DESARROLLO DE APLICACIONES J2ME PARA POCKET PC

con WebSphere Studio Device Developer

Tutor: **Mª Celeste Campo Vázquez**
Autor: **Roberto Díaz Morales**

1 INDICE

1 INDICE 2

2 OBJETIVOS 3

3 INTRODUCCIÓN 4

4 J2ME 5

5 KVM/CLDC/MIDP 7

6 INSTALACIÓN DEL ENTORNO EN WINDOWS 8

7 PROYECTO J2M3/MIDP DE EJEMPLO 12

7.1 CREACIÓN 12

7.2 SIMULACIÓN 14

7.3 PRUEBA EN DISPOSITIVO REAL 20

8 PROYECTO J2ME/MIDP PROPIO 24

8.1 LA APLICACIÓN 24

8.2 CREACIÓN 25

8.3 SIMULACIÓN 29

8.4 PRUEBA EN DISPOSITIVO REAL 32

9 ANEXO I (CÓDIGO FUENTE) 35

9.1 BuscPMIDlet.java 35

9.2 BParC.java 47

9.3 RecordGuard.java 52

9.4 Temporizador.java 52

9.5 Temporizador2.java 53

9.6 Temporizador3.java 54

2 OBJETIVOS

El objetivo de este documento es desarrollar un tutorial para desarrollar aplicaciones de J2ME para dispositivos cuyo sistema operativo sea Pocket PC (como PDA, teléfonos móviles, etc ...), utilizando como herramienta de desarrollo la aplicación IBM Websphere Studio Device Developer.

En primer lugar se hará una introducción sobre la tecnología J2ME, luego se explicará la instalación del entorno en el sistema operativo Windows, tras esto se explicará la creación, compilación y simulación de una de las aplicaciones de ejemplo que contiene en entorno de desarrollo y posteriormente su instalación y ejecución del ejemplo en un dispositivo real.

Una vez explicado todo esto se realizará una aplicación propia repitiendo los pasos anteriores.

3 INTRODUCCIÓN

A medida que la tecnología aumenta, se exigen más funcionalidades a los dispositivos para satisfacer las necesidades de los usuarios. Por este motivo Sun ha estructurado la tecnología Java 2 dirigiéndose a sectores distintos:

- Java 2 Enterprise Edition (J2EE): Soluciones de empresa e-commerce, e-business.
- Java 2 Standard Edition (J2ME): Soluciones de PCs de sobremesa: applets, aplicaciones de usuario.
- Java 2 Micro Edition (J2ME): Dispositivos de consumo y embebidos y dispositivos móviles.

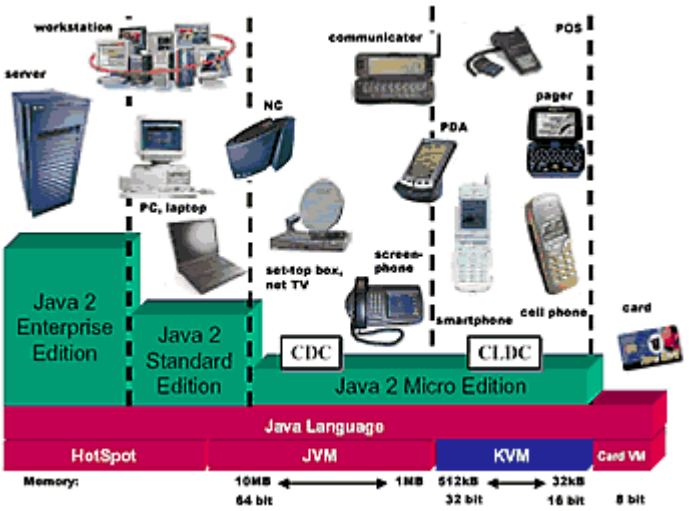


Figura 1. Ediciones del Java 2 y el Mercado objetivo de cada uno. Grafico Sun Microsystems

La tercera de estas tres tecnologías surgió por la necesidad de añadir funcionalidades a los dispositivos móviles debido al éxito que estaban teniendo estos (existen aproximadamente un billón de teléfonos móviles) y poder desarrollar aplicaciones para PDAs, teléfonos móviles, agendas electrónicas y todo tipo de dispositivos que cumplan unas determinadas especificaciones.

4 J2ME

Debido a que Java 2 Micro Edition está pensado para utilizarse en un amplio rango de dispositivos, la arquitectura de las aplicaciones J2ME es la siguiente, estructurándose en tres niveles:

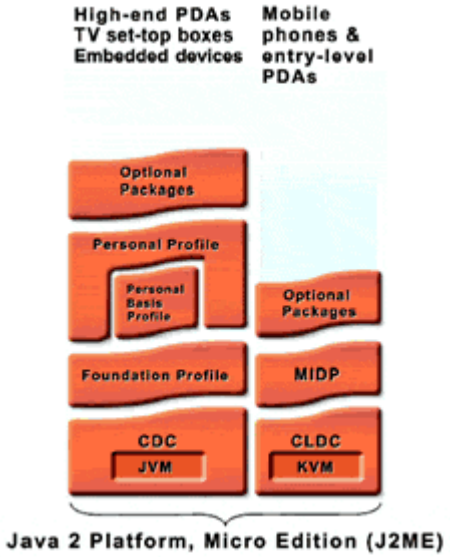


Figura 2. Arquitectura del Java 2. Grafico: Sun Microsystems

J2ME está dirigido a un amplio rango de dispositivos, para conseguir flexibilidad y adaptación se estructura en tres niveles:

- Maquina virtual
- Configuración (mínimo conjunto de clases disponibles, engloba un segmento horizontal del mercado).
- Perfil (Clases adicionales para un segmento vertical de mercado).

Un dispositivo puede soportar múltiples perfiles.

Máquinas virtuales:

- CVM (C Virtual Machine): Orientada a dispositivos embebidos y electrónica de consumo (electrodomésticos, TV digital, set-top box), como JVM pero con mejor uso de la memoria y procesadores de 32 bits, está ligada a la configuración CDC.
- KVM (Kilo Virtual Machine): Diseñada desde cero para dispositivos con poca memoria, capacidad de proceso limitada y con conexión a red intermitente, está ligada a la configuación CLDC.

Configuraciones:

- CDC (Connected Device Configuration): Orientado a dispositivos con 512 KB de ROM, 256 de RAM, conexión a red fija, soporte completo a la especificación de JVM e interfaz de usuario relativamente limitado.
- CLDC (Connected Limited Device Configuration): Orientado a dispositivos con 16 MHz a 32 MHz, limitaciones de consumo (baterías), conectividad a red (inalámbrica), restricciones importantes en el interfaz de usuario.

Perfiles:

Para CDC:

- J2ME Foundation Profile: Para dispositivos sin interfaz gráfico.
- J2ME Personal Profile: Perfil gráfico básico, evolución del Personal Java.
- J2ME RMI Profile: Soporte a RMI para dispositivos limitados.

Para CLDC:

- J2ME Mobile Information Device Profile (MIDP): Perfil para dispositivos inalámbricos.
- J2ME PDA Profile: Perfil para agendas personales electrónicas.

5 KVM/CLDC/MIDP

Esta es la arquitectura que vamos a utilizar en las aplicaciones de este tutorial.

Cubre la maquina virtual, soporte al lenguaje Java, modelo de seguridad, entrada/salida, soporte a conexiones de red e internalización.

Tenemos diversos paquetes:

Los heredados de j2SE:

- java.lang.*
- java.io.*
- java.util.*

Clases específicas introducidas por CLDC:

- javax.microedition.io .*: Aquí están las clases encargadas de las conexiones de red.
- javax.microedition.lcdui .*: Aquí están las clases encargadas de las interfaces gráficas de usuario.
- javax.microedition.midlet .*: Define la interacción entre la aplicación y el entorno en que corre.
- javax.microedition.rms.*: Aquí están las clases encargadas del almacenamiento y recuperación de datos.

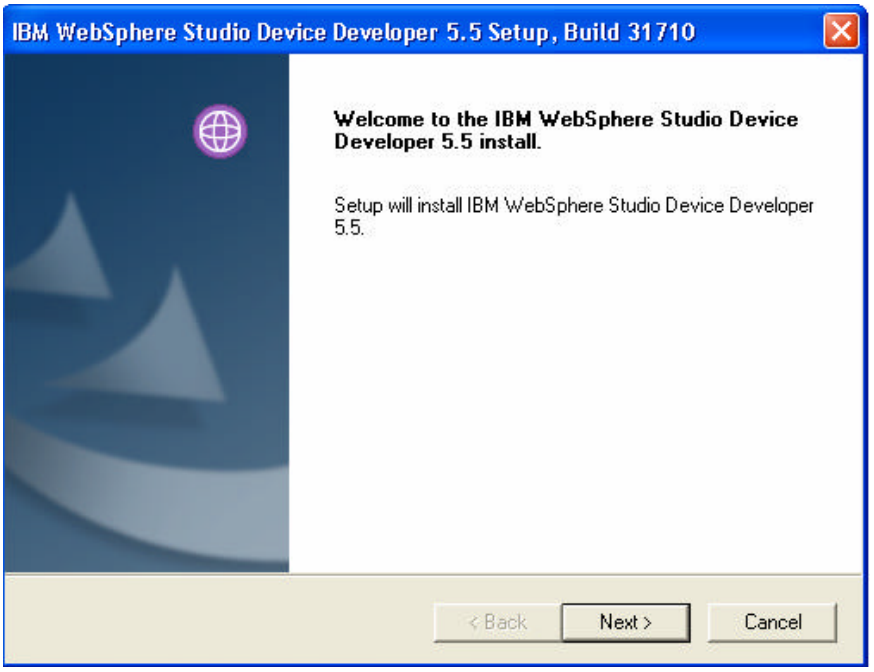
Las principales cosas que no cubre respecto a J2SE son:

- No soporta tipos en punto flotante (float).
- No soporta finalización.
- Limitaciones en el manejo de errores.
- No soporta Java Native Interface (JNI).
- No soporta reflexión (reflection).
- No soporta cargadores de clase definitdos por el usuario.
- No soporta grupos de hilos ni demonios (thread groups, daemon groups).
- Verificación de código en dos fases: preverifier y standard Java bytecode annotations.

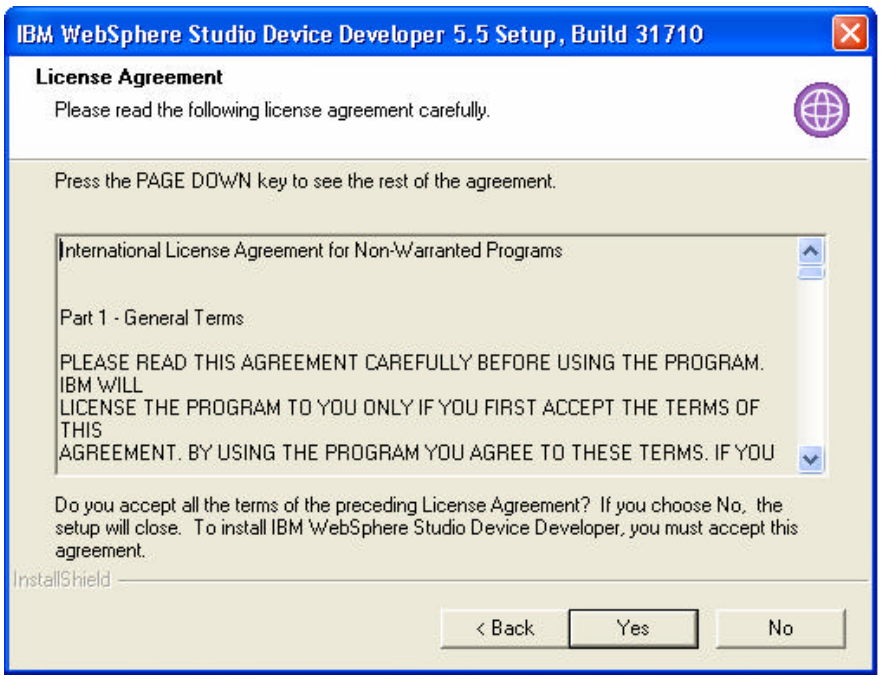
6 INSTALACIÓN DEL ENTORNO EN WINDOWS

Al instalar el entorno de desarrollo, si se realiza desde un CD probablemente al introducirlo sea autoarrancable, en caso contrario debemos ejecutar Setup.exe para comenzar su instalación.

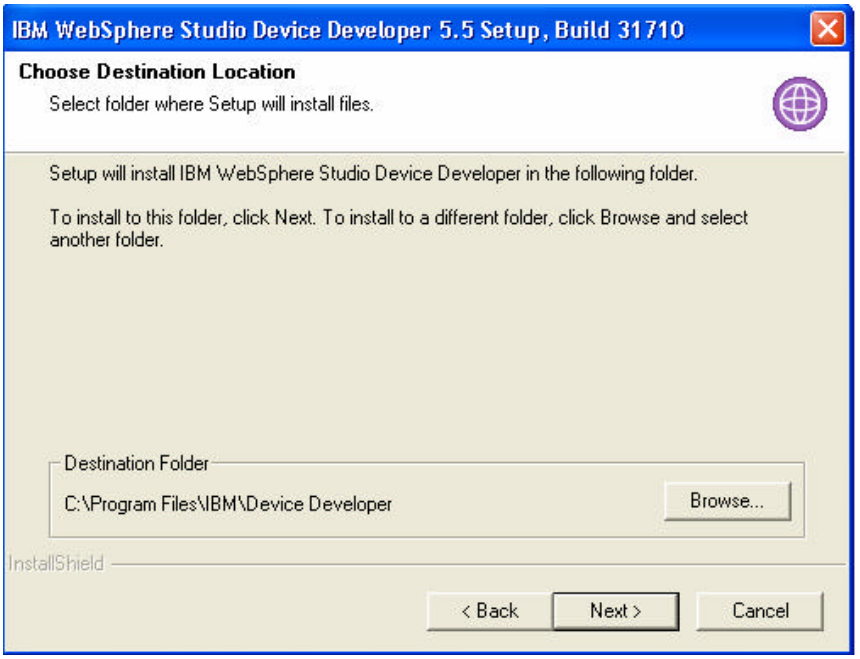
Primero nos aparece una pantalla indicándonos que vamos a instalar IBM WebSphere Studio Device Developer 5.5.



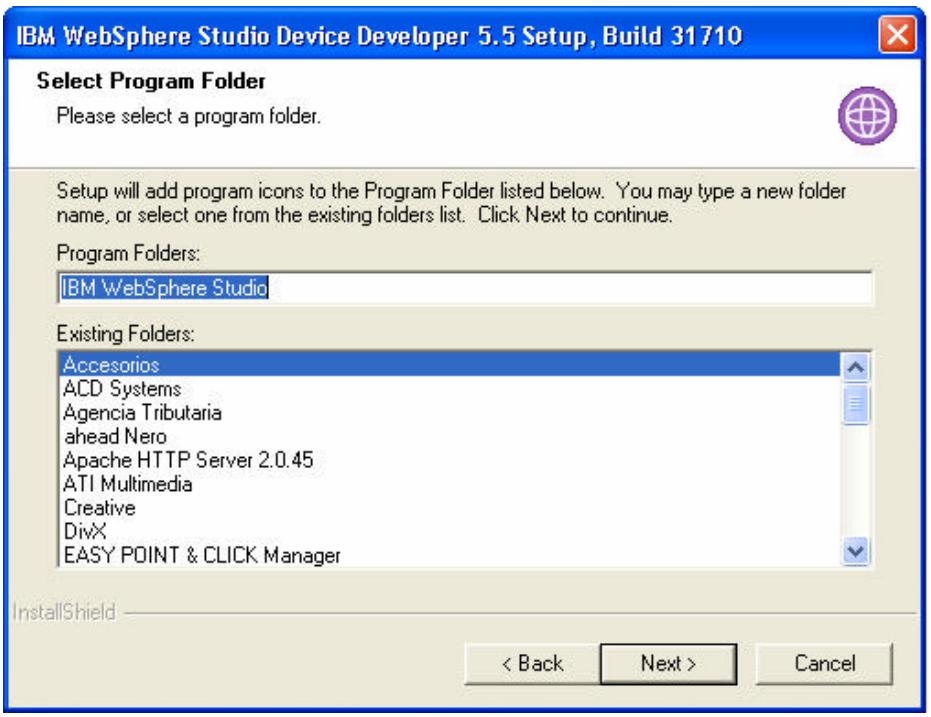
Tras pulsar el botón 'Next >' vamos a la pantalla donde aparece la licencia de este producto:



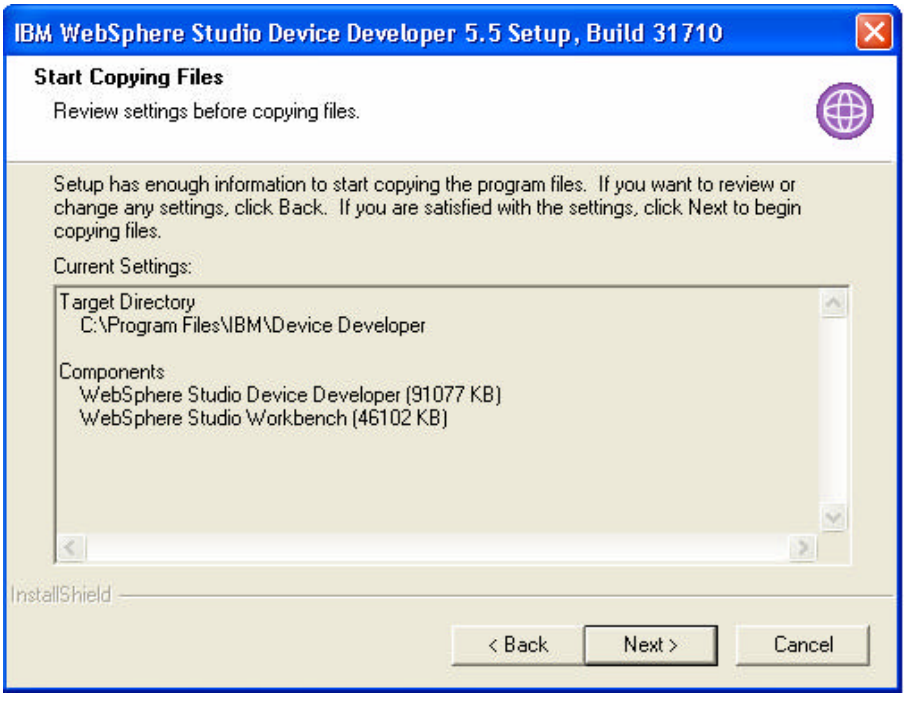
Tras pulsar el botón ‘Yes’ para aceptar la licencia vamos a otra pantalla donde elegimos el directorio de instalación, por defecto será ‘C:\Program Files\IBM\Device Developer’, para cambiarlo pulsamos sobre el botón ‘Browse’ y seleccionamos el directorio de instalación.



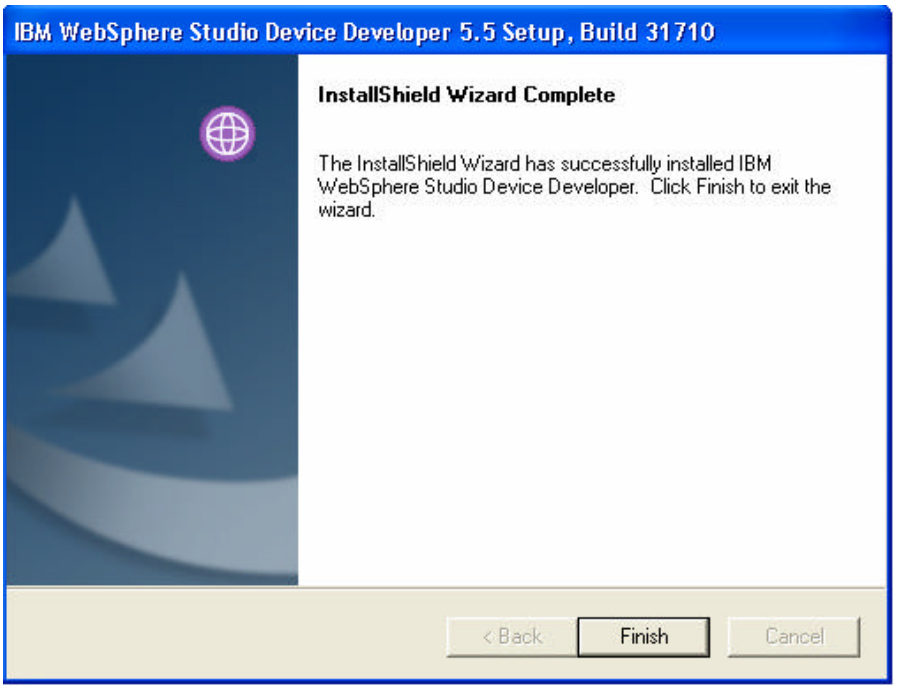
Al pulsar el botón ‘Next’ vamos a la pantalla para añadir el programa al menú de inicio:



Al pulsar el botón 'Next' vamos a la pantalla donde se nos informa sobre lo que ocupa el producto y el directorio donde va a ser instalado:



Finalmente si pulsamos el botón 'Next' la herramienta de desarrollo se instalará en nuestro PC, cuando finalice la instalación se nos mostrará una pantalla indicándonoslo:

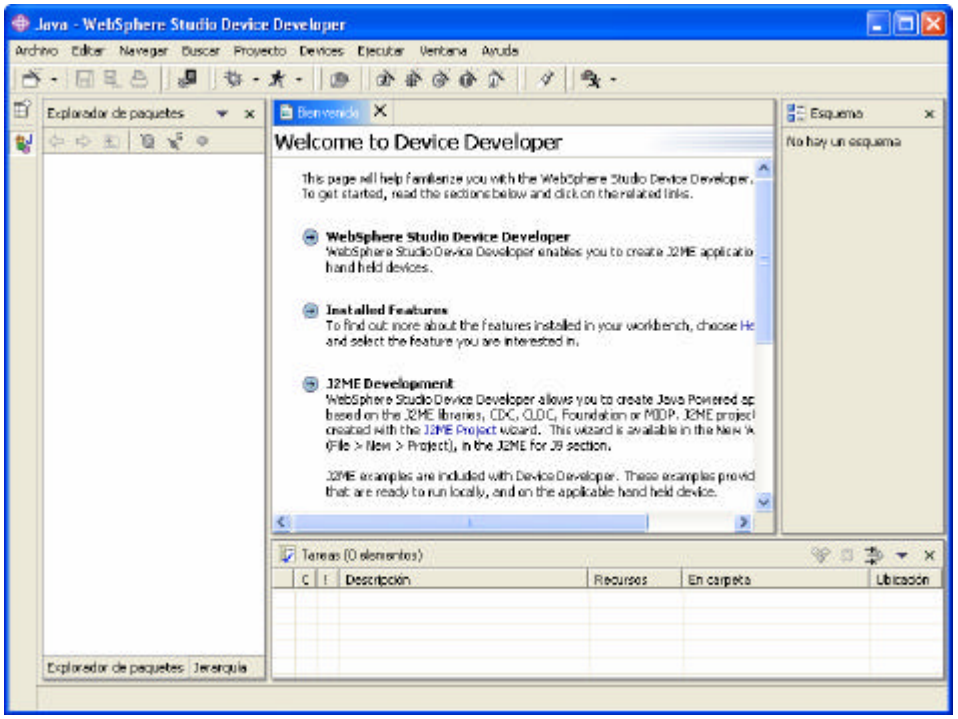


Al pulsar ‘Finish’ terminaremos la instalación y se crearán los accesos directos.

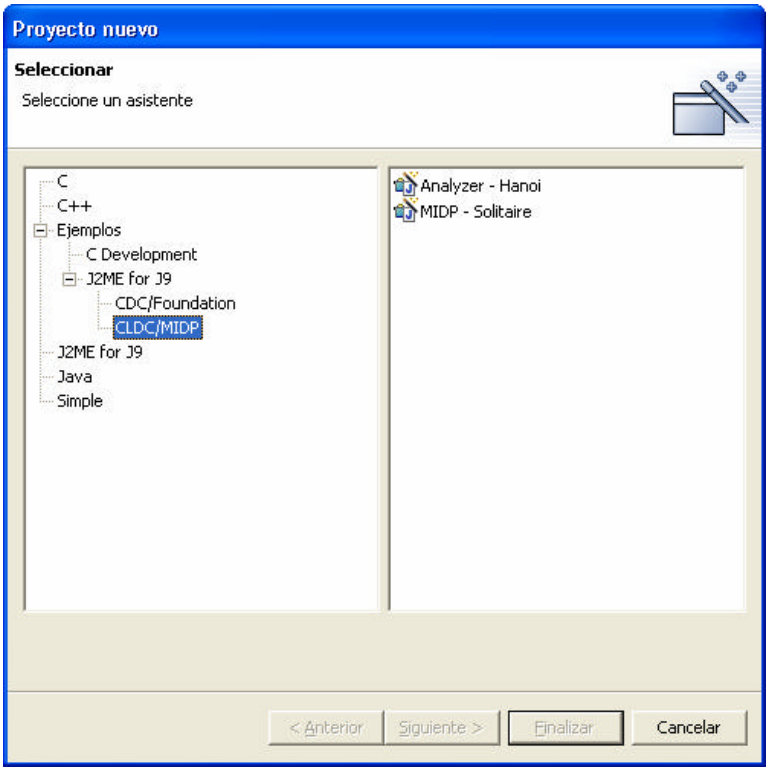
7 PROYECTO J2ME/MIDP DE EJEMPLO

7.1 CREACIÓN:

Ejecutamos la aplicación de desarrollo, que tiene la siguiente forma:



Para la creación de un proyecto de ejemplo, seleccionamos el menú ‘Archivo’ y dentro de la opción ‘Nuevo’ elegimos ‘Proyecto’, se nos abrirá una pantalla como esta:



Elegimos la opción CLDC/MIDP dentro de "J2ME for J9" y seleccionamos "MIDP-Solitaire" del cuadro de la derecha y al pulsar el botón 'Finalizar' se nos creará el proyecto.

Vemos que en la parte de la izquierda tendremos un panel con la estructura de nuestro proyecto:

Dentro de la primera carpeta 'src' se encuentran las clases (SolitaireCanvas.java y SolitaireMIDlet.java), un archivo de texto (SolitaireHelp.txt) que necesita la aplicación para luego poder mostrarla por la pantalla del dispositivo y el archivo jxeLink.rules que es de configuración.

Después están los paquetes que necesitamos para una aplicación CLDC/MIDP:
C:\Program Files\IBM\Device Developer\wsdd5.0\ive\runtimes\winn32\common\lib\jclMidp\classes.zip
C:\Program Files\IBM\Device Developer\wsdd5.0\ive\runtimes\common\ive\lib\charconv.zip

La carpeta "Solitaire.MidletsInfo" contiene el archivo Solitaire-Classes.jxeLinkOptions que contiene opciones de configuración y la carpeta "META-INF" que contiene el archivo MANIFEST.MF que contiene diversos datos de nuestra aplicación, como el nombre de la aplicación, las versiones del CLDC y MIDP, etc.

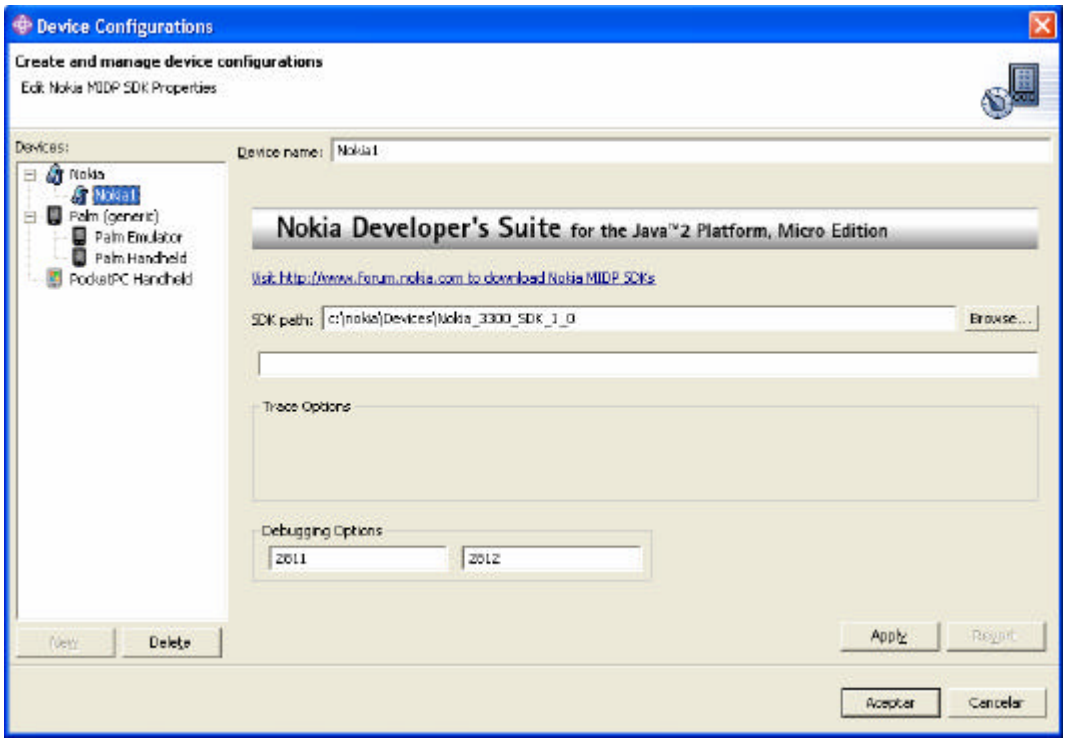
Finalmente están los archivos Solitaire.jad (que da valor a ciertos atributos sobre el nombre, la versión, el tamaño del Jar, …) y wsddbuild.xml que se utiliza para crear los archivos necesarios para la plataforma en la que lo queramos instalar después.

7.2 SIMULACIÓN:

Esta aplicación de desarrollo no nos permite simular en un “Pocket PC”, debido a esto tendremos que simularlo en otro tipo de dispositivo como un móvil o una Palm:

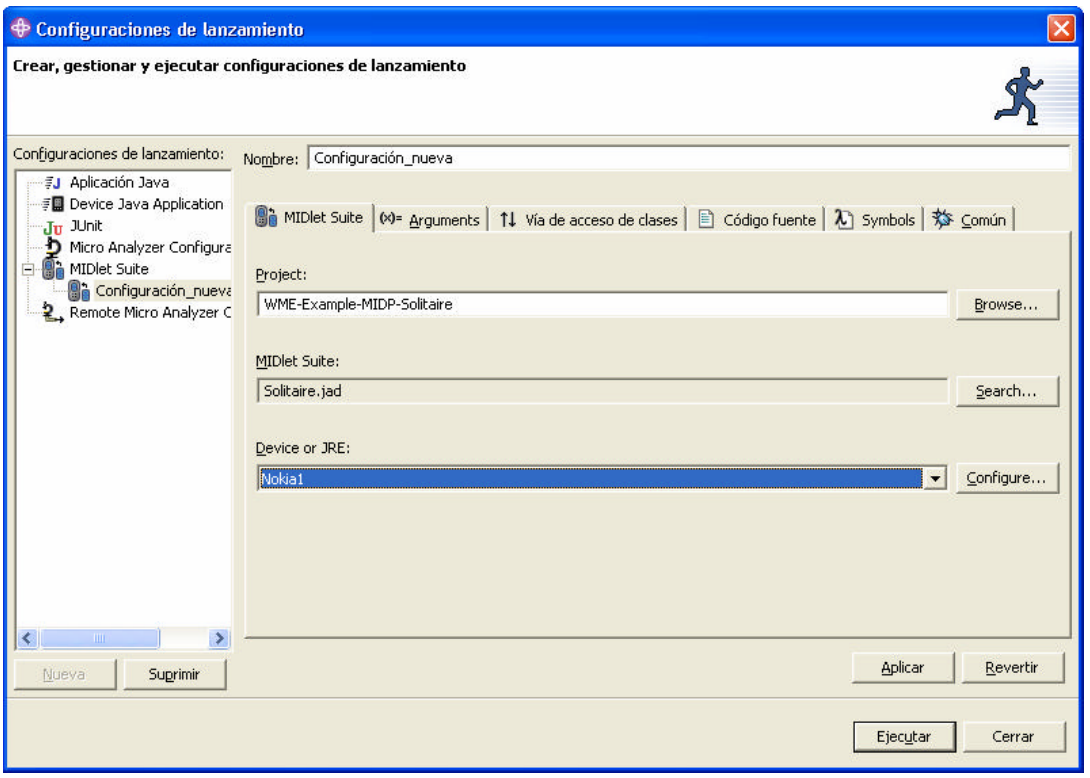
7.2.1 Móvil:

Necesitaremos instalar para esto el SDK de algún teléfono móvil para su simulación si no queremos utilizar el móvil que viene por defecto, primero crearemos una configuración del dispositivo, tenemos que ir al menú ‘Devices’ y pinchar sobre ‘Configure…’:



Seleccionamos en el cuadro de la izquierda Nokia y debajo pulsamos “New”, entonces prodremos poner un nombre al dispositivo “Device name” y especificamos el path del SDK, en nuestro caso hemos instalado el SDK Nokia 3300 que hemos descargado de la página cuya URL aparece encima de dónde introducimos el path, cuando introduzcamos todos los datos pulsamos ‘Aceptar’.

Después de esto nos vamos al menú ‘Ejecutar’ y pinchamos sobre ‘Ejecutar…’ y se nos abre una pantalla:



Seleccionamos MIDlet Suite y damos a ‘Nueva’ en el cuadro de la izquierda, después de esto rellenamos en nombre el nombre para nuestra configuración, en ‘Proyect’ seleccionamos el proyecto que queramos ejecutar en caso de tener más de uno, en MIDlet-Suite seleccionamos Solitaire.jad y en ‘Device or JRE’ seleccionamos el dispositivo que queramos (Default para un móvil por defecto o si no el que hallamos creado antes).

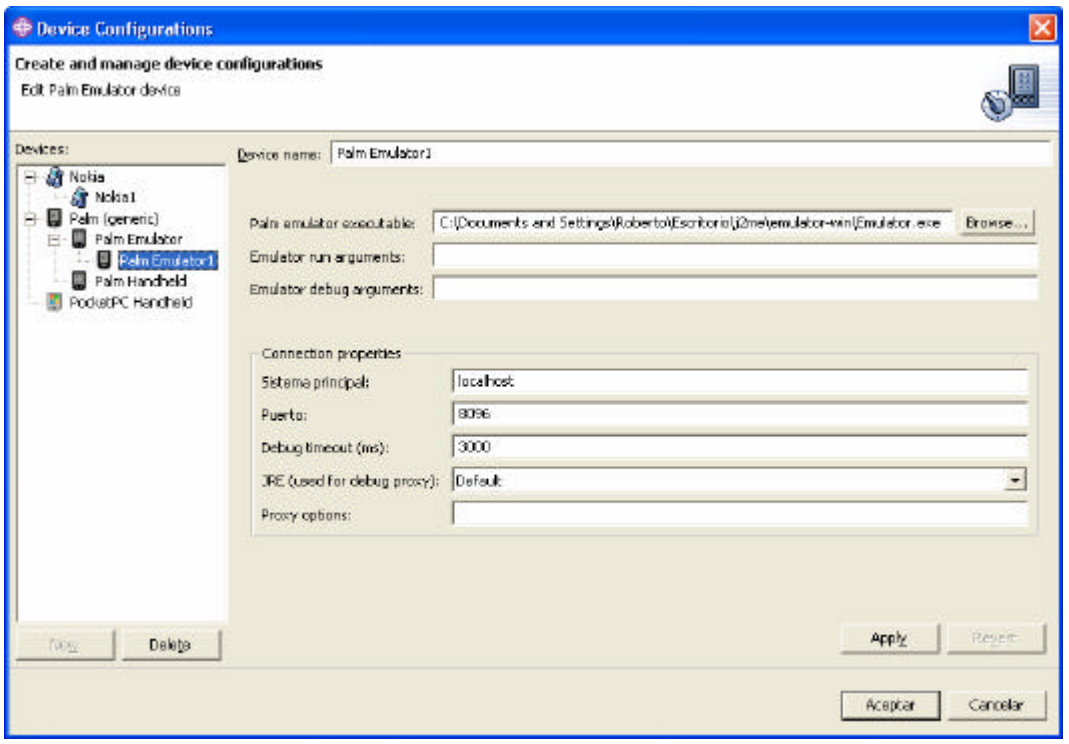
Tras esto pulsamos ejecutar y se lanzará nuestra aplicación:

Aquí tenemos dos ejemplos de la simulación, a la izquierda con el SDK Nokia 3300 y a la derecha con el simulador que viene con el entorno de desarrollo, con el Nokia no podemos jugar ya que el juego necesita que el dispositivo tenga pantalla táctil y este dispositivo no la tiene.

7.2.2 Palm:

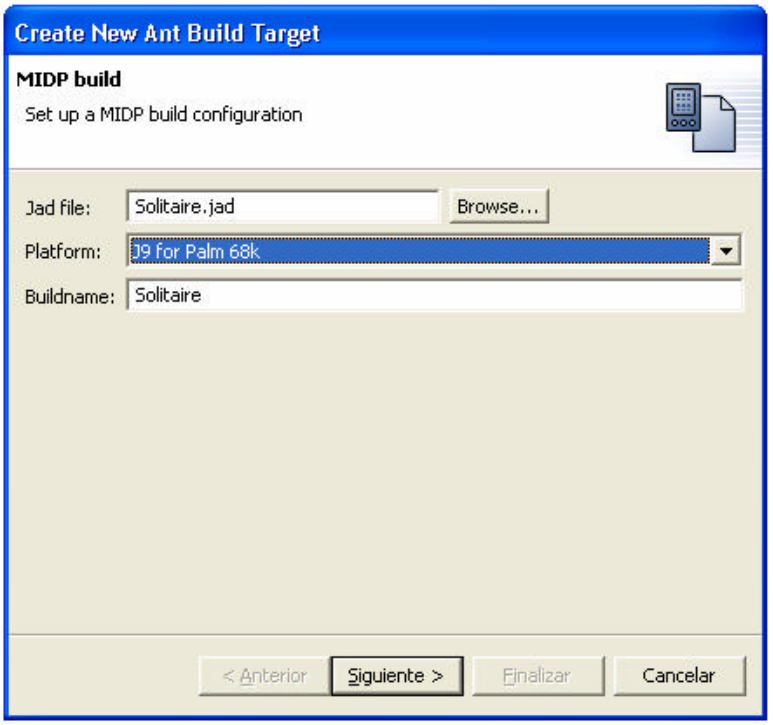
Necesitaremos instalar para esto previamente un emulador de Palm.

Luego iremos al menú ‘Devices’ y pincharemos sobre ‘Configure…’:

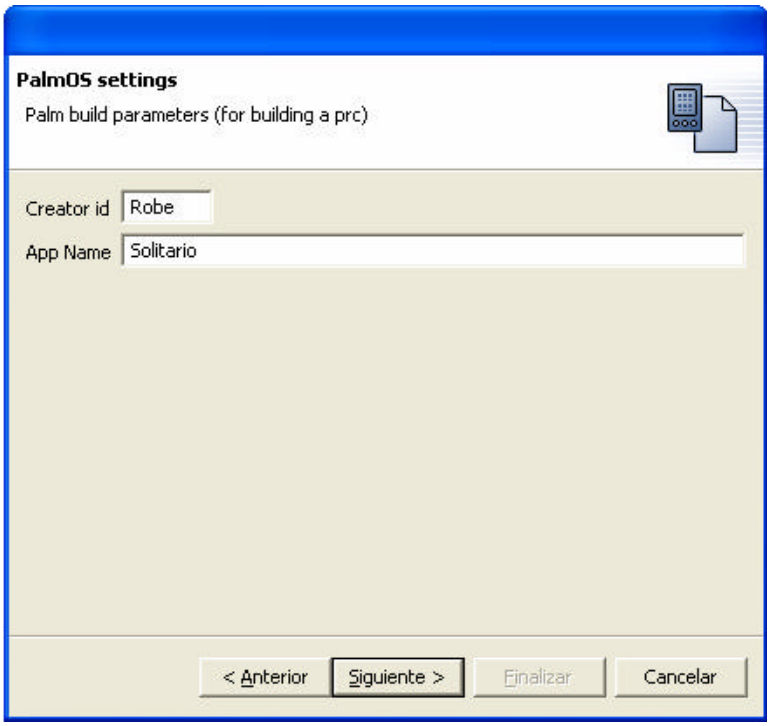


Debemos rellenar el campo sobre dónde está el ejecutable darle un nombre al dispositivo y pulsar el botón ‘Aceptar’ para la creación de este.

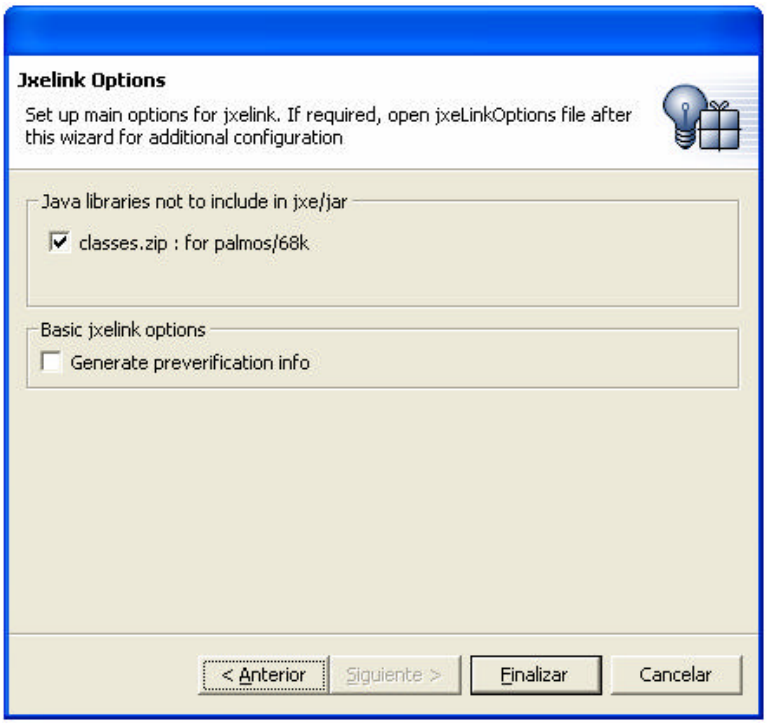
Después de esto debemos pinchar sobre el archivo ‘wsddbuild.xml’ para que se abra en la pantalla central, veremos que aparece un botón en el que aparece ‘Add Build’, pinchamos sobre él y rellenamos el formulario que aparece:



Dentro del campo 'Platform' debemos indicar "J9 for Palm 68k", luego debemos poner el jad de la aplicación y un nombre, tras esto damos a 'Siguiente >':

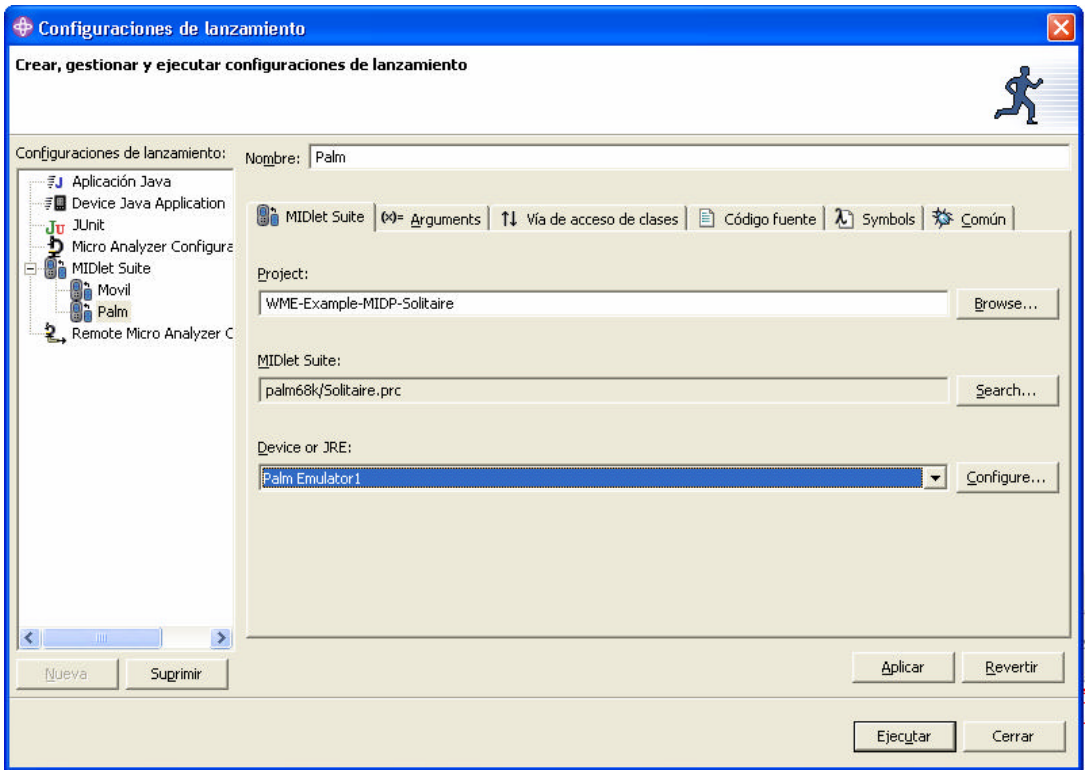


Ahora debemos indicar un identificador de cuatro letras y un nombre para la aplicación y pulsar en 'Siguiente >':

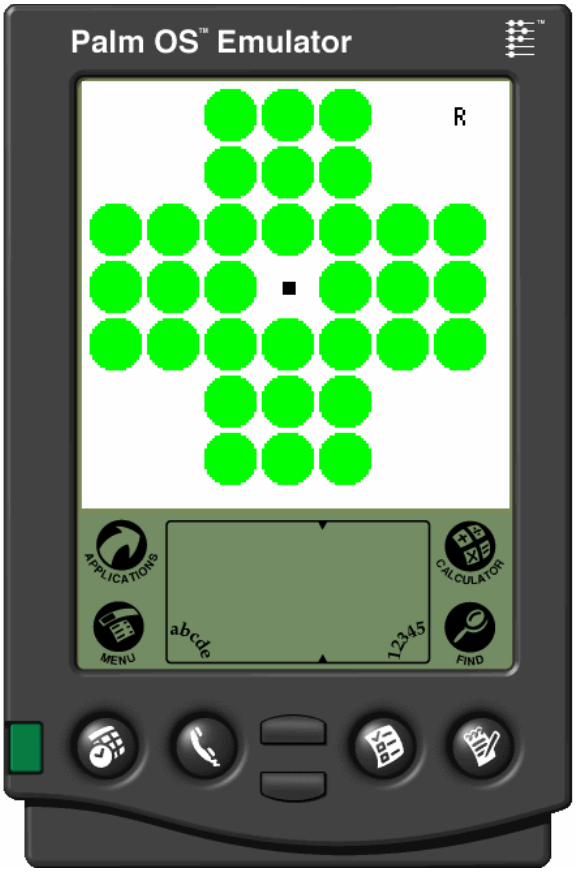


Seleccionamos las opciones que queramos y pulsamos en ‘Finalizar’ para que se construya lo necesario para su simulación sobre palm.

Tras realizar todo vamos al menú ‘Ejecutar’ y pinchamos sobre la opción ‘Ejecutar…’:



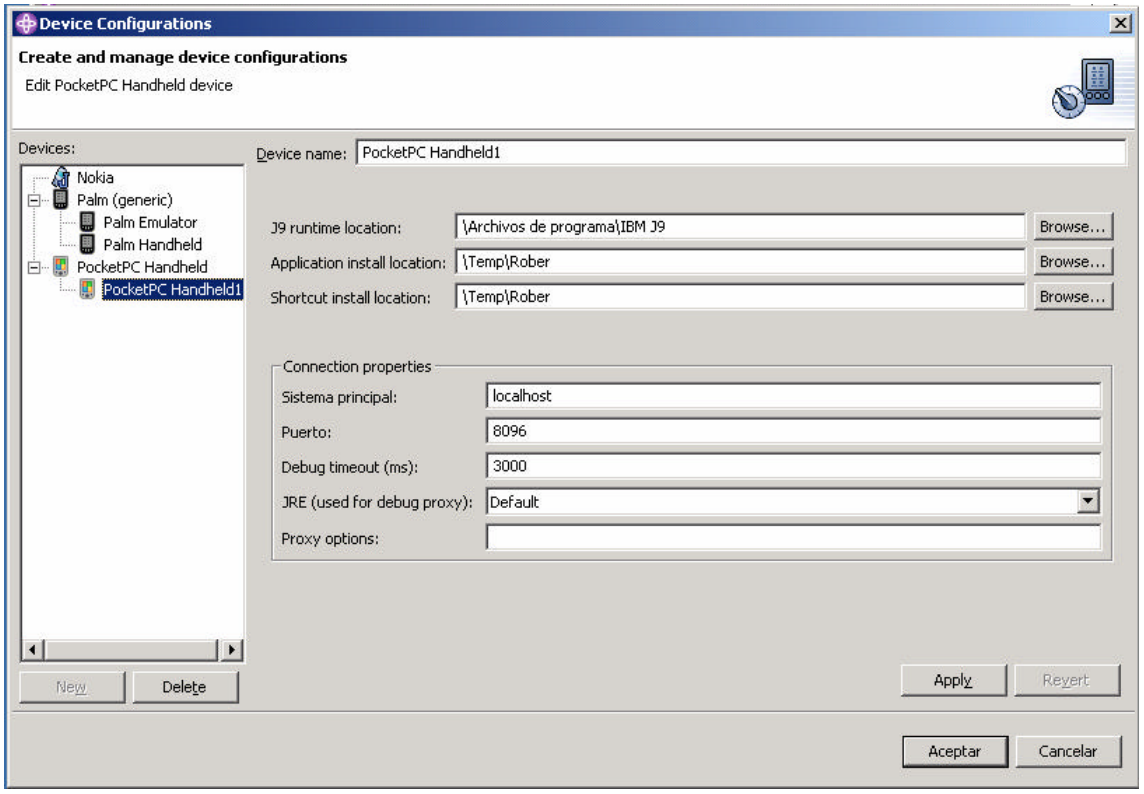
Aquí en la parte de la izquierda seleccionamos ‘MUDlet Suite’ y damos al botón ‘Nuevo’, seleccionamos el proyecto, el “MIDlet Suite” que se nos ha creado para la palm y el emulador que hemos creado, tras esto pulsamos ejecutar y podremos simular la aplicación que creamos, este es un ejemplo en algunos de los posibles emuladores:



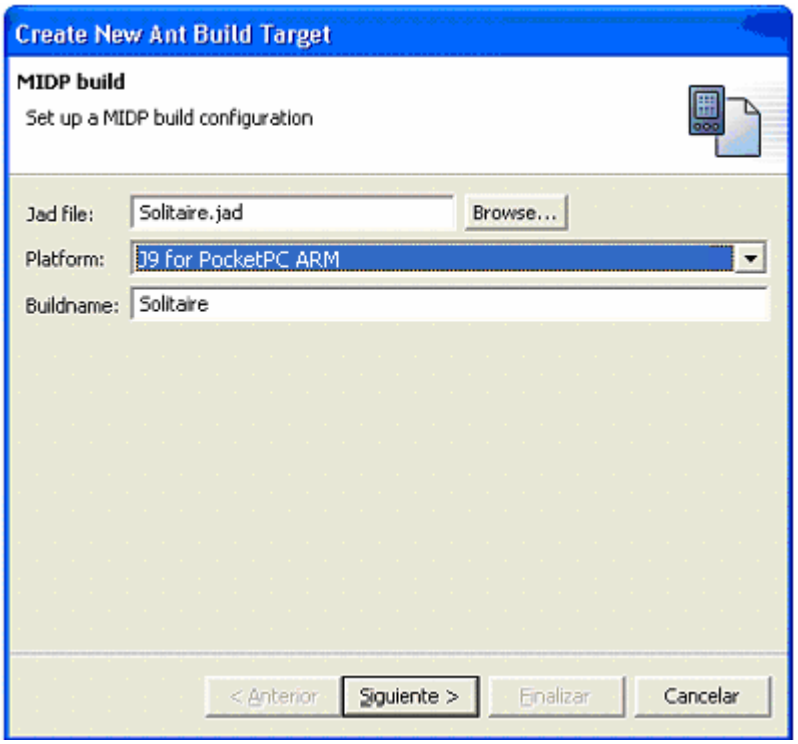
7.3 PRUEBA EN DISPOSITIVO REAL:

Primero iremos al menú ‘Devices’ y pincharemos sobre ‘Configure…’:

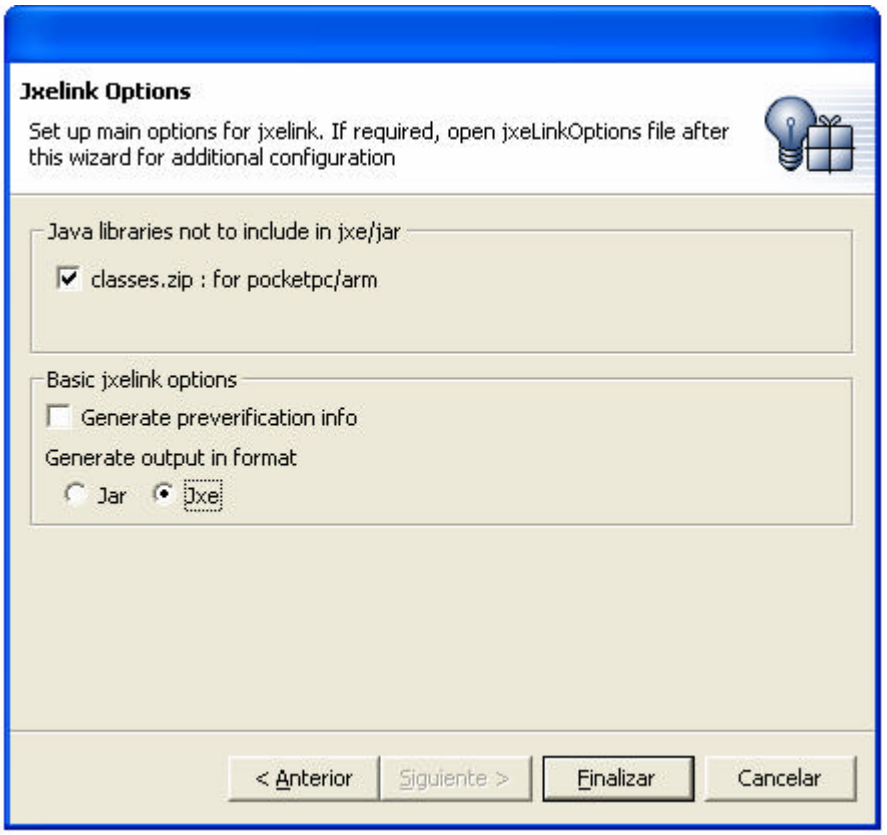
Seleccionamos ‘PocketPC Handheld’ y pinchamos sobre el botón ‘New’ e indicamos todo lo referente a dónde tiene instalado el J9 la PDA y el path dónde quéremos que se instale nuestra aplicación. También debemos introducir los datos de la conexión (dirección, puerto, …). Cuando hayamos introducido los datos pulsamos ‘Aceptar’.



Después de esto debemos pinchar sobre el archivo 'wsddbuild.xml' para que se abra en la pantalla central, veremos que aparece un botón en el que aparece 'Add Build', pinchamos sobre él y rellenamos el formulario que aparece:

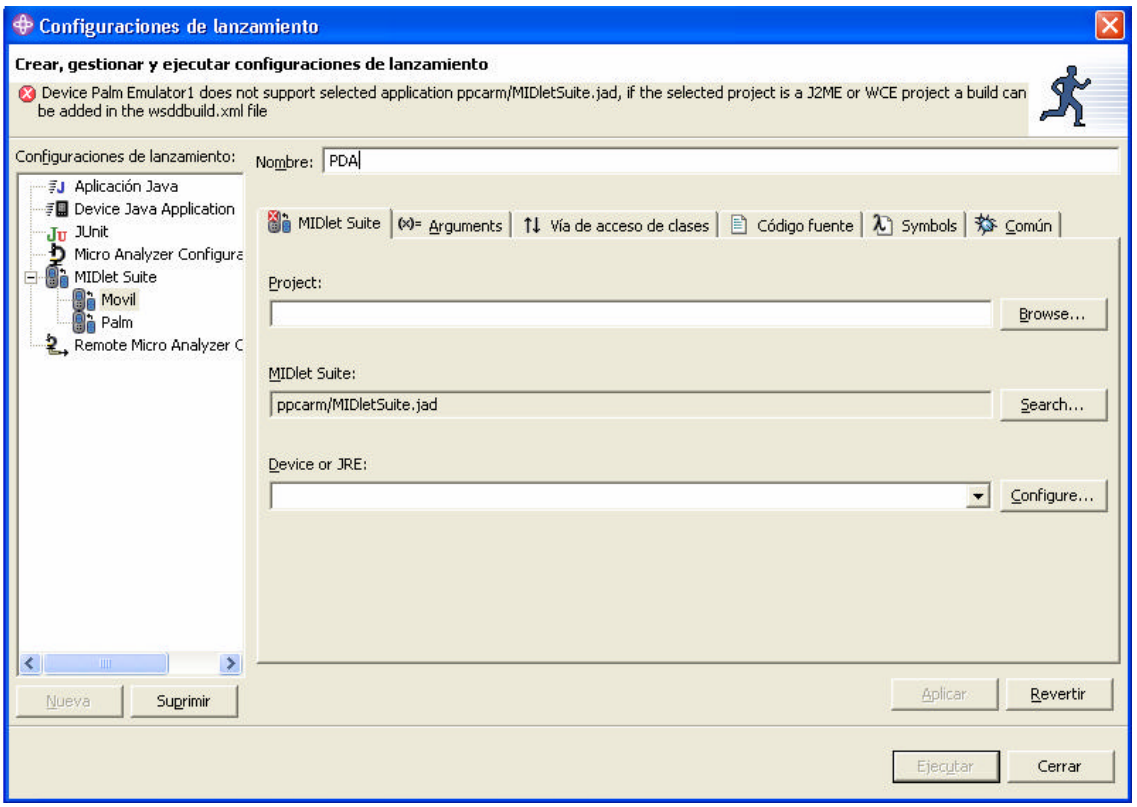


Dentro del campo ‘Platform’ debemos indicar “J9 for PocketPC ARM”, luego debemos poner el jad de la aplicación y un nombre, tras esto damos a ‘Siguiente >’:



Seleccionamos las opciones que queramos y pulsamos en ‘Finalizar’ para que se construya lo necesario para su simulación sobre PocketPC.

Tras realizar todo vamos al menú ‘Ejecutar’ y pinchamos sobre la opción ‘Ejecutar…’:



Aquí en la parte de la izquierda seleccionamos ‘MUDlet Suite’ y damos al botón ‘Nuevo’, seleccionamos el proyecto, el “MIDlet Suite” que se nos ha creado para el PocketPC y el dispositivo que hemos creado, tras esto pulsamos ejecutar y la aplicación se instalará en la PDA y podremos ejecutarla.

8 PROYECTO J2ME/MIDP PROPIO

8.1 LA APLICACIÓN:

La aplicación que vamos a realizar consiste en un juego que consiste en encontrar parejas de imágenes.

La aplicación dispondrá de tres botones:

- 'Jugar': Mediante el cual comenzaremos la partida.
- 'Menú': Mediante el cual iremos al menú de la aplicación.
- 'Salir': Mediante el cual finalizaremos la ejecución de la aplicación.

El menú dispondrá a su vez de una lista con tres opciones:

- 'Ayuda': Donde se explicarán las instrucciones de la partida.
- 'Descargar imágenes': Opción mediante la cual si disponemos de una conexión a internet HTTP podemos descargar nuevas imágenes para nuestras partidas.
- 'Records': Donde aparecen las cuatro mejores puntuaciones que hemos tenido.

Jugar:

Nos aparecerá un tablero de 3x4 en cada casilla hay una imagen, en total hay doce casillas y habrá seis imágenes que aparecerán dos veces cada una, en la parte inferior aparecerá una barra que irá disminuyendo de tamaño, cuando la barra se agote las imágenes se darán la vuelta y tendremos que encontrar todas las parejas antes de que una nueva barra que aparece debajo se agote.

Si encontramos todas las parejas pasamos a la siguiente pantalla, que será igual pero con un tiempo inferior para recordar y encontrar las parejas. Los puntos que recibiremos serán por parejas encontradas y por pantallas finalizadas, según completemos pantallas valdrán más puntos las cosas. Si cuando se nos agote el tiempo y no hallamos podido encontrar las parejas tenemos una de las cuatro mejores puntuaciones nos aparecerá un cuadro de texto para introducir nuestro nombre y aparecer en los records.

Descargar imágenes:

Nos aparecerá un cuadro de texto en el que introduciremos una URL (por ejemplo: http:// 163.127.144.103:80). Esta URL debe apuntar a una página que contenga código html del tipo:

<A HREF="http://80.103.135.226:80/animales/"> Animales</A>

<A HREF="http://80.103.135.226:80/banderas/">Banderas</A>

Debe apuntar a carpetas que contengan 13 imágenes llamadas 1.png, 2.png …., 13.png. Las doce primeras serán imágenes que pueden aparecer durante la partida y la número trece será la que aparezca cuando las imágenes estén dadas la vuelta.

En el ejemplo anterior aparecería una lista en la que aparecería:

Animales

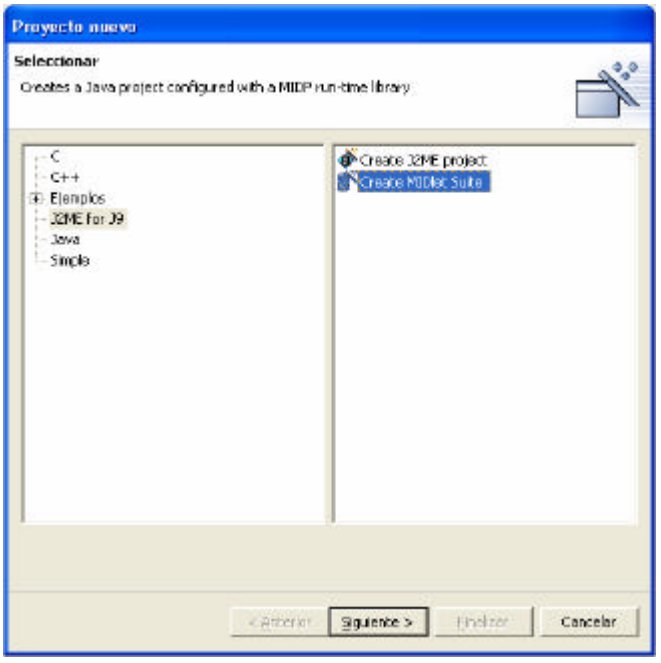
Banderas

Seleccionaríamos uno y se descargarían las imágenes.

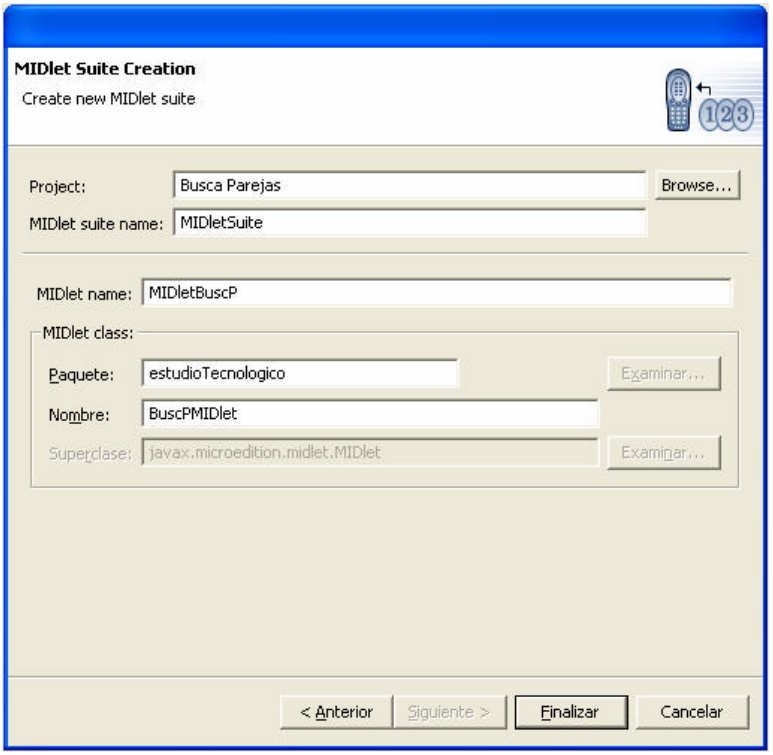
8.2 CREACIÓN:

Debemos ir al menú ‘Archivo’ y dentro de ‘Nuevo’ seleccionar proyecto.

Nos aparecerá una pantalla en la que en la parte de la izquierda debemos seleccionar ‘J2ME for J9’ y después en la parte de la derecha seleccionar la opción ‘Create MIDlet Suite’ que acaba de aparecer.



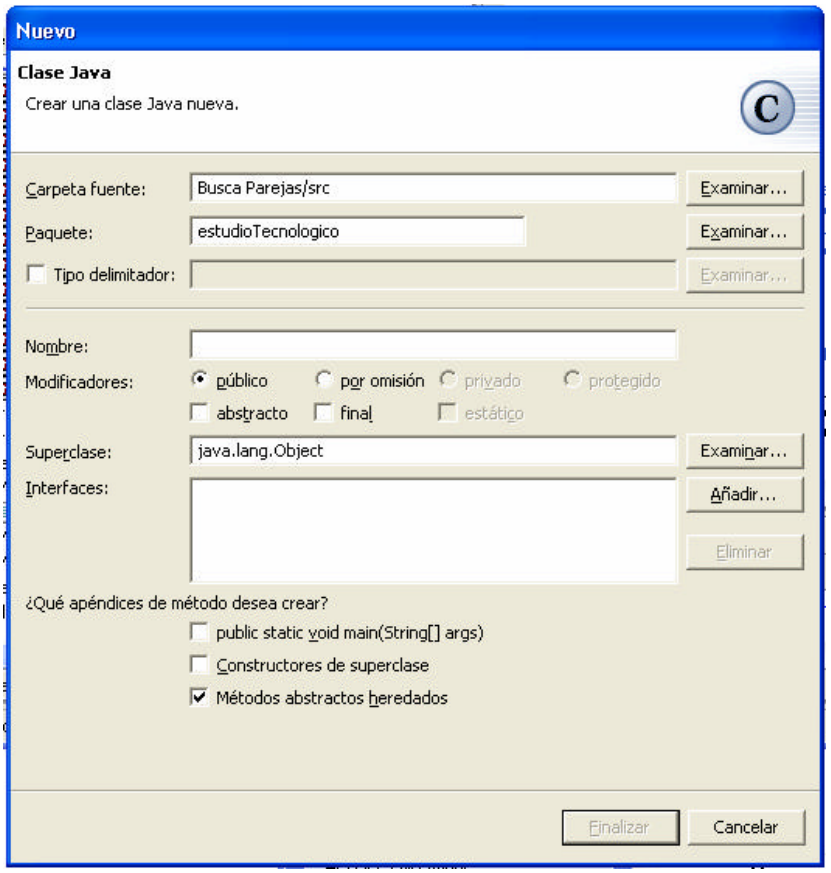
Tras seleccionar esto pulsamos ‘Siguiente >’ y aparecemos en una pantalla en la que seleccionamos el nombre del proyecto, en nuestro caso ‘Busca Parejas’, el nombre del MIDletSuite, el nombre del midlet y los datos de la clase del MIDlet, en nuestro caso hemos llamado al paquete ‘estudioTecnologico’ y a la clase principal ‘BuscPMIDlet’.



Tras rellenar los campos pulsamos ‘Finalizar’.

Podemos ver que se nos ha creado una estructura de este tipo:

Pinchando dos veces sobre BuscPMIDlet.java en la parte de la izquierda editamos el archivo, en el Anexo I viene el código de las clases, para crear una clase nueva pinchamos con el botón derecho sobre estudioTecnologico y seleccionamos 'Nuevo' y luego 'Clase' y nos aparece un pantalla en la que rellenamos los datos principales de la clase.



Una vez que creamos todas las clases del anexo tal y como aparecen (las clases BuscPMIDlet.java, BParC.java, RecordGuard.java, Temporizador.java, Temporizador2.java y Temporizador3.java) crearemos los archivos que necesitamos.

Necesitamos trece imágenes, las doce primeras serán las posibles imágenes a encontrar y la número trece será la imagen que aparecerá cuando todavía no se han encontrado las parejas.

En el ejemplo, por defecto vamos a utilizar estas imágenes de banderas de la unión europea:

Para la imagen que vamos a utilizar en las imágenes que están dadas la vuelta vamos a utilizar esta imagen:

Creamos las imágenes, las llamamos 1.png, 2.png, ….,12.png y la última 13.png, luego la forma más sencilla para llevarlas al "IBM Websphere Studio Device Developer" es seleccionarlas desde una carpeta y a la izquierda del programa en el arbol de directorios dónde pone 'estudioTecnologico'.

También necesitamos dos imágenes, una para cuando no aparece nada, llamada Portada.png y otra llamada Cargando.png para cuando está empezando una partida, vamos a utilizar estas:

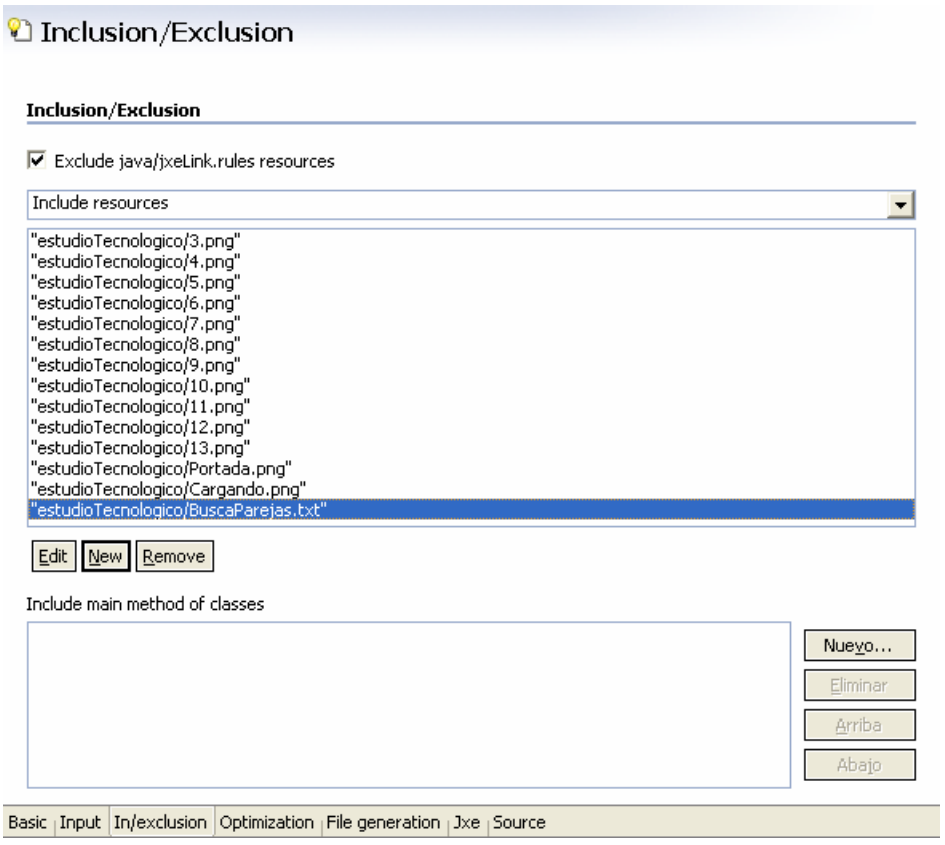
Finalmente creamos un archivo de texto llamado BuscaParejas.txt en el que aparezcan las instrucciones del juego y lo llevamos también a 'estudioTecnologico' de la misma forma, en el archivo escribiremos esto:

```
---------------------------
Licensed Materials - Property of Roberto,
(c) Copyright Roberto Corp. 2001, 2003
All Rights Reserved
---------------------------

Ayuda de BuscaParejas

El juego consiste en que se muestran una serie de imágenes durante un
periodo de tiempo.
```

```
Tras esto las imágenes se dan la vuelta y tienes que encontrar todas
las parejas antes de que se acabe el tiempo.

Cada vez que encuentras todas se inicia otra partida en la que tienes
menos tiempo.

Cuanto más parejas obtengas y menos tiempo tardes más puntos
conseguirás.

¡Buena suerte!
```

Lo último que debemos hacer es indicar en el archivo MIDletSuite-Classes.jxeLinkOptions.

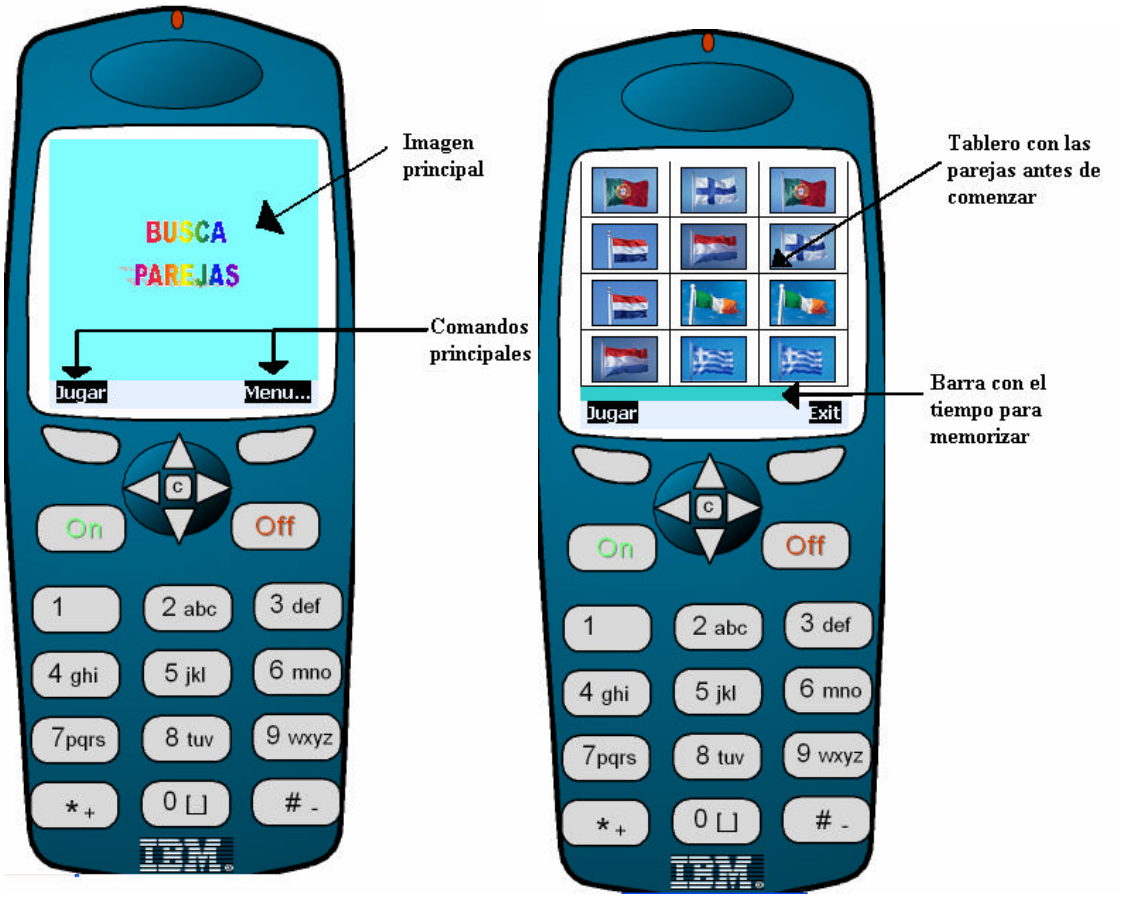
Editamos este archivo y abajo pulsamos en la pestaña 'In/Exclusión', en el desplegable de arriba seleccionamos 'Include resources' y añadimos todos los archivos que necesitamos (las imágenes y el txt):

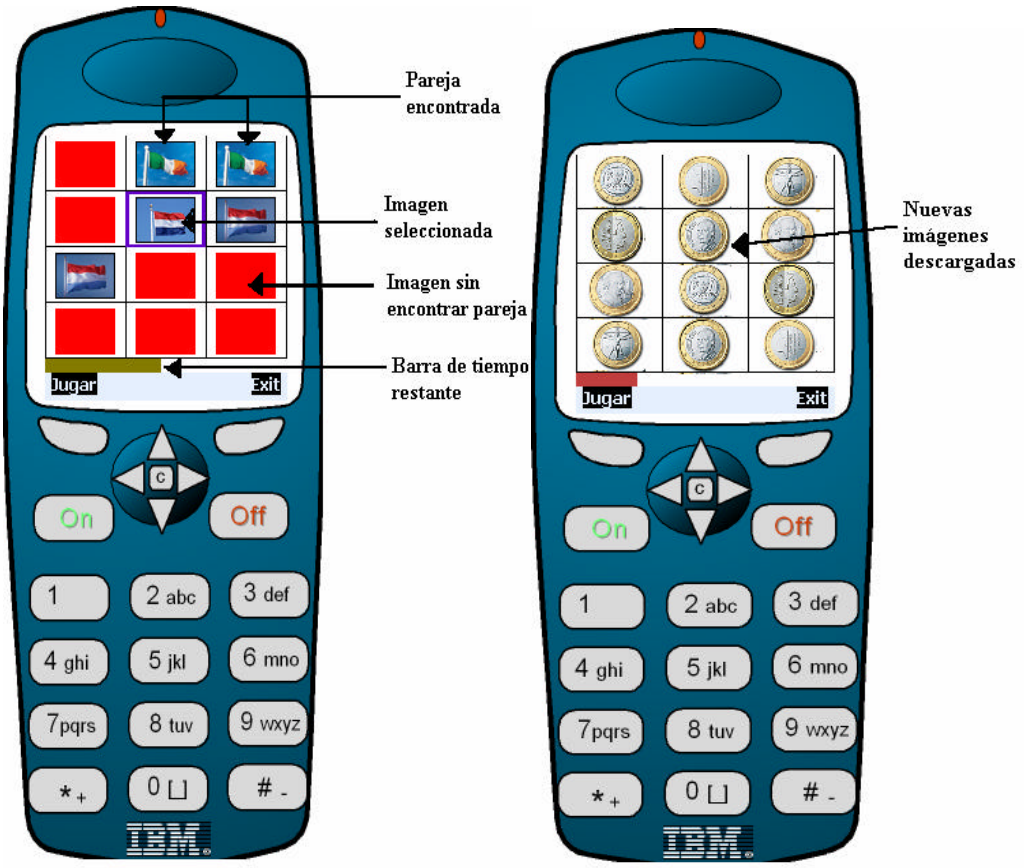


8.3 SIMULACIÓN:

La aplicación de desarrollo no nos permite simular en un "Pocket PC", debido a esto tendremos que simularlo en otro tipo de dispositivo como un móvil o una Palm, en

el apartado 7.2 se explica cómo simular con distintos móviles o en un emulador de palm. Ahora vamos a ver cómo simularlo con el teléfono móvil que viene por defecto, nos vamos al menú 'Ejecutar' y pinchamos sobre 'Ejecutar…', rellenamos los campos del formulario que aparece con el nombre del proyecto, el .jad y en el dispositivo seleccionamos 'Default', finalmente pulsamos el botón 'Ejecutar':





Para poder simular la parte de descargar imágenes se ha instalado el servidor apache en un ordenador con dirección IP 62.83.143.152, y se ha puesto como página principal esta página web:

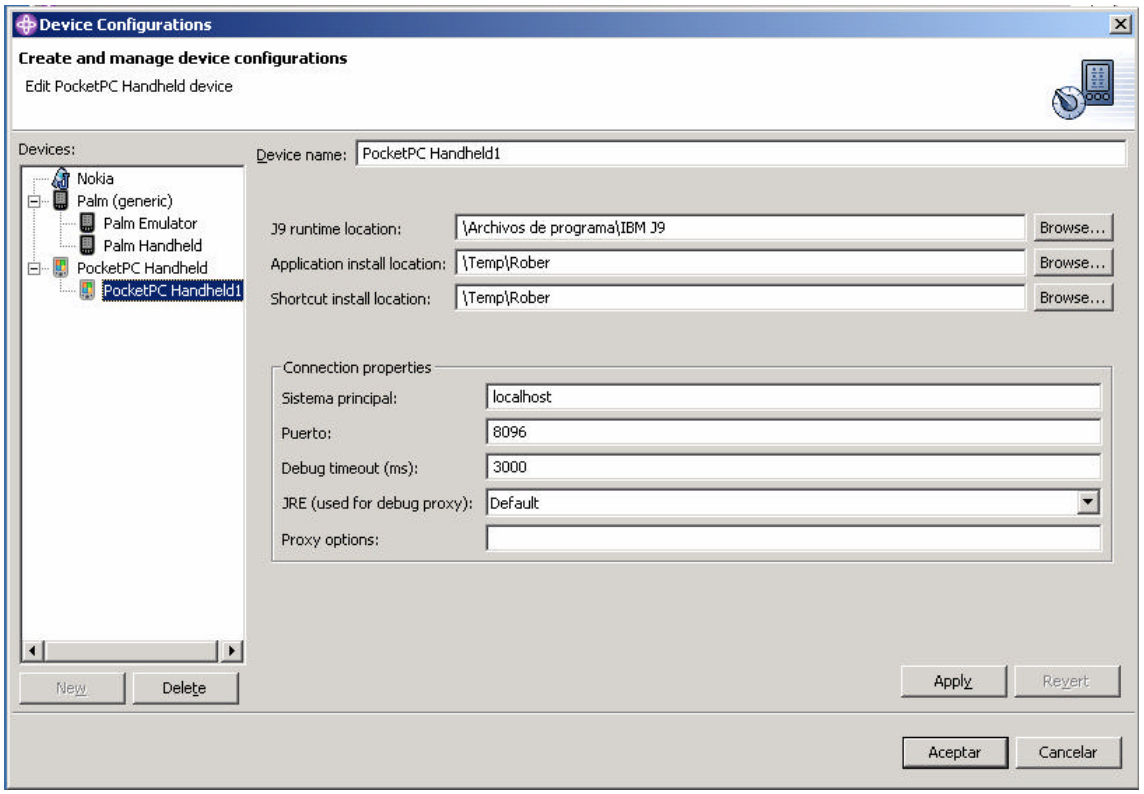
```
<HTML>
<HEAD>
<TITLE>
Imagenes para las partidas
</TITLE>
</HEAD>
<BODY>
<A HREF="http://62.83.143.152:80/euros/">Euros</A><BR>
<A HREF="http://62.83.143.152:80/banderas/">Banderas</A><BR>
</BODY>
</HTML>
```

Como se puede ver hay dos enlaces a dos sitios con imágenes, en una hemos puesto las que ya teníamos y en la otra imágenes de distintas monedas de euros europeas, para ello hemos creado dos carpetas dentro del servidor web, una llamada euros y otra llamada banderas y hemos guardado las imágenes correspondientes.

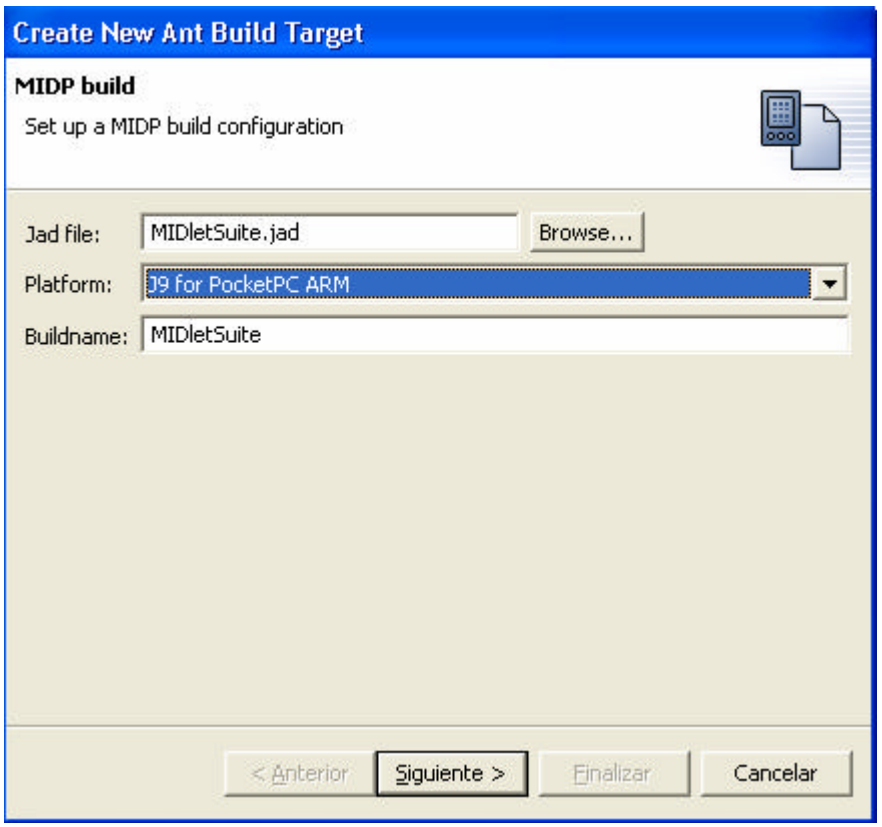
8.4 PRUEBA EN DISPOSITIVO REAL:

Primero iremos al menú 'Devices' y pincharemos sobre 'Configure…':

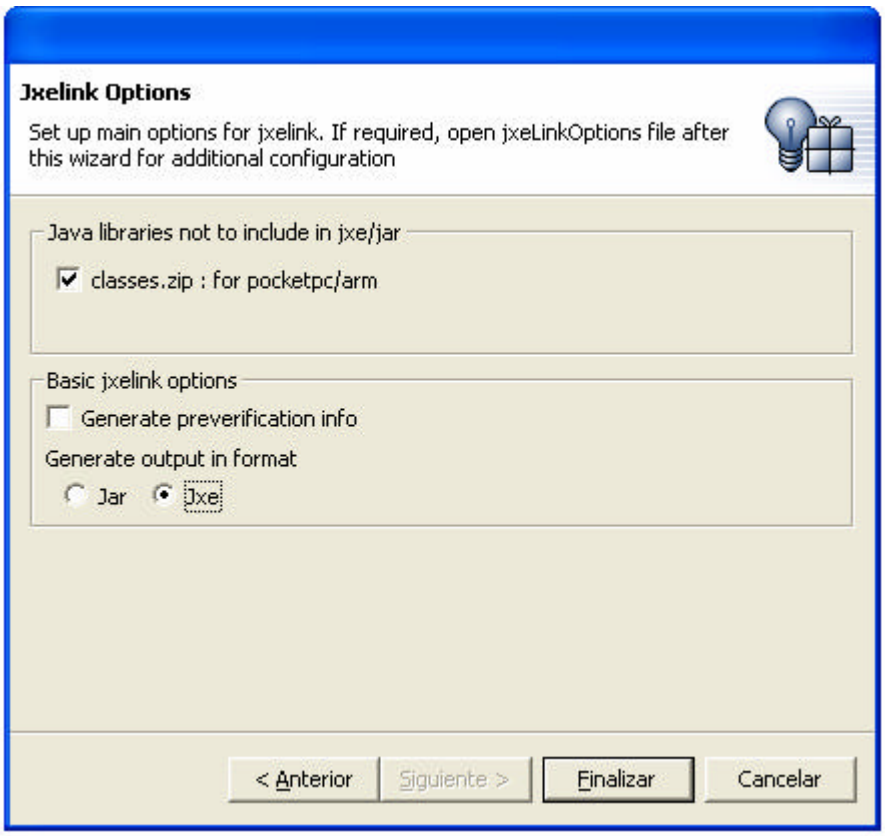
Seleccionamos 'PocketPC Handheld' y pinchamos sobre el botón 'New' e indicamos todo lo referente a dónde tiene instalado el J9 la PDA y el path dónde quéremos que se instale nuestra aplicación. También debemos introducir los datos de la conexión (dirección, puerto, …). Cuando hayamos introducido los datos pulsamos 'Aceptar'.



Después de esto debemos pinchar sobre el archivo 'wsddbuild.xml' para que se abra en la pantalla central, veremos que aparece un botón en el que aparece 'Add Build', pinchamos sobre él y rellenamos el formulario que aparece:

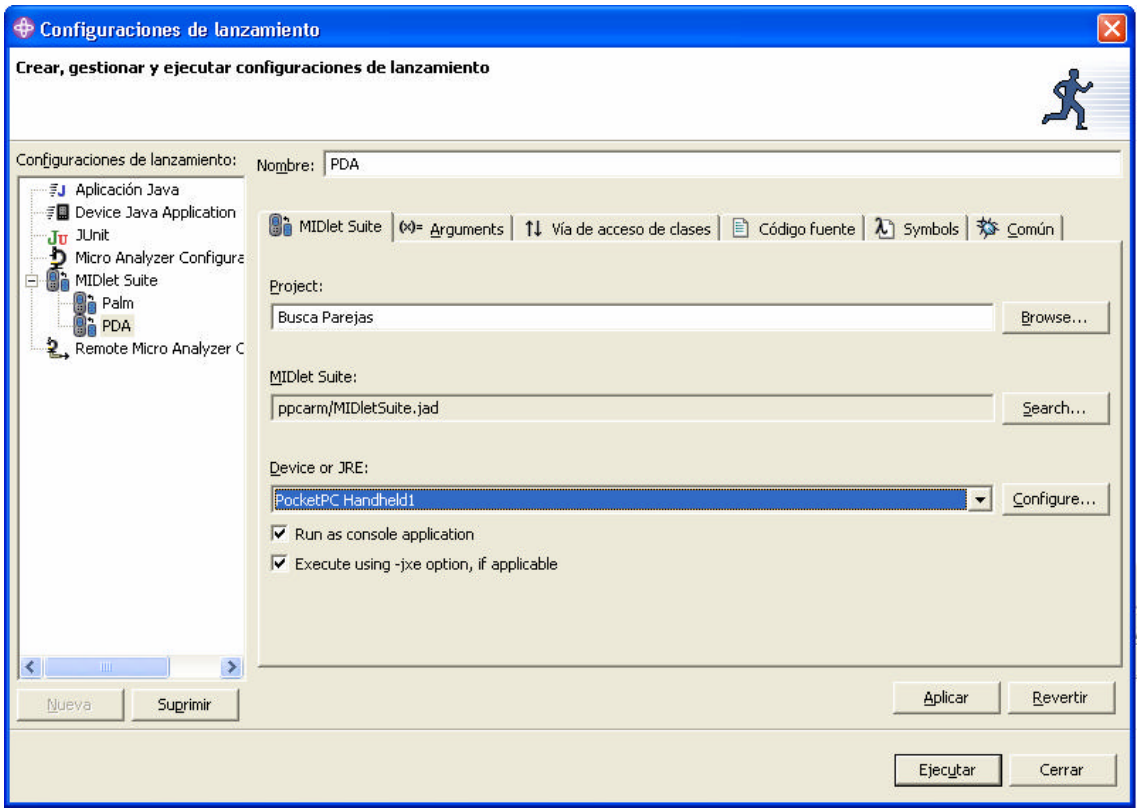


Dentro del campo 'Platform' debemos indicar "J9 for PocketPC ARM", luego debemos poner el jad de la aplicación y un nombre, tras esto damos a 'Siguiente >':



Seleccionamos las opciones que queramos y pulsamos en ‘Finalizar’ para que se construya lo necesario para su simulación sobre PocketPC.

Tras realizar todo vamos al menú ‘Ejecutar’ y pinchamos sobre la opción ‘Ejecutar…’:



Aquí en la parte de la izquierda seleccionamos ‘MUDlet Suite’ y damos al botón ‘Nuevo’, seleccionamos el proyecto, el “MIDlet Suite” que se nos ha creado para el PocketPC y el dispositivo que hemos creado, tras esto pulsamos ejecutar y la aplicación se instalará en la PDA y podremos utilizar nuestra aplicación.

9 ANEXO I(CÓDIGO FUENTE)

9.1) BuscPMIDlet.java

```
   package estudioTecnologico;

      // Clases importadas
   import java.io.*;
   import javax.microedition.lcdui.*;
   import javax.microedition.io.*;
   import javax.microedition.midlet.MIDlet;
   import javax.microedition.midlet.MIDletStateChangeException;
   import java.util.Vector;
   import java.util.Random;
   import java.util.Timer;
   import java.util.TimerTask;
   import javax.microedition.rms.*;
/**
* @author Roberto Díaz Morales
*
* Esta es la clase principal de la aplicación, contiene las variables
* de la aplicación y se encarga de gestionar las distintas pantallas
* que deben aparecer y la transición entre ellas.
*/
    public class BuscPMIDlet extends MIDlet implements CommandListener
{

   //Elementos de las interfaces gráficas
      public Display         fDisplay;
      public BParC                  fCanvas;
      public static TextBox nombre = null;
      private TextBox      fHelpPanel;
      public Form                   formulario;
      public List                   lista;
      public List                   listaBusqueda;
   //Comandos
      private Command      fJugarCmd    = new Command("Jugar",
Command.SCREEN, 1);
      private Command      fAceptarCmd = new
Command("Ok",Command.SCREEN,1);
      private Command      fOkCmd = new
Command("Ok",Command.SCREEN,1);
      private Command         fURLCmd = new
Command("Aceptar",Command.SCREEN,1);
      private Command      fOpcionCmd = new
Command("Aceptar",Command.SCREEN,1);
      private Command      fHelpBackCmd = new
Command("Back",Command.BACK,2);
      public  Command         fMenuCmd = new
Command("Menu",Command.SCREEN,2);
      private Command         fInicializarCmd = new
Command("Salir",Command.SCREEN,2);
      private Command      fExitCmd = new Command("Exit",
Command.SCREEN, 3);
   //Elementos para la pantalla de error
      public String        fHelpText;
   //Elementos para las mejores puntuaciones
```

```
   public int                    numrecords;
   public RecordStore      partidaguardada = null;
   public RecordGuard      puntrecords[]= new RecordGuard[4];
//Elementos para cargar imagenes
   public Vector                 menuIm;
   public Vector                 enlaces;
   public Image                  imagenes[]=new Image[13];
   public Image                  principal;
   public Image                  Cargando;
//Temporizador para la partida
   public Timer                  temporiz;
//Elementos de la partida
   public int                    estado;
   public boolean                elegido;
   public boolean                elegido2;
   public Image                  tablero[][]=new Image[4][3];
   public Image                  partida[][]=new Image[4][3];
   public Random                 rand=new Random();
   public int                    elegidoX;
   public int                    elegidoY;
   public int                    elegido2X;
   public int                    elegido2Y;
   public int                    tiempoVista=6000;
   public int                    tiempoJuego=30000;
   public int                    puntos=0;
   public int                    pantalla=1;
   public int                    numeroAciertos=0;


/**
 * Este es el primer método que se ejecuta al iniciar la aplicación
 * se encarga de hacer que salga por pantalla el canvas y añade los
 * comandos principales (Jugar, menú y salir).
 */
    public void startApp() throws MIDletStateChangeException {

      if (fDisplay != null)
         return;
      estado=0;
      fDisplay = Display.getDisplay(this);

      int numberOfColors = fDisplay.numColors();

      fCanvas = new BParC(numberOfColors,this);

      fCanvas.addCommand(fJugarCmd);
      fCanvas.addCommand(fMenuCmd);
      fCanvas.addCommand(fExitCmd);

      fCanvas.setCommandListener(this);
      fDisplay.setCurrent(fCanvas);
      inicializarImagenes();
      cargarRecords();
      fCanvas.repaint();
   }

/**
 * Este método no hace nada, pero como esta clase implementa a la
 * clase MIDlet debemos implementar el método.
 */
    protected void pauseApp() {
```

```
      }

   /**
     * Este método se ejecuta cuando finaliza la aplicación, se
     * encarga de guardar las mejores puntuaciones.
     */
       protected void destroyApp(boolean flag){
         guardarRecords();
      }

/**
  * Este método sirve para volver a poner los valores de las variables
  * dispuestos para iniciar una partida nueva
  */
       public void inicializarValores(){
         estado = 1;
         puntos = 0;
         pantalla = 1;
         numeroAciertos = 0;
         tiempoVista=6000;
         tiempoJuego=30000;
         elegido = false;
         elegido2 = false;
         inicializarTablero();
         fCanvas.repaint();
      }

/**
  * Este método se encarga de comprobar si el usuario ha acertado ya
  * todas las parejas de una pantalla, en ese caso actualiza las
  * variables para dejar todo preparado para la siguiente pantalla.
  */
       public void comprobar(){
         if(numeroAciertos>=6){
            numeroAciertos=0;
            temporiz.cancel();
            puntos = puntos+(pantalla*10);
            estado=5;
            pantalla++;
            if (tiempoJuego>=7000) tiempoJuego=tiempoJuego-2000;
            if (tiempoVista>=2200) tiempoVista=tiempoVista-350;
            elegido=false;
            elegido2=false;
            inicializarTablero();
            fCanvas.repaint();
         }
      }

   /**
    * Este método se encarga de inicializar la pantalla poniendo
    * las variables con sus valores correspondientes e iniciando
    * el temporizador que muestras las parejas antes de empezar
    * a buscarlas.
    */
       public void      empezarPartida(){
         estado=2;
         fCanvas.repaint();
         Temporizador temp = new Temporizador(this,1);
         temporiz = new Timer();
         temporiz.schedule((TimerTask)temp,100);
      }
```

```
   /**
    * Este método se encarga de cargar las imágenes que tiene por
    * defecto el juego.
    */
       public void inicializarImagenes(){
         for (int a=1;a<14;a++){
            try{
               imagenes[a-
1]=Image.createImage("estudioTecnologico/"+a+".png");
            }
                catch(Exception e){
                  System.out.println("No consigo leer la imagen");
               }
         }
         try{
            principal =
Image.createImage("estudioTecnologico/Portada.png");
         }
             catch(Exception e){
               System.out.println("No puedo cargar la imagen
Principal.png");
            }
         try{
            Cargando =
Image.createImage("estudioTecnologico/Cargando.png");
         }
             catch(Exception e){
               System.out.println("No puedo cargar Cargando.png");
            }
      }

/**
 * Este método se encarga de colocar todas las parejas en el tablero
 * de forma aleatoria para cada pantalla.
 */
       public void inicializarTablero(){
         tablero = new Image[4][3];
         partida = new Image[4][3];
         for(int a = 0;a<4;a++){
            for(int e=0;e<3;e++){
               partida[a][e]=imagenes[12];
            }
         }
         int aleatorios[]=new int[6];
         int aux;
         boolean valorCorrecto;
         for(int a=0;a<6;a++){
            valorCorrecto = true;
            aux=(rand.nextInt()/100+21474836)*12/42949673;
            for(int e=0;e<a;e++){
               if (aleatorios[e]==aux) valorCorrecto = false;
            }
            if (aux>=0 & aux<12 & valorCorrecto == true){
               aleatorios[a]=aux;
            }
            else{
               a=a-1;
            }
         }
         for (int a=0;a<6;a++){
```

```
            boolean nocolocado=true;
            while(nocolocado){
               nocolocado = colocarImagen(aleatorios[a]);
            }
            nocolocado=true;
            while(nocolocado){
               nocolocado = colocarImagen(aleatorios[a]);
            }
         }
      }

/**
* Este método elige una posición aleatoria del tablero, si está
* ocupada devuelve true y si está libre coloca la imagen que le
* pasamos como parámetro y devuelve false.
*/
       public boolean colocarImagen(int img){

         boolean devolver=true;
         int X = (rand.nextInt()/100+21474836)*3/42949673;
         int Y = (rand.nextInt()/100+21474836)*4/42949673;

         if (tablero[Y][X] == null){
            tablero[Y][X]= imagenes[img];
            devolver=false;
         }
         return devolver;
      }

/**
 * Este método se encarga de coger de la memoria las mejores
 * puntuaciones de las partidas anteriores.
 */
       public void cargarRecords(){
         try{

         // Abrimos el RecordStore
            partidaguardada =
RecordStore.openRecordStore("partida",true);
            numrecords = 0;

            try{
               String record1 = new
String(partidaguardada.getRecord(1));
               int puntos1 = (new Integer(0)).parseInt(
                  new String(partidaguardada.getRecord(2)),10);
               RecordGuard recordg1 = new
RecordGuard(record1,puntos1);
               puntrecords[0]=recordg1;
            }
                catch(Exception e){
                  RecordGuard recordg1 = new RecordGuard("",0);
                  puntrecords[0]=recordg1;
               }
            numrecords = 1;

            try{
               String record2 = new
String(partidaguardada.getRecord(3));
               int puntos2 = (new Integer(0)).parseInt(
                  new String(partidaguardada.getRecord(4)),10);
```

```
                RecordGuard recordg2 = new
RecordGuard(record2,puntos2);
                puntrecords[1]=recordg2;
             }
                 catch(Exception e){
                   RecordGuard recordg1 = new RecordGuard("",0);
                   puntrecords[1]=recordg1;
                }
             numrecords = 2;

             try{
                String record3 = new
String(partidaguardada.getRecord(5));
                int puntos3 = (new Integer(0)).parseInt(
                   new String(partidaguardada.getRecord(6)),10);
                RecordGuard recordg3 = new
RecordGuard(record3,puntos3);
                puntrecords[2]=recordg3;
             }
                 catch(Exception e){
                   RecordGuard recordg1 = new RecordGuard("",0);
                   puntrecords[2]=recordg1;
                }
             numrecords = 3;

             try{
                String record4 = new
String(partidaguardada.getRecord(7));
                int puntos4 = (int)(new Integer(0)).parseInt(
                   new String(partidaguardada.getRecord(8)),10);
                RecordGuard recordg4 = new
RecordGuard(record4,puntos4);
                puntrecords[3]=recordg4;
             }
                 catch(Exception e){
                   RecordGuard recordg1 = new RecordGuard("",0);
                   puntrecords[3]=recordg1;
                }
             numrecords = 4;
          }
              catch(Exception e){
             }
          try{
             partidaguardada.closeRecordStore();
          }

              catch(Exception e){
             }

       }

       /**
        * Este método se encarga de guardar en la memoria las mejores
        * puntuaciones de partidas anteriores.
        */
        public void guardarRecords(){

          try{
             RecordStore.deleteRecordStore("partida");
             partidaguardada =
RecordStore.openRecordStore("partida",true);
```

```
        }
            catch(Exception e){
           }
        try{
           byte a[];
           Integer entero = new Integer(0);

           a = puntrecords[0].nombre.getBytes();
           partidaguardada.addRecord(a,0,a.length);
           a = entero.toString(puntrecords[0].puntos).getBytes();
           partidaguardada.addRecord(a,0,a.length);

           a = puntrecords[1].nombre.getBytes();
           partidaguardada.addRecord(a,0,a.length);
           a = entero.toString(puntrecords[1].puntos).getBytes();
           partidaguardada.addRecord(a,0,a.length);

           a = puntrecords[2].nombre.getBytes();
           partidaguardada.addRecord(a,0,a.length);
           a = entero.toString(puntrecords[2].puntos).getBytes();
           partidaguardada.addRecord(a,0,a.length);

           a = puntrecords[3].nombre.getBytes();
           partidaguardada.addRecord(a,0,a.length);
           a = entero.toString(puntrecords[3].puntos).getBytes();
           partidaguardada.addRecord(a,0,a.length);
        }
            catch(Exception e){
           }
        try{
           partidaguardada.closeRecordStore();
        }

            catch(Exception e){
           }

     }

  /**
   * Este método se utiliza para que cuando alguien obtiene un nuevo
   * record situarlo en la posición correcta.
   */
      public void ordenarRecord(){

        RecordGuard rec = new RecordGuard(nombre.getString(),puntos);
        RecordGuard aux;

        if(numrecords != 0){
           for (int i=0;i < numrecords;i++){
              if(rec.puntos >= puntrecords[i].puntos){
                 aux = puntrecords[i];
                 puntrecords[i] = rec;
                 rec = aux;

              }

           }
           if (numrecords < 4){
              puntrecords[numrecords]= rec;
              numrecords = numrecords+1;
```

```
            }

         }
         else{
            puntrecords[0] = rec;
            numrecords = 1;

         }
         guardarRecords();
         cargarRecords();
      }

   /**
    * Este método se utiliza para realizar las acciones pertinentes
    * cuando alguien presiona alguno de los comandos.
    */
       public void commandAction(Command c, Displayable s){

         if (c == fExitCmd) {
            destroyApp(false);
            notifyDestroyed();
            return;
         }

         if (null != fHelpBackCmd) {
            if (c == fHelpBackCmd) {
               fDisplay.setCurrent(fCanvas);
               return;
            }
         }

         if (c == fOpcionCmd) {
            enlaceMenu();
            return;
         }
         if(c==fAceptarCmd){
            ordenarRecord();
            inicializarValores();
            estado=0;
            fCanvas.addCommand(fMenuCmd);
            fDisplay.setCurrent(fCanvas);
         }


         if(c==fOkCmd){
            cargarEnlace();
         }
         if(c==fInicializarCmd){
            inicializarValores();
            estado=0;
            fCanvas.addCommand(fMenuCmd);
            fDisplay.setCurrent(fCanvas);
         }

         if(c==fJugarCmd){
            inicializarValores();
            empezarPartida();
            fCanvas.removeCommand(fMenuCmd);
         }
         if (c == fMenuCmd) {
            pantallaMenu();
```

```
         }
         if (c == fURLCmd) {
            try{
               leerDeWeb(nombre.getString());
               displayBusqueda();
            }
                catch(Exception e){

               }
         }
      }

/**
 * Este método se encarga de coger las nuevas imágenes cuando tenemos
 * elegimos una opción dentro de las posibles que tiene la URL
 */
       public void cargarEnlace(){

         int indice = listaBusqueda.getSelectedIndex();
         HttpConnection c = null;
         InputStream is = null;
         Image imagenesAux[]=new Image[12];
         try{
            for(int i=1;i<14;i++){
               c = (HttpConnection)Connector.open(
                  (String)enlaces.elementAt(indice)+i+".png");
               is = c.openInputStream();
               String type = c.getType();
               int len = (int)c.getLength();
               if (len > 0) {
                  byte[] data = new byte[len];
                  int actual = is.read(data);
                  imagenesAux[i-
1]=Image.createImage(data,0,data.length);
               }
               else {
                  int ch;
                  while ((ch = is.read()) != -1) {

                  }
               }
               if (is != null) is.close();
               if (c != null) c.close();
            }
            imagenes = imagenesAux;
         }
             catch(Exception e){

            }
         fDisplay.setCurrent(fCanvas);
      }

/**
 * A este método le pasamos una URL y nos dice el menú de imagenes
 * que tenemos para elegir.
 */
       public void leerDeWeb(String URL) throws IOException{
         HttpConnection c = null;
         InputStream is = null;
         menuIm = new Vector();
         enlaces = new Vector();
```

```
        String pagina;
        String enlace;
        String nombreEnlace;
        try {
           c = (HttpConnection)Connector.open(URL);
           is = c.openInputStream();
           String type = c.getType();
           int len = (int)c.getLength();
           if (len > 0) {
              byte[] data = new byte[len];
              int actual = is.read(data);
              pagina = new String(data);
              int principio =0;
              int fin = 0;
              while(principio < pagina.length()){
                 principio = pagina.indexOf("A HREF=",principio);
                 if (principio != -1){
                    fin = pagina.indexOf(">",principio);
                    enlace = pagina.substring(principio+8,fin-1);
                    enlaces.addElement(enlace);
                    principio = fin+1;
                    fin = pagina.indexOf("</A><BR>",principio);
                    nombreEnlace = pagina.substring(principio,fin);
                    menuIm.addElement(nombreEnlace);
                 }
                 else principio= pagina.length();
              }
           }
           else {
              int ch;
              while ((ch = is.read()) != -1) {

              }
           }
        }
            catch(Exception e){

           }
        finally {
           if (is != null)
              is.close();
           if (c != null)
              c.close();
        }

     }

  /**
   * Este método nos muestra la pantalla que aparece cuando tenemos
   * un nuevo record para introducir nuestro nombre.
   */
      public void pantallaGanador(){

        nombre = new TextBox("Nombre", "", 512,0);
        nombre.addCommand(fAceptarCmd);
        nombre.addCommand(fInicializarCmd);
        fDisplay.setCurrent(nombre);
        nombre.setCommandListener(this);


     }
```

```
    /**
     * Este método se encarga de sacar la pantalla del menú
     */
        public void pantallaMenu(){
          formulario = null;
          lista = new List("    MENU",List.IMPLICIT);

          lista.insert(0,"Ayuda",null);
          lista.insert(1,"Descargar imagenes",null);
          lista.insert(2,"Records",null);

          lista.addCommand(fOpcionCmd);
          lista.addCommand(fHelpBackCmd);
          fDisplay.setCurrent(lista);

          lista.setCommandListener(this);

        }

        /**
         * Este método se encarga de ver qué opción hemos elegido del
         * menú y realizar las acciones pertinentes.
         */
         public void enlaceMenu(){

           int indice = lista.getSelectedIndex();

           switch (indice){
              case 0:
                 displayHelp();
                 break;
              case 1:
                 try{
                    pantallaURL();
                 }
                     catch(Exception e){

                    }

                 break;
              case 2:
                 Records();
                 //pantallaMenu();
                 break;
           }
        }

    /**
     * Esta pantalla se encarga de sacar la pantalla dónde aparecen las
     * mejores puntuaciones.
     */
        public void Records(){
          formulario = new Form("RECORDS ");
          for (int i=0;i < numrecords;i++){
             formulario.append(
                puntrecords[i].nombre+"  "+puntrecords[i].puntos+"
puntos\n");
          }
          formulario.addCommand(fMenuCmd);
          formulario.setCommandListener(this);
```

```
            fDisplay.setCurrent(formulario);
         }

         /**
          * Este método se encarga de sacar la pantalla dónde aparece la
          * ayuda del juego.
          */
          public void displayHelp() {

            if (null == fHelpText) fHelpText = getHelpText();

            if (null == fHelpPanel) {
               fHelpPanel = new TextBox(
                  "BuscaParejas
Help",fHelpText,fHelpText.length(),TextField.ANY);
               fHelpPanel.addCommand(fMenuCmd);
               fHelpPanel.setCommandListener(this);
            }

            fDisplay.setCurrent(fHelpPanel);

         }

      /**
       * Este método se encarga de dibujar la pantalla para introducir
       * la URL para la búsqueda de imágenes.
       */
          public void pantallaURL(){

            nombre = new TextBox("Introduce URL", "", 512,0);
            nombre.addCommand(fURLCmd);
            nombre.addCommand(fMenuCmd);
            fDisplay.setCurrent(nombre);
            nombre.setCommandListener(this);


         }

      /**
       * Este método se encarga de dibujar la pantalla dónde aparece el
       * menú con las distintas opciones que ha salido de la URL para
       * buscar imágenes
       */
          public void displayBusqueda(){

         // Colocamos el titulo
            listaBusqueda = new List("Tipos de imagenes",List.IMPLICIT);

         // Colocamos los distintos menus
            for(int i = 0;i<menuIm.size();i++){
               listaBusqueda.insert(i,(String)menuIm.elementAt(i),null);
            }
            // Añadimos los comandos

            listaBusqueda.addCommand(fOkCmd);
            listaBusqueda.addCommand(fMenuCmd);
            fDisplay.setCurrent(listaBusqueda);

         // Hacemos que los menus escuchen eventos
            listaBusqueda.setCommandListener(this);
```

```
      }

/**
  * Este método se encarga de obtener el texto de la ayuda del fichero
  * dónde está escrito.
  */

       private String getHelpText() {
         InputStream  is;
         StringBuffer sb;

         is = getClass().getResourceAsStream("BuscaParejas.txt");
         if (null == is)
            return "Help is not available: text not found.";

         sb = new StringBuffer(200);

         try {
            byte[] buffer = new byte[200];
            int    read   = is.read(buffer);

            while (read > 0) {
               String string = new String(buffer,0,read);
               string = string.replace('\r',' ');
               sb.append(string);
               read = is.read(buffer);
            }

            is.close();

            return sb.toString();
         }
             catch (IOException e) {
               return "Help is not available: error reading text.";
            }

      }

   }
```

2) BParC.java

```
package estudioTecnologico;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.TimerTask;
import java.util.Timer;

/**
* @author Roberto Díaz Morales
*
* Esta clase es la encargada de dibujar el tablero y realizar las
* acciones oportunas cuando presionamos con el puntero en la pantalla.
*/
    public class BParC extends Canvas {

    // Estas son las variables sobre el tamaño
      int fcanvasWidth;
```

```
      int fcanvasHeight;
      private int     fPegWidth;
      private int     fPegHeight;
      private int     fNumberOfColors;
      final static private int  gRows = 4;
      final static private int  gCols = 3;
    // Este es el MIDlet de la aplicación
      BuscPMIDlet padre;

   /**
    * Este método es el constructor, se le pasa como parámetro
    * el número de colores y el MIDlet de la aplicación
    */
       BParC(int numberOfColors,BuscPMIDlet midl) {
         super();
         padre  = midl;
         fcanvasWidth  = getWidth();
         fcanvasHeight = getHeight()-10;
         System.out.println("Anchura ="+fcanvasWidth);
         System.out.println("Altura ="+fcanvasHeight);
         fNumberOfColors = numberOfColors;
         fPegWidth       = fcanvasWidth  / gCols;
         fPegHeight      = fcanvasHeight / gRows;


      }

/**
 * Este método se encarga de realizar las acciones oportunas cuando el
 * usuario presiona con el puntero sobre la pantalla, como por ejemplo
 * al elegir una imagen del tablero o pasar a la siguiente pantalla
 * cuando el usuario a finalizado una.
 */
       public void pointerPressed(int x, int y) {
         if(padre.estado==3){
            int coordX=x/fPegWidth;
            int coordY=y/fPegHeight;


if(((padre.partida[coordY][coordX]).equals(padre.imagenes[12]))){

               if(padre.elegido){
                  if((padre.tablero[coordY][coordX]).equals(
                  padre.tablero[padre.elegidoY][padre.elegidoX])){
                     if(coordX==padre.elegidoX &&
coordY==padre.elegidoY){
                     }
                     else{
                        padre.partida[padre.elegidoY][padre.elegidoX]=
                        padre.tablero[padre.elegidoY][padre.elegidoX];

padre.partida[coordY][coordX]=padre.tablero[coordY][coordX];
                        padre.numeroAciertos++;
                        padre.puntos = padre.puntos+padre.pantalla;
                     }
                     padre.elegido=false;
                  }
                  else{
                     padre.elegido2=true;
                     padre.elegido2X=coordX;
                     padre.elegido2Y=coordY;
```

```
                    Temporizador3 temp = new
Temporizador3(this.padre);
                    Timer t = new Timer();
                    t.schedule((TimerTask)temp,400);
                 }

              }
              else{

                 padre.elegido=true;
                 padre.elegidoX=coordX;
                 padre.elegidoY=coordY;
              }
           }

        }
        if(padre.estado==4){
           if(padre.puntrecords[3].puntos<padre.puntos){
              padre.pantallaGanador();
           }
           else{
              padre.inicializarValores();
              padre.estado=0;
              addCommand(padre.fMenuCmd);
           }
        }
        if(padre.estado==5){
           padre.empezarPartida();
        }
        repaint();
        padre.comprobar();
     }

/**
* Este método se encarga de dibujar el tablero, las imágenes y los
* textos de GAME OVER o el que aparece entre una pantalla y la
* siguiente.
*/
      protected void paint(Graphics g) {
        g.setColor(255,255,255);
        g.fillRect(0,0,fcanvasWidth,fcanvasHeight+10);
        g.setColor(0,0,0);

        if(padre.estado==0){

           g.drawImage(padre.principal, fcanvasWidth/2,
            fcanvasHeight/2+5, (Graphics.VCENTER|Graphics.HCENTER));
        }

        if(padre.estado==1){
           g.drawImage(padre.Cargando, fcanvasWidth/2,
            fcanvasHeight/2+5, (Graphics.VCENTER|Graphics.HCENTER));
        }

        if(padre.estado==2){
           Image imagen;

           try{
              for(int a=0;a<3;a++){
                 for(int e=0;e<4;e++){
```

```
                    g.drawImage(padre.tablero[e][a],
fPegWidth/2+a*fPegWidth,
                     fPegHeight/2+fPegHeight*e,
(Graphics.VCENTER|Graphics.HCENTER));
                  }
               }
            }
                catch(Exception e){

               }
            for(int e=0;e<5;e++){
               g.drawLine(0,fPegHeight*e-1,fcanvasWidth-
1,fPegHeight*e-1);
            }
            for(int a=0;a<4;a++){
               g.drawLine(fPegWidth*a,0,fPegWidth*a,fcanvasHeight-1);
            }
         int ancho =
(fcanvasWidth*Temporizador.numero*100)/padre.tiempoVista;
         int colorin = ancho*254/fcanvasWidth;
         System.out.println("colorin "+colorin);
         g.setColor(colorin%255,(255-colorin)%255,(255-colorin)%255);
         g.fillRect(0,fcanvasHeight,fcanvasWidth-ancho,10);
         g.setColor(0,0,0);
         }

         if(padre.estado==3){
            Image imagen;
            try{
               for(int a=0;a<3;a++){
                  for(int e=0;e<4;e++){
                     g.drawImage(padre.partida[e][a],
fPegWidth/2+a*fPegWidth,
                      fPegHeight/2+fPegHeight*e,
(Graphics.VCENTER|Graphics.HCENTER));
                  }
               }
            }
                catch(Exception e){

               }
            for(int e=0;e<5;e++){
               g.drawLine(0,fPegHeight*e-1,fcanvasWidth-
1,fPegHeight*e-1);
            }
            for(int a=0;a<4;a++){
               g.drawLine(fPegWidth*a,0,fPegWidth*a,fcanvasHeight-1);
            }
            if(padre.elegido){
               g.setColor(255,0,0);

g.drawImage(padre.tablero[padre.elegidoY][padre.elegidoX],
                fPegWidth/2+padre.elegidoX*fPegWidth,
                   fPegHeight/2+fPegHeight*padre.elegidoY,
                   (Graphics.VCENTER|Graphics.HCENTER));

g.fillRect(padre.elegidoX*fPegWidth,padre.elegidoY*fPegHeight,
                fPegWidth,fPegHeight/10);

g.fillRect(padre.elegidoX*fPegWidth,padre.elegidoY*fPegHeight,
                fPegWidth/10,fPegHeight);
```

```
                g.fillRect(((padre.elegidoX+1)*fPegWidth),
                (padre.elegidoY*fPegHeight),(fPegWidth/10),fPegHeight);
                g.fillRect((padre.elegidoX*fPegWidth),

((padre.elegidoY+1)*fPegHeight),fPegWidth,(fPegHeight/10));
                g.setColor(0,0,0);
             }
             if(padre.elegido2){

g.drawImage(padre.tablero[padre.elegido2Y][padre.elegido2X],
                 fPegWidth/2+padre.elegido2X*fPegWidth,
                    fPegHeight/2+fPegHeight*padre.elegido2Y,
                    (Graphics.VCENTER|Graphics.HCENTER));
                g.setColor(255,0,0);
                g.fillRect(padre.elegido2X*fPegWidth,
                 padre.elegido2Y*fPegHeight,fPegWidth,fPegHeight/10);
                g.fillRect(padre.elegido2X*fPegWidth,
                 padre.elegido2Y*fPegHeight,fPegWidth/10,fPegHeight);
                g.fillRect(((padre.elegido2X+1)*fPegWidth),

(padre.elegido2Y*fPegHeight),(fPegWidth/10),fPegHeight);
                g.fillRect((padre.elegido2X*fPegWidth),

((padre.elegido2Y+1)*fPegHeight),fPegWidth,(fPegHeight/10));
                g.setColor(0,0,0);
             }
          int ancho =
(fcanvasWidth*Temporizador2.numero*100)/padre.tiempoJuego;
          int colorin = ancho*254/fcanvasWidth;
          System.out.println("colorin "+colorin);
          g.setColor(colorin%255,(255-colorin)%255,0);
          g.fillRect(0,fcanvasHeight,fcanvasWidth-ancho,10);
          g.setColor(0,0,0);

          }
          if(padre.estado==4){
             Font f = Font.getDefaultFont();
             g.setFont(f);
             String str;
             if(padre.puntrecords[3].puntos<padre.puntos){
                str = "GAME OVER, nuevo record";
             }
             else{
                str = "GAME OVER";
             }
             g.drawString(str, fcanvasWidth/2, fcanvasHeight/4 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);
             String str3 = "Puntos = "+(padre.puntos);
             g.drawString(str3, fcanvasWidth/2, fcanvasHeight/2 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);
             String str2 = "Pulse la pantalla para continuar ...";
             g.drawString(str2, fcanvasWidth/2, 3*fcanvasHeight/4 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);

          }
          if(padre.estado==5){
             Font f = Font.getDefaultFont();
             g.setFont(f);
             String str = "Pantalla "+(padre.pantalla-1)+" completa";
             g.drawString(str, fcanvasWidth/2, fcanvasHeight/4 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);
```

```
            String str3 = "Puntos = "+(padre.puntos);
            g.drawString(str3, fcanvasWidth/2, fcanvasHeight/2 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);
            String str2 = "Pulse la pantalla para continuar ...";
            g.drawString(str2, fcanvasWidth/2, 3*fcanvasHeight/4 +
f.getBaselinePosition(), g.BASELINE|g.HCENTER);

         }
      }
   }
```

3) RecordGuard.java

```
   package estudioTecnologico;
//Esta clase representa uno de los records de las partidas anteriores

    public class RecordGuard{
   // Este atributo indica el nombre introducido por la persona que ha
conseguido el record
      public String nombre;
   // Estas son el nº de bombas que le quedaban por usar
      public int puntos;

   // Este es el constructor al que se le pasan como parámetro los dos
atributos.
       public RecordGuard(String nom,int punt){
         nombre = nom;
         puntos = punt;

      }


   }
```

4) Temporizador.java

```
   package estudioTecnologico;
// Esta clase sirve para controlar el tiempo que se muestran
// las imagenes de las parejas antes de cada pantalla
   import java.util.TimerTask;
   import java.util.Timer;


    public class Temporizador extends TimerTask{

   // Este es el midlet de la aplicación
      BuscPMIDlet padre;
   // Este entero sirve para calcular la barra que aparece abajo
        public static  int numero = 0;

        /*
         * Este es el constructuor de la clase.
         */
       public Temporizador(BuscPMIDlet p, int n){
         padre=p;
         numero = n;
      }
/*
 * Este método se ejecuta a los 100 ms de crearse el objeto,
```

```
 * si todavía no ha finalizado el tiempo actualiza los valores
 * y crea otro objeto y si ya ha finalizado actualiza los valores
 * para que comencemos a jugar en la pantalla
 */
       public void run(){

           if(numero*100<padre.tiempoVista){
         Temporizador temp = new Temporizador(this.padre,numero+1);
         Timer temporiz = new Timer();
         temporiz.schedule((TimerTask)temp,100);

         padre.fCanvas.repaint();
           }else{
           numero=0;
         padre.estado=3;
         Temporizador2 temp = new Temporizador2(this.padre,0);
         padre.temporiz = new Timer();
         padre.temporiz.schedule(temp,100);
         padre.fCanvas.repaint();
           }

      }

   }
```

5) Temporizador2.java

```
   package estudioTecnologico;
// Esta clase sirve para controlar el tiempo que dura
// cada pantalla
   import java.util.TimerTask;
   import java.util.Timer;


    public class Temporizador2 extends TimerTask{
   // Este es el midlet de la aplicación
      BuscPMIDlet padre;
   // Este entero sirve para calcular la barra que aparece abajo
        public static  int numero = 0;

        /*
         * Este es el constructuor de la clase.
         */
       public Temporizador2(BuscPMIDlet p, int n){
         padre=p;
         numero = n;
      }

      /*
       * Este método se ejecuta a los 100 ms de crearse el objeto,
       * si todavía no ha finalizado el tiempo actualiza los valores
       * y crea otro objeto y si ya ha finalizado actualiza los
valores
       * para que termine la partida
       */
       public void run(){
        if(numero*100<padre.tiempoJuego){
         Temporizador2 temp = new Temporizador2(this.padre,numero+1);
         Timer temporiz = new Timer();
         temporiz.schedule((TimerTask)temp,100);

         padre.fCanvas.repaint();
```

```
            }else{
             numero = 0;
             if(padre.estado==3){
          padre.estado=4;
             }
             padre.fCanvas.repaint();
        }
      }

   }
```

6) Temporizador3.java

```
   package estudioTecnologico;
// Esta clase sirve para controlar el tiempo que dura
// una pareja dada la vuelta si no hemos acertado
   import java.util.TimerTask;
   import java.util.Timer;


    public class Temporizador3 extends TimerTask{

   // Este es el midlet padre
      BuscPMIDlet padre;

        /*
         * Este es el constructuor de la clase.
         */

      public Temporizador3(BuscPMIDlet p){
         padre=p;
      }

      // Cuando finaliza el tiempo que hemos puesto se actualizan
      // los valores
       public void run(){
         padre.elegido=false;
         padre.elegido2=false;
         padre.fCanvas.repaint();

      }

   }
```