporta punto flotante.
- Escrita en lenguaje C.
- Posee alta portabilidad.

Por último, la arquitectura de J2ME, puntualmente la configuración CLDC, que es la que nos interesa, se

bifurca en 2 perfiles: PDA Profile y MID Profile.

Un perfil es un conjunto de APIs orientado a un ámbito de aplicación determinado. El perfil estipula las librerías necesarias para implementar una aplicación en una clase abstracta de dispositivos, como por ejemplo en teléfonos celulares. Los perfiles engloban un grupo de dispositivos según la funcionalidad que proporcionan. Para teléfonos celulares existe un único perfil llamado MIDP (Mobile Information Device Profile). MIDP nos provee las clases y funciones necesarias para que el usuario ejecute una aplicación en el celular.

A continuación podemos apreciar gráficamente la arquitectura de ejecución de J2ME:



Para que quede mas clara la diferencia entre un perfil y una configuración podemos decir que una configuración implica la máquina virtual a implementar y las librerías disponibles para una familia de dispositivos con características en común, como pagers, PDAs o celulares, mientras que un perfil establece las APIs que se utilizarán para programar en un tipo de dispositivo genérico, por ejemplo, en un teléfono celular. Es por ello que debemos tener en claro que configuración usar y que perfil. En nuestro caso la configuración será CLDC y el perfil MIDP.

Creación del MIDlet a través de NetBeans 6.5

Bueno mucha teoría, seguro quieren ver de que trata el código, paciencia, ya arrancamos. Vimos ya que un Midlet es un programa capaz de correr en un teléfono celular. Así como a las aplicaciones Java que corren en un navegador web se las llama Applets y a las que se ejecutan en un servidor Servlets, a las que se ejecutan utilizando el perfil MIDP se las denominan MIDlets.

Para crear nuestro cronómetro debemos disponer del siguiente software:

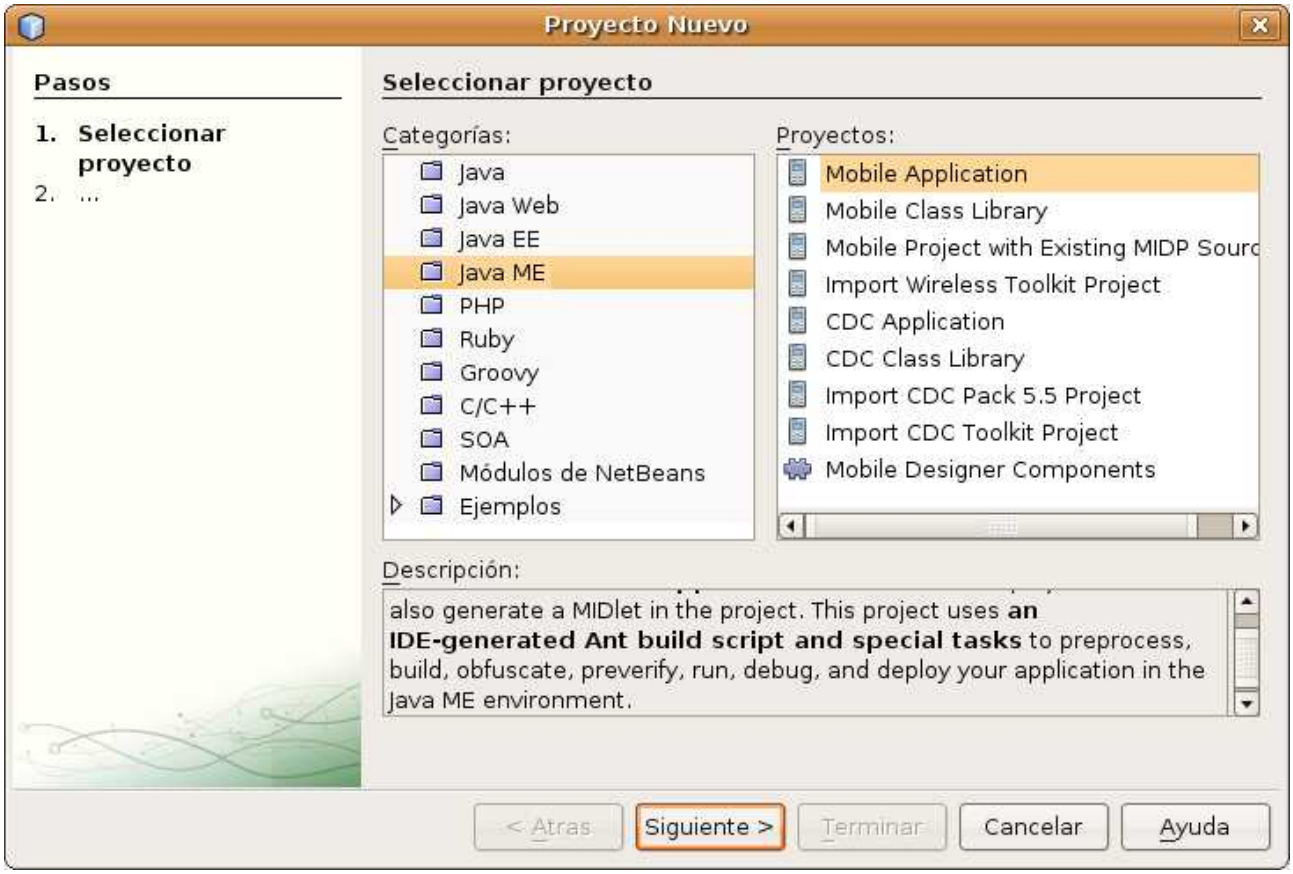
- Java SE Runtime Environment (JRE) 6 Update 11
- Java SE Development Kit (JDK) 6 Update 11
- NetBeans IDE 6.5 con plugin JME

Todos el software está disponible gratuitamente en la web de Sun www.sun.com.

Los dos IDEs más populares que soportan J2ME son Eclipse y NetBeans. Para la construcción de la aplicación que propongo en este artículo opte por NetBeans, dicho sea de paso, NetBeans ha cumplido 10 años en octubre de 2008, lo cual se refleja en la pantalla de bienvenida de la versión 6.5.

Punto de partida

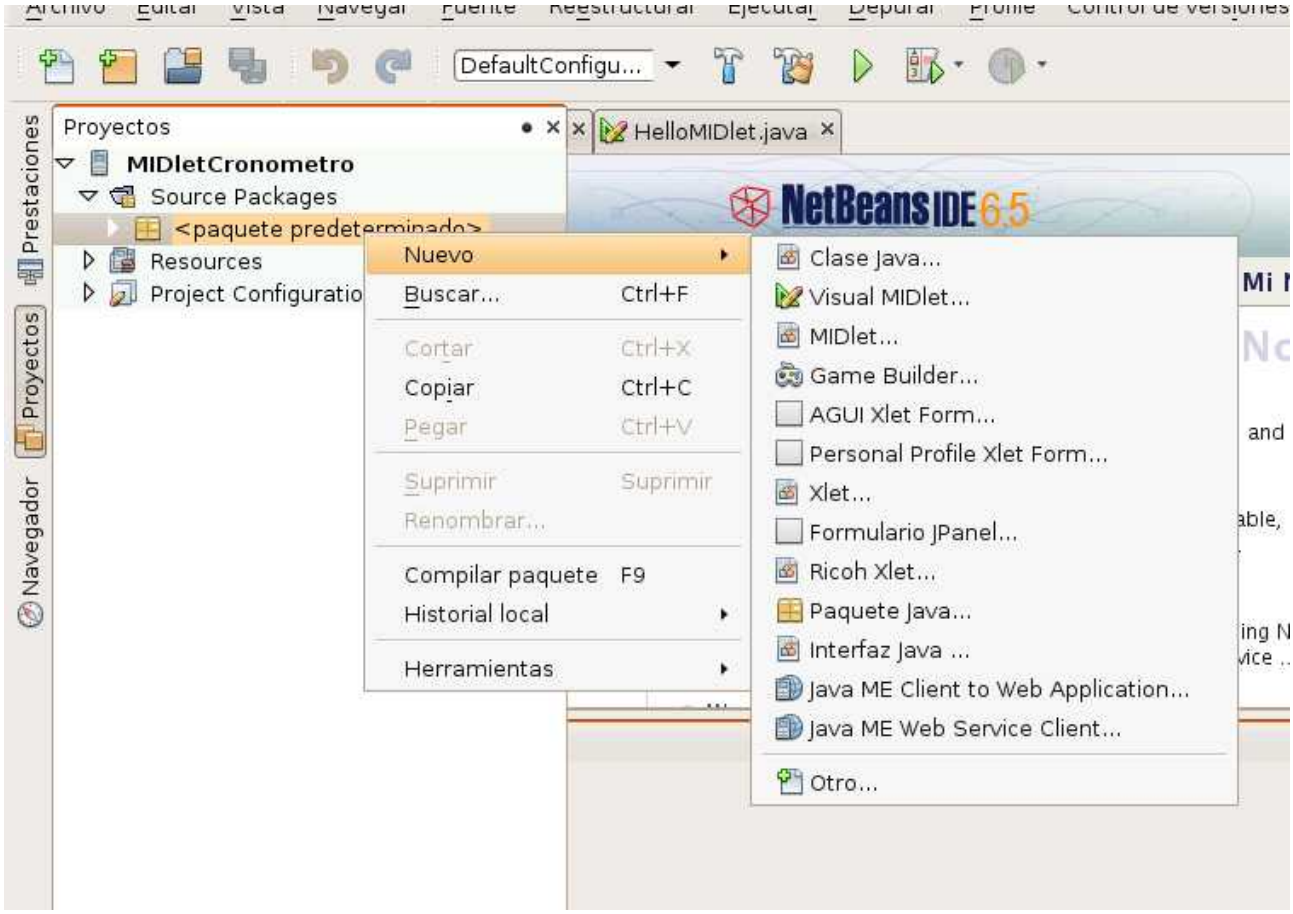
Iniciamos NetBeans y vamos al menú Proyecto Nuevo, en la ventana, que se muestra a continuación, seleccionamos en categorías, J2ME y en Proyectos, Mobile Application:



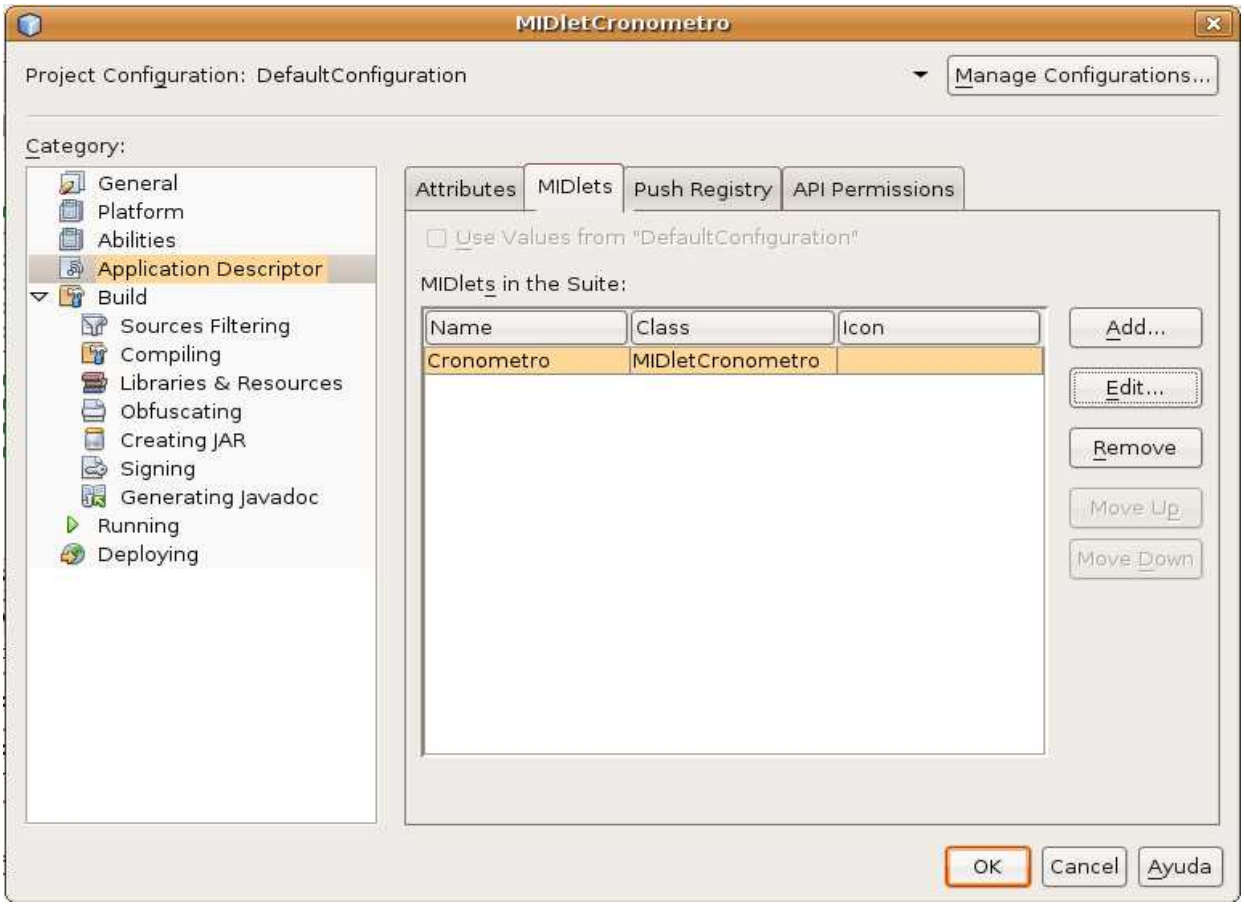
A continuación nos pedirá el nombre del proyecto, ponemos aquí MIDletCronometro y quitamos el tilde en Create Hello MIDlet para evitar que nuestro MIDlet ya arranque con la programación básica del típico "Hola Mundo". Llegamos ahora al punto en que tenemos que seleccionar la versión tanto de la configuración como del perfil, también es posible seleccionar un emulador de celular para probar la aplicación. Como mi celular trabaja con la configuración CLDC 1.0 y el perfil MIDP 2.0 son estos los parámetros que selecciono. Clic en siguiente, clic en terminar y por fin estamos listos para teclear código.

La programación J2ME es orientada a objetos, escapa del alcance de este artículo explicar que es una clase, un objeto, un método, una instancia, un constructor y herencia de clases. Si no comprenden estos términos les aconsejo que investiguen antes de continuar con la lectura de este artículo.

Es momento ahora de crear una clase cuyo nombre debe ser el mismo del proyecto y debe heredar de la clase MIDlet. Para crear esta clase nos dirigimos a la ventana Proyectos y seleccionamos Nueva Clase Java desde el menú contextual de Source Packages, <paquete predeterminado>. La captura siguiente refleja lo dicho anteriormente:



Antes de continuar, es importante no olvidar configurar las propiedades del MIDlet para que la clase de inicio sea la recién creada, si no lo hacemos nuestra clase nunca se instanciará. Para ello vamos al menú Archivo → Project Properties, para luego dirigirnos a la categoría Application Descriptor, y, previa selección de la solapa MIDlet, pulsamos sobre el botón Add para seleccionar nuestra única clase. Asi debería quedarnos:



Volvemos a la ventana de código que ya contiene la definición de la clase MiDletCronometro. Es momento ahora de importar a nuestro proyecto las clases necesarias para, en primer lugar, implementar un MIDlet, y en segundo lugar, diseñar la interfaz gráfica de alto nivel. También indicaremos que nuestro MIDlet hereda de la clase MIDlet, y será nuestra clase MIDletCronometro quien manejará los eventos que disparen cada uno de lo botones de comandos que próximamente incorporaremos. El código entonces inicia así:

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

public class MIDletCronometro extends MIDlet implements CommandListener {

}
```

Una clase que hereda de la clase MIDlet estará en uno de tres estados: Activo, Pausado o Destruido. Es necesario implementar estos métodos aún si no contienen programación, lo hacemos en el interior de la clase MIDletCronometro (dentro de las llaves):

```
public void startApp(){

}


public void pauseApp(){

}

public void destroyApp(boolean unconditional){

}
```

Bien, seguramente el IDE debe estar resaltando lineas a lo loco, esto es normal, ya que queda mucho por programar y el código está incompleto, de a poco los errores irán desapareciendo a medida que ingresemos programación, paciencia.

Vamos ahora a definir los objetos de nuestra clase. Pensemos entonces que objetos necesitamos para implementar nuestro cronómetro:

- Un objeto Display, que tendrá una referencia al display o pantalla del celular donde correrá la aplicación.
- Un objeto Form, que será el formulario que visualizaremos en la pantalla. Sobre este formulario incorporaremos el resto de los objetos que menciono a continuación.
- 4 objetos Command (botones para desencadenar procesos), uno para iniciar el cronómetro, otro para salir del programa, otro para detener el cronómetro y otro para reiniciarlo.
- Un objeto TextField que será en donde se visualizará el tiempo transcurrido.
- Un objeto Cronometro, que no existe en java y lo debemos crear nosotros. Por ahora pensemos que un objeto cronómetro debe ser capaz de iniciar el conteo, detenerlo y mostrar el tiempo transcurrido en algún otro objeto.

Estos objetos los declaramos como privados de nuestra clase y los instanciamos desde el constructor de la misma:

```
public class MIDletCronometro extends MIDlet implements CommandListener {
   private Form formulario;
   private Display pantalla;
   private Command cmdIniciar;
   private Command cmdSalir;
   private Command cmdParar;
   private Command cmdReiniciar;
   private TextField txtVisor;
   private Cronometro cronometro;

   public MIDletCronometro(){
       pantalla=Display.getDisplay(this);
       formulario=new Form("Cronómetro");
       cmdIniciar=new Command("Iniciar",Command.OK,0);
       cmdSalir=new Command("Salir",Command.EXIT,1);
       cmdParar=new Command("Parar",Command.STOP,0);
       cmdReiniciar=new Command("Reiniciar",Command.OK,1);
       txtVisor=new TextField("Cronómetro","00:00:00:00",15,TextField.ANY);
       formulario.addCommand(cmdSalir);
       formulario.addCommand(cmdIniciar);
       formulario.setCommandListener(this);
       formulario.append(txtVisor);
   }

   public void startApp(){
       pantalla.setCurrent(formulario);
   }

   public void pauseApp(){
   }

   public void destroyApp(boolean unconditional){
   }
```

Si son observadores habrán notado que cada vez que la aplicación se inicia se llama al método startApp y el formulario se muestra en pantalla.

Vamos ahora a programar que ocurrirá cada vez que se pulse un objeto Command. Habíamos indicado que nuestra clase principal, MIDletCronometro, sería la encargada de procesar cada uno de los eventos que generen los Command. Si no lo recuerdan lo habíamos indicado con implements CommandListener en la linea de código que define la clase. Pues bien, si nuestra clase va cumplir esa función debe disponer entonces del método commandAction el cual tiene dos parámetros, uno es el objeto Command que activó el evento y el otro es un objeto de la clase displayable que indica en donde se encontraba es objeto. Los invito a que analicen el siguiente código:

```
public void commandAction(Command c, Displayable d){
    if (c==cmdSalir){
          destroyApp(false);
          notifyDestroyed();
    }
    else if(c==cmdIniciar){
         cronometro=new Cronometro(txtVisor);
         cronometro.Iniciar();
         formulario.removeCommand(cmdIniciar);
         formulario.removeCommand(cmdSalir);
         formulario.addCommand(cmdParar);
         formulario.addCommand(cmdReiniciar);
         formulario.setCommandListener(this);
    }
    else if (c==cmdParar){
         cronometro.Parar();
         formulario.removeCommand(cmdParar);
         formulario.removeCommand(cmdReiniciar);
```

```
            formulario.addCommand(cmdSalir);
            formulario.addCommand(cmdIniciar);
            formulario.setCommandListener(this);
        }

        else if (c==cmdReiniciar){
            cronometro.Parar();
            cronometro=new Cronometro(txtVisor);
            cronometro.Iniciar();
        }
}
```

Puntualmente, se trata de detectar cuál de los cuatro Command disparó el evento, para así obrar según corresponda. Cabe aclarar que nunca estarán los cuatro Command al mismo tiempo en pantalla. Al iniciar el programa veremos los Command Salir e Iniciar. Si pulsamos Iniciar se quitan estos dos del formulario y se agregan los Command Parar y Reiniciar, esto se aplica al objeto Form mediante el método removeCommand y addCommand respectivamente. Si pulsamos Parar, volvemos a mostrar los Command Salir e Iniciar y si pulsamos Reiniciar, seguimos mostrando los mismos Command, ya que las opciones siguen siendo las mismas, Parar o Reiniciar.

Otra cosa que resalta es cómo se programa sobre un objeto que aún no tenemos implementado, pero del que sí sabemos, como dijimos antes, que debe ser capaz de iniciar el conteo, detenerlo e informar en algún objeto el tiempo transcurrido. Repasemos que pasa con el objeto cronometro cuando se presiona el Command Iniciar:

```
            cronometro=new Cronometro(txtVisor);
            cronometro.Iniciar();
```

Se crea una instancia de la clase Cronometro, y a su vez le pasamos como parámetro un objeto para que sepa en donde debe mostrar el tiempo transcurrido. Veamos ahora que pasa cuando se presiona sobre Parar:

```
            cronometro.Parar();
```

Simplemente le tiramos la pelota al objeto, llamamos al método Parar() y el sabrá que hacer... Y se pulsa Reiniciar?

```
        cronometro.Parar();
        cronometro=new Cronometro(txtVisor);
        cronometro.Iniciar();
```

Lo detenemos y volvemos a crear una instancia del objeto.

Eso es todo el MIDlet debería funcionar, siempre y cuando esté implementada la clase Cronometro y responda a los mismo métodos que utilizamos. Esto lo haremos a continuación.

Creando la clase Cronometro

La clase Cronometro debe ser programada a continuación de la clase MIDletCronometro, luego de la llave de cierre. Esta clase debe ser capaz de trabajar cada cierto intervalo de tiempo. Por ejemplo, se me ocurre que el cronómetro muestre las centésimas de segundo. Entonces cada cierta cantidad de milisegundos deseo informar el tiempo en curso del cronometro. Es por ello que importaremos a nuestro proyecto dos clases que permiten ejecutar tareas periódicamente cada cierta cantidad de milisegundos, estas clases son Timer y TimerTask. Así quedará entonces el inicio de nuestro código:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.Timer;
```

```
import java.util.TimerTask;
```

Es momento de definir que la clase Cronometro hereda de TimerTask. La clase completa es la siguiente:

```
class Cronometro extends TimerTask {
    private long contador=0;
    private Timer timerVelocidad;
    private TextField txt;

    public Cronometro(TextField t){
       txt=t;
    }

    public void run(){
       contador=contador + 23;    //incremento en 23 centésimas el tiempo
       txt.setString(DameFormatoHora(contador));
    }

   public void Iniciar(){
       contador=0;
       timerVelocidad=new Timer();
       timerVelocidad.schedule(this,0, 230); //cada 230 milisegundos dispara run()
   }

   public void Parar(){
      timerVelocidad.cancel();
  }

   public String DameFormatoHora(long millis){
       String Hora;
       long centesimas, segundos,minutos,horas;
       centesimas=millis%100;
       segundos=(millis/100) % 60;
       minutos=(millis/100)/60;
       minutos=minutos%60;
       horas=(((millis/100)/60)/60);

       if (horas<10)
          Hora="0" + horas + ":" ;
       else
          Hora= horas + ":";

       if (minutos<10)
          Hora=Hora + "0" + minutos + ":";
       else
          Hora= Hora + minutos +  ":";

       if (segundos<10)
          Hora=Hora + "0" + segundos + ":";
       else
          Hora= Hora + segundos +  ":";

       if (centesimas<10)
         Hora=Hora + "0" + centesimas;
       else
         Hora= Hora + centesimas;

       return Hora ;
   }

}
```

Del código anterior debo destacar que el tiempo transcurrido se acumula en una objeto del tipo long representando centésimas de segundo. En un objeto Timer diremos que objeto TimerTask vamos a disparar y con que regularidad, es decir cada cuantos milisegundos, aquí propongo 230, o sea 23 centésimas. Como

nuestra clase hereda de TimerTask debemos implementar el método run que es el método temporizado de toda clase TimerTask. Por último se informa en el Objeto TextField que mantiene una referencia al objeto que recibimos como parámetro, el tiempo transcurrido. Como no encontré una clase que me permita convertir de milisegundos a un formato de hh:mm:ss decidí crear un método llamado DameFormatoHora que no es para nada complicado de entender.

Eso es todo, el cronómetro está terminado. Si funciona como deseamos y estamos conformes con su funcionamiento podemos echar un vistazo en el directorio dist (ubicado en el directorio donde se aloja el proyecto) allí estarán los archivo .jad y .jar esperando ser instalados en sus celulares.

Pablo Mileti
pablomileti@gmail.com