PROGRAMACIÓN EN JAVA

TEMA 5: Menús y barras de herramientas

Bibliografía

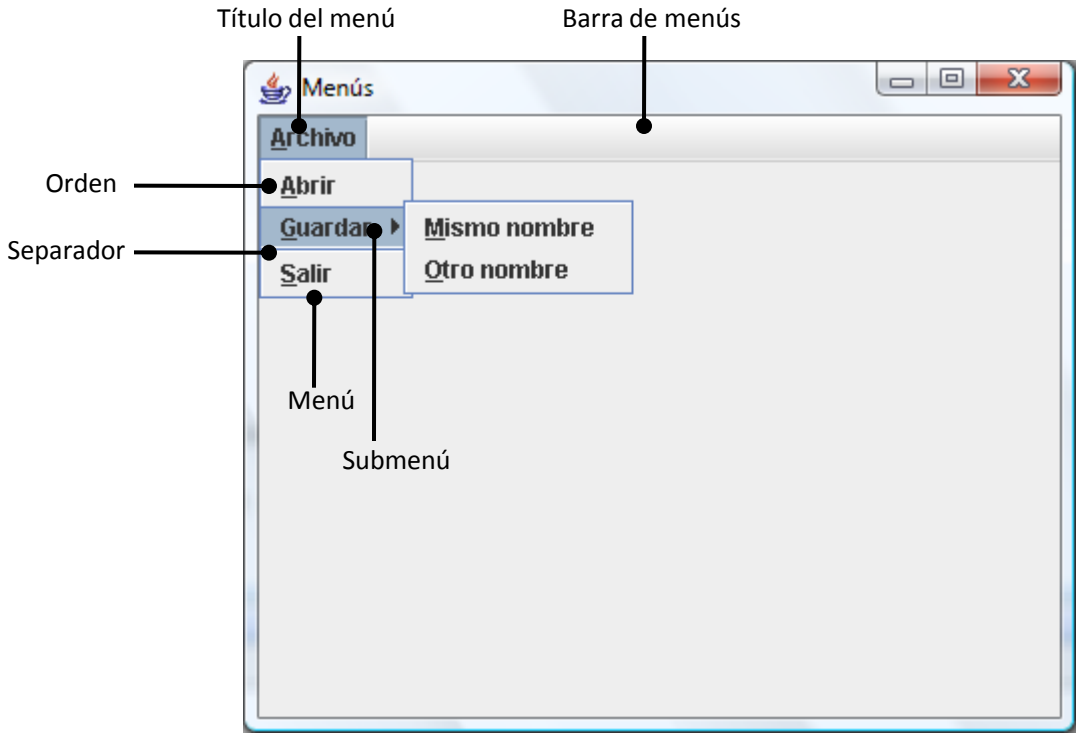
- "JAVA 2 Interfaces gráficas y aplicaciones para Internet – 2ª Edición"
  Fco. Javier Ceballos. Ed. Ra-Ma, 2006
  Capítulo: 3

Contenido

- Menús
- Diseño de una barra de menús
- Responder a eventos
- Aceleradores
- Barra de herramientas
- Utilizar imágenes en los botones de la barra
- Diseño de una barra de herramientas

Menús

- Proveen al usuario de un conjunto de órdenes, lógicamente relacionadas, agrupadas bajo un mismo título
- El conjunto de todos los títulos correspondientes a los menús diseñados aparecerán en la barra de menús
- Los elementos de un menú pueden ser: Órdenes, submenús y separadores



Menús (cont.)

- Los separadores son útiles para separar, en función de su actividad, grupos de órdenes de un mismo menú
- Cuando hacemos clic sobre:
  - Órdenes: *. Se ejecuta una acción o se despliega una caja de diálogo
    *. Una orden seguida de 3 puntos (…) indica que se desplegará una caja de diálogo
  - Submenús: *. Despliega una nueva lista de elementos

Diseño de una barra de menús

- Disponemos de las siguientes clases: *. **JMenuBar** (Barras de menús)
  *. **JMenu** (Menús de la barra)
  *. **JMenuItem** (Elementos de los menús)
  *. **JSeparator** (Separadores)

- Los pasos a ejecutar para añadir una barra de menús son los siguientes:

  1. Crear una barra de menús y añadirla al formulario

```
//...
private void initComponents()
{
        //...
        mbarBarraDeMenus = new JMenuBar();  //Creamos la barra de menús
        //...
        setJMenuBar(mbarBarraDeMenus);       //La añadimos al formulario
        //...
}
//...
private JMenuBar barBarraDeMenus = null;
//...
```

Diseño de una barra de menús (cont.)

2. Añadir los menús a la barra de menús
    2.1. Construimos un objeto de la clase **JMenu**
    2.2. Establecemos sus propiedades
    2.3. Lo añadimos a la barra de menús

```
//…
private void initComponents()
{
        //…
        mnuArchivo = new JMenu();
        mnuArchivo.setMnemonic('A');
        mnuArchivo.setText("Archivo");
        mbarBarraDeMenus.add(mnuArchivo);
        //…
}
//…
private JMenu mnuArchivo = null;
//…
```

Diseño de una barra de menús (cont.)

3. Añadir al menú los elementos que lo componen
    3.1. Construimos un objeto de la clase **JMenuItem**
    3.2. Establecemos sus propiedades
    3.3. Lo añadimos al menú

```
//…
private void initComponents()
{
        //…
        mnuItAbrir = new JMenuItem();
        mnuItAbrir.setMnemonic('A');
         mnuItAbrir.setText("Abrir…");
        mnuArchivo.add(mnuItAbrir);
        //…
}
//…
private JMenuItem mnuItAbrir = null;
//…
```

Diseño de una barra de menús (cont.)

4. Crear un submenú (opcional): No es más que un objeto **JMenu** añadido como elemento a otro objeto **JMenu**

```
//…
private void initComponents()
{
        //…
        mnuGuardar = new JMenu();
        mnuGuardar.setMnemonic('G');
        mnuGuardar.setText("Guardar");
        mnuArchivo.add(mnuGuardar);
        //…
}
//…
private JMenu mnuGuardar = null;
//…
```

Diseño de una barra de menús (cont.)

5. Añadir un separador (opcional): Un separador es un objeto de la clase **JSeparator**

```
//…
private void initComponents()
{
        //…
        sepSeparadorUno = new JSeparator();
        mnuArchivo.add(sepSeparadorUno);
        //…
}
//…
private void JSeparator sepSeparadorUno = null;
//…
```

NOTA: Los menús serán añadidos a la barra de menús en el orden en el que se ejecuten los métodos **add** (Idéntico para los elementos de un menú)

Responder a eventos

- Para responder a los eventos producidos al hacer clic en una orden de un menú, debemos asignar a dicha orden un manejador de eventos (ME) de acción
- Interfaz: ActionListener, Evento: ActionEvent, Método: actionPerformed

```
//…
private void initComponents()
{
        //…
        mnuItSalir.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        mnuItSalirActionPerformed(e);
                }
        });
        //…
}
//…
public void mnuItSalirActionPerformed(ActionEvent e)
{
        System.exit(0);
}
//…
```

Responder a eventos (cont.)

- Otra alternativa podría ser crear un único ME de acción y asociarlo con cada uno de las órdenes de nuestro menú
- Después, este ME solicitaría una u otra respuesta en función de la orden que haya generado el evento

```
//...
private void initComponents()
{
		//...
		//Crear un ME de eventos de acción
		ActionListener al = new ActionListener()
		{
				public void actionPerformed(ActionEvent e)
				{
						//Obtener el ID del elemento en el que se hiso clic
						Object item = evt.getSource();
						//Invocar al controlador adecuado
						if(item == mnuItAbrir)
								mnuItAbrirActionPerformed(e);
						if(item == mnuItSalir)
								mnuItSalirActionPerformed(e);
				}
		};
		//Asignar el ME "al" a cada uno de los elementos del menú
		mnuItAbrir.addActionListener(al);
		mnuItSalir.addActionListener(al);
		//...
}
//...
private void mnuItAbrirActionPerformed(ActionEvent e)
{
		//Código para la orden abrir
}
private void mnuItSalirActionPerformed(ActionEvent e)
{
		//Código para la orden salir
}
//...
```

Aceleradores

- Son muy similares a los nemónicos, excepto que se puede especificar cualquier combinación de teclas de la forma:

  [{ Ctrl | Shift | Alt }] + {cualquier tecla}

- Los aceleradores actúan independientemente de que el componente esté visible o no
- Ejemplo:
  - Si asociamos el acelerador *Ctrl + A* con la orden *Abrir* del menú *Archivo*, siempre que se pulse esa combinación de teclas se ejecutará la orden *Abrir*, independientemente de que el menú *Archivo* esté o no desplegado
- Los aceleradores sólo pueden estar asociados con elementos de un menú (**JMenuItem**), no con menús (**JMenu**)
- Ejemplo:
  - Código que asocia el acelerador *Ctrl + A* con la orden *Abrir* del menú *Archivo*

```
//...
mnuItArchivo.setAccelerator(keyStore.getKeyStore(keyEvent.VK_A,InputEvent.CTRL_MASK));
//...
```

Barra de herramientas

- Es una ventana que normalmente contiene botones gráficos (puede contener: Cajas de texto, Listas desplegables, etc.)
- Normalmente se añadirá un botón a una barra de herramientas para evitar que al usuario la molestia de abrir un menú para ejecutar una determinada orden
- En *Swing* una barra de herramientas es un objeto de la clase **JToolBar**

Utilizar imágenes en los botones de la barra

- Para asignar una imagen a un botón hay que enviarle el mensaje **setIcon**, la respuesta a este mensaje será la ejecución del método **setIcon**
- Éste recibe un parámetro que se corresponde con una referencia de tipo **Icon**
- Puesto que **Icon** es una interfaz, cada vez que se invoque a **setIcon** se puede pasar como argumento cualquier objeto de alguna clase que implemente esa interfaz (**ImageIcon**)
- La clase **ImageIcon** tiene varios constructores; uno de ellos es:

  public ImageIcon(URL localización);

- Este constructor recibe como argumento el **URL** correspondiente al lugar dónde se localiza el recurso que se desea obtener y crea un objeto **ImageIcon** que empaqueta dicho recurso
- Para obtener este **URL**, utilizamos el método **getResource** de la clase **Class**
- Un objeto **Class** puede representar a cualquier clase o interfaz inmersa en una aplicación JAVA en ejecución (podemos usar sus métodos para obtener información relativa a la misma)
- Ejemplo:

```
//…
Class objeto = getClass();  //this.getClass()
java.net.URL url = objeto.getResource(“/Ejemplo/Abrir.gif”);
javax.swing.ImageIcon imgGif = new javax.swing.ImageIcon(url);
btnbarAbrir.setIcon(imgGif);
//…
```

Diseño de una barra de herramientas

1. Creamos una barra de herramientas, establecemos su administrador de diseño y la añadimos al formulario

```
//…
private void initComponents()
{
        //…
        tbarBarraDeHerramientas = new JToolBar();
        //…
         tbarBarraDeHerramientas.setLayout(null);
        getContentPane().add(tbarBarraDeHerramientas);
         tbarBarraDeHerramientas.setBounds(0,0,492,36);
        //…
}
//…
private JToolBar tbarBarraDeHerramientas = null;
```

Diseño de una barra de herramientas (cont.)

2. Añadimos los componentes necesarios a la barra de herramientas

```
//…
private void initComponents()
{
        //…
        btnbarAbrir = new JButton();
        //…
        btnbarAbrir.setIcon(new ImageIcon(getClass().getResource("/Ejemplo/Abrir.gif")));
        btnbarAbrir.setToolTipText("Abrir");
        btnbarAbrir.setFocusPainted(false);
        btnbarAbrir.setMargin(new java.awt.Insets(0,0,0,0));
        //…
         tbarBarraDeHerramientas.add(btnbarAbrir);
        btnbarAbrir.setBounds(6,6,25,25);
        //…
}
//…
private JButton btnbarAbrir = null;
//…
```

NOTA: El método setFocusPainted permite que el foco sea pintado, o no, cuando se selecciona el botón. Si el argumento es true, el foco se pintará y si es false, no

Diseño de una barra de herramientas (cont.)

3. Finalmente, asociamos con cada botón el mismo ME y el mismo controlador (método que responde al evento) que tienen asociados las órdenes del menú

```
//…
private void initComponents()
{
        //…
        btnbarAbrir.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        mnuItAbrirActionPerformed(e);
                }
        });
        //…
}
//…
```

Menús emergentes

- Para visualizar un menú emergente hay que crear un objeto **JPopupMenu** e invocar a su método **show**
- Por ejemplo, suponga que se desea crear un menú emergente con la orden *Salir*, que se visualice al hacer clic con el botón derecho del ratón sobre el formulario
- Para ello debemos realizar los siguientes pasos:

1. Añadimos a nuestra clase una variable de tipo **JPopupMenu**

```
//...
public class Ejemplo extends JFrame
{
        public Ejemplo()
        {
                initComponents();
        }
        //...
        private JPopupMenu pumMenuEmergente = null;
        //...
}
```

Menús emergentes (cont.)

2. Creamos un objeto **JPopupMenu** referenciado por la variable declarada en el apartado anterior

```
//…
public class Ejemplo extends JFrame
{
        private void initComponents ()
        {
                //…
                pumMenuEmergente = new JpopupMenu();
                //…
        }
        //…
        private JPopupMenu pumMenuEmergente = null;
        //…
}
```

Menús emergentes (cont.)

3. Añadimos los elementos que van a formar parte del menú emergente (Pueden ser: **JMenuItem**, **JSeparator**, etc.) y los asociamos al menú emergente

```
//…
public class Ejemplo extends JFrame
{
        private void initComponents ()
        {
                //…
                 //Creamos el objeto JMenuItem
                pumMnuItSalir = new JMenuItem();
                pumMnuItSalir.setText("Salir");
                //Añadimos el elemento JMenuItem al menú emergente
                pumMenuEmergente.add(pumMnuItSalir);
                //…
        }
        //…
        private JMenuItem pumMnuItSalir = null;
        //…
}
```

Menús emergentes (cont.)

4. Asociamos el ME de acción con el elemento **JMenuItem** añadido en el apartado anterior

```
//…
public class Ejemplo extends JFrame
{
        private void initComponents ()
        {
                //…
                 //Asignamos el ME con la orden Salir del menú emergente
                 pumMnuItSalir.addActionListener(new ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                System.exit(0);
                        }
                });
                //…
        }
        //…
}
```

Menús emergentes (cont.)

5. Asociamos un ME de tipo **MouseAdapter** con el formulario para poder detectar cuándo el usuario hace clic con el botón derecho del ratón

```
//…
public class Ejemplo extends JFrame
{
        private void initComponents ()
        {
                //…
                /*Asociamos un ME de tipo MouseAdapter con el formulario,
                   para detectar cuando el usuario hace clic con el botón
                   derecho*/
                addMouseListener(new MouseAdapter()
                {
                        public void mousePressed(MouseEvent e)
                        {
                                frameMousePressed(e);
                        }
                });
                //…
        }
        //…
}
```

Menús emergentes (cont.)

6. Editamos el método *frameMousePressed* para que cada vez que se pulse el botón del ratón, se muestre el menú emergente

```
//...
public class Ejemplo extends JFrame
{
        private void initComponents ()
        {
                //...
        }
        public void frameMousePressed(MouseEvent e)
        {
                //Si se pulsó el botón derecho, visualizar el menú emergente
                if(e.getButton()==MouseEvent.BUTTON3)
                        pumMenuEmergente.show(this, e.getX(), e.getY());

        }
        //...
}
```

- El método **getButton** devuelve un entero que identifica el botón pulsado del ratón (BUTTON1, BUTTON2, BUTTON3)
- El método show permite visualizar el menú emergente, este tiene la siguiente sintaxis:

  public void show(Component *origen*, int *x*, int *y*);

  - *origen*: Nombre del componente que define el espacio de coordenadas
  - *x* e *y*: Son las coordenadas referidas a la esquina superior izquierda y relativas al componente origen, que definen donde será visualizado el menú