Índice:

Swing – JcomboBox

El control JComboBox permite seleccionar un String de una lista.

Para inicializar los String que contendrá el JComboBox debemos llamar al método addItem tantas veces como elementos queremos cargar.

Un evento muy útil con este control es cuando el operador selecciona un Item de la lista. Para capturar la selección de un item debemos implementar la interface ItemListener que contiene un método llamada itemStateChanged.

Problema 1:

Cargar en un JComboBox los nombres de varios colores. Al seleccionar alguno mostrar en la barra de título del JFrame el String seleccionado.

Programa:

```
import javax.swing.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ItemListener{
    private JComboBox combo1;
    public Formulario() {
        setLayout(null);
        combo1=new JComboBox();
        combo1.setBounds(10,10,80,20);
        add(combo1);
        combo1.addItem("rojo");
        combo1.addItem("vede");
        combo1.addItem("azul");
        combo1.addItem("amarillo");
        combo1.addItem("negro");
        combo1.addItemListener(this);
    }

    public void itemStateChanged(ItemEvent e) {
        if (e.getSource()==combo1) {
            String seleccionado=(String)combo1.getSelectedItem();
            setTitle(seleccionado);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
```

```
        formulario1.setBounds(0,0,200,150);
        formulario1.setVisible(true);
    }
}
```

Indicamos a la clase que implementaremos la interface ItemListener:

```
public class Formulario extends JFrame implements ItemListener{
```

Declaramos un objeto de la clase ComboBox:

```
    private JComboBox combo1;
```

En el constructor creamos el objeto de la clase JComboBox:

```
        combo1=new JComboBox();
```

Posicionamos el control:

```
        combo1.setBounds(10,10,80,20);
```

Añadimos el control al JFrame:

```
        add(combo1);
```

Añadimos los String al JComboBox:

```
        combo1.addItem("rojo");
        combo1.addItem("vede");
        combo1.addItem("azul");
        combo1.addItem("amarillo");
        combo1.addItem("negro");
```

Asociamos la clase que capturará el evento de cambio de item (con this indicamos que esta misma clase capturará el evento):

```
        combo1.addItemListener(this);
```

El método itemStateChanged que debemos implementar de la interface ItemListener tiene la siguiente sintaxis:

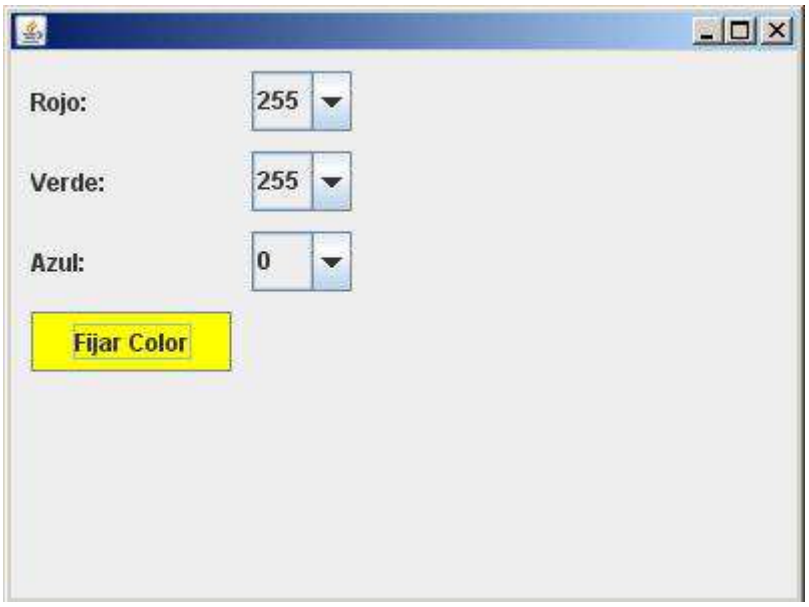
```
    public void itemStateChanged(ItemEvent e) {
        if (e.getSource()==combo1) {
            String seleccionado=(String)combo1.getSelectedItem();
            setTitle(seleccionado);
        }
    }
```

Para extraer el contenido del item seleccionado llamamos al método getSelectemItem() el cual retorna un objeto de la clase Object por lo que debemos indicarle que lo transforme en String:

```
            String seleccionado=(String)combo1.getSelectedItem();
```

Problema 2:

Disponer tres controles de tipo JComboBox con valores entre 0 y 255 (cada uno representa la cantidad de rojo, verde y azul). Luego al presionar un botón pintar el mismo con el color que se genera combinando los valores de los JComboBox.



Programa:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JLabel label1,label2,label3;
    private JComboBox combo1,combo2,combo3;
    private JButton boton1;
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Rojo:");
        label1.setBounds(10,10,100,30);
        add(label1);
        combo1=new JComboBox();
        combo1.setBounds(120,10,50,30);
        for(int f=0;f<=255;f++) {
            combo1.addItem(String.valueOf(f));
        }
        add(combo1);
        label2=new JLabel("Verde:");
        label2.setBounds(10,50,100,30);
        add(label2);
        combo2=new JComboBox();
        combo2.setBounds(120,50,50,30);
        for(int f=0;f<=255;f++) {
            combo2.addItem(String.valueOf(f));
        }
        add(combo2);
        label3=new JLabel("Azul:");
```

```
        label3.setBounds(10,90,100,30);
        add(label3);
        combo3=new JComboBox();
        combo3.setBounds(120,90,50,30);
        for(int f=0;f<=255;f++) {
            combo3.addItem(String.valueOf(f));
        }
        add(combo3);
        boton1=new JButton("Fijar Color");
        boton1.setBounds(10,130,100,30);
        add(boton1);
        boton1.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String cad1=(String)combo1.getSelectedItem();
            String cad2=(String)combo2.getSelectedItem();
            String cad3=(String)combo3.getSelectedItem();
            int rojo=Integer.parseInt(cad1);
            int verde=Integer.parseInt(cad2);
            int azul=Integer.parseInt(cad3);
            Color color1=new Color(rojo,verde,azul);
            boton1.setBackground(color1);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,400,300);
        formulario1.setVisible(true);
    }
}
```

Importamos el paquete java.awt ya que el mismo contiene la clase Color:

```
import java.awt.*;
```

Implementaremos la interface ActionListener ya que tenemos que cambiar el color del botón cuando se lo presione y no haremos actividades cuando cambiemos items de los controles JComboBox:

```
public class Formulario extends JFrame implements ActionListener{
```

Definimos los siete objetos requeridos en esta aplicación:

```
    private JLabel label1,label2,label3;
    private JComboBox combo1,combo2,combo3;
    private JButton boton1;
```

En el constructor creamos los objetos, primero el control label1 de la clase JLabel:

```
        label1=new JLabel("Rojo:");
        label1.setBounds(10,10,100,30);
        add(label1);
```

Lo mismo hacemos con el objeto combo1:

```
        combo1=new JComboBox();
        combo1.setBounds(120,10,50,30);
```

Para añadir los 256 elementos del JComboBox disponemos un for y previa a llamar al método addItem convertimos el entero a String:

```
        for(int f=0;f<=255;f++) {
            combo1.addItem(String.valueOf(f));
        }
        add(combo1);
```

En el método actionPerformed cuando detectamos que se presionó el botón procedemos a extraer los tres item seleccionados:

```
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            String cad1=(String)combo1.getSelectedItem();
            String cad2=(String)combo2.getSelectedItem();
            String cad3=(String)combo3.getSelectedItem();
```

Los convertimos a entero:

```
            int rojo=Integer.parseInt(cad1);
            int verde=Integer.parseInt(cad2);
            int azul=Integer.parseInt(cad3);
```

y creamos finalmente un objeto de la clase Color, el constructor de la clase Color requiere que le pasemos tres valores de tipo int:

```
            Color color1=new Color(rojo,verde,azul);
```

Para cambiar el color de fondo del control JButton debemos llamar al método setBackground y pasarle el objeto de la clase Color:

```
            boton1.setBackground(color1);
```

Problemas propuestos

1. Solicitar el ingreso del nombre de una persona y seleccionar de un control JComboBox un país. Al presionar un botón mostrar en la barra del título del JFrame el nombre ingresado y el país seleccionado.

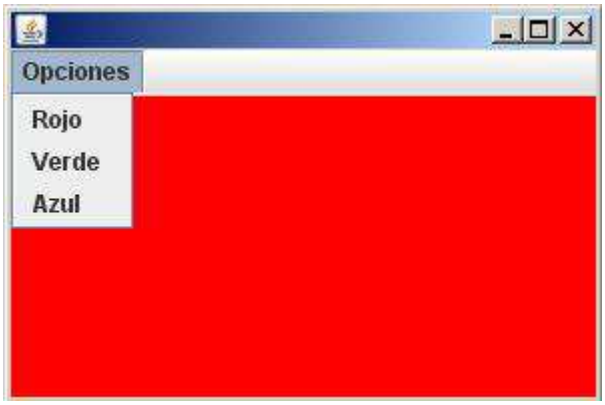
Swing - JMenuBar, JMenu, JMenuItem

Ahora veremos como crear un menú de opciones y la captura de eventos de los mismos. Cuando necesitamos implementar un menú horizontal en la parte superior de un JFrame requerimos de un objeto de la clase JMenuBar, uno o más objetos de la clase JMenu y por último objetos de la clase JMenuItem.

Par la captura de eventos debemos implementar la interface ActionListener y asociarlo a los controles de tipo JMenuItem, el mismo se dispara al presionar con el mouse el JMenuItem.

Problema 1:

Confeccionaremos un menú de opciones que contenga tres opciones que permita cambiar el color de fondo del JFrame a los colores: rojo, verde y azul.



Programa:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JMenuBar mb;
    private JMenu menu1;
    private JMenuItem mi1,mi2,mi3;
    public Formulario() {
        setLayout(null);
        mb=new JMenuBar();
        setJMenuBar(mb);
        menu1=new JMenu("Opciones");
        mb.add(menu1);
        mi1=new JMenuItem("Rojo");
        mi1.addActionListener(this);
        menu1.add(mi1);
        mi2=new JMenuItem("Verde");
        mi2.addActionListener(this);
        menu1.add(mi2);
        mi3=new JMenuItem("Azul");
        mi3.addActionListener(this);
        menu1.add(mi3);
    }
```

```
    public void actionPerformed(ActionEvent e) {
        Container f=this.getContentPane();
        if (e.getSource()==mi1) {
            f.setBackground(new Color(255,0,0));
        }
        if (e.getSource()==mi2) {
            f.setBackground(new Color(0,255,0));
        }
        if (e.getSource()==mi3) {
            f.setBackground(new Color(0,0,255));
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(10,20,300,200);
        formulario1.setVisible(true);
    }
}
```

Importamos el paquete javax.swing ya que en el mismo se encuentran las tres clases JMenuBar, JMenu y JMenuItem:

```
import javax.swing.*;
```

Importamos java.awt donde se encuentra la clase Color:

```
import java.awt.*;
```

Para la captura de eventos mediante la interface ActionListener debemos importar el paquete java.awt.event:

```
import java.awt.event.*;
```

Declaramos la clase Formulario, heredamos de la clase JFrame e indicamos que implementaremos la interface ActionListener:

```
public class Formulario extends JFrame implements ActionListener{
```

Definimos un objeto de la clase JMenuBar (no importa que tan grande sea un menú de opciones solo se necesitará un solo objeto de esta clase):

```
    private JMenuBar mb;
```

Definimos un objeto de la clase JMenu (esta clase tiene por objeto desplegar un conjunto de objetos de tipo JMenuItem u otros objetos de tipo JMenu:

```
    private JMenu menu1;
```

Definimos tres objetos de la clase JMenuItem (estos son los que disparan eventos cuando el operador los selecciona:

```
    private JMenuItem mi1,mi2,mi3;
```

En el constructor creamos primero el objeto de la clase JMenuBar y lo asociamos al JFrame llamando al método setJMenuBar:

```
        mb=new JMenuBar();
        setJMenuBar(mb);
```

Seguidamente creamos un objeto de la clase JMenu, en el constructor pasamos el String que debe mostrar y asociamos dicho JMenu con el JMenuBar llamando al método add de objeto de tipo JMenuBar (Es decir el objeto de la clase JMenu colabora con la clase JMenuBar):

```
        menu1=new JMenu("Opciones");
        mb.add(menu1);
```

Ahora comenzamos a crear los objetos de la clase JMenuItem y los añadimos al objeto de la clase JMenu (también mediante la llamada al método addActionListener indicamos al JMenuItem que objeto procesará el clic):

```
        mi1=new JMenuItem("Rojo");
        mi1.addActionListener(this);
        menu1.add(mi1);
```

Lo mismo hacemos para los otros dos JMenuItem:

```
        mi2=new JMenuItem("Verde");
        mi2.addActionListener(this);
        menu1.add(mi2);
        mi3=new JMenuItem("Azul");
        mi3.addActionListener(this);
        menu1.add(mi3);
```

En el método actionPerformed primero obtenemos la referencia al panel asociado con el JFrame:

```
    public void actionPerformed(ActionEvent e) {
        Container f=this.getContentPane();
```

Luego mediante if verificamos cual de los tres JMenuItem fue seleccionado y a partir de esto llamamos al método setBackground del objeto de la clase Container):

```
        if (e.getSource()==mi1) {
            f.setBackground(new Color(255,0,0));
        }
        if (e.getSource()==mi2) {
            f.setBackground(new Color(0,255,0));
        }
        if (e.getSource()==mi3) {
            f.setBackground(new Color(0,0,255));
        }
    }
```

Problema 2:

Confeccionaremos un menú de opciones que contenga además del JMenu de la barra otros dos objetos de la clase JMenu que dependan del primero.
Uno debe mostrar dos JMenuItem que permitan modificar el tamaño del JFrame y el segundo también debe mostrar dos JMenuItem que permitan cambiar el color de fondo.

Programa:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener{
    private JMenuBar mb;
    private JMenu menu1,menu2,menu3;
    private JMenuItem mi1,mi2,mi3,mi4;
    public Formulario() {
        setLayout(null);
        mb=new JMenuBar();
        setJMenuBar(mb);
        menu1=new JMenu("Opciones");
        mb.add(menu1);
        menu2=new JMenu("Tamaño de la ventana");
        menu1.add(menu2);
        menu3=new JMenu("Color de fondo");
        menu1.add(menu3);
        mi1=new JMenuItem("640*480");
        menu2.add(mi1);
        mi1.addActionListener(this);
        mi2=new JMenuItem("1024*768");
        menu2.add(mi2);
        mi2.addActionListener(this);
        mi3=new JMenuItem("Rojo");
        menu3.add(mi3);
        mi3.addActionListener(this);
        mi4=new JMenuItem("Verde");
        menu3.add(mi4);
        mi4.addActionListener(this);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==mi1) {
            setSize(640,480);
        }
```

```
        if (e.getSource()==mi2) {
            setSize(1024,768);
        }
        if (e.getSource()==mi3) {
            getContentPane().setBackground(new Color(255,0,0));
        }
        if (e.getSource()==mi4) {
            getContentPane().setBackground(new Color(0,255,0));
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,300,200);
        formulario1.setVisible(true);
    }
}
```

Definimos un objeto de la clase JMenuBar, 3 objetos de la clase JMenu y finalmente 4 objetos de la clase JMenuItem:

```
    private JMenuBar mb;
    private JMenu menu1,menu2,menu3;
    private JMenuItem mi1,mi2,mi3,mi4;
```

Es importante notar el orden de creación de los objetos y como los relacionamos unos con otros.
Primero creamos el JMenuBar y lo asociamos con el JFrame:

```
        mb=new JMenuBar();
        setJMenuBar(mb);
```

Creamos el primer JMenu y lo pasamos como parámetro al JMenuBar mediante el método add:

```
        menu1=new JMenu("Opciones");
        mb.add(menu1);
```

Ahora creamos el segundo objeto de la clase JMenu y lo asociamos con el primer JMenu creado:

```
        menu2=new JMenu("Tamaño de la ventana");
        menu1.add(menu2);
```

En forma similar creamos el tercer objeto de la clase JMenu y lo asociamos con el primer JMenu creado:

```
        menu3=new JMenu("Color de fondo");
        menu1.add(menu3);
```

Finalmente comenzamos a crear los objetos de la clase JMenuItem y los dos primeros los asociamos con el segundo JMenu:

```
        mi1=new JMenuItem("640*480");
        menu2.add(mi1);
```

```
mi1.addActionListener(this);
mi2=new JMenuItem("1024*768");
menu2.add(mi2);
mi2.addActionListener(this);
```

También hacemos lo mismo con los otros dos objetos de tipo JMenuItem pero ahora los asociamos con el tercer JMenu:

```
mi3=new JMenuItem("Rojo");
menu3.add(mi3);
mi3.addActionListener(this);
mi4=new JMenuItem("Verde");
menu3.add(mi4);
mi4.addActionListener(this);
```

En el método actionPerformed si se presiona el mi1 procedemos a redimensionar el JFrame llamando al método setSize y le pasamos dos parámetros que representan el nuevo ancho y alto de la ventana:

```
if (e.getSource()==mi1) {
    setSize(640,480);
}
```

De forma similar si se presiona el segundo JMenuItem cambiamos el tamaño de la ventana a 1024 píxeles por 768:

```
if (e.getSource()==mi2) {
    setSize(1024,768);
}
```

Para cambiar de color de forma similar al problema anterior mediante el método getContentPane obtenemos la referencia al objeto de la clase Container y llamamos al método setBackground para fijar un nuevo color de fondo:

```
if (e.getSource()==mi3) {
    getContentPane().setBackground(new Color(255,0,0));
}
if (e.getSource()==mi4) {
    getContentPane().setBackground(new Color(0,255,0));
}
```

Problemas propuestos

1. Mediante dos controles de tipo JTextField permitir el ingreso de dos números. Crear un menú que contenga una opción que redimensione el JFrame con los valores ingresados por teclado. Finalmente disponer otra opción que finalice el programa (Finalizamos un programa java llamando al método exit de la clase System: System.exit(0))

Swing - JCheckBox

El control JCheckBox permite implementar un cuadro de selección (básicamente un botón de dos estados)

Problema 1:

Confeccionar un programa que muestre 3 objetos de la clase JCheckBox con etiquetas de tres idiomas. Cuando se lo selecciona mostrar en el título del JFrame todos los JCheckBox seleccionados hasta el momento.

Programa:

```
import javax.swing.*;
import javax.swing.event.*;
public class Formulario extends JFrame implements ChangeListener{
    private JCheckBox check1,check2,check3;
    public Formulario() {
        setLayout(null);
        check1=new JCheckBox("Inglés");
        check1.setBounds(10,10,150,30);
        check1.addChangeListener(this);
        add(check1);
        check2=new JCheckBox("Francés");
        check2.setBounds(10,50,150,30);
        check2.addChangeListener(this);
        add(check2);
        check3=new JCheckBox("Alemán");
        check3.setBounds(10,90,150,30);
        check3.addChangeListener(this);
        add(check3);
    }

    public void stateChanged(ChangeEvent e){
        String cad="";
        if (check1.isSelected()==true) {
            cad=cad+"Inglés-";
        }
        if (check2.isSelected()==true) {
            cad=cad+"Francés-";
        }
        if (check3.isSelected()==true) {
```

```
            cad=cad+"Alemán-";
        }
        setTitle(cad);
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,300,200);
        formulario1.setVisible(true);
    }
}
```

Lo primero y más importante que tenemos que notar que para capturar el cambio de estado del JCheckBox hay que implementar la interface ChangeListener que se encuentra en el paquete:

```
import javax.swing.event.*;
```
y no en el paquete:
```
import java.awt.event.*
```

Cuando declaramos la clase JFrame indicamos que implementaremos la interface ChangeListener:

```
public class Formulario extends JFrame implements ChangeListener{
```

Definimos tres objetos de la clase JCheckBox:

```
    private JCheckBox check1,check2,check3;
```

En el constructor creamos cada uno de los objetos de la clase JCheckBox y llamamos al método addChangeListener indicando quien procesará el evento de cambio de estado:

```
        check1=new JCheckBox("Inglés");
        check1.setBounds(10,10,150,30);
        check1.addChangeListener(this);
        add(check1);
```

El método que debemos implementar de la interface ChangeListener es:

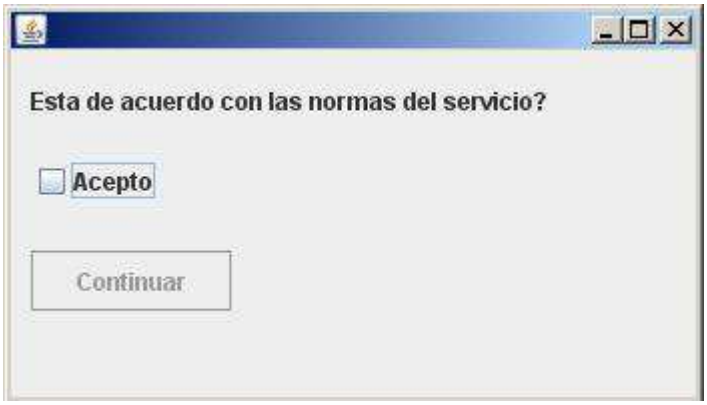
```
    public void stateChanged(ChangeEvent e){
```

En este mediante tres if verificamos el estado de cada JCheckBox y concatenamos los String con los idiomas seleccionados:

```
        String cad="";
        if (check1.isSelected()==true) {
            cad=cad+"Inglés-";
        }
        if (check2.isSelected()==true) {
            cad=cad+"Francés-";
        }
        if (check3.isSelected()==true) {
            cad=cad+"Alemán-";
        }
        setTitle(cad);
```

Problema 2:

Disponer un control JLabel que muestre el siguiente mensaje: "Esta de acuerdo con las normas del servicio?", luego un JCheckBox y finalmente un objeto de tipo JButton desactivo. Cuando se tilde el JCheckBox debemos activar el botón.



Programa:

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
public class Formulario extends JFrame implements ActionListener,
ChangeListener{
    private JLabel label1;
    private JCheckBox check1;
    private JButton boton1;
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Esta de acuerdo con las normas del
servicio?");
        label1.setBounds(10,10,400,30);
        add(label1);
        check1=new JCheckBox("Acepto");
        check1.setBounds(10,50,100,30);
        check1.addChangeListener(this);
        add(check1);
        boton1=new JButton("Continuar");
        boton1.setBounds(10,100,100,30);
        add(boton1);
        boton1.addActionListener(this);
        boton1.setEnabled(false);
    }

    public void stateChanged(ChangeEvent e) {
        if (check1.isSelected()==true) {
            boton1.setEnabled(true);
        } else {
            boton1.setEnabled(false);
        }
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==boton1) {
            System.exit(0);
        }
```

```
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,350,200);
        formulario1.setVisible(true);
    }
}
```

Importamos los paquetes donde se encuentran las interfaces para captura de eventos de objetos de tipo JButton y JCheckBox:

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

También importamos el paquete donde están definidas las clase JFrame, JButton y JCheckBox:

```
import javax.swing.*;
```

Como debemos implementar dos interfaces las debemos enumerar después de la palabra implements separadas por coma:

public class Formulario extends JFrame implements ActionListener, ChangeListener{

Definimos los tres objetos:

```
    private JLabel label1;
    private JCheckBox check1;
    private JButton boton1;
```

En el constructor creamos el objeto de tipo JLabel:

```
    public Formulario() {
        setLayout(null);
        label1=new JLabel("Esta de acuerdo con las normas del
servicio?");
        label1.setBounds(10,10,400,30);
        add(label1);
```

El objeto de tipo JCheckBox:

```
        check1=new JCheckBox("Acepto");
        check1.setBounds(10,50,100,30);
        check1.addChangeListener(this);
        add(check1);
```

y también creamos el objeto de tipo JButton y llamando al método setEnabled con un valor false luego el botón aparece desactivo:

```
        boton1=new JButton("Continuar");
        boton1.setBounds(10,100,100,30);
        add(boton1);
        boton1.addActionListener(this);
        boton1.setEnabled(false);
```

Cuando se cambia el estado del control JCheckBox se ejecuta el método stateChanged donde verificamos si está seleccionado procediendo a activar el botón en caso negativo lo desactivamos:

```
public void stateChanged(ChangeEvent e) {
    if (check1.isSelected()==true) {
        boton1.setEnabled(true);
    } else {
        boton1.setEnabled(false);
    }
}
```

El método actionPerformed se ejecuta cuando se presiona el objeto de tipo JButton (debe estar activo para poder presionarlo):

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource()==boton1) {
        System.exit(0);
    }
}
```

Problemas propuestos

1. Disponer tres objetos de la clase JCheckBox con nombres de navegadores web. Cuando se presione un botón mostrar en el título del JFrame los programas seleccionados.

Swing - JRadioButton

Otro control visual muy común es el JRadioButton que normalmente se muestran un conjunto de JRadioButton y permiten la selección de solo uno de ellos. Se los debe agrupar para que actúen en conjunto, es decir cuando se selecciona uno automáticamente se deben deseleccionar los otros.

Problema 1:

Confeccionar un programa que muestre 3 objetos de la clase JRadioButton que permitan configurar el ancho y alto del JFrame.

Programa:

```
import javax.swing.*;
import javax.swing.event.*;
public class Formulario extends JFrame implements ChangeListener{
    private JRadioButton radio1,radio2,radio3;
    private ButtonGroup bg;
    public Formulario() {
        setLayout(null);
        bg=new ButtonGroup();
        radio1=new JRadioButton("640*480");
        radio1.setBounds(10,20,100,30);
        radio1.addChangeListener(this);
        add(radio1);
        bg.add(radio1);
        radio2=new JRadioButton("800*600");
        radio2.setBounds(10,70,100,30);
        radio2.addChangeListener(this);
        add(radio2);
        bg.add(radio2);
        radio3=new JRadioButton("1024*768");
        radio3.setBounds(10,120,100,30);
        radio3.addChangeListener(this);
        add(radio3);
        bg.add(radio3);
    }
```

```
    public void stateChanged(ChangeEvent e) {
        if (radio1.isSelected()) {
            setSize(640,480);
        }
        if (radio2.isSelected()) {
            setSize(800,600);
        }
        if (radio3.isSelected()) {
            setSize(1024,768);
        }
    }

    public static void main(String[] ar) {
        Formulario formulario1=new Formulario();
        formulario1.setBounds(0,0,350,230);
        formulario1.setVisible(true);
    }
}
```

Importamos los dos paquetes donde están definidas las clases e interfaces para la captura de eventos:

```
import javax.swing.*;
import javax.swing.event.*;
```

Heredamos de la clase JFrame e implementamos la interface ChangeListener para capturar el cambio de selección de objeto de tipo JRadioButton:

```
public class Formulario extends JFrame implements ChangeListener{
```

Definimos tres objetos de la clase JRadioButton y uno de tipo ButtonGroup:

```
    private JRadioButton radio1,radio2,radio3;
    private ButtonGroup bg;
```

En el constructor creamos primero el objeto de la clase ButtonGroup:

```
        bg=new ButtonGroup();
```

Creamos seguidamente el objeto de la clase JRadioButton, definimos su ubicación, llamamos al método addChangeListener para informar que objeto capturará el evento y finalmente añadimos el objeto JRadioButton al JFrame y al ButtonGroup:

```
        radio1=new JRadioButton("640*480");
        radio1.setBounds(10,20,100,30);
        radio1.addChangeListener(this);
        add(radio1);
        bg.add(radio1);
```

Exactamente hacemos lo mismo con los otros dos JRadioButton:

```
        radio2=new JRadioButton("800*600");
        radio2.setBounds(10,70,100,30);
        radio2.addChangeListener(this);
        add(radio2);
```

```
        bg.add(radio2);
        radio3=new JRadioButton("1024*768");
        radio3.setBounds(10,120,100,30);
        radio3.addChangeListener(this);
        add(radio3);
        bg.add(radio3);
```

En el método stateChanged verificamos cual de los tres JRadioButton está seleccionado y procedemos a redimensionar el JFrame:

```
    public void stateChanged(ChangeEvent e) {
        if (radio1.isSelected()) {
            setSize(640,480);
        }
        if (radio2.isSelected()) {
            setSize(800,600);
        }
        if (radio3.isSelected()) {
            setSize(1024,768);
        }
    }
```

Problemas propuestos

1. Permitir el ingreso de dos números en controles de tipo JTextField y mediante dos controles de tipo JRadioButton permitir seleccionar si queremos sumarlos o restarlos. Al presionar un botón mostrar en el título del JFrame el resultado de la operación.