Programación II

Carla TARAMASCO

Profesora e Investigadora

DECOM & CNRS

mail : alumnos_uv@yahoo.cl

3 septembre 2013

Programación Gráfica

- Ventana
- Clic en una ventana
- Primer componente : un botón
- Componentes
- Primer dibujo
- Manejo de dimensiones

Programación Gráfica

Para crear una ventana gráfica disponemos en el paquete llamado *javax.swing*, de una clase clase *JFrame* que posee un constructor sin argumentos.
JFrame ven = new JFrame() ;
Esto crea un objeto del tipo JFrame dejando su referencia en ven.

- Para hacer visible esta ventana : *ven.setVisible(true) ;*
- Para darle el tamaño a la ventana : *ven.setSize(300,150) ;* , altura de 150 pixeles y largo de 300.
- Para mostrar un texto en la barra de titulo : *ven.setTitle("Mi primera ventana") ;*

Cree un programa que cree una ventana, la visualice, le de un tamaño y le agregue un titulo.

Programación Gráfica

```
import javax.swing.*;
public class Grafica {
        public static void main (String arg []){
                JFrame ven = new JFrame();
                ven.setSize(300,150);
                ven.setTitle("Mi primera ventana");
                ven.setVisible(true);
        }}
```

Programación Gráfica

Ventana

El usuario podrá

- cambiar el tamaño de la ventana
- desplazar la ventana
- reducirla
- maximizarla

Programación Gráfica

Parar el programa

Una vez que el método main a llegado a su final la ventana gráfica sigue abierta. Dado que un programa java puede contar con diversos procesos independientes llamados *threads*.
Aqui el *threads* principal corresponde al metodo main y un *threads* utilizador lanza la ventana grafica. Cuando ha terminado el metodo main solo el *threads* principal se interrumpe.
Cerrar la ventana grafica no finalizara el *threads*. Mas adelante veremos como finalizar este *threads*.
Use : *CTL C* para finalizar.

Programación Gráfica

Creación de una ventana personalizada

Anteriormente hemos creado un objeto JFrame y utilizado sus funcionalidades.
Para personalizar la ventana y asociarle campos o funcionalidades suplementarias. Para eso será necesario definir una clase derivada de JFrame y crear un objeto de este nuevo tipo.
Transforme el programa desarrollado anteriormente (sin agregar aun nuevas funcionalidades) creando una clase derivada de JFrame.

Programación Gráfica

a la clase base

```
import javax.swing.*;
class MiVentana extends JFrame{
        public MiVentana() //constructor
        {       setTitle("Mi primera ventana");
                setSize(300,150);}}

public class Grafica1 {
        public static void main (String arg []){
                JFrame ven = new MiVentana();
                ven.setVisible(true);
        }}
```

Programación Gráfica

Acciones sobre las características de una ventana

Naturalmente será posible cambiar los atributos de la ventana por ejemplo : a partir de datos ingresados por el usuario. Otros métodos :

1. Posición de la ventana y sus dimensiones : *setBounds(10,40,300,200) ;* con esta instrucción la esquina superior izquierda de la ventana esta en el pixel $10, 40$ y sus dimensiones son $300 * 200$.
2. Modificar el color de fondo *setBackground (Color.red) (import java.awt.Color ;)*
3. Obtener el tamaño actual *getsize()*

Programación Gráfica

Acciones sobre las características de una ventana

Realice un programa que cree ventanas con dimensiones y color definido por el usuario.
Cree una clase derivada de JFrame con un titulo y dimensiones iniciales y luego solicite al usuario que las redefina y que ingrese el color de fondo de la ventana.

Programación Gráfica

Acciones sobre las características de una ventana

```
import javax.swing.*;
import java.util.Scanner;
import java.awt.Color;

class MiVentana extends JFrame{
        public MiVentana() //constructor
        {       setTitle("Mi primera ventana");
                setSize(300,150);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}}
                //setDefaultCloseOperation(DISPOSE_ON_CLOSE);}}
```

Programación Gráfica

Acciones sobre las características de una ventana

```
import javax.swing.*;
import java.util.Scanner;
import java.awt.Color;

class MiVentana extends JFrame{
        public MiVentana() //constructor
        {       setTitle("Mi primera ventana");
                setSize(300,150);
                //setDefaultCloseOperation(EXIT_ON_CLOSE);}}
//setDefaultCloseOperation(DISPOSE_ON_CLOSE);}}


public class Grafica3 {
        public static void main (String arg []){
                //JFrame ven = new MiVentana();
                Scanner leer =new Scanner(System.in);

                while (true) {
                        JFrame ven = new MiVentana();
                        System.out.println("Ingrese el largo : ");
                        int largo = leer.nextInt();
                        System.out.println("Ingrese el ancho : ");
                        int ancho = leer.nextInt();
                        System.out.println("Elija entre el rojo, verde y azul
                                                para el color de fondo : ");
                        String color = leer.next();
```

Programación Gráfica

Acciones sobre las características de una ventana

```
System.out.println("Ingrese el nuevo titulo : (0 si desea salir) ");
String titulo = leer.next();
ven.setSize(largo,ancho);
ven.setTitle(titulo);
if (color.equals("rojo")){
ven.setBackground(Color.red);}
if (color.equals("verde")){
ven.setBackground(Color.green);}
if (color.equals("azul")){
ven.setBackground(Color.blue);}
ven.setVisible(true);
if (titulo.equals("0"))
break;  }}}
```

Programación Gráfica

Cerrar ventanas

Si el usuario cierra la ventana gráfica simplemente la deja invisible, semejante a decir *setVisible(false)*. Podríamos usar el método *setDefaultCloseOperation* con alguno de los argumentos siguientes :

- *DO_NOTHING_ON_CLOSE* : no hacer nada.
- *HIDE_ON_CLOSE* : ocultar la ventana (por defecto).
- *DISPOSE_ON_CLOSE* : destruir el objeto ventana.
- *EXIT_ON_CLOSE* : sale de la aplicación usando el metodo system exit.

Programación Gráfica

Clic en una ventana

La programación gráfica se basa en eventos que son creados por componentes que se introducen en la ventana gráfica como por ejemplo : menú, botones, ...
En java todos los eventos tienen una fuente, es decir, un objeto botón, menú, ventana, ...otro. Por ahora será la ventana principal.
Para tratar un evento se le asocia a la fuente un objeto, la clase implementa una interfaz particular que corresponde a una categoría de eventos. Decimos así que este objeto es un escuchador o *listener* de esa categoría de eventos. Los escuchadores son interfaces de java.

Programación Gráfica

Clic en una ventana

Por ejemplo : Existe una categoría *evento mouse* que se puede tratar con un objeto de esa clase implementando la interfaz *MouseListener*. Esta tiene cinco métodos :

1. *mousePressed*
2. *mouseReleased*
3. *mouseEntered*
4. *mouseExited*
5. *mouseClicked*

Programación Gráfica

Clic en una ventana

Una clase susceptible de instanciar un objeto escuchador de los diferentes eventos deberá corresponder al esquema siguiente :

```
class Ventana extends JFrame implements MouseListener
public Ventana (){
setTitle("Ventana con click");
setBounds(10,20,300,200);
```

Dado que a interfaz *MouseListener* tiene 5 métodos será necesario redefinir todos los métodos aunque se dejen vacíos.

Programación Gráfica

Click en una ventana

Para tratar un click en la ventana, será necesario redefinir no vacío el metodo *mouseCliked*. A cada método se le asociar un objeto del tipo *MouseEvent* y a la clase se le asocia un objeto del tipo *MouseListener* usando *addMouseListener(objetListener)*, *objetListener* es un objeto de una clase de tipo *MouseListener* Podemos incluir este objeto en el constructor de la siguiente manera :

```
class VentanaMia extends JFrame implements MouseListener{
        public VentanaMia (){
                setTitle("Ventana con click");
                setBounds(10,20,300,200);
                addMouseListener(this); //la ventana sera su propio
                //escuchador de eventos Mouse
                setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
```

Programación Gráfica

Click en una ventana

Haga un programa que escriba un mensaje en consola cada vez que el usuario haga click en la ventana.
Para esto debe importar el paquete *java.awt.event* que gestionara los eventos.
El método *mouseClicked* debe ser publico dado que una clase no puede restringir el acceso de métodos ya implementados.

Programación Gráfica

Click en una ventana

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener

class VentanaMia extends JFrame implements MouseListener{
public VentanaMia (){
setTitle("Ventana con click");
setBounds(10,20,300,200);
addMouseListener(this); //la ventana sera su propio
//escuchador de eventos Mouse
setDefaultCloseOperation(EXIT_ON_CLOSE);
}
public void mouseClicked(MouseEvent ev){
//metodo para click
System.out.println("click en la ventana");
}
public void mousePressed(MouseEvent ev){}
public void mouseReleased(MouseEvent ev){}
public void mouseEntered(MouseEvent ev){}
public void mouseExited(MouseEvent ev){}
}
public class Ventana{
public static void main (String arg[]){
VentanaMia ven=new VentanaMia();
ven.setVisible(true);
}}
```

Programación Gráfica

Usar la información asociada al evento

El argumento del método *mouseClicked* es un objeto de tipo : *MouseEvent*. Esta clase corresponde a la categoria de eventos manejados por la interfaz *MouseListener*.
Java crea un objeto de esa clase automaticamente luego de un click y lo envia a escuchador deseado. Este contiene diversas informaciones, en particular las coordenadas del *mouse* que son accesibles a traves de los metodos *getX* y *getY*.
Adapte el programa precedente desplegando las coordenadas del *mouse* al hacer click en la ventana.

Programación Gráfica

Click en una ventana

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener

class VentanaMia extends JFrame implements MouseListener{
public VentanaMia (){
setTitle("Ventana con click");
setBounds(10,20,300,200);
addMouseListener(this); //la ventana sera su propio
//escuchador de eventos Mouse
setDefaultCloseOperation(EXIT_ON_CLOSE);
}
public void mouseClicked(MouseEvent ev){
//metodo para click
        int x=ev.getX();
        int y=ev.getY();

System.out.println("click en la ventana en las coordenadas " + x + "," + y );
}
public void mousePressed(MouseEvent ev){}
public void mouseReleased(MouseEvent ev){}
public void mouseEntered(MouseEvent ev){}
public void mouseExited(MouseEvent ev){}
}
public class Ventana1{
public static void main (String arg[]){
VentanaMia ven=new VentanaMia();
ven.setVisible(true);
}}
```

Programación Gráfica

Resumen gestión de eventos

Un evento generado por una fuente es tratado por otro objeto llamado escuchador asociado previamente a la fuente. El objeto escuchador podrá ser cualquier objeto incluso un objeto fuente. En particular a una categoria dada *Xxx* se le asocia siempre un objeto escuchador de eventos del tipo *XxxEvent* usando el metodo *addXxxListener*. Cada vez que una categoria dispone de varios metodos podemos :

- redefinir todos los metodos de la interfaz correspondiente *XxxListener* (*implements* debe figurar en la cabeza de la clase del escuchador), ciertos metodos podrán ser definidos vacíos.
- llamar a una clase derivada de una clase adaptador *XxxAdapter* y definir solo los métodos que nos interesan

Programación Gráfica

Primer componente : un botón

Para crear un botón usamos el constructor de la clase *JButton* :
JButton miBoton ;
miBoton = new JButton("Mi Primer Boton") ;
Un objeto del tipo *JFrame* esta formado de :

- una raiz
- un contenido
- un vidrio

En el contenido incluiremos los diferentes componentes.

Programación Gráfica

Primer componente : un botón

El metodo *getContentPane* de la clase *JFrame* referencia al contenido de tipo *Container*.
Container c = getContentPane() ;
El metodo *add* de la clase Container permitira agregar un componente a un objeto de ese tipo.
Para agregar el botón al contenido del objeto de tipo Container debemos :
c.add(miBoton) ;
Una manera condensada sería :
getContentPane().add(miBoton) ;

Programación Gráfica

Primer componente : un botón

El constructor de la ventana con un botón sería :

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener

class VenBoton extends JFrame {
public VenBoton (){
        setTitle("Primer Boton");
        setSize(300,200);
        miBoton=new JButton ("Prueba");
        getContentPane().add(miBoton);}

        JButton miBoton;}
```

Programación Gráfica

Primer componente : un botón

Un botón es visible por defecto. Al mostrar la ventana vemos que el botón esta pero ocupa todo el espacio disponible. La disposición de componentes en una ventana la organiza un gestor de disposición (*Layout Manager*).
Existen diversos gestores (en forma de clase). Por defecto Java usa *BorderLayout* con el cual, si no hay otra información, un componente ocupa toda la ventana.
Un gestor interesante es *FlowLayout* que dispone de diversos componentes mostrandolos como texto, uno después de otro en una linea y luego en la siguiente.

Programación Gráfica

Primer componente : un botón

Para elegir el gestor simplemente se aplica el metodo *setLayout* al objeto contenido de la ventana (al igual que el componente botón).
Para obtener el gestor :
getContentPane().setLayout(new Layout()) ;
Escriba un programa que cree un botón en la ventana gráfica.

Programación Gráfica

Primer componente : un botón

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener
import java.awt.*; //FlowLayout y Container
class VeBoton extends JFrame {
        public VeBoton (){
                setTitle("Mi primer boton");
                setBounds(150,360,300,200);
                //setSize(300,200);
                setBackground(Color.red);
                miBoton = new JButton("prueba");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
}
        private JButton miBoton;}
public class MainBoton{
public static void main (String arg[]){
VeBoton venB=new VeBoton();
venB.setVisible(true);}}
```

Programación Gráfica

Acción sobre un botón : un evento

De las misma manera que con la ventana debemos :

- Crea un escuchador que será un objeto de una clase que implemente la interfaz *ActionListener*. Esta interfaz tiene 1 método en la categoría *Action* llamado *actionPerformed*.
- Asociar el escuchador al botón por medio del metodo *addActionListener*

Modifique el programa anterior para que muestre un mensaje cada vez que haga click sobre el botón.

Programación Gráfica

Acción sobre un botón : un evento

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener
import java.awt.*; //FlowLayout y Container
class VeBotonClick extends JFrame implements ActionListener {
        public VeBotonClick (){
                setTitle("Mi primer boton");
                setBounds(150,360,300,200);
                //setSize(300,200);
                setBackground(Color.red);
                miBoton = new JButton("prueba");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton);
                miBoton.addActionListener(this);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
}
        public void actionPerformed(ActionEvent ev){
                System.out.println("Accion sobre el boton Prueba");}

        private JButton miBoton;}
public class MainBotonClick{
public static void main (String arg[]){
VeBotonClick venBC=new VeBotonClick ();
venBC.setVisible(true);}}
```

Programación Gráfica

Resumen : Acción en un botón

- Para actuar sobre un componente a partir del teclado este debe estar seleccionado. Solo un componente puede estar seleccionado a la vez. Incluso una simple acción en la barra de espacio mostrara el mensaje.
- La categoria de eventos *Action* tiene un solo metodo *actionPerformed*.

Programación Gráfica

Gestión de multiples componentes

Si queremos agregar nuevos botones usando el gestor *FlowLayout*, los botones se muestran secuencialmente en el orden que se agregan.
En lo que respecta a la gestión de acciones sobre los botones cada acción sobre un componente puede disponer de su propio objeto escuchador.
Para verificar la fuente de un evento puede usar los método *getSource* o *getActionCommand*
Modifique el código anterior para que cree 2 botones que realicen la misma acción (mostrar un mensaje en pantalla)

Programación Gráfica

Acciones sobre dos botones : un evento

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent y MouseListener
import java.awt.*; //FlowLayout y Container
class Ve2BotonClick extends JFrame implements ActionListener {
        public Ve2BotonClick (){
                setTitle("Dos botones");
                setBounds(150,360,300,200);
                //setSize(300,200);
                setBackground(Color.red);
                miBoton1 = new JButton("boton 1");
                miBoton2 = new JButton("boton 2");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton1);
                getContentPane().add(miBoton2);
                miBoton1.addActionListener(this);
                miBoton2.addActionListener(this);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}

        public void actionPerformed(ActionEvent ev){
                System.out.println("Accion sobre uno de los botones");}

        private JButton miBoton1, miBoton2;}
public class Main2BotonClick{
public static void main (String arg[]){
Ve2BotonClick venBC=new Ve2BotonClick ();
venBC.setVisible(true);}}
```

Programación Gráfica

Acciones sobre dos botones : dos eventos

Obtenga la fuente del evento usando *getSource*

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container

class Ve2Boton2Click extends JFrame implements ActionListener {
        public Ve2Boton2Click (){
                setTitle("Dos botones");
                setBounds(150,360,300,200);
                //setSize(300,200);
                setBackground(Color.red);
                miBoton1 = new JButton("boton 1");
                miBoton2 = new JButton("boton 2");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton1);
                getContentPane().add(miBoton2);
                miBoton1.addActionListener(this);
                miBoton2.addActionListener(this);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
```

Programación Gráfica

Acciones sobre dos botones : getSource

```
public void actionPerformed(ActionEvent ev){
if (ev.getSource()== miBoton1)
System.out.println("Accion sobre el boton 1");
if (ev.getSource()== miBoton2)
System.out.println("Accion sobre el boton 2");}

private JButton miBoton1, miBoton2;}
public class Main2Boton2Click{
        public static void main (String arg[]){
                Ve2Boton2Click venBC=new Ve2Boton2Click ();
                venBC.setVisible(true);}}
```

Programación Gráfica

Acciones sobre dos botones : getSource

El método *getSource* nos permite identificar la fuente de un evento (para esto debemos aplicarlo a cada botón).
El método *getActionCommand* presente solamente en la clase *ActionEvent* permite obtener la cadena de caracteres asociada a la fuente de un evento.
Aplique este método al programa anterior para capturar la cadena de caracteres asociada a los botones 1 y 2. Por defecto la cadena de comando asociada a un botón es su etiqueta.
Para imponer una cadena de comando debemos usar el metodo *setActionCommand*

Programación Gráfica

Acciones sobre dos botones : getActionCommand y setActionCommand

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container
class Ve2Boton2Click2 extends JFrame implements ActionListener {
        public Ve2Boton2Click2 (){
                setTitle("Dos botones");
                setBounds(150,360,300,200);
                setBackground(Color.red);
                miBoton1 = new JButton("Boton 1");
                miBoton2 = new JButton("Boton 2");
                miBoton1.setActionCommand("primero");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton1);
                getContentPane().add(miBoton2);
                miBoton1.addActionListener(this);
                miBoton2.addActionListener(this);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
        public void actionPerformed(ActionEvent ev){
                String nombre = ev.getActionCommand();
                System.out.println("Accion sobre el " + nombre);}
        private JButton miBoton1, miBoton2;}
public class Main2Boton2Click2{
        public static void main (String arg[]){
                Ve2Boton2Click2 venBC=new Ve2Boton2Click2 ();
                venBC.setVisible(true);}}
```

Programación Gráfica

Clase escuchador diferente de la ventana

Escuchador diferente de la ventana :

- una clase escuchador por botón
- una clase escuchador para todos los botones

Programación Gráfica

Una clase escuchador por botón

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container
class VentBotons extends JFrame {
        public VentBotons (){
                setTitle("Dos botones");
                setBounds(150,360,300,200);
                setBackground(Color.red);
                miBoton1 = new JButton("Boton 1");
                miBoton2 = new JButton("Boton 2");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton1);
                getContentPane().add(miBoton2);
                EscuchadorBoton1 escu1= new EscuchadorBoton1();
                EscuchadorBoton2 escu2= new EscuchadorBoton2();
                miBoton1.addActionListener(escu1);
                miBoton2.addActionListener(escu2);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
                private JButton miBoton1, miBoton2;}
```

Programación Gráfica

Una clase escuchador por botón

```
class EscuchadorBoton1 implements ActionListener {
        public void actionPerformed(ActionEvent ev){
                System.out.println("Accion sobre el boton 1");}}

class EscuchadorBoton2 implements ActionListener {
        public void actionPerformed(ActionEvent ev){
                System.out.println("Accion sobre el boton 2");}}

public class Main2Escuchador{
        public static void main (String arg[]){
                VentBotons venB=new VentBotons ();
                venB.setVisible(true);}}
```

Programación Gráfica

Una clase escuchador para los dos botones

Podemos disponer un solo método *actionPerformed* común a los dos botones. Para identificar el botón de la acción podemos usar *getActionCommand*.

Programación Gráfica

Una clase escuchador para los botones

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container
class VentBotons1 extends JFrame {
        public VentBotons1 (){
                setTitle("Dos botones");
                setBounds(150,360,300,200);
                setBackground(Color.red);
                miBoton1 = new JButton("Boton 1");
                miBoton2 = new JButton("Boton 2");
                getContentPane().setLayout(new FlowLayout());
                getContentPane().add(miBoton1);
                getContentPane().add(miBoton2);
                EscuchadorBoton escu1= new EscuchadorBoton();
                EscuchadorBoton escu2= new EscuchadorBoton();
                miBoton1.addActionListener(escu1);
                miBoton2.addActionListener(escu2);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
                private JButton miBoton1, miBoton2;}
class EscuchadorBoton implements ActionListener {
        public void actionPerformed(ActionEvent ev){
                String nombre=ev.getActionCommand();
                System.out.println("Accion sobre el " + nombre);}}
public class Main1Escuchador{
        public static void main (String arg[]){
                VentBotons1 venB=new VentBotons1 ();
                venB.setVisible(true);}}
```

Programación Gráfica

Dinámica de componentes

En los ejemplos precedentes los botones fueron creados al mismo tiempo que la ventana y quedaban visibles y activos. Como :

- crear un nuevo componente
- suprimir un componente
- desactivar un componente
- reactivar un componente

Programación Gráfica

Dinámica de componentes

1. Para **crear** un nuevo componente sabemos que debemos usar el metodo *add*, pero en el caso que la ventana ya este visible, será necesario decirle al gestor que recalcule la posición de los componentes en la ventana usando :
   - *revalidate* para el componente
   - *validate* para su contenido
2. Para **suprimir** un componente usamos el metodo *remove* y volvemos a llamar a *validate*.
3. Para **desactivar** usaremos el metodo *content.setEnabled (false)*, para **reactivar** usaremos *content.setEnabled (true)*, para saber si el componente **esta activo** usaremos *content.isEnabled ()*

Programación Gráfica

Dinámica de componentes : Crear botones

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container
class VentBotonDin extends JFrame {
        public VentBotonDin (){
                setTitle("Botones Dinamicos");
                setBounds(150,360,300,200);
                setBackground(Color.red);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                miBoton = new JButton("Crear Botones");
                contenido.add(miBoton);
                EscuchadorBoton escu= new EscuchadorBoton(contenido);
                miBoton.addActionListener(escu);
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
                private JButton miBoton;}
class EscuchadorBoton implements ActionListener {
        public EscuchadorBoton (Container contenido){
                this.contenido=contenido;}
        public void actionPerformed(ActionEvent ev){
                JButton nuevoBoton = new JButton("Boton");
                contenido.add(nuevoBoton);
                contenido.validate(); //para recalcular
        } private Container contenido;}
public class MainCrearBoton{
        public static void main (String arg[]){
                VentBotonDin venBD=new VentBotonDin ();
                venBD.setVisible(true);}}
```

Programación Gráfica

Dinámica de componentes

Desarrolle un programa que muestre un numero dado por el usuario de botones.
El usuario al hacer click sobre un botón lo desactiva.
Imprima :

- un mensaje en la consola para especificar cual botón desactivo
- un mensaje en la consola con el estado de los botones.

Programación Gráfica

Dinámica de componentes : Activar-Desactivar botones

```
import javax.swing.*; // JFrame
import java.awt.event.*; //MouseEvent, MouseListener y ActionEvent
import java.awt.*; //FlowLayout y Container
import java.util.Scanner;
class VentBotonDin1 extends JFrame implements ActionListener{
        public VentBotonDin1 (int nbotones){
                setTitle("Activar/Desactivar Botones");
                setBounds(150,360,300,200);
                setBackground(Color.red);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                tabBoton=new JButton[nbotones];
                for (int i=0;i<nbotones;i++){
                        tabBoton[i]=new JButton ("Boton"+i);
                        contenido.add(tabBoton[i]);
                        tabBoton[i].addActionListener(this);}
                setDefaultCloseOperation(EXIT_ON_CLOSE);}
```

Programación Gráfica

Dinámica de componentes : Activar-Desactivar botones

```
public void actionPerformed(ActionEvent ev){
System.out.print("Estado de los botones : ");
for (int i=0;i<tabBoton.length;i++)
System.out.print(tabBoton[i].isEnabled() + "  ");
System.out.println();
JButton fuente = (JButton) ev.getSource();
System.out.println("Ahora desactivamos el boton " + fuente.getActionCommand());
fuente.setEnabled(false);       }
private JButton tabBoton[];}
public class Main1CrearBoton2{
        public static void main (String arg[]){
                Scanner kyd=new Scanner(System.in);
                System.out.println("Ingrese la cantidad de botones");
                int nbotones=kyd.nextInt();
                VentBotonDin1 venBD=new VentBotonDin1(nbotones);
                venBD.setVisible(true);}}
```

Programación Gráfica

Mi primer dibujo

Para realizar un dibujo en una ventana lo mas recomendable será evitar dibujar directamente en la ventana (JFrame) y usar un panel, es decir, un objeto de la clase *JPanel*.
Los paneles pueden estar en un contenedor y ser a la vez contenedor de otros componentes (no como los botones que no pueden contener otros componentes).
Un panel es como una *subventana* sin titulo no bordes, es simplemente un rectángulo del color de la ventana.
Un panel no puede existir de manera autonoma (como una ventana) deberá ser asociado por el metodo *add* a un contenedor.

Programación Gráfica

Mi primer dibujo- Crear un panel

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
class MiVentana extends JFrame{
        public MiVentana (){ //constructor
      ....
        panel = new JPanel();
        getContentPane().add(panel);}
} private JPanel panel;}
```

Programación Gráfica

Mi primer dibujo- Crear un panel

Con el código anterior si bien creamos un panel no lo vemos ya que por defecto es del mismo color y tamaño que la ventana. Usando *setBackground() cambie el color del panel para que sea visible.*

Programación Gráfica

Mi primer dibujo- Crear un panel

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
class MiVentana extends JFrame {
        public MiVentana (){//constructor
                setTitle("Ventana = panel rojo");
                setSize(300,150);
                panel = new JPanel();
                panel.setBackground(Color.red);
                getContentPane().add(panel);}
 private JPanel panel;}
public class MainPanel0{
        public static void main (String args[]){
                MiVentana ven= new MiVentana();
                ven.setVisible(true);
        }}
```

Programación Gráfica

Mi primer dibujo - Dibujar en el panel

Para obtener un dibujo permanente en un componente es necesario redefinir el metodo *paintComponent* que será llamado automaticamente por Java cada vez que un componente deba ser redibujado.
Como debemos redefinir un metodo de la clase *JPanel* debemos crear el panel como un objeto de una clase derivada de *JPanel*.
La cabecera del metodo *paintComponent* a redefinir es :
void paintComponent (Graphics g)
La clase *Graphics* encapsula todas las informaciones y métodos para dibujar sobre un componente (color de fondo, color de la linea, estilo, tipos de letra, tamaño, ...).

Programación Gráfica

Mi primer dibujo- Dibujar una linea en el panel

Si queremos dibujar una linea en el panel debemos solamente llamar a metodo *drawLine* usando el objeto *g*.
g.drawLine(15,10,100,50)
Esta instrucción hace una linea de punto 15,10 al punto 15+100, 10+50.
Las coordenadas se expresan en pixeles relativas a la esquina superior izquierda del componente.

Programación Gráfica

Dibujar una linea en el panel

Para trabajar con el metodo *paintComponent* debemos llamarlos explicitamente de la clase ascendente *JPanel*.
super.paintComponent(g)
Desarrolle un programa que muestre una linea en un panel rojo que ocupa toda la ventana.

Programación Gráfica

Mi primer dibujo - Crear un panel

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
class MiVentana extends JFrame {
        public MiVentana (){//constructor
                setTitle("Dibujando una linea");
                setSize(300,150);
                panel = new Panel();
                panel.setBackground(Color.red);
                getContentPane().add(panel);}
 private JPanel panel;}
class Panel extends JPanel{
        public void paintComponent(Graphics g){
                super.paintComponent(g);
                g.drawLine(15,10,200,300);}}
public class MainPanel1{
        public static void main (String args[]){
                MiVentana ven= new MiVentana();
                ven.setVisible(true);}}
```

Programación Gráfica

Dibujar en el panel

Si se quiere dibujar y redibujar directamente en un panel se debe usar el metodo *repaint* que llama al método *paintComponent* actualizando el contenido del panel.
Dentro del gestor de contenido por defecto llamado *BorderLayout* (que usa todo el espacio para insertar un componente) existe la posibilidad de dejar los componente no solo al centro sino que tambien sobre los 4 bordes de la ventana.
Para eso solo será necesario en el metodo *add* un argumento *"North", "South","East" o "West"* .

Programación Gráfica

Dibujar en el panel

Desarrolle un programa que tenga 2 botones (uno en la parte de arriba y otro en la parte de abajo del contenido de la ventana).
El primer boton dibujará un circulo en un panel y el segundo dibujara en el mismo panel un rectángulo que remplazara eventualmente el circulo.
Al inicio del programa no se muestra nada en el panel.
El panel ocupara la parte libre del contenido (el centro).

Programación Gráfica

Mi primer dibujo - Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
class MiVentana extends JFrame implements ActionListener {
        public MiVentana (){//constructor
                setTitle("Dibujando en el panel");
                setSize(300,150);
                Container contenido = getContentPane();
                //creo el panel para el dibujo
                panel = new Panel();
                panel.setBackground(Color.cyan);
                contenido.add(panel);
                //crear el boton "rectangulo"
                rectangulo = new JButton("Rectangulo");
                contenido.add(rectangulo, "West");
                rectangulo.addActionListener(this);
                //crear el boton "ovalo"
                ovalo = new JButton("Ovalo");
                contenido.add(ovalo, "East");
                ovalo.addActionListener(this);}
```

Programación Gráfica

Mi primer dibujo - Ejercicio

```
public void actionPerformed(ActionEvent ev){
if (ev.getSource()==rectangulo)
panel.setRectangulo();
if (ev.getSource()==ovalo)
panel.setOvalo();
panel.repaint();}
private Panel panel;
private JButton rectangulo, ovalo;}

class Panel extends JPanel{
        public void paintComponent(Graphics g){
                super.paintComponent(g);
                if (ovalo) g.drawOval(80,20,120,60);
                if (rectangulo) g.drawRect(80,20,120,60);}
        public void setOvalo(){ovalo=true;rectangulo=false;}
        public void setRectangulo(){ovalo=false;rectangulo=true;}
        private boolean rectangulo=false,ovalo=false;}

public class MainPanelDibujo{
        public static void main (String args[]){
                MiVentana ven= new MiVentana();
                ven.setVisible(true);}}
```

Programación Gráfica

Campos de texto

Los campos de texto son una zona rectangular en la cual el usuario puede entrar o modificar texto (de una sola linea).
Se obtienen instanciando un objeto *JTextField*.
Su constructor debe obligatoriamente indicar un tamaño (que indica el numero de caracteres, estos caracteres dependen del tipo de letra).
La construcción de campos de texto en un objeto del tipo JFrame :

```
JTextField entrada1,entrada2 ;
entrada1 = new JTextField(20) ;
entrada2 = new JTextField("texto inicial",15) ;
```

Programación Gráfica

Campos de texto

Para obtener la información que figura en un campo de texto usamos :
String cadena = entrada1.getText() ;
Para definir si el campo de texto será editable usaremos :
entrada1.setEditable(false) ;
entrada1.setEditable(true) ;
Para modificar durante la ejecución usarmos (como ya lo vimos anteriormente) :
entrada1.revalidate() ;

Programación Gráfica

Ejercicio - Campos de texto

Desarrolle un programa que le proponga al usuario un campo de texto y un botón con la etiqueta ”Copiar”.
Cada acción sobre el botón provoca la copia en un segundo campo de texto (no editable) del contenido del primer campo de texto.

Programación Gráfica

Ejercicio - Campos de texto

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;

class MiVentana extends JFrame implements ActionListener {
        public MiVentana (){//constructor
                setTitle("Campo de texto");
                setSize(300,150);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                entrada = new JTextField(20);
                contenido.add(entrada);
                entrada.addActionListener(this);
                boton = new JButton("Copiar");
                contenido.add(boton);
                boton.addActionListener(this);
                copia = new JTextField(20);
                copia.setEditable(false);
                contenido.add(copia);}
```

Programación Gráfica

Ejercicio - Campos de texto

```
public void actionPerformed(ActionEvent ev){
if (ev.getSource()==boton){
String texto = entrada.getText();
copia.setText(texto);
}}
private JTextField entrada, copia;
private JButton boton;}

public class MainPanelTexto{
        public static void main (String args[]){
                MiVentana ven= new MiVentana();
                ven.setVisible(true);}}
```

Programación Gráfica

Ejercicio - Campos de texto

Desarrolle un programa que repita las acciones realizadas en un campo de texto en otro. De alguna manera un campo de texto espejo.
Use un objeto del tipo *Document* para conservar el contenido del componente. Las modificaciones al objeto *Document* generan uno de los 3 eventos de la categoria *Document*. El escuchador sera *DocumentListener*
Encabezados de metodos :
public void insertUpdate(DocumentEvent ev)
public void removeUpdate(DocumentEvent ev)
public void changedUpdate(DocumentEvent ev)

Programación Gráfica

Ejercicio - Campos de texto

```
import javax.swing.*; // JFrame JPanel
import javax.swing.event.*;
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;


class MiVentanas extends JFrame implements DocumentListener {
        public MiVentanas (){//constructor
                setTitle("Espejo");
                setSize(300,150);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                entrada = new JTextField(20);
                contenido.add(entrada);

                entrada.getDocument().addDocumentListener(this);
                copia = new JTextField(20);
                copia.setEditable(true);
                contenido.add(copia);}
```

Programación Gráfica

Mi primer dibujo - Ejercicio

```
public void insertUpdate(DocumentEvent Document){
String texto = entrada.getText();
copia.setText(texto);
}
public void removeUpdate(DocumentEvent Document){
String texto = entrada.getText();
copia.setText(texto);
}
public void changedUpdate(DocumentEvent Document){}
private JTextField entrada, copia;}

public class MainPanel2Texto{
        public static void main (String args[]){
                MiVentanas ven= new MiVentanas();
                ven.setVisible(true);}}
```

Programación Gráfica

Listas

Las listas son componente que permiten elegir uno o varios valores en una lista predefinida. Para crear una lista se le entrega un arreglo de cadenas al constructor.
String [] colores ={*"rojo","verde","azul","blanco", "gris"*} .
JList lista = new JList(colores)
Para seleccionar un elemento usaremos :
lista.setSelectedIndex(2) ;
Selecciona el indice en la posición 2.

Programación Gráfica

Tipos de Listas

- SINGLE_SELECTION : selecciona un solo valor.
- SINGLE_INTERVAL_SELECTION : selecciona un grupo de valores seguidos.
- MULTIPLE_INTERVAL_SELECTION : selecciona una cantidad de valores. (Valor por defecto)

Para modificar la selección usaremos :
lista.setSelectionMode(SINGLE_SELECTION)

Programación Gráfica

Listas - Barra de desplazamiento

Para incluir una barra o panel de desplazamiento debemos :
JScrollPane barra = new JScrollPane(lista) ;
getContentPane().add(barra) ;
Por defecto mostrara ocho valores, si la lista tiene menos valores la barra no aparecera. Para hacerla visible sera necesario :
lista.setVisibleRowCount(3) ;

Programación Gráfica

Listas - Acceder a la información seleccionada

Para una lista con SINGLE_SELECTION el método *getSelectedValue* entrega el unico valor seleccionado (para los otros tipos de lista este método entrega solo el primer valor seleccionado).
El resultado es del tipo *Object* y no *String* por lo que será necesario aplicar una conversion de tipo explicita :
String cadena= (String) lista.getSelectedValue() ; Para obtener todos los valores seleccionados usaremos :
getSelectedValues()

Programación Gráfica

Listas

```
Object [] valores = lista.getSelectedValues();
for (int i =0; i<valores.length;i++)
        System.out.println((String) valores[i];)
```

Para conocer la posición del valor seleccionado
: *int* [] *getSelectIndices())*

Programación Gráfica

Listas - Eventos

A diferencia de otros componente las listas no generan eventos *Action*. El escuchador apropiado es : *ListSelectionListener* que dispone de un solo metodo :
public void valueChanged(ListSelectionEvent ev).
Para evitar la redundancia (que se produce en la fase de transición) en la selección usaremos : *getValuesIsAdjusting*

```
public void valueChanged(ListeSelectionEvent e)
{if (e!.getValuesIsAdjusting()){
//acceso a la informacion seleccionada
}}
```

Programación Gráfica

Listas - Ejercicio

Desarrolle un programa que cree una lista con el nombre de 5 colores. Al seleccionar uno y muchos colores debe imprimirlos en la consola.

Programación Gráfica

Listas - Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
import javax.swing.event.*;

class MiVentana extends JFrame implements ListSelectionListener {
        public MiVentana (){//constructor
                setTitle("Campo de texto");
                setSize(300,150);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                lista = new JList(colores);
                contenido.add(lista);
                lista.addListSelectionListener(this);}
```

Programación Gráfica

Listas - Ejercicio

```
public void valueChanged(ListSelectionEvent ev){
if (!ev.getValueIsAdjusting()){
System.out.println("Accion sobre la lista");
Object [] valores = lista.getSelectedValues();
for (int i =0;i<valores.length;i++)
System.out.println((String) valores[i]);
}}

private String [] colores ={"rojo","verde","amarillo", "azul", "otro"};
private JList lista;}

public class Lista{
        public static void main (String args[]){
                MiVentana ven= new MiVentana();
                ven.setVisible(true);}}
```

Programación Gráfica

Etiquetas

Un componente del tipo *JLabel* permite mostrar en un contenedor un texto no modificable por el usuario. *JLabel texto = new JLabel(”Texto Inicial ”)* Este componente no tiene borde ni color de fondo. Para modificar el texto de una etiqueta usuaremos : *texto.setText(”Nueva etiqueta”) ;*

Programación Gráfica

Ejercicio

Desarrolle un programa en el cual muestre en permanencia el numero de click realizados por un usuario sobre un botón.

Programación Gráfica

Listas - Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
import javax.swing.event.*;

class Ventana extends JFrame implements ActionListener {
        public Ventana() {
                setTitle("Combo");
                setSize(300,150);

                //ven.addActionListener(this);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                boton=new JButton("Contar");
                boton.addActionListener(this);
                contenido.add(boton);
                nClick=0;
                contar = new JLabel("Numero de click sobre el boton: "+nClick);
                contenido.add(contar);}
```

Programación Gráfica

Listas - Ejercicio

```
public void actionPerformed(ActionEvent ev) {
nClick ++;
contar.setText("Numero de click sobre el boton: "+nClick);}

private JButton boton;
private JLabel contar;
private int nClick;}

public class BotonClick{
        public static void main (String args[]){
                Ventana ven= new Ventana();
                ven.setVisible(true);}}
```

Programación Gráfica

Ejercicio

Desarrolle un programa con dos botones uno que incremente y el otro disminuya un contador. El contador debe estar en el contenido de la ventana use *JLabel* para mostrar el contador.

Programación Gráfica

Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
import javax.swing.event.*;

class Ejemplo extends JFrame implements ActionListener {
        public Ejemplo() {
                setTitle("Botones+acciones");
                setSize(300,150);

                //ven.addActionListener(this);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());

                botonMas = new JButton("mas");
                botonMenos = new JButton("menos");
                botonMenos.addActionListener(this);
                botonMas.addActionListener(this);
                contenido.add(botonMenos);
                contenido.add(botonMas);
                label = new JLabel("0", JLabel.CENTER);
                contenido.add(label);
```

Programación Gráfica

Ejercicio

```
}
public void actionPerformed(ActionEvent event) {
if(event.getSource() == botonMas)
{count++;}
else {count--;}
label.setText(Integer.toString(count));}

private int count = 0;
private JLabel label;
private JButton botonMenos;
private JButton botonMas;}
public class Exemple11{
        public static void main (String args[]){
                Ejemplo ven= new Ejemplo();
                ven.setVisible(true);}}
```

Programación Gráfica

Combo-box

Los ComboBox están asociados a un campo de texto no editable. Cuando el componente no esta seleccionado se muestra solo el campo de texto, al seleccionar el componente se despliega la lista.
El usuario puede elegir un valor en la lista. Por defecto el texto asociado a un comboBox no es editable.
Construcción de un comboBox :
String [] colores ={"rojo","verde","azul","blanco", "gris"} .
JComboBox= combo = new JComboBox(colores) ;
combo.setEditable(true) ;
combo.setMaximumRowCount(4) ;
combo.selectionIndex(2) ;

Programación Gráfica

Combo-box - acceso a los datos

El método *getSelectedItem* entrega los valores seleccionados, este metodo es como el método *getSelectedValues* de las listas.
Object valor = combo.getSelectedItem() ;
El método *getSelectedItem() ;* entrega el rango de valores seleccionado.

Programación Gráfica

Combo-box - eventos

Los comboBox generan evento *Action* al seleccionar un elemento en la lista. Ademas un comboBox genera eventos *Item* en cada modificación de selección lo que será tratado con el escuchador *ItemListener* que tiene un solo método :
public void itemStateChanged(ItemEvent e)
En un comboBox simple siempre tenemos dos eventos (suprimir la selección y nueva selección) ya sea en un campo de texto o en una lista.

Programación Gráfica

Combo-box - eventos

Desarrolle un programa que genere un comboBox con 5 colores que sea editable en el cual podamos incluir en curso de ejecución un nuevo color.

Programación Gráfica

Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
import javax.swing.event.*;

class Combo extends JFrame implements ActionListener, ItemListener {
        public Combo() {
                setTitle("Combo");
                setSize(300,150);

                //ven.addActionListener(this);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                combo=new JComboBox(colores);
                combo.setEditable(true);
                contenido.add(combo);
                combo.addActionListener(this);
                combo.addItemListener(this);
                }
```

Programación Gráfica

Ejercicio

```
public void actionPerformed(ActionEvent ev) {
System.out.println("accion combo");
Object valor = combo.getSelectedItem();
System.out.println((String) valor);}

public void itemStateChanged(ItemEvent ev) {
if (ev.getStateChange() == ItemEvent.SELECTED){
System.out.println("item combo");
Object valor = combo.getSelectedItem();
System.out.println((String) valor);}}
private String [] colores= {"rojo","verde","azul","negro","blanco"};
private JComboBox combo;}

public class EjemploCombo{
        public static void main (String args[]){
                Combo ven= new Combo();
                ven.setVisible(true);}}
```

Programación Gráfica

Combo-box - evolución dinamica de una lista de un combo

Los bomboBox disponen de método que permiten agregar un nuevo valor al final de la lista : *combo.addItem("naranja") ;*, agrega un elemento al final de la lista.
El método *addItemAt("naranja", 2) ;* agrega un elemento en la posición dos.
El método *removeItem("gris") ;* suprime un valor existente.

Programación Gráfica

Ejercicio

Modifique el programa anterior para que los valores ingresados por el usuario sean adjuntados a la lista de valores del comboBox. Para distinguir una selección de un ingreso de datos, usaremos el método *getSelectedIndex*

Programación Gráfica

Ejercicio

```
import javax.swing.*; // JFrame JPanel
import java.awt.*; //FlowLayout y Container
import java.awt.event.*;
import javax.swing.event.*;

class Combo extends JFrame implements ActionListener {
        public Combo() {
                setTitle("Combo dinamico");
                setSize(300,150);

                //ven.addActionListener(this);
                Container contenido = getContentPane();
                contenido.setLayout(new FlowLayout());
                combo=new JComboBox(colores);
                combo.setEditable(true);
                combo.setMaximumRowCount(8);
                contenido.add(combo);
                combo.addActionListener(this); }
```

Programación Gráfica

Ejercicio

```
public void actionPerformed(ActionEvent ev) {
System.out.println("accion combo - ");
Object valor = combo.getSelectedItem();
int rango = combo.getSelectedIndex();
if (rango==-1){
System.out.println("ingreso de un nuevo valor " + valor);
combo.addItem(valor);}
else {
System.out.println("selecciona un valor existente " + valor + "de rango " + rango);}
}

private String [] colores= {"rojo","verde","azul","negro","blanco"};
private JComboBox combo;}

public class EjemploComboDin{
        public static void main (String args[]){
                Combo ven= new Combo();
                ven.setVisible(true);}}
```

Programación Gráfica

Eventos Focus

Los eventos Focus son tratados por el escuchador :
FocusListener que tiene dos métodos :
public void focusGained(FocusEvent ev) ;
public void focusLost(FocusEvent ev) ;
En general, se tratara a la vez la validación, por ejemplo : en un campo de texto, y la perdida de foco.

Programación Gráfica

Ejercicio : Una aplicación completa

Desarrolle un programa que permita al usuario dibujar formas (rectángulo y/o ovalo) en una ventana, sus dimensiones y el color de fondo. Use combo, campo de texto y checkBox.
Las dimensiones, comunes a las diferentes formas, son ingresadas en campos de texto (los valores obtenidos de tipo String deben ser convertidos con *Integer.parseInt* a enteros).
El color de fondo sera elegido en un combo.
Para seleccionar la figura puede usar una lista o un *JCheckBox* : *ovalo = new JCheckBox("Ovalo") ;*.

Programación Gráfica

Ejercicio : Una aplicación completa

Se aconseja dibujar en un panel con el gestor *FlowLayout* para que quede en el centro, para facilitar las cosas ponga los otros componentes en un segundo panel usando el gestor por defecto para poner los controles abajo (*South*).
Debe considerar los eventos *Focus* para validar el contenido de los campos de texto.
Use los eventos acción para los campos de texto y los checkBox.
Use los eventos Item para el combo. Para la comunicación entre el objeto ventana y el objeto panel del dibujo use, y por lo cual cree, los métodos de modificación : *setLargo, setAncho, setOvalo, setRectangulo*

Programación Gráfica

Ejercicio

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

class MiVentana extends JFrame implements ActionListener,ItemListener, FocusListener{

static public final String[] nomColores=
{"cyan","rojo","verde","amarillo","azul"};
static public final Color[] colores=
        {Color.cyan,Color.red,Color.green,Color.yellow,Color.blue};
```

Programación Gráfica

Ejercicio

```
public MiVentana(){
setTitle("Figuras");
setSize(450,300);
Container contenido = getContentPane();
//panel para dibujo
panDib=new PanelDibujo();
contenido.add(panDib);
panDib.setBackground(Color.cyan);
//panel para comandos
panCom=new JPanel();
contenido.add(panCom,"South");
//elegir color
comboColorFondo=new JComboBox(nomColores);
panCom.add(comboColorFondo);
comboColorFondo.addItemListener(this);
//elegir dimensiones
JLabel dim=new JLabel("Dimensiones");
panCom.add(dim);
txtLargo = new JTextField("100",5);
txtLargo.addActionListener(this);
txtLargo.addFocusListener(this);
panCom.add(txtLargo);
txtAncho = new JTextField("100",5);
txtAncho.addActionListener(this);
txtAncho.addFocusListener(this);
panCom.add(txtAncho);
```

Programación Gráfica

Ejercicio

```
// elegir formas
ovalo = new JCheckBox("Ovalo");
panCom.add(ovalo);
ovalo.addActionListener(this);
rectangulo = new JCheckBox("Rectangulo");
panCom.add(rectangulo);
rectangulo.addActionListener(this);
setDefaultCloseOperation (EXIT_ON_CLOSE);}
```

Programación Gráfica

Ejercicio

```
public void actionPerformed(ActionEvent ev){
if (ev.getSource()==txtLargo) setLargo();
if (ev.getSource()==txtAncho) setAncho();
if (ev.getSource()==ovalo) panDib.setOvalo(ovalo.isSelected());
if (ev.getSource()==rectangulo) panDib.setRectangulo(rectangulo.isSelected());
panDib.repaint();}

public void focusLost (FocusEvent e){
if (e.getSource() == txtLargo){setLargo();
System.out.println("perdida de foco en largo");
panDib.repaint();}
if (e.getSource() == txtAncho){setAncho();
System.out.println("perdida de foco en ancho");
panDib.repaint();}}

public void focusGained (FocusEvent e){}
```

Programación Gráfica

Ejercicio

```
private void setLargo(){
String cad = txtLargo.getText();
System.out.println("largo" + cad);
panDib.setLargo(Integer.parseInt(cad));}

private void setAncho(){
String cad = txtAncho.getText();
System.out.println("ancho" + cad);
panDib.setAncho(Integer.parseInt(cad));}

public void itemStateChanged(ItemEvent e){
String color=(String) comboColorFondo.getSelectedItem();
panDib.setColor(color);}

private PanelDibujo panDib;
private JPanel panCom;
private JComboBox comboColorFondo;
private JTextField txtAncho, txtLargo;
private JCheckBox ovalo, rectangulo;
}
```

Programación Gráfica

Ejercicio

```
class PanelDibujo extends JPanel{
        public void paintComponent(Graphics g){
                super.paintComponent(g);
                if(ovalo) g.drawOval(10,10,10+largo,10+ancho);
                if(rectangulo) g.drawRect(10,10,10+largo,10+ancho);}
        public void setRectangulo (boolean b){rectangulo=b;}
        public void setOvalo (boolean b){ovalo=b;}
        public void setLargo (int l){largo=l;}
        public void setAncho (int a){ancho=a;}
        public void setColor (String c){
                for (int i=0;i<MiVentana.nomColores.length;i++)
                        if(c==MiVentana.nomColores[i])
                                setBackground(MiVentana.colores[i]);}
        private boolean rectangulo = false,ovalo=false;
        private int largo=50, ancho=50;}
public class Final {
        public static void main (String arg []){
                MiVentana ven=new MiVentana();
                ven.setVisible(true);
        }}
```