Tema 2

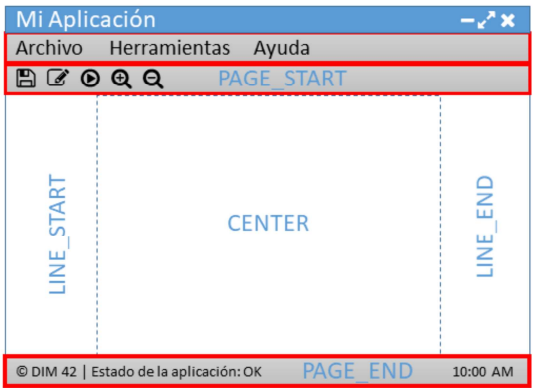
- Layouts: Orientaciones para su uso
- Menús y navegación
- Acciones
- Arboles

Práctica: Visor de ficheros de texto

Aplicar layouts (casos típicos)

Layout general:

- Menu superior: Establecer en el JFrame con setJMenuBar()
- Barra de herramientas: Región norte del panel de contenido (BorderLayout)
- Barra de estado inferior: Región sur del panel de contenido (BorderLayout)



- Una aplicación típica de Windows suele tener un menú superior, una barra de herramientas, una barra de estado y una zona central de trabajo.
  - La barra de menú se establece en el propio objeto JFrame.
  - La barra de herramientas se crea con un objeto JToolBar, y puede ubicarse en cualquiera de los lados del panel de contenido, aunque normalmente estará en la parte superior.
  - Para la barra de estado no existe ningún componente específico. Para ello debemos aplicar el BorderLayout al panel de contenido, y usar la región "PAGE_END" para insertar un JPanel de la altura deseada y con un borde o fondo que lo distinga. De esa forma usaremos la zona "CENTER" como zona de trabajo.

Aplicar layouts (casos típicos)

Menú o Panel lateral:

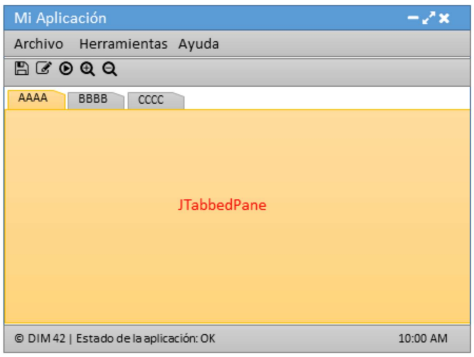
- Ancho fijo: Región "LINE_START" o "LINE_END" del panel de contenido (BorderLayout)
- Ancho variable: Panel con división vertical (JSplitPane)

- Si nuestra aplicación tiene un menu o panel lateral (izquierda o derecha), tenemos 2 opciones para implementarlo:
  - Usar la región "LINE_START" o "LINE_END" del BorderLayout. Esto proporciona una zona de ancho fijo.
  - Usar un panel con división vertical (JSplitPane). Esto permite al usuario redimensionar la zona del panel lateral a su gusto.

Aplicar layouts (casos típicos)

Zona de trabajo multidocumento:

- Pestañas



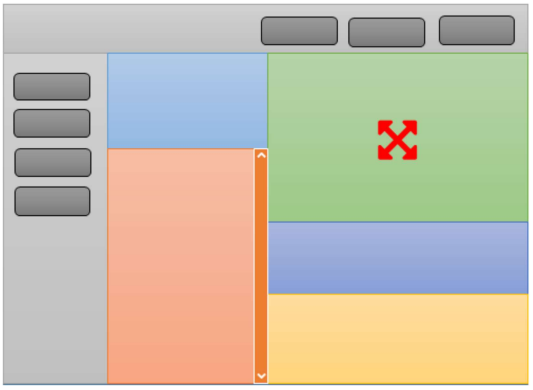
- Ventanas (modo MDI)

- En la zona de trabajo, debemos ubicar los componentes funcionales de nuestra aplicación, normalmente derivados de JPanel.
  - Si necesitamos trabajar con varios documentos o componentes en la zona de trabajo, y cambiar facilmente de uno a otro, podemos hacerlo de 2 formas:
    - Mediante pestañas para cada documento o componente, al estilo "Lotus Notes", con el componente JTabbedPane.
    - Mediante ventanas para cada documento o componente, al estilo MDI, con el componente JInternalFrame. Para ello el panel de contenido debe ser de tipo JDesktopPanel.

Aplicar layouts (casos típicos)

Paneles con contenido multiple

- Múltiples componentes internos → **GridBagLayout**
- Al menos un componente debe adaptarse al tamaño disponible
- Paneles de contenido variable → Capacidad de Scroll con **JScrollPane**
- Paneles auxiliares → **BoxLayout** o **FlowLayout**

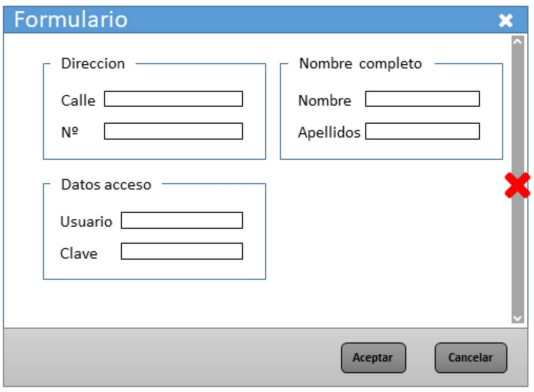


- Cada panel o frame que alberga un componente, debe a su vez establecer su propio layout.
    - Si necesitamos disponer multiples paneles y controles en diversas posiciones, el layout más adecuado es el GridBagLayout, que parte de un grid básico, y permite unir filas y columnas para crear areas diferenciadas.
    - Hay que tener siempre en cuenta que el contenido debe adaptarse al tamaño de la ventana. Al menos una de las áreas debe ser capaz de expandirse al tamaño disponible.
    - Los paneles de contenido variable, como visores de documentos, arboles, listas, tablas, deben tener capacidad de hacer scroll, con el componente JScrollPane.
    - Para paneles auxiliares suelen ser utiles los layout tipo BoxLayout (horizontal o vertical), y FlowLayout.

Aplicar layouts (casos típicos)

Formularios / Diálogos:

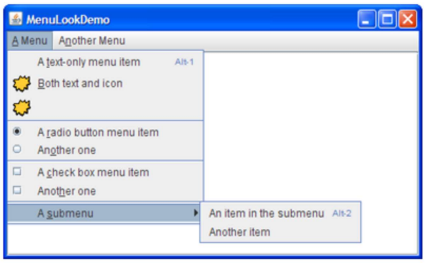
- Opciones en parte inferior (BorderLayout)
- Tamaño fijo. Si no cabe todo:
  - Dividir contenido en pestañas
  - No usar scroll
- Usar grupos de controles relacionados
- Combinar layouts si es necesario

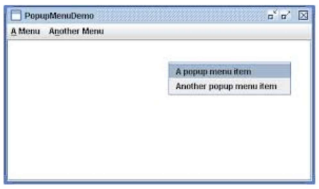


- Para diálogos y formularios:
  - Se deben ubicar las opciones del diálogo en una barra de botones, normalmente en la parte inferior.
  - El tamaño debe ser fijo. Si el contenido del formulario no cabe en el area disponible, es conveniente dividirlo en varios paneles y mostrarlo en pestañas superiores o laterales. En formularios no es conveniente que la zona de trabajo use el scroll.
    - Es conveniente usar grupos de controles para separar las zonas del formulario. Para ello meter los controles en un panel, y aplicar un borde de tipo "title border". Los grupos de controles se pueden distribuir mediante un BoxLayout o un FlowLayout.
    - Pueden combinarse varios layouts para producir efectos especiales. Por ejemplo para distribuir controles en varias columnas (un tablero Kanvan, por ejemplo), se puede usar un BoxLayout horizontal para crear las columnas, y dentro de cada una usar un BoxLayout vertical para distribuir los componentes de cada columna.

Menús

- Componentes:
  - Menú → Clase **JMenu**
  - Elemento de menú → Clase **JMenuItem**
- Se ubican en la barra de menú, o aparecen como menú contextual
- Elementos de menú:
  - Texto (con o sin imagen)
  - CheckBox
  - RadioButton
  - Otro menú anidado
- Teclas de acceso rápido





Los menús se ubican en la barra de menú, o bien se muestran como menú contextual.

Los elementos del menú pueden ser tipo texto, con o sin imagen, radio buttons, checkboxes, así como otros menus anidados.

Se pueden asociar teclas de acceso rápido a una opción de menú

Crear un menú

1. Crear el objeto JMenu
2. Crear un objeto JMenuItem por cada opción de menú (texto)
3. Añadir los objetos JMenuItem al objeto JMenu

Menus anidados:

1. Crear el menú anidado y sus ítems
2. Añadir el menú anidado al menú padre

Elementos de menú no textuales:

- CheckBox: Usar *JCheckboxMenuItem*
- RadioButton: Usar *JRadioButtonMenuItem*

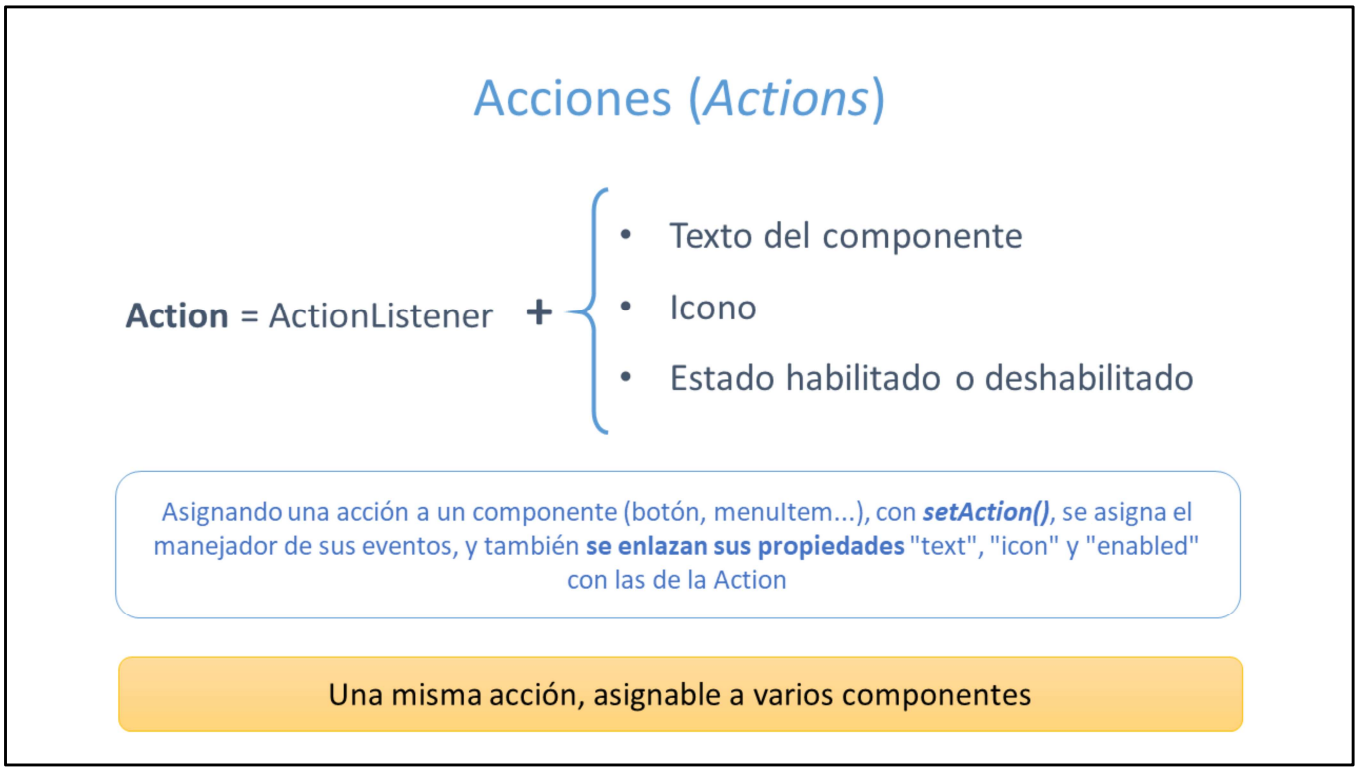
```
JMenu menu = new JMenu();
// Menu de texto
JMenuItem mNuevo = new JMenuItem("Nuevo
   fichero")

// Menu con texto e icono
JMenuItem mGuardar = new JMenuItem("Guardar",
   new ImageIcon("floppydisk.png"));

// Menu con checkbox
JCheckboxMenuItem mSoloLectura = new
   JCheckboxMenuItem("Fichero de solo
   lectura");

// Añadir elementos al menú
menu.add(mNuevo);
menu.add(mGuardar);
menu.add(mSoloLectura);
```

Respuesta a selección de elementos de menu

1. Asignar un **ActionCommand** a cada ítem del menú.
2. Crear un manejador de eventos de tipo **ActionListener**, y asignarlo a todos los ítems de menú.
3. En el método *actionPerformed()*, comprobar el *ActionCommand* de la fuente del evento.

➢ *Se puede asignar el mismo ActionCommand a varios ítems de menú o a botones.*

```
// Asignar actionCommand a cada item del menu
mNuevo.setActionCommand("nuevoFichero");
mGuardar.setActionCommand("guardarFichero");
mSoloLectura.setActionCommand("soloLectura");

// Crear manejador
ActionListener listener= new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e) {
          switch (e.getActionCommand()) {
             case "nuevoFichero":
                   // Tareas para crear fichero
                   break;
             case "guardarFichero":
                   // Tareas para guarder fichero
                   break;
             case "soloLectura":
                   // Marcar fichero solo lectura
                   break;
             ...
    }
};

// Asignar listener a los elementos de menu
mNuevo.setActionListener(listener);
mGuardar.setActionListener(listener);
mSoloLectura.setActionListener(listener);
```



Las acciones son objetos derivados de ActionListener, que además definen un texto, un icono y un estado habilitado o deshabilitado.

Cuando se asigna una acción a un botón o elemento de menu, además de manejar el evento click, se enlazan las propiedades de texto, icono y "enabled" del componente a los valores que tenga la acción.

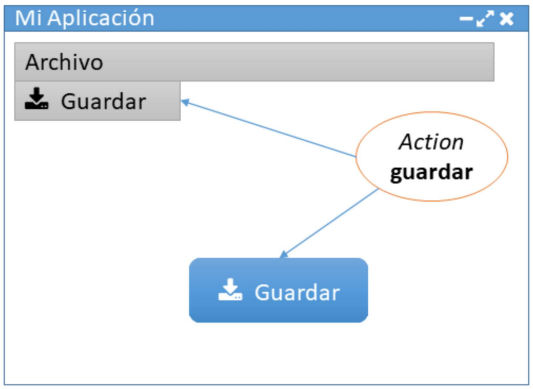
En una aplicación podemos usar 2 tipos de acciones:

- Acciones personalizadas: Cuando queremos asociar el mismo ActionListener y una apariencia similar a un elemento de menu y a un botón en una barra de herramientas. Para ello debemos crear una clase derivada de AbstractAction, asignarle el texto e icono deseados, e implementar el método *actionPerformed()*.

- Acciones predefinidas en componentes Swing: Algunos componentes de Swing definen sus propias acciones, a modo de "comandos", y que puedo asignar a un botón o elemento de menu para que dicha acción se lance. Por ejemplo, los componentes de texto disponen de multiples acciones para las típicas tareas de edición de texto: copiar, pegar, etc. De esa forma podría asociar la acción "Copiar" a un elemento de mi menu. En estos casos, el comportamiento ya viene definido en la accion.

- Podemos asignar una acción a un componente:
    - En el constructor, para botones y opciones de menú.
    - Mediante el método setAction(Action).

Crear una acción personalizada para boton y menu

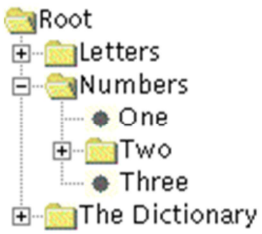



```
Action guardar = new AbstractAction("Guardar",
  iconoGuardar) {
  @Override
  public void actionPerformed(ActionEvent e) {
        // Código para guardar...
  }
};

// Asignar la acción en el constructor
JButton btGuardar = new JButton(guardar);

// Asignar la acción tras la creación
JMenuItem menuGuardar = new JMenuItem();
menuGuardar.setAction(guardar);

// Deshabilitando la acción...
guardar.setEnabled(false); // ...deshabilita
  el menú y el botón
```

Se pueden crear acciones personalizadas, derivando de la clase AbstractAction.

Arboles (JTree)

- Componentes:
  - Arbol → **JTree**
  - Nodo → **TreeNode** (*DefaultMutableTreeNode*)
- Se construye :
  - A partir de un **nodo raíz**
  - O con un HashTable, array de objetos o Vector
- Nodos hijos:
  - Tipo "Rama": pueden tener a su vez nodos hijos
  - Tipo "Hoja"(*leaf*): no tienen hijos, son nodos finales
- Los nodos se cargan recursivamente con ***add()***
- Tipos de nodos según contenido asociado:
  - Nodo simple: texto
  - Nodo complejo: objeto de cualquier tipo. Como texto en el árbol se aplica "*toString()*" del objeto.



Arboles (JTree)

- Otras capacidades:
  - Selección simple o múltiple
  - Arrastrar y soltar
  - L&F define iconos por defecto, pero se pueden personalizar
  - El texto de cada nodo puede ser editado por el usuario

- Eventos:
  - Cambio de nodo seleccionado
  - Expandir o comprimir cualquier nodo
  - Añadir, modificar o quitar nodos