ACTIVIDAD GUIADA 12
Componentes de Menús

Barras de menú

Las barras de menú son un método para agrupar muchas opciones en un espacio reducido. Por lo general se ubican en la parte superior de la ventana y contienen un conjunto de opciones que se agrupan por categorías.

Para crear una barra de menú en un frame utilizamos la clase **`JMenuBar`**. Para agregar la barra al frame se utiliza el método **`setJMenuBar()`** del mismo (**no se utiliza un administrador de contenido** como con el resto de componentes).

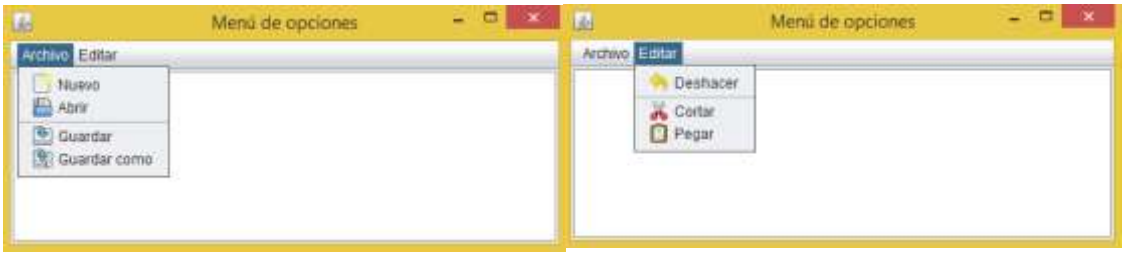
Un menú está formado por una serie de objetos **`JMenu`**. Cada **`JMenu`** contiene un grupo de acciones a realizar, y al pulsar en ellos se desplegarán dichas acciones. El constructor **`JMenu(String nombre)`** permite crear un menú con el nombre indicado. Los menús se añaden a la barra de menú con el método **`add()`** de **`JMenuBar()`**

Cada menú está formado por una serie de ítems de menú (**`JMenuItem`**) que representan las acciones a realizar. Los ítems de menú pueden crearse con los siguientes constructores:

- **`JMenuItem(String nombre)`**: Crea el menú ítem con el nombre indicado.
- **`JMenuItem(String nombre, ImageIcon icono)`**: Crea el menú ítem con el icono y el nombre indicado.

Podemos añadir ítems de menú con el método **`add()`** de **`JMenu`**. También podemos añadir separadores entre distintos ítems de menú con el método **`addSeparator()`** de **`JMenu`**.

El siguiente código muestra un frame que contiene una barra de menús con dos menús.



Se han utilizado los iconos de la librería Tango, accesible desde el siguiente enlace: http://tango.freedesktop.org/Tango_Icon_Library

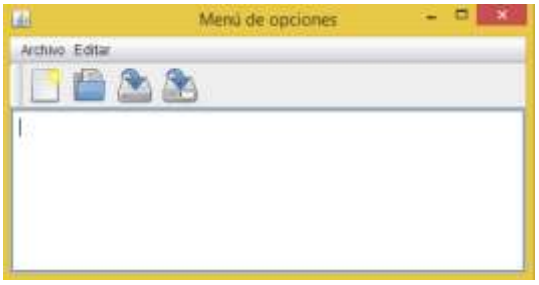
```
import java.awt.BorderLayout;
import javax.swing.*;

public class MenuOpcion extends JFrame {

    public MenuOpcion(){
        super("Menú de opciones");
        setLookAndFeel();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Crear iconos
        ImageIcon iconNuevo = new ImageIcon(
                            getClass().getResource("/resources/document-new.png"));
        ImageIcon iconAbrir = new ImageIcon(
                            getClass().getResource("/resources/document-open.png"));
        ImageIcon iconGuardar = new ImageIcon(
                            getClass().getResource("/resources/document-save.png"));
        ImageIcon iconGuardarComo = new ImageIcon(
                        getClass().getResource("/resources/document-save-as.png"));
        ImageIcon iconCortar = new ImageIcon(
                            getClass().getResource("/resources/edit-cut.png"));
        ImageIcon iconPegar = new ImageIcon(
                            getClass().getResource("/resources/edit-paste.png"));
        ImageIcon iconDeshacer = new ImageIcon(
                            getClass().getResource("/resources/edit-undo.png"));
        // Crear menu
        JMenuItem miNuevo = new JMenuItem("Nuevo",iconNuevo);
        JMenuItem miAbrir = new JMenuItem("Abrir", iconAbrir);
        JMenuItem miGuardar = new JMenuItem("Guardar", iconGuardar);
        JMenuItem miGuardarComo = new JMenuItem("Guardar como",iconGuardarComo);

        JMenuItem miCortar = new JMenuItem("Cortar",iconCortar);
        JMenuItem miPegar = new JMenuItem("Pegar",iconPegar);
        JMenuItem miDeshacer = new JMenuItem("Deshacer",iconDeshacer);

        JMenu menuArchivo = new JMenu("Archivo");
        JMenu menuEditar = new JMenu("Editar");

        menuArchivo.add(miNuevo);
        menuArchivo.addSeparator();
        menuArchivo.add(miGuardar);
        menuArchivo.add(miGuardarComo);
        menuEditar.add(miDeshacer);
        menuEditar.addSeparator();
        menuEditar.add(miCortar);
        menuEditar.add(miPegar);

        JMenuBar menuBar = new JMenuBar();
        menuBar.add(menuArchivo);
        menuBar.add(menuEditar);
```

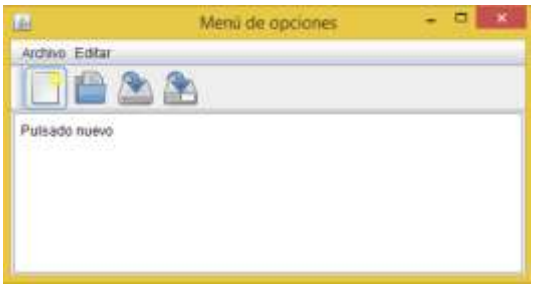
```
        // Preparar interface usuario
        JTextArea editar = new JTextArea(8, 40);
        JScrollPane scroll = new JScrollPane(editar);
        BorderLayout admin = new BorderLayout();
        setLayout(admin);

        setJMenuBar(menuBar);
        add(scroll,BorderLayout.CENTER);
        pack();
        setVisible(true);
}

    public static void main(String[] args) {
      MenuOpcion frMenuOpcion=new MenuOpcion();
    }
}
```

Barra de herramientas

La siguiente aplicación combina el uso de un menú de opciones con una barra de herramientas. Como se verá posteriormente, lo más común es que el menú de opciones contenga todas las opciones que podemos realizar en el programa, mientras que la barra de herramientas contenga sólo algunas (habitualmente las que se considera más importantes o las que el usuario e la aplicación considere).

En este ejemplo se insertan en la barra de herramientas las acciones *Nuevo, Abrir, Guardar y Guardar Como*, implementadas mediante los correspondientes botones.

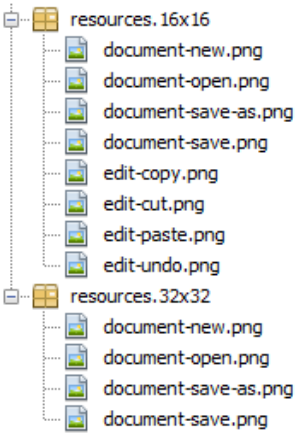




Se han utilizado dos tipos de iconos distintos, ambos incluidos en la colección:

- Iconos de 16x16 píxeles para los ítems de menú.
- Iconos de 32x32 píxeles para los botones de la barra de herramientas.

Ambos se han organizado convenientemente en subpaquetes dentro del paquete *resources*:



Como algunas de las opciones están duplicadas (tanto a través del menú de opciones como de la barra de herramientas), sería conveniente establecer un método para tratarlas de manera simultánea en el método asociado al evento. En ejemplos anteriores comprobábamos directamente cuál era el objeto causante del evento.

En este ejercicio se ha utilizado el método **`setActionCommand(String comando)`**, disponible en la mayoría de componentes Swing. Este método nos permite establecer un código de comando (el que nosotros consideremos) que permite identificar la acción asociada a un botón o una opción de menú, de modo que luego la podamos usar para identificar la fuente del evento en el manejador asociado al mismo.

Por ejemplo, en el ítem de menú *Nuevo* establecemos el siguiente comando:

```
miNuevo.setActionCommand("Nuevo");
```

Y lo mismo hacemos en el botón correspondiente de la barra de herramientas:

```
btnNuevo.setActionCommand("Nuevo");
```

De este modo estamos asociando la misma opción a ambos componentes. Posteriormente, en el método asociado al evento **`actionPerformed()`** [que recogerá las pulsaciones en los dos] comprobaremos si el comando ha sido el especificado:

```
public void actionPerformed(ActionEvent ae) {
  if(ae.getActionCommand().equals("Nuevo")
     accionNuevo();
```

El método **`accionNuevo()`** implementa las acciones correspondientes (en nuestro ejemplo simplemente escribir en el área de texto).

*NOTA: En el ejemplo se utiliza una estructura de tipo **`switch…case`** para hacer la comprobación, dado que se han definido comandos para el resto de acciones del programa que están duplicadas tanto en el menú como en la barra de herramientas (Abrir, Guardar, Guardar como)*

```
import java.awt.BorderLayout;
import java.awt.event.*;
import javax.swing.*;

public class MenuOpcionBarra extends JFrame implements ActionListener{
   JTextArea editor;
    public MenuOpcionBarra(){
        super("Menú de opciones");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Crear iconos
        ImageIcon iconNuevo = new ImageIcon(
                    getClass().getResource("/resources/16x16/document-new.png"));
        ImageIcon iconNuevoGrande = new ImageIcon(
                    getClass().getResource("/resources/32x32/document-new.png"));
        ImageIcon iconAbrir = new ImageIcon(
                    getClass().getResource("/resources/16x16/document-open.png"));
        ImageIcon iconAbrirGrande = new ImageIcon(
                    getClass().getResource("/resources/32x32/document-open.png"));
        ImageIcon iconGuardar = new ImageIcon(
                    getClass().getResource("/resources/16x16/document-save.png"));
        ImageIcon iconGuardarGrande = new ImageIcon(
                    getClass().getResource("/resources/32x32/document-save.png"));
        ImageIcon iconGuardarComo = new ImageIcon(
              getClass().getResource("/resources/16x16/document-save-as.png"));
        ImageIcon iconGuardarComoGrande = new ImageIcon(
              getClass().getResource("/resources/32x32/document-save-as.png"));
        ImageIcon iconCortar = new ImageIcon(
              getClass().getResource("/resources/16x16/edit-cut.png"));
        ImageIcon iconCortarGrande = new ImageIcon(
              getClass().getResource("/resources/32x32/edit-cut.png"));
        ImageIcon iconPegar = new ImageIcon(
              getClass().getResource("/resources/16x16/edit-paste.png"));
        ImageIcon iconPegarGrande = new ImageIcon(
              getClass().getResource("/resources/32x32/edit-paste.png"));
        ImageIcon iconDeshacer = new ImageIcon(
              getClass().getResource("/resources/16x16/edit-undo.png"));
        ImageIcon iconDeshacerGrande = new ImageIcon(
              getClass().getResource("/resources/32x32/edit-undo.png"));
        // Crear menu
        JMenuItem miNuevo = new JMenuItem("Nuevo",iconNuevo);
        miNuevo.setActionCommand("Nuevo");
        miNuevo.addActionListener(this);
        JMenuItem miAbrir = new JMenuItem("Abrir", iconAbrir);
        miAbrir.setActionCommand("Abrir");
        miAbrir.addActionListener(this);
        JMenuItem miGuardar = new JMenuItem("Guardar", iconGuardar);
        miGuardar.setActionCommand("Guardar");
        miGuardar.addActionListener(this);
        JMenuItem miGuardarComo = new JMenuItem("Guardar como",iconGuardarComo);
        miGuardarComo.setActionCommand("Guardar como");
        miGuardarComo.addActionListener(this);
```

```
        JMenuItem miCortar = new JMenuItem("Cortar",iconCortar);
        JMenuItem miPegar = new JMenuItem("Pegar",iconPegar);
        JMenuItem miDeshacer = new JMenuItem("Deshacer",iconDeshacer);

        JMenu menuArchivo = new JMenu("Archivo");
        JMenu menuEditar = new JMenu("Editar");

        menuArchivo.add(miNuevo);
        menuArchivo.add(miAbrir);
        menuArchivo.addSeparator();
        menuArchivo.add(miGuardar);
        menuArchivo.add(miGuardarComo);
        menuEditar.add(miDeshacer);
        menuEditar.addSeparator();
        menuEditar.add(miCortar);
        menuEditar.add(miPegar);

        JMenuBar menuBar = new JMenuBar();
        menuBar.add(menuArchivo);
        menuBar.add(menuEditar);

//Crear botones de la barra de herramientas
        JButton btnNuevo=new JButton(iconNuevoGrande);
        btnNuevo.setActionCommand("Nuevo");
        btnNuevo.addActionListener(this);
        JButton btnAbrir=new JButton(iconAbrirGrande);
        btnAbrir.setActionCommand("Abrir");
        btnAbrir.addActionListener(this);
        JButton btnGuardar=new JButton(iconGuardarGrande);
        btnGuardar.setActionCommand("Guardar");
        btnGuardar.addActionListener(this);
        JButton btnGuardarComo=new JButton(iconGuardarComoGrande);
        btnGuardarComo.setActionCommand("Guardar como");
        btnGuardarComo.addActionListener(this);

        //Crear la barra de herramientas
        JToolBar barraHerramientas=new JToolBar();
        barraHerramientas.add(btnNuevo);
        barraHerramientas.add(btnAbrir);
        barraHerramientas.add(btnGuardar);
        barraHerramientas.add(btnGuardarComo);

        // Preparar interface usuario
        editor = new JTextArea(8, 40);
        JScrollPane scroll = new JScrollPane(editor);
        BorderLayout admin = new BorderLayout();
        setLayout(admin);

        setJMenuBar(menuBar);
        add(barraHerramientas, BorderLayout.NORTH);
        add( scroll,BorderLayout.CENTER);

         pack();
         setVisible(true);
}
```

```
    public static void main(String[] args) {
      MenuOpcionBarra frMenuOpcionBarra=new MenuOpcionBarra();
    }

    @Override
    public void actionPerformed(ActionEvent ae) {
        switch(ae.getActionCommand()){
            case "Nuevo":{
                accionNuevo();
                break;
            }
            case "Abrir":{
                accionAbrir();
                break;
            }
            case "Guardar":{
                accionGuardar();
                break;
            }
            case "Guardar como":{
                accionGuardarComo();
                break;
            }
        }

    }

    private void accionNuevo(){
        editor.setText("Pulsado nuevo");
    }
    private void accionAbrir(){
         editor.setText("Pulsado abrir");
    }
    private void accionGuardar(){
        editor.setText("Pulsado guardar");
    }
    private void accionGuardarComo(){
        editor.setText("Pulsado guardar como");
    }
}
```

Menú contextual

Los menús contextuales son menús que aparecen ocultos hasta que el usuario realiza una acción determinada (habitualmente pulsar el botón derecho del ratón). En ese momento el menú aparece en el sitio donde hemos pulsado. Reciben dicho nombre porque lo más habitual es que dependan del contexto, es decir, del sitio donde pulsemos.

Los menús contextuales en Swing se crean mediante el componente **`JPopupMenu`**. A este componente le podemos añadir ítems de menú al igual que hacemos con los menús:

- **`add(JMenuItem menuItem)`**. Añade el ítem de menú correspondiente.
- **`addSeparator()`**. Añade una barra separadora.

Para asignar el menú contextual a un elemento, crearemos un escuchador de ratón que implemente al menos los métodos **`mousePressed()`** y **`mouseReleased()`** -si bien hay autores que recomiendan implementarlos todos. Dentro de los mismos, accederemos al objeto **`MouseEvent`** y mediante su método **`isPopUpTrigger()`** comprobaremos si el botón que se ha pulsado es el empleado en el Sistema Operativo actual para mostrar el menú contextual. En caso afirmativo, establecemos la ubicación del menú, que habitualmente es el sitio donde hemos pulsado, y mostramos el mismo.

Posteriormente asociamos el **escuchador** al componente donde queremos que se muestre el menú.

El siguiente código añade un menú contextual al área de texto del ejemplo que hemos realizado con un menú y una barra de herramientas. El primer paso es definir el menú como una propiedad del frame:

```
JPopupMenu popup;
```

En el constructor del mismo, lo inicializamos y añadimos los ítems de menú correspondientes:

```
popup=new JPopupMenu();
JMenuItem miCortarContextual=new JMenuItem("Cortar");
miCortarContextual.setActionCommand("Cortar");
JMenuItem miCopiarContextual=new JMenuItem("Copiar");
miCortarContextual.setActionCommand("Copiar");

popup.add(miCortarContextual);
popup.addSeparator();
popup.add(miCopiarContextual);
```

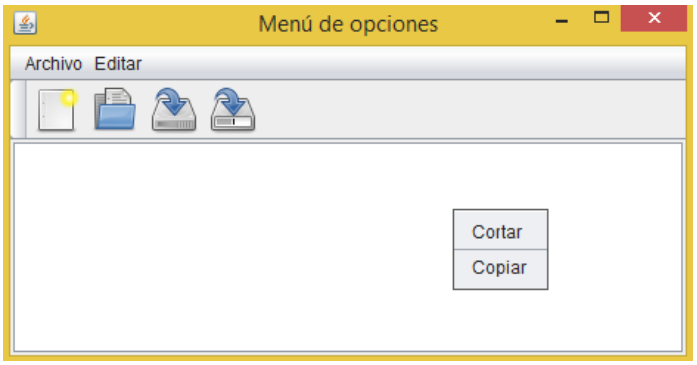
Creamos el listener como clase interna, de modo que podemos acceder al menú:

```
private class PopUpListener extends MouseAdapter {

 public void mousePressed(MouseEvent e) {
       muestraPopUp(e);
    }

    public void mouseReleased(MouseEvent e) {
        muestraPopUp(e);
    }

    private void muestraPopUp(MouseEvent e) {
        if (e.isPopupTrigger()) {
            popup.show(e.getComponent(),
                       e.getX(), e.getY());
        }
    }
   }
```

Y como último paso hacemos que el área de texto sea escuchador de los eventos de ratón:

```
editor.addMouseListener(new PopUpListener());
```

Si ejecutamos el programa, veremos que si pulsamos con el botón derecho sobre el área de texto se mostrará el menú contextual:



No hemos asociado ninguna acción al menú contextual, pues en el ejemplo anterior tampoco lo habíamos hecho para las acciones del menú *Editar*. Tendríamos que tratar el caso correspondiente en el `ActionListener` que habíamos implementado.

Por otra parte, nótese que hemos duplicado el código, dado que hemos añadido dos `JMenuItem` con las acciones de *Cortar* y *Copiar* (los correspondientes al menú y al popUp). Tenemos dos opciones para resolver este problema:

- Crear una función que nos devuelva el `JMenuItem` personalizado, y a la cual invocaríamos cada vez que queremos crear uno.
- Crear nuestros propios componentes ítems de menú, haciendo que hereden de la clase base `JMenuItem`. Este enfoque lo seguiremos en la UD3.