ACTIVIDAD GUIADA 7
Casillas de verificación y botones de opción

Tienen en común que pueden contener dos posibles valores: seleccionados o no seleccionados.

Las **casillas de verificación** se usan para hacer elecciones simples en una interfaz, como *si-no* u *on-off*. Los **botones de opción** se agrupan juntos de forma que sólo se puede seleccionar uno al mismo tiempo.

Las casillas de verificación (la clase **`JCheckBox`**) aparecen como casillas etiquetadas o no etiquetadas que contienen una marca cuando está seleccionada y nada si es de otra forma. Los botones de opción (de la clase **`JRadioButton`**) aparecen como círculos que contienen un punto cuando están seleccionados y se encuentran vacíos de lo contrario.

Tanto las clases **`JCheckBox`** como las **`JRadioButton`** tienen diversos métodos:

- **`setSelected(boolean)`***:* Selecciona el componente si el argumento es **`true`** y lo deselecciona si es de otro modo.
- **`isSelected()`***:* Devuelve un **`boolean`** que indica si el componente está seleccionado actualmente.

Los siguientes constructores están disponibles para la clase **`JCheckBox`**:

- **`JCheckBox(String)`***:* Una casilla de comprobación con la etiqueta de texto especificada.
- **`JCheckBox(String, boolean)`***:* Una casilla de comprobación con la etiqueta de texto especificada que se selecciona si el segundo argumento es *true*.
- **`JCheckBox(Icon)`***:* Una casilla de comprobación con el icono gráfico que se ha especificado. Posteriormente veremos cómo especificar el icono.
- **`JCheckBox(Icon, boolean)`***:* Una casilla de comprobación con el icono gráfico que se ha especificado que se selecciona si el segundo argumento es *true*.
- **`JCheckBox(String, Icon)`***:* Una casilla de comprobación con la etiqueta de texto y el icono gráfico especificados.
- **`JCheckBox(String, Icon, boolean)`***:* Una casilla de comprobación con la etiqueta de texto y el icono gráfico especificado que se selecciona si el tercer argumento es *true*.

La clase **`JRadioButton`** tiene constructores con los mismos argumentos y funcionalidad.

Las casillas de comprobación y botones de radio por sí mismos son no exclusivos, lo que significa que si tiene cinco casillas de comprobación en un contenedor, las cinco pueden estar marcadas o desmarcadas al mismo tiempo. Para hacerlas exclusivas, como deberían ser los botones de radio, es necesario organizar los componentes relacionados en grupos.

Para organizar diversos botones de radio en un grupo, permitiendo que sólo se pueda seleccionar uno a la vez, debemos crear un objeto de clase **ButtonGroup**, como se muestra en la siguiente instrucción:

```
ButtonGroup grpColores = new ButtonGroup();
```

El objeto **ButtonGroup** sigue la permite agrupar un conjunto de botones. Llamaremos al método del grupo **add(Component)** para añadir el componente especificado al grupo.
El siguiente ejemplo crea un grupo y dos botones de radio que le pertenecen:

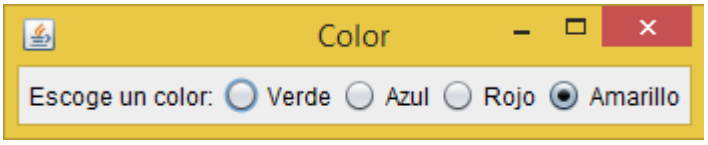
```
ButtonGroup grpColores=new ButtonGroup();

JRadioButton rbVerde=new JRadioButton("Verde",false);
grpColores.add(rbVerde);

JRadioButton rbAzul=new JRadioButton("Azul",true);
grpColores.add(rbAzul);
```

El objeto **grpColores** se usa para agrupar los botones de radio **btnVerde** y **btnAzul**. Se selecciona el objeto **btnAzul**, que tienen la etiqueta "*Azul*". Solo un miembro del grupo se puede seleccionar cada vez, si se selecciona un componente, el objeto **grpColores** se asegura de que todos los objetos en el grupo están deseleccionados.

En el código siguiente la aplicación contiene cuatro botones de radio diferentes en un grupo. Es necesario añadir los botones uno a uno al frame. El objeto **grpColores** sólo nos vale para agrupar de forma lógica los botones, pero no es un contenedor físico que contiene a los mismos.



```
public class Color extends JFrame {
 public Color() {
    super("Color");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    FlowLayout admin = new FlowLayout();
    setLayout(admin);

    JLabel lblColores=new JLabel("Escoge un color:");
    add(lblColores);

    ButtonGroup grpColores=new ButtonGroup();

    JRadioButton rbVerde=new JRadioButton("Verde",false);
    grpColores.add(rbVerde);

    JRadioButton rbAzul=new JRadioButton("Azul",false);
    grpColores.add(rbAzul);

    JRadioButton rbRojo=new JRadioButton("Rojo",false);
    grpColores.add(rbRojo);

    JRadioButton rbAmarillo=new JRadioButton("Amarillo",true);
    grpColores.add(rbAmarillo);

    add(rbVerde);
    add(rbAzul);
    add(rbRojo);
    add(rbAmarillo);
    pack();
}

public static void main(String[] arguments) {
    new Color().setVisible(true);
}

}
```

Una posible modificación al código anterior sería introducir todos los botones en un array, de este modo no manejamos tantas variables y podemos añadir todos los botones al panel en un bucle. El cambio no supone ningún cambio en la funcionalidad:

```
public class Color extends JFrame {
  private JRadioButton[] rbColores=new JRadioButton[4];

 public Color() {
...
    rbColores[0]=new JRadioButton("Verde",false);
    rbColores [1]=new JRadioButton("Azul",false);
    rbColores [2]=new JRadioButton("Rojo",false);
    rbColores [3]=new JRadioButton("Amarillo",true);

    for(JRadioButton rbColor:rbColores){
        grpColores.add(rbColor);
        add(rbColor);
    }
    ...
```

Un componente **JCheckBox** es una casilla junto a una línea de texto que el usuario puede marcar o desactivar. Las siguientes instrucciones crean un objeto **JCheckBox** y lo añaden al frame donde se encuentra:

```
JCheckBox chkHamburguesa = new JCheckBox("Hamburguesa");
add(chkHamburguesa);
```

El argumento del constructor **JCheckBox()** indica el texto que se va a mostrar junto a la casilla. Si quisiéramos que la casilla estuviera marcada usaríamos esta opción:

```
JCheckBox chkHamburguesa = new JCheckBox("Hamburguesa",true);
add(chkHamburguesa);
```

Es posible presentar un **JCheckBox** de forma individual o en grupo. En un grupo de casillas de verificación, sólo se puede seleccionar una por vez. Para que un objeto **JCheckBox** forme parte de un grupo, es necesario crear un objeto **ButtonGroup**:

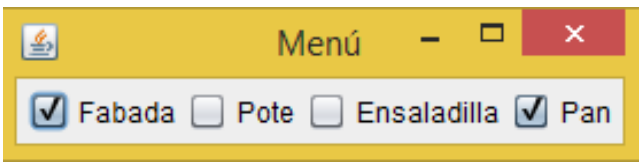
```
    JCheckBox chkFabada=new JCheckBox("Fabada",true);
    JCheckBox chkPote=new JCheckBox("Pote");
    JCheckBox chkEnsaladilla=new JCheckBox("Ensaladilla");
    JCheckBox chkPan=new JCheckBox("Pan",true);

    ButtonGroup grpComidas=new ButtonGroup();

    grpComidas.add(chkFabada);
    grpComidas.add(chkFabada);
    grpComidas.add(chkEnsaladilla);

    add(chkFabada);
    add(chkFabada);
    add(chkEnsaladilla);
    add(chkPan);
```

Se crean tres casillas de verificación agrupadas bajo el objeto **grpComidas**, de tipo **ButtonGroup**. La casilla Fabada aparece inicialmente marcada pero si el usuario marca otra de las casillas, se desactiva automáticamente. En la figura siguiente podemos ver cómo quedaría al ejecutarlo:



El código completo sería:

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

public class Menu extends JFrame {
    public Menu(){
     super("Menú");

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    FlowLayout admin = new FlowLayout();
    setLayout(admin);

    JCheckBox fabada=new JCheckBox("Fabada",true);
    JCheckBox pote=new JCheckBox("Pote");
    JCheckBox ensaladilla=new JCheckBox("Ensaladilla");
    JCheckBox pan=new JCheckBox("Pan",true);

    ButtonGroup comidas=new ButtonGroup();

    comidas.add(fabada);
    comidas.add(pote);
    comidas.add(ensaladilla);

    add(fabada);
    add(pote);
    add(ensaladilla);
    add(pan);

    pack();
    setVisible(true);
    }

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

JToggleButton

Los **JToggleButton** son botones que pueden quedarse pulsados.

A través del método ***isSelected*** podemos saber si un JToggleButton está seleccionado. También puedes usar el método **setSelected** para seleccionar o no un botón de este tipo.

Realmente, estos botones no suelen ser muy usados, ya que pueden ser sustituidos por JCheckBox que son más conocidos.

```
JToggleButton tglbtn = new JToggleButton("Botón de activación");
tglbtn.setSelected(true);     // seleccionado inicialmente
tglbtn.setBounds(10, 259, 166, 23);
contentPane.add(tglbtn);
```