Informática Móvil

Diseño de aplicaciones con J2ME

Interfaz de Usuario: API de Alto Nivel

José Ramón Arias García
Ignacio Marín Prendes

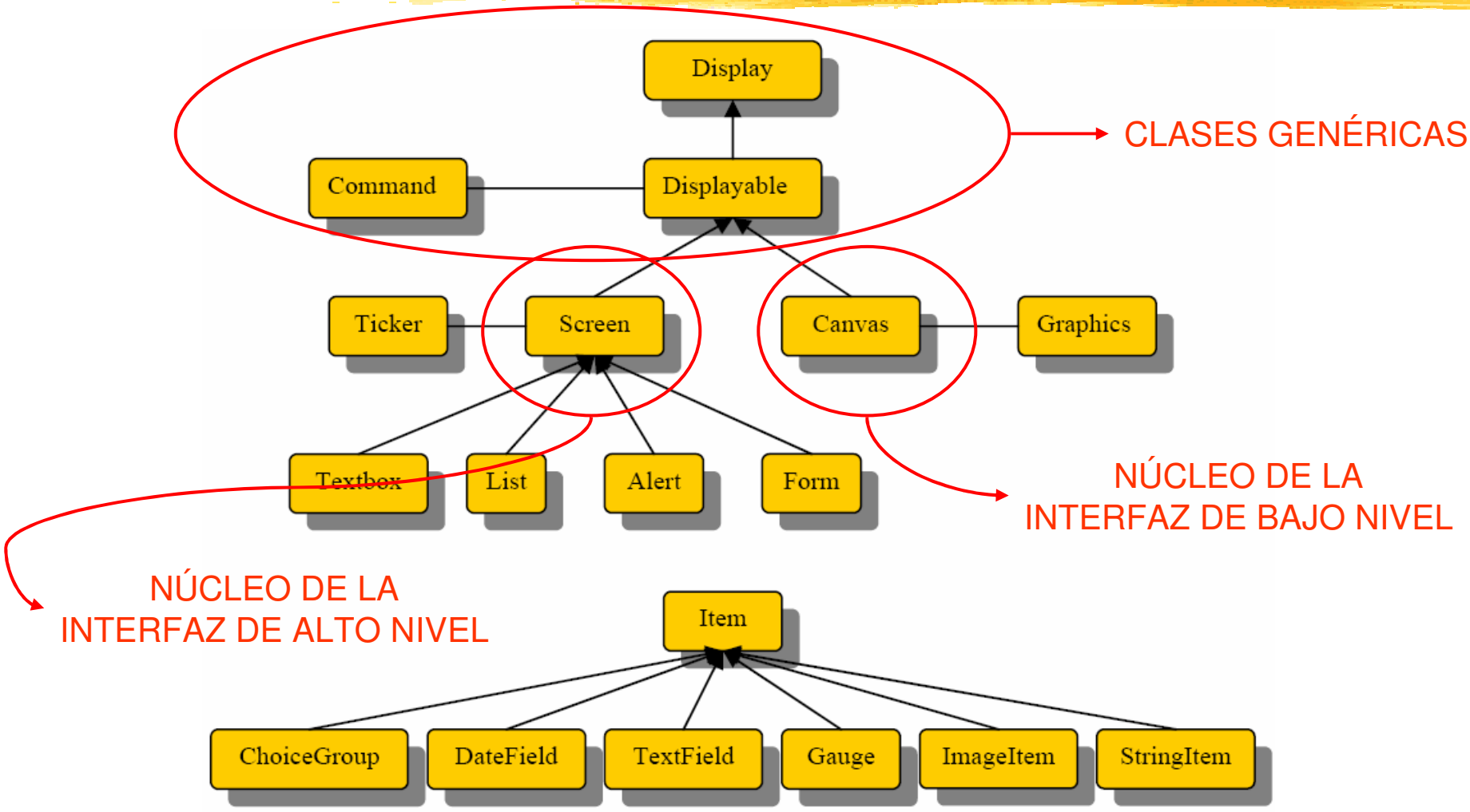
UNIVERSIDAD DE OVIEDO
Área de Arquitectura y Tecnología de Computadores

Curso 2004/2005

Interfaz gráfica de usuario

- Una vez visto el ciclo de vida de un MIDlet (javax.microedition.midlet)
- Interfaz gráfica de usuario (javax.microedition.lcdui)
- LCDUI = Liquid Cristal Display User Interface
- Incorpora:
  - **Clases de interfaz de usuario de alto nivel**
    - Botones, cajas de texto, formularios, …
    - Implementados por los distintos dispositivos
    - Las APIs de alto nivel permiten la portabilidad
    - Aspecto distinto según el dispositivo
    - Útiles para aplicaciones de negocio
  - **Clases de interfaz de usuario de bajo nivel**
    - Control total de la pantalla
    - Control de eventos de bajo nivel (rastreo de pulsaciones, p.e.)
    - Útiles para creación de juegos

Interfaz gráfica de usuario



Clases derivadas de Display e Item

Clases genéricas

- Clase Display (public class Display)
  - Maneja pantalla y dispositivos de entrada
  - Todo MIDlet poseerá, al menos, un objeto Display
  - Puede incluir tantos objetos Displayable como se desee

Métodos	Descripción
void callSerially(Runnable r)	Retrasa la ejecución del método run() del objeto *r* para no interferir con los eventos de usuario.
boolean flashBacklight(int duracion)	Provoca un efecto de *flash* en la pantalla.
int getBestImageHeight(int imagen)	Devuelve el mejor alto de imagen para un tipo dado.
int getBestImageWidth(int imagen)	Devuelve el mejor ancho de imagen para un tipo dado.
int getBorderStyle(bolean luminosidad)	Devuelve el estilo de borde actual.
int getColor(int color)	Devuelve un color basado en el parámetro pasado.
Displayable getCurrent()	Devuelve la pantalla actual.
static Display getDisplay(MIDlet m)	Devuelve una referencia a la pantalla del MIDlet *m*.
boolean isColor()	Devuelve *true* o *false* si la pantalla es de color o b/n.
int numAlphaLevels()	Devuelve el número de niveles *alpha* soportados.
int numColors()	Devuelve el número de colores aceptados por el MID.
void setCurrent(Alert a, Displayable d)	Establece la pantalla *d* despues de la alerta *a*
void setCurrent(Displayable d)	Establece la pantalla actual
void setCurrent(Item item)	Establece la pantalla en la zona dónde se encuentre el *item*
boolean vibrate(int duracion)	Realiza la operación de vibración del dispositivo.

Clases genéricas

- Clase Displayable (public **abstract** class Display)
  - Clase abstracta: no puede ser instanciada
  - Heredan de ella Screen y Canvas
  - Incluye métodos encargados de manejar eventos de pantalla y de añadir y eliminar comandos

Métodos	Descripción
void addComand(Command cmd)	Añade el Command *cmd*.
int getHeight()	Devuelve el alto de la pantalla.
Ticker getTicker()	Devuelve el T*icker* (cadena de texto que se desplaza) asignado a la pantalla.
String getTitle()	Devuelve el título de la pantalla.
int getWidth()	Devuelve el ancho de la pantalla.
bolean isShown()	Devuelve *true* si la pantalla está activa.
void removeCommand(Command cmd)	Elimina el Command *cmd*.
void setCommandListener(CommandListener l)	Establece un *listener* para la captura de eventos.
void setTicker(Ticker ticker)	Establece un *Ticker* a la pantalla.
void setTitle(String s)	Establece un título a la pantalla.
protected void sizeChanged(int w, int h)	El AMS llama a este método cuándo el área disponible para el objeto Displayable es modificada.

Clases genéricas

- Constructor de la aplicación
  - Obtención de objeto Display
    - Display pantalla = Display.getDisplay(this)
    - this es el objeto MIDlet (nuestra aplicación)
  - Creación de los objetos gráficos que aparecerán en pantalla
  - Importante
    - **Puede haber varias pantallas con múltiples objetos Displayable cada una**
- Método startApp()
  - Hacer referencia a la pantalla activa mediante el método setCurrent()
  - Importante
    - startApp() se ejecuta al inicio Y al volver de estado PAUSA
      - Los objetos se crean en el constructor, no aquí

Clases genéricas

- Clase Command (public class Command)
  - Mantiene información sobre un evento
  - Similar a un botón en Windows
  - Detecta un evento y ejecuta una acción
  - Al crear un objeto Command, se deben definir tres parámetros
    - **ETIQUETA**: Cadena de texto mostrada en pantalla con la que el usuario identificará al Command
    - **TIPO DE OBJETO**: Indica una apariencia y aspecto específicos (ver tabla)
    - **PRIORIDAD**: Útil para que el AMS establezca un orden de aparición de los distintos Command en la pantalla

Tipo	Descripción
BACK	Petición para volver a la pantalla anterior
CANCEL	Petición para cancelar la acción en curso
EXIT	Petición para salir de la aplicación
HELP	Petición para mostrar información de ayuda
ITEM	Petición para introducir el comando en un "item" en la pantalla
OK	Aceptación de una acción por parte del usuario
SCREEN	Para Commands de propósito más general
STOP	Petición para parar una operación

Clases genéricas

- Clase Command (public class Command)
  - Ejemplo de creación: `Command botonAceptar = new Command("Aceptar", Command.BACK, 1);`
  - Métodos de la clase:

Métodos	Descripción
public int getCommandType()	Devuelve el tipo del Command.
public String getLabel()	Devuelva la etiqueta del Command.
public String getLongLabel()	Devuelve la etiqueta larga del Command.
public int getPriority()	Devuelve la prioridad del Command.

  - Tras crear el objeto Command, hay que definir la acción que puede hacer, implementando la interfaz CommandListener
  - INTERFAZ = Clase sólo con métodos declarados todos como abstract
    - También puede definir constantes
  - Es misión del programador programar los métodos
  - La interfaz **CommandListener** sólo incluye un método:
    - commandAction(Command c, Displayable d)
    - Implementa el código correspondiente a las acciones a ejecutar cuando el Command c contenido en el objeto Displayable d reciba un evento

Interfaz de usuario de alto nivel

- Clase Screen
  - Superclase de la que derivan todas las clases de la IU de alto nivel
  - public **abstract** class Screen **extends** Displayable
  - En MIDP 1.0 contiene cuatro métodos que permiten obtener y definir título y ticker
    - Ticker: cadena de texto que se desplaza por la pantalla de derecha a izquierda
    - setTitle(string s), getTitle(), setTicker(Ticker ticker), getTicker()
  - En MIDP 2.0, estos cuatro métodos han sido incluidos en la clase Displayable

Interfaz de usuario de alto nivel

- Clase Alert
  - public class Alert **extends** Screen
  - Pantalla de aviso al usuario
    - Típicamente, diálogo de error
    - Suele estar formado por un título, texto y, a veces, imágenes
    - Dos constructores
      - Alert(String titulo)
      - Alert(String tit, String txt, Image img, AlertType tipo)
    - Se puede definir tiempo de vida del aviso (2 tipos)
      - **Modal**: Hasta que cancela el usuario
        - Alert.setTimeOut(Alert.FOREVER)
      - **No Modal**: Durante un tiempo definido
        - Alert.setTimeOut(tiempo)

Interfaz de usuario de alto nivel

- Clase Alert
  - Se puede elegir el tipo de alerta a mostrar

Tipo	Descripción
ALARM	Aviso de una petición previa
CONFIRMATION	Indica la aceptación de una acción
ERROR	Indica que ha ocurrido un error
INFO	Indica algún tipo de información
WARNING	Indica que puede ocurrir algún probl

  - Cada tipo tiene asociado un sonido
  - Se puede reproducir el sonido sin tener que crear un objeto Alert
    - AlertType.CONFIRMATION.playSound(display)

Interfaz de usuario de alto nivel

- Clase List
  - public class List **extends** Screen **implements** Choice
  - Pantalla que incluye una lista de opciones
  - Interfaz Choice
    - Representa la posibilidad de modelar la elección entre diversas opciones
  - Choice permite 3 formas de elegir opciones → 3 tipos de List

Tipo	Descripción
EXCLUSIVE	Lista en la que un sólo elemento puede ser seleccionado a la vez.
IMPLICIT	Lista en la que la selección de un elemento provoca un evento.
MULTIPLE	Lista en la que cualquier número de elementos pueden ser seleccionados al mismo tiempo.

  - 2 constructores:
    - Lista vacía
      - List(String titulo, int listType)
    - Lista con un conjunto inicial de opciones (y de imágenes asociadas, si se desea
      - List(String titulo, int listType, String[] eltos, Image[] imagenes)
      - Importante: eltos.length = imagenes.length

Interfaz de usuario de alto nivel

- Tipos de listas
  - Implícitas
    - Al seleccionar un elemento, se provoca una acción
    - Útil para el clásico menú de opciones
    - La acción se implementa con commandAction(Command c, Displayable d)
  - Exclusivas
    - Sólo se puede seleccionar un elemento a la vez (RadioButton)
    - La selección de opción no implica una acción
    - Hay que usar un Command para salvar la opción seleccionada
  - Múltiples
    - Se pueden seleccionar todos los elementos deseados
    - Análoga a exclusiva (no implica acción; uso de Command para guardar el contexto)

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

public class ListaImplicita extends MIDlet implements CommandListener{
        Command atras, salir; //Declaracion de Variables
        Display pantalla;
        List menu;
        Form formul, formu2, formu3;

        public ListaImplicita(){
                pantalla = Display.getDisplay(this); //Creacion de pantallas
                menu = new List("Menú",List.IMPLICIT);
                menu.insert(0,"Opcion3",null);
                menu.insert(0,"Opcion2",null);
                menu.insert(0,"Opcionl",null);
                atras = new Command("Atras",Command.BACK,1);
                salir = new Command("Salir",Command.EXIT,1);
                menu.addCommand(salir);
                formul = new Form("Formulario 1");
                formu2 = new Form("Formulario 2");
                formu3 = new Form("Formulario 3");
                formul.addCommand(atras);
                formu2.addCommand(atras);
                formu3.addCommand(atras);
                menu.setCommandListener(this);
        }

        public void startApp() {
                pantalla.setCurrent(menu); // Pongo el menu en pantalla
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void commandAction(Command c, Displayable d){
                if (c == menu.SELECT_COMMAND){ // Si selecciono
                        switch(menu.getSelectedIndex()){ //opcion del menu
                                case 0:{ pantalla.setCurrent(formul);break;}
                                case 1:{ pantalla.setCurrent(formu2);break;}
                                case 2:{ pantalla.setCurrent(formu3);break;}
                        }
                 }
                else if (c == atras){ //Selecciono comando "Atrás"
                        pantalla.setCurrent(menu);
                      }
                else if (c == salir){ // Selecciono salir de la aplicacion
                        destroyApp(false);
                        notifyDestroyed();
                }
        }
}
```

EJEMPLO
LISTA IMPLÍCITA

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

public class ListaExclusiva extends MIDlet implements CommandListener {
        Display pantalla;
        Command salir, salvar;
        List menu;

        public ListaExclusiva(){
                pantalla = Display.getDisplay(this);
                salir = new Command("Salir",Command.EXIT,1);
                salvar = new Command("Salvar",Command.ITEM,1);
                String opciones[] = {"Opcion1","Opcion2","Opcion3"};
                menu = new List("Lista exclusiva",List.EXCLUSIVE,opciones,null);
                menu.addCommand(salvar);
                menu.addCommand(salir);
                menu.setCommandListener(this);
        }

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

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void commandAction(Command c, Displayable d){
                if (c == salvar){
                        int opcionelegida = menu.getSelectedIndex();
                        //salvar opciones en memoria persistente.
                        System.out.println("Opcion elegida n°"+(opcionelegida+1));
                }
                else{
                        destroyApp(false);
                        notifyDestroyed();
                }
         }
}
```

EJEMPLO LISTA EXCLUSIVA

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

public class ListaMultiple extends MIDlet implements CommandListener {
        Display pantalla;
        List menu;
        Command salir, salvar;

        public ListaMultiple(){
                pantalla = Display.getDisplay(this);
                salir = new Command("Salir",Command.EXIT,1);
                salvar = new Command("Salvar",Command.ITEM,1);
                menu = new List("Lista Multiple",List.MULTIPLE);
                menu.insert(menu.size(),"Opcion1",null);
                menu.insert(menu.size(),"Opcion2",null);
                menu.insert(menu.size(),"Opcion3",null);
                menu.addCommand(salir);
                menu.addCommand(salvar);
                menu.setCommandListener(this);
        }

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

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void commandAction(Command c, Displayable d){
                if (c == salvar){
                   boolean seleccionados[] = new boolean[menu.size()];
                   menu.getSelectedFlags(seleccionados);
                   for(int i = 0;i<menu.size();i++){
                         System.out.println(menu.getString(i) +
                           (seleccionados[i] ? " seleccionada" : " no seleccionada"));
                   }
                }
                else{
                   destroyApp(false);
                   notifyDestroyed();
                }
        }
}
```

EJEMPLO
LISTA MÚLTIPLE

Interfaz de usuario de alto nivel

- Métodos de la clase List

Métodos	Descripción
int append(String texto,Image imagen)	Añade un elemento al final de la lista
void delete(int posición)	Elimina el elemento de la posición especificada
void deleteAll()	Elimina todas las entradas de la lista.
void insert(int pos, String texto, Image im)	Inserta un elemento en la posición especificada
int getFitPolicy()	Devuelve el modo en el que se muestran las entradas de la lista por pantalla
Font getFont(int pos)	Devuelve la fuente del elemento *pos*.
Image getImage(int pos)	Obtiene la imagen de una posición determinada.
int getSelectedFlags(bolean[] array)	Almacena el estado de selección en un array.
int getSelectedIndex()	Obtiene el índice del elemento seleccionado.
String getString(int pos)	Obtiene el texto del elemento indicado por *pos*.
boolean isSelected(int pos)	Determina si está seleccionado el elemento
void removeCommand(Command cmd)	Elimina el Command *cmd*.
void set(int pos, String texto, Image im)	Reemplaza el elemento de la posición *pos*
void setFitPolicy(int modo)	Establece el modo de posicionar las entradas de la lista por pantalla.
void setFont(int pos, Font fuente)	Establece la fuente de la entrada indicada en *pos*
void setSelectCommand(Command cmd)	Selecciona el Command a usar.
int setSelectedFlags(bolean[] array)	Reemplaza el estado de selección por el de *array*
int setSelectedIndex(int pos, bolean selec)	Reemplaza el estado de selección
int size()	Obtiene el número de elementos

Interfaz de usuario de alto nivel

- Clase TextBox
  - public class TextBox **extends** Screen
  - Pantalla que permite editar texto en ella
  - Hay que especificar su capacidad (nº máximo de caracteres)
  - Si se excede la capacidad del dispositivo → scroll
  - La capacidad devuelta por el constructor puede ser distinta a la solicitada
    - getMaxSize() devuelve la capacidad para un TextBox ya creado
  - Restricciones del texto: gestionadas por la clase TextField

Valor	Descripción
ANY	Permite la inserción de cualquier carácter.
CONSTRAINT_MASK	Se usa cuándo necesitamos determinar el valor actual de las restricciones
EMAILADDR	Permite caracteres válidos para direcciones de correo electrónico.
NUMERIC	Permite sólo números.
PASSWORD	Oculta los caracteres introducidos mediante una máscara para proporcionar privacidad.
PHONENUMBER	Permite caracteres válidos sólo para números de teléfono
URL	Permite caracteres válidos sólo para direcciones URL.

Interfaz de usuario de alto nivel

- Clase TextBox
  - Constructor
    - TextBox(String titulo, String texto, int tamaño, int restricciones)
  - Ejemplo de llamada
    - Crea un TextBox que sólo admite números y además oculta el texto introducido

```
TextBox cajatexto = new TextBox(“Contraseña”,””,30,
TextField.NUMERIC | TextField.PASSWORD)
```

Interfaz de usuario de alto nivel

- Clase TextBox
  - Métodos

Métodos	Descripción
void delete(int desplazamiento, int longitud)	Borra caracteres del TextBox.
int getCaretPosition()	Devuelve la posición del cursor en pantalla.
int getChars(char[] datos)	Copia el contenido del TextBox en *datos*.
int getConstraints()	Devuelve las restricciones de entrada.
int getMaxSize()	Devuelve el tamaño máximo del TextBox.
String getString()	Devuelve el contenido del TextBox.
void insert(char[] datos, int des, int long, int pos)	Inserta un subrango de caracteres de *datos* en el TextBox.
void insert(char[] datos, int pos)	Inserta la cadena de caracteres *datos* en una posición determinada.
void setChars(char[] datos, int des, int long)	Reemplaza el contenido del TextBox por un subconjunto de caracteres de *datos*.
void setConstraints(int restricciones)	Establece las restricciones de entrada.
void setInitialInputMode(String caracteres)	Establece un tipo de entrada inicial.
int setMaxSize(int capacidad)	Establece el tamaño máximo del TextBox.
void setString(String texto)	Establece el contenido del TextBox.
int size()	Devuelve el número de caracteres.

Interfaz de usuario de alto nivel

- Clase Form
  - public class Form **extends** Screen
  - Contenedor de un número indeterminado de objetos
  - Esos objetos tienen que derivar de la clase Item
  - Si se insertan demasiados objetos → scroll
  - Referencia a cada Item: mediante índices (desde 0 hasta Form.size()-1)
  - Un mismo Item no puede estar en más de un Form a la vez

Interfaz de usuario de alto nivel

- Clase Form
  - Métodos

Métodos	Descripción
Int append(Image imagen)	Añade una imagen al formulario.
int append(Item item)	Añade un Item al formulario.
int append(String texto)	Añade un String al formulario.
void delete(int num)	Elimina el Item que ocupa lo posición *num*.
void deleteAll()	Elimina todos los Items del formulario.
Item get(int num)	Devuelve el Item que se encuentra en la posición *num*.
int getHeight()	Devuelve la altura en pixels del área disponible (sin realizar *scroll*) para los Items.
int getWidth()	Devuelve la anchura en pixels del área disponible (sin realizar *scroll*) para los Items.
void insert(int num, Item item)	Inserta un Item justo antes del que ocupa la posición *num*.
void set(int num, Item item)	Reemplaza el Item que ocupa la posición *num*.
void setItemStateListener(ItemStateListener listener)	Establece un 'listener' que captura los eventos que produzcan cualquier Item del formulario.
int size()	Devuelve el número de Items del formulario.

Interfaz de usuario de alto nivel

- Clase Form
  - Gestión de eventos
    - Similar a Command
    - El MIDlet debe implementar ItemStateListener
      - Contiene un solo método abstracto: itemStateChanged(Item item)
      - Análogo a implementar CommandListener e incluir en el MIDlet el código de CommandListener(Command c, Displayable d)

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

public class ManejoItems extends MIDlet implements ItemStateListener, CommandListener{
        Display pantalla;
        Form formulario;
        TextField txt;
        Command salir;

        public ManejoItems(){
                pantalla = Display.getDisplay(this);
                formulario = new Form("");
                txt = new TextField("Introduce datos","",70,TextField.ANY);
                salir = new Command("Salir",Command.EXIT,1);
                formulario.append(txt);
                formulario.addCommand(salir);
                formulario.setItemStateListener(this);
                formulario.setCommandListener(this);
        }

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

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void commandAction(Command c, Displayable d){
                if (c == salir){
                        destroyApp(false);
                        notifyDestroyed();
                }
        }

        public void itemStateChanged(Item i){
                if (i == txt){
                        System.out.println("Evento detectado en el TextBox");
                }
         }
}
```

EJEMPLO
GESTIÓN EVENTOS
EN UN FORM

Interfaz de usuario de alto nivel

- Clase StringItem
  - public class StringItem **extends** Item
  - Clase más simple que deriva de Item
  - Texto no modificable
  - 2 constructores
    - StringItem(String etiqueta, String texto)
    - StringItem(String etiqueta, String texto, int apariencia)
  - Apariencia: Item.PLAIN, Item.HYPERLINK, Item.BUTTON
    - StringItem(etiqueta, texto, Item.PLAIN) = 1er constructor

Interfaz de usuario de alto nivel

- Clase StringItem
  - Métodos

Métodos	Descripción
int getAppearanceMode()	Nos devuelve la apariencia del texto
Font getFont()	Nos devuelve la Fuente del texto
String getText()	Nos devuelve el texto del StringItem
void setFont(Font fuente)	Establece la Fuente del texto.
void setText(String texto)	Establece el texto del StringItem

Interfaz de usuario de alto nivel

- Clase ImageItem
  - public class ImageItem **extends** Item
  - Inclusión de imágenes en un Form
  - El usuario no puede interactuar con la imagen
  - 2 constructores
    - ImageItem(String etiqueta, Image imagen, int layout, String textoalt)
    - ImageItem(String etiqueta, Image imagen, int layout, String textoalt, int apariencia)
  - textoalt: cadena de texto alternativa a la imagen si ésta excede la capacidad de la pantalla
  - layout: posición de la imagen en la pantalla

Interfaz de usuario de alto nivel

- Clase ImageItem
  - Posibilidades de layout

Valor	Descripción
LAYOUT_LEFT	Imagen posicionada a la izquierda
LAYOUT_RIGHT	Imagen posicionada a la derecha
LAYOUT_CENTER	Imagen centrada
LAYOUT_DEFAULT	Posición por defecto
LAYOUT_NEWLINE_AFTER	Imagen posicionada tras un salto de línea
LAYOUT_NEWLINE_BEFORE	Imagen posicionada antes de un salto de línea

  - Métodos

Métodos	Descripción
String getAltText()	Nos devuelve la cadena de texto alternativa.
Int getAppearanceMode()	Nos devuelve la apariencia
Image getImage()	Nos devuelve la imagen
Int getLayout()	Nos devuelve el posicionado de la imagen
void setAltText(String textoalt)	Establece un texto alternativo
void setImage(Image imagen)	Establece una nueva imagen
void setLayout(int layout)	Establece un nuevo posicionado en pantalla

Interfaz de usuario de alto nivel

- Clase TextField
  - public class TextField **extends** Item
  - Campo de texto insertable en un Form y editable
  - Diferencias con TextBox
    - Ha de ser insertado en un formulario (TextBox es autónomo)
    - TextField deriva de Item; TextBox, de Screen
    - Eventos controlados por itemStateChanged(Item item); los del TextBox, a través de commandAction(Command c, Displayable d)
  - Similitudes
    - Restricciones en las entradas (ANY, NUMERIC, PASSWORD, ...)
  - Constructor
    - TextField(String etiqueta, String texto, int capacidad, int restricciones)

Interfaz de usuario de alto nivel

- Clase TextField
  - Métodos

Métodos	Descripción
void delete(int desplazamiento, int longitud)	Borra caracteres del TextField.
int getCaretPosition()	Devuelve la posición del cursor en pantalla.
int getChars(char[] datos)	Copia el contenido del TextField en *datos*.
int getConstraints()	Devuelve las restricciones de entrada.
int getMaxSize()	Devuelve el tamaño máximo del TextField.
String getString()	Devuelve el contenido del TextField.
void insert(char[] datos, int des, int long, int pos)	Inserta un subrango de caracteres de *datos* en el TextField.
void insert(char[] datos, int pos)	Inserta la cadena de caracteres *datos* en una posición determinada.
void setChars(char[] datos, int des, int long)	Reemplaza el contenido del TextField por un subconjunto de caracteres de *datos*.
void setConstraints(int restricciones)	Establece las restricciones de entrada.
void setInitialInputMode(String caracteres)	Establece un tipo de entrada inicial.
int setMaxSize(int capacidad)	Establece el tamaño máximo del TextField.
void setString(String texto)	Establece el contenido del TextField.
int size()	Devuelve el número de caracteres.

Interfaz de usuario de alto nivel

- Clase DateField
  - public class DateField **extends** Item
  - Gestión de fechas y horas en un Form
  - Hace uso de la clase java.util.Date
  - Constructor
    - DateField(String etiqueta, int modo)
      - Modo: DATE, TIME, DATE_TIME
    - DateField(String etiqueta, int modo, java.util.TimeZone zonahoraria)

CREACIÓN DE UN DATEFIELD CON LA FECHA Y HORA ACTUALES

```
Date fechaactual = new Date();     // Crear objeto Date con fecha actual
DateField fecha = new DateField("Fecha",DateField.DATE_TIME);
fecha.setDate(fechaactual);        // Establecer en el objeto la fecha actual
```

Interfaz de usuario de alto nivel

- Clase ChoiceGroup
  - public class ChoiceGroup **extends** Item **implements** Choice
  - Grupo de elementos seleccionables
  - Análogo a List, pero dentro de un Form
  - Constructor
    - ChoiceGroup(String etiqueta, int tipo)
    - ChoiceGroup(String etiq, int tipo, String[] elementos, Image[] imagenes)

Interfaz de usuario de alto nivel

- Clase ChoiceGroup
  - Métodos

Métodos	Descripción
int append(String texto,Image imagen)	Añade un elemento al final de la lista
void delete(int posición)	Elimina el elemento de la posición especificada
void deleteAll()	Elimina todas las entradas de la lista.
void insert(int pos, String texto, Image im)	Inserta un elemento en la posición especificada
int getFitPolicy()	Devuelve el modo en el que se muestran las entradas de la lista por pantala
Font getFont(int pos)	Devuelve la fuente del elemento *pos*.
Image getImage(int pos)	Obtiene la imagen de una posición determinada.
int getSelectedFlags(bolean[] array)	Almacena el estado de selección en un array.
int getSeleetedIndex()	Obtiene el índice del elemento seleccionado.
String getString(int pos)	Obtiene el texto del elemento indicado por *pos*.
boolean isSelected(int pos)	Determina si está seleccionado el elemento
void set(int pos, String texto, Image im)	Reemplaza el elemento de la posición *pos*
void setFitPolicy(int modo)	Establece el modo de posicionar las entradas de la lista por pantalla.
void setFont(int pos, Font fuente)	Establece la fuente de la entrada indicada en *pos*
int setSelectedFlags(bolean[] array)	Reemplaza el estado de selección por el de *array*
int setSelectedIndex(int pos, bolean selec)	Reemplaza el estado de selección
int size()	Obtiene el número de elementos

Interfaz de usuario de alto nivel

- Clase ChoiceGroup
  - Gestión de eventos:
  1. A través de itemStateChanged()
     - Cuando el usuario selecciona una opción, el formulario registra un ItemStateListener y se realiza una llamada a ItemStateChanged()
     - En ItemStateChanged(), se puede comprobar qué elemento ha sido seleccionado y cuál no, realizando las acciones pertinentes
     - Cada vez que se cambia una selección, se llama a ItemStateChanged()
  2. A través de commandAction(), como con los objetos List

Interfaz de usuario de alto nivel

- Clase Gauge
  - public class Gauge extends Item
  - Indicador de progreso mediante un gráfico de barras
  - Representa valores entre 0 y un valor máximo
  - Constructor
    - Gauge(String etiqueta, boolean interactivo, int valormax, int valorinicial)
  - Dos tipos
    - **Interactivo**: el usuario puede modificar el valor actual del Gauge (mediante botones y siempre dentro de [0-valormax])
    - **No interactivo**: la aplicación modifica el valor actual mediante Gauge.setValor(int valor)
      - Muy util para indicar el progreso de una tarea

Interfaz de usuario de alto nivel

- Recomendaciones
  - Hasta ahora:
    - Control de la interacción de usuario mediante los métodos de las interfaces CommandListener e ItemStateListener
  - Ejemplo: MIDlet con 15 pantallas
    - Constructor, CommandAction, itemStatusChanged grandes y difíciles de depurar
  - Modularizar la aplicación
    - Un archivo .java por pantalla que incluye todas las clases participantes en la misma

Informática Móvil

Diseño de aplicaciones con J2ME

Interfaz de Usuario: API de Alto Nivel

José Ramón Arias García
Ignacio Marín Prendes

UNIVERSIDAD DE OVIEDO
Área de Arquitectura y Tecnología de Computadores

Curso 2004/2005