JCheckBox.

Este elemento es una casilla de verificación, la forma más habitual de instanciar un objeto de esta clase es:

JCheckBox casilla = JcheckBox(“Texto de la casilla”);

Las casillas de verificación tiene dos estados: seleccionado o no seleccionado. Podemos comprobar a través de método isSelected() si está o no seleccionada; si lo está devuelve true y si no devuelve false.

Podemos asociarle un evento para detectar cuando cambia el estado, es decir si el usuario lo marca o no:

```
class escuchador_casilla implements ItemListener{

    public void itemStateChanged( ItemEvent evento ){
    //código que se ha de ejecutar

    }
}
```

La clase implementa ItemListener, es donde están definidos todos los posibles eventos ante los que puede responder una casilla de verificación. En concreto para responder cuando el usuario hace click y cambia el estado se ha de reescribir el método itemStateChanged.

Para añadir ese escuchador al objeto:

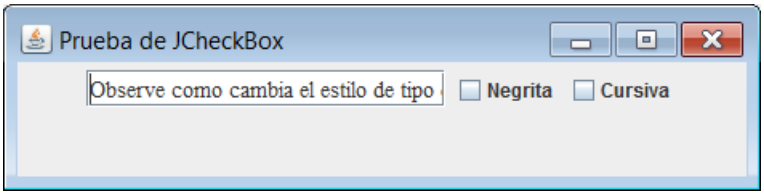
casilla.addItemListener(new escuchador_casilla());

Veamos un ejemplo de cierta complejidad en el que se usan casillas de verificación:

```
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JCheckBox;

public class MarcoCasillaVerificacion extends JFrame{
    private JTextField campoTexto; // muestra el texto en tipos de letra cambiantes
    private JCheckBox negritaJCheckBox; // para seleccionar/deseleccionar negrita
    private JCheckBox cursivaJCheckBox; // para seleccionar/deseleccionar cursiva

    // El constructor de MarcoCasillaVerificacion agrega objetos JCheckBox a JFrame
    public MarcoCasillaVerificacion(){
        setTitle( "Prueba de JCheckBox" );
        setLayout( new FlowLayout() );

        // establece JTextField y su tipo de letra
        campoTexto = new JTextField( "Observe como cambia el estilo de tipo de letra", 20 );
        campoTexto.setFont( new Font( "Serif", Font.PLAIN, 14 ) );
        add( campoTexto ); // agrega campoTexto a JFrame

        negritaJCheckBox = new JCheckBox( "Negrita" ); // crea casilla de verificación "negrita"
        cursivaJCheckBox = new JCheckBox( "Cursiva" ); // crea casilla de verificación "cursiva"
        add( negritaJCheckBox ); // agrega casilla de verificación "negrita" a JFrame
        add( cursivaJCheckBox ); // agrega casilla de verificación "cursiva" a JFrame

        // listener de  JCheckBox
        ManejadorCheckBox manejador = new ManejadorCheckBox();
```

```
        negritaJCheckBox.addItemListener( manejador );
        cursivaJCheckBox.addItemListener( manejador );
    }

    // clase para el escuchador de los JCheckBox, en este caso no es un ActionListener como en los botones, es un ItemListener
    private class ManejadorCheckBox implements ItemListener{
        private int valNegrita = Font.PLAIN; // controla el estilo de tipo de letra negrita
        private int valCursiva = Font.PLAIN; // controla el estilo de tipo de letra cursiva

        // cuando cambia el estado de un JCheckBox:
        public void itemStateChanged( ItemEvent evento ){
            // procesa los eventos de la casilla de verificación "negrita"
             if ( evento.getSource() == negritaJCheckBox )
                valNegrita = negritaJCheckBox.isSelected() ? Font.BOLD : Font.PLAIN;

            // procesa los eventos de la casilla de verificación "cursiva"
            if ( evento.getSource() == cursivaJCheckBox )
                valCursiva = cursivaJCheckBox.isSelected() ? Font.ITALIC : Font.PLAIN;

            // establece el tipo de letra del campo de texto
            campoTexto.setFont( new Font( "Serif", valNegrita + valCursiva, 14 ) );
        }
    }
}
```

El resultado del código anterior:



JRadioButton.

Los botones de opción (que se declaran con la clase JRadioButton) son similares a las casillas de verificación, en cuanto a que tienen dos estados: seleccionado y no seleccionado.

Sin embargo, los botones de opción generalmente aparecen como un grupo, en el cual sólo un botón de opción puede estar seleccionado en un momento dado . Al seleccionar un botón de opción distinto en el grupo se obliga a que todos los demás botones de opción del grupo se deseleccionen. Los botones de opción se utilizan para representar un conjunto de opciones mutuamente exclusivas (es decir, no pueden seleccionarse varias opciones en el grupo al mismo tiempo).

Para poder agrupar uno o más botones de opción estos se deben asociar a un un objeto ButtonGroup. Un objeto ButtonGroup no es un objeto visible pera hace que los botones de opción asociados a él actúen juntos y se comporten de forma mutuamente exclusiva.

Por ejemplo si queremos dos botones de opción agrupados podríamos hacer:

```
JRadioButton bo1 = new JRadioButton( "Opción 1", true );

JRadioButton bo2 = new JRadioButton( "Opción 2", false);

ButtonGroup grupoOpciones = new ButtonGroup();


grupoOpciones.add(bo1);

grupoOpciones.add(bo1);
```

Aun botón de opción se le puede añadir un manejador de eventos para detectar si el usuario lo ha pulsado, en este caso se usa exactamente el mismo que para JcheckBox:

```
class escuchador_botonopcion implements ItemListener{
```

```
        public void itemStateChanged( ItemEvent evento ){
        //código que se ha de ejecutar

        }
    }
```

Y para añadirlo a los botones de opción:

```
bo1.addItemListener( new escuchador_botonopcion() );
bo2.addItemListener( new escuchador_botonopcion() );
```

A continuación un ejemplo similar al anterior pero con botones de opción:

```
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;

public class MarcoBotonOpcion extends JFrame
{
private JTextField campoTexto; // se utiliza para mostrar los cambios en el tipo de letra
private Font tipoLetraSimple; // tipo de letra para texto simple
private Font tipoLetraNegrita; // tipo de letra para texto en negrita
private Font tipoLetraCursiva; // tipo de letra para texto en cursiva
private Font tipoLetraNegritaCursiva; // tipo de letra para texto en negrita y cursiva
```

```
private JRadioButton simpleJRadioButton; // selecciona texto simple

private JRadioButton negritaJRadioButton; // selecciona texto en negrita

private JRadioButton cursivaJRadioButton; // selecciona texto en cursiva

private JRadioButton negritaCursivaJRadioButton; // negrita y cursiva

private ButtonGroup grupoOpciones; // grupo de botones que contiene los botones de opción



public MarcoBotonOpcion()

{

setTitle( "Prueba de RadioButton" );

setLayout( new FlowLayout() ); // establece el esquema del marco


campoTexto = new JTextField( "Observe el cambio en el estilo del tipo de letra", 28);

add( campoTexto ); // agrega campoTexto a JFrame


// crea los botones de opción

simpleJRadioButton = new JRadioButton( "Simple", true );

negritaJRadioButton = new JRadioButton( "Negrita", false );

cursivaJRadioButton = new JRadioButton( "Cursiva", false );

negritaCursivaJRadioButton = new JRadioButton( "Negrita/Cursiva", false );

add( simpleJRadioButton ); // agrega botón simple a JFrame

add( negritaJRadioButton ); // agrega botón negrita a JFrame

add( cursivaJRadioButton ); // agrega botón cursiva a JFrame

add( negritaCursivaJRadioButton ); // agrega botón negrita y cursiva


// crea una relación lógica entre los objetos JRadioButton

grupoOpciones = new ButtonGroup(); // crea ButtonGroup

grupoOpciones.add( simpleJRadioButton ); // agrega simple al grupo

grupoOpciones.add( negritaJRadioButton ); // agrega negrita al grupo

grupoOpciones.add( cursivaJRadioButton ); // agrega cursiva al grupo

grupoOpciones.add( negritaCursivaJRadioButton ); // agrega negrita y cursiva


// crea objetos tipo de letra
```

```
tipoLetraSimple = new Font( "Serif", Font.PLAIN, 14 );
tipoLetraNegrita = new Font( "Serif", Font.BOLD, 14 );
tipoLetraCursiva = new Font( "Serif", Font.ITALIC, 14 );
tipoLetraNegritaCursiva = new Font( "Serif", Font.BOLD + Font.ITALIC, 14 );
campoTexto.setFont( tipoLetraSimple ); // establece tipo letra inicial a simple

// registra eventos para los objetos JRadioButton
simpleJRadioButton.addItemListener(
new ManejadorBotonOpcion( tipoLetraSimple ) );
negritaJRadioButton.addItemListener(
new ManejadorBotonOpcion( tipoLetraNegrita ) );
cursivaJRadioButton.addItemListener(
new ManejadorBotonOpcion( tipoLetraCursiva ) );
negritaCursivaJRadioButton.addItemListener(
new ManejadorBotonOpcion( tipoLetraNegritaCursiva ) );
} // fin del constructor de MarcoBotonOpcion
// clase para  eventos de botones de opción
private class ManejadorBotonOpcion implements ItemListener
{
private Font tipoLetra; // tipo de letra asociado con este componente de escucha
public ManejadorBotonOpcion( Font f )
{
tipoLetra = f; // establece el tipo de letra de este componente de escucha
} // fin del constructor ManejadorBotonOpcion

// maneja los eventos de botones de opción
public void itemStateChanged( ItemEvent evento )
{
campoTexto.setFont( tipoLetra ); // establece el tipo de letra de campoTexto
}
}
}
```

El resultado de esa ventana sería: