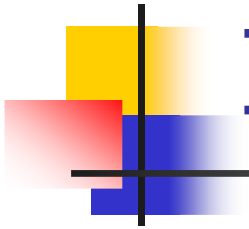
Interfaz Gráfica de Usuario (GUI)

Unidad: 2
Laboratorio de Programación

Universidad Nacional de la Patagonia Austral
Unidad Académica Río Gallegos



Indice

- Los administradores de diseño (Layouts)
- Características y ventajas
- Tipos de Layouts: FlowLayout, BorderLayout, GridLayout
- Ejemplos
- Gestión de eventos
- Modelo: fuentes y escuchadores
- Funcionamiento
- Ejemplos
- Interfaces
- Adapters

Los administradores de diseño

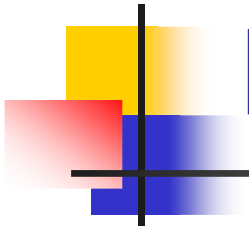
Layout Managers

Layout Managers: características

- Estos administradores, determinan el tamaño y posición del componente en el contenedor.
- Cuando el contenedor (container) necesita posicionar un componente, invoca al layout manager para que este lo haga.

Layout Managers

- Ventajas
  - Permiten representar de una manera ordenada los componentes en la pantalla.
  - Cuando usamos los administradores no nos tenemos que preocupar, por ejemplo, de que se redimensione una ventana, ya que el propio administrador se encargará de reorganizar los componentes en el Container.



Layout Managers

- Tipos de Layouts
  - FlowLayout
  - BorderLayout
  - GridLayout
  - CardLayout
  - GridBagLayout

Layout Managers

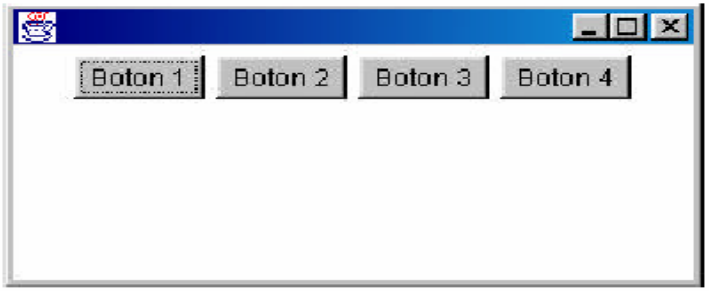
- Como definirlos?
  - Para utilizar un layout manager específico, se usa el método de la clase Container:
    - setLayout(AdministradorDeDiseño)
  - Por ejemplo:
    - setLayout(new BorderLayout());
  - Si no se especifica un layout manager se usa el default.
    - Para la clase Panel (superclase de la clase Applet) el layout manager por default es: FlowLayout.
    - Para la clase Frame, el layout manager por default es: BorderLayout.

Flow Layout

- Es el que tienen los paneles por defecto.
- Los objetos se van colocando en filas en el mismo orden en que se añadieron al contenedor.
- Cuando se llena una fila se pasa a la siguiente.
- Tiene tres posibles constructores:
  - **FlowLayout();.**
  - **FlowLayout(FlowLayout.LEFT[RIGTH][CENTER]);**
    - Crea el layout sin añadirle los componentes, con los bordes de unos pegados a otros
  - **FlowLayout(FlowLayout.LEFT, gap_horizontal, gap_vertical);**
    - Indica la alineación de los componentes: a la izquierda, derecha o centro con espacio entre los componentes.

Flow Layout

- Ejemplo:



Código:

```
import java.awt.Frame;
import java.awt.Button;
import java.awt.FlowLayout;
public class VentanaFlowLayout extends Frame {
   public VentanaFlowLayout() {
         Button boton1 = new Button("Boton 1");
         Button boton2 = new Button("Boton 2");
         Button boton3 = new Button("Boton 3");
         Button boton4 = new Button("Boton 4");
         this.setLayout(new FlowLayout());
         this.add(boton1);
         this.add(boton2);
         this.add(boton3);
         this.add(boton4);
         this.setSize(300,150);
         this.show();
   }
   public static void main(String[] args) {
          new VentanaFlowLayout();
   }
}
```

Border Layout

- Sitúa los elementos en una de estas 5 orientaciones: Norte, Sur, Este, Oeste y Centro
- Es el layout por defecto de los Frames
- Posee dos constructores:
  - BorderLayout();
  - BorderLayout(int gap_horizontal, int gap_vertical);
    - Creará el layout dejando los gaps horizontales y verticales entre sus distintas zonas.
- Para añadir más paneles o componentes (usando el método add), tenemos que indicar la región donde queremos añadir.
  - Panel.add(componente_a_añadir, BorderLayout.REGION_que_QUERAMOS);

Border Layout

- Ejemplo:

Código:

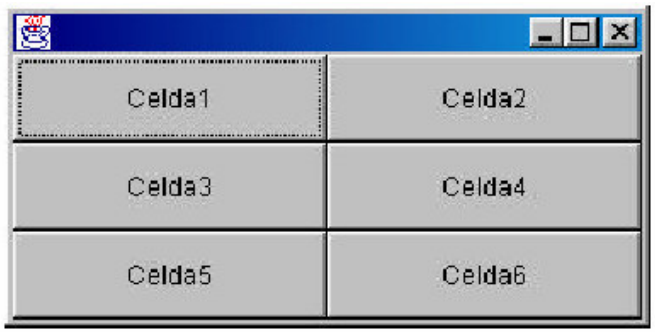
```
import java.awt.Frame;
import java.awt.Button;
import java.awt.BorderLayout;
public class VentanaBorderLayout extends Frame {
        public VentanaBorderLayout() {
                Button boton1 = new Button("Norte");
                Button boton2 = new Button("Sur");
                Button boton3 = new Button("Este");
                Button boton4 = new Button("Oeste");
                Button boton5 = new Button("Centro");
                this.setLayout(new BorderLayout());
                this.add(boton1, "North");
                this.add(boton2, "South");
                this.add(boton3, "East");
                this.add(boton4, "West");
                this.add(boton5, "Center");
                this.setSize(300,150);
                this.show();
        }
        public static void main(String[] args) {
                new VentanaBorderLayout();
        }
}
```

Grid Layout

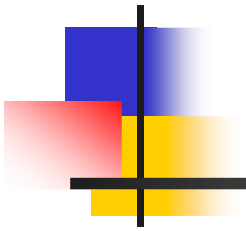
- Divide el contenedor en una cuadrícula, cada una de estas albergará un componente.
- Todas las casillas son del mismo tamaño.
- Los componentes se agregan de izquierda a derecha y de arriba hacia abajo.
- Dos posibles constructores:
  - **GridLayout(int filas, int columnas);**
    - Creará un layout en forma de malla con un número de columnas y filas igual al especificado.
  - **GridLayout(int columnas, int filas, int gap_horizontal, int gat_vertical);**
    - Especifica espaciados verticales y horizontales entre las cuadrículas. El espaciado se mide en píxeles.

Grid Layout

- Ejemplo:




```
import java.awt.Frame;
import java.awt.Button;
import java.awt.GridLayout;
public class VentanaGridLayout extends Frame {
        public VentanaGridLayout() {
                Button boton1 = new Button("Celda1");
                Button boton2 = new Button("Celda2");
                Button boton3 = new Button("Celda3");
                Button boton4 = new Button("Celda4");
                Button boton5 = new Button("Celda5");
                Button boton6 = new Button("Celda6");
                this.setLayout(new GridLayout(3,2));
                this.add(boton1); this.add(boton2); this.add(boton3);this.add(boton4);
                this.add(boton5);this.add(boton6); this.setSize(300,150); this.show();
        }
        public static void main(String[] args) {
                new VentanaGridLayout();
        }
}
```



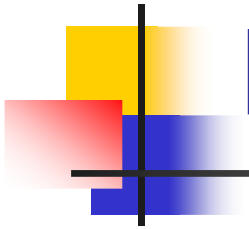
Gestión de Eventos

Introducción

- Programación Tradicional:
  - Generalmente, no existen modelos de gestión de eventos definidos.
  - El programa debe, mediante un bucle, preocuparse de monitorizar constantemente las acciones que realiza el usuario.
  - Ejemplo: Si el usuario pulsa ENTER, el programa no lo detecta a no ser que explícitamente lo pregunte mediante un getChar()
- Programación en Java (y en otros lenguajes actuales)
  - Existen modelos de eventos definidos.
  - El programa es avisado automáticamente de todas aquellas acciones (eventos) en los que esté interesado.
  - Ejemplo: Si el usuario pulsa un botón, el entorno avisa al programa invocando un método especial que éste tiene reservado para la notificación de ese evento ("pulsar botón").

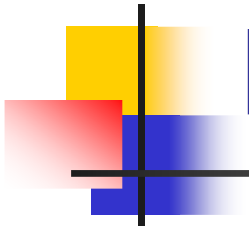
Modelo de Eventos

- El modelo de eventos de Java distingue dos elementos:
  - Fuentes de Eventos
    - Elementos sobre los que se producen los eventos.
    - Ejemplos: un botón, una lista, un panel,...
  - Escuchadores de Eventos
    - Elementos que reciben las notificaciones de los eventos



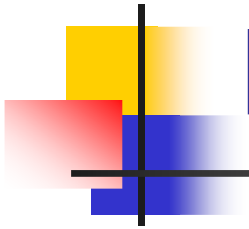
Eventos

- Cuando el usuario desarrolla una acción en la interfaz (click con el mouse, al presionar una tecla, etc.), genera un evento.
- Los eventos son objetos que describen lo que ha ocurrido.
- Existen un conjunto de diferentes tipos de clases de eventos que describen categorías de acciones de los usuarios.



Fuente de Eventos

- Una fuente de evento es el generador de un evento.
- Por ejemplo, un click del mouse sobre una componente Button, genera un ActionEvent, con el botón como fuente del evento.
- El ActionEvent es un objeto que contiene información sobre el evento que ha sido generado.



Eventos

- ¿Quién puede ser una fuente de eventos?
  - Cualquier componente de AWT (o SWING) sobre el que se puedan producir eventos (prácticamente todos)
- ¿Quién puede ser un escuchador de eventos?
  - Cualquier objeto (perteneciente a una clase) que implemente alguno de los interfaces definidos en Java para la notificación de los eventos.

Funcionamiento

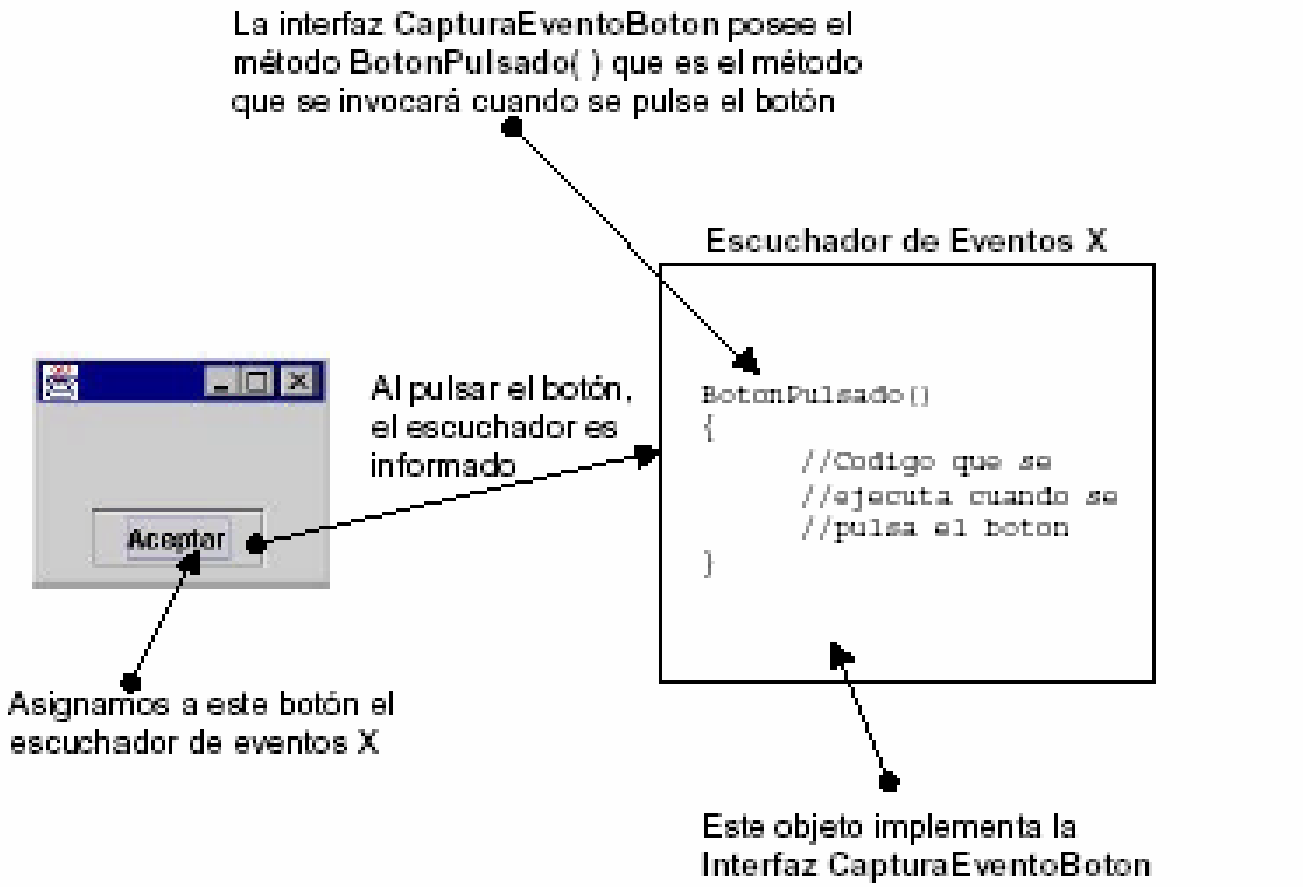
- Toda "Fuente de Eventos" debe tener asignado un "Escuchador de Eventos" que reciba las notificaciones de sus eventos.
- Cuando se produce un evento sobre la fuente de eventos, su escuchador es informado.
- Para ello, se invoca el método que el escuchador tenga definido para la notificación de ese tipo de evento.
- El escuchador, dentro de ese método, tendrá el código necesario para tratar ese evento.

Funcionamiento…

- ¿Cómo se asigna un escuchador a una fuente de eventos?
  <FuenteEvento>.add<EventoListener>(<Escuchador>)

- Ejemplo:
  - b.**addActionListener**(**new** EscuchaBoton());

Ejemplo gráfico



Ejemplo:

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

public class TestButton {
        private Frame f;
        private Button b;

        public TestButton() {
                f = new Frame("Test");
                b = new Button("Press Me!");
                }
        public void launchFrame() {
                b.addActionListener(new EscuchaBoton());
                f.add(b, BorderLayout.CENTER);
                f.pack();
                f.setVisible(true);
                }
```

```
public static void main(String[] args) {
        TestButton tb = new TestButton();
        tb.launchFrame();
        }
}
```

Ejemplo...

```
class EscuchaBoton implements ActionListener {
    public void actionPerformed(ActionEvent e) {
    System.out.println("Action occurred");
    System.out.println("Button's command is: " +
      e.getActionCommand());
    }
}
```

Ejemplo...

Este ejemplo tiene las siguientes características:

- La clase Button tiene un addActionListener (ActionListener).
- La interfaz ActionListener define el método actionPerformed, el cual recibe un ActionEvent.
- Una vez creado, un objeto Button puede tener un listener para ActionEvents por medio del método addActionListener().

Ejemplo…

- Cuando se da un click al objeto Button, un ActionEvent es enviado. El ActionEvent es recibido por el método actionPerformed() de algún ActionListener que está registrado en el botón por medio del método addActionListener().

- El método getActionCommand() de la clase ActionEvent retorna el nombre del comando asociado con esta acción, en este caso se relaciona al objeto botón creado, por lo cual retorna "Button Pressed".

Eventos: notificación

- Interfaces para la notificación de eventos:
  - Interfaces que contienen métodos que serán invocados cuando se produzca un determinado tipo de evento.
  - Cada interfaz está especializado en capturar un tipo de eventos determinado.
  - Lista de algunas Interfaces y sus métodos

Categorías, Interfaces y Métodos

Categoria	Nombre de la interfaz	Métodos
Action	ActionListener	actionPerformed(ActionEvent)
Item	ItemListener	itemStateChanged(ItemEvent)
Mouse	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
Mouse Motion	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
Key	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
Focus	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)

Categorías, Interfaces y Métodos…

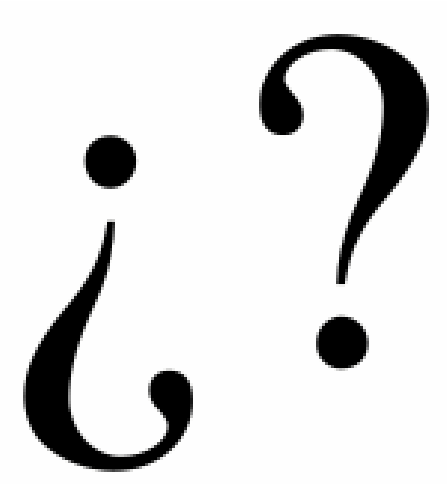
Adjustment	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
Component	ComponentListener	componenrtMoved(ComponentEvent) componenrtHidden(ComponentEvent) componenrtResized(ComponentEvent) componenrtShown(ComponentEvent)
Window	WindowListener	window Closing(WindowEvent) window Opened(WindowEvent) window Iconified(WindowEvent) window Deiconified(WindowEvent) window Closed(WindowEvent) window Activated(WindowEvent) window Deactivated(WindowEvent)
Container	Containeristener	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
Text	TextListener	textValueChanged(TextEvent)

Eventos Adapters

- Adapters: Son clases que tienen definidos todos los métodos de un interfaz concreto.
  - La implementación de dichos métodos está vacía.

- Llamando a un Adapter y sobreescribiendo los métodos necesarios conseguimos el mismo resultado que implementando directamente el interfaz.
  - Heredando de un adapter *extends.*
  - Implementando directamente la interfaz *implements.*

Consultas...



UNPA
Universidad Nacional de la Patagonia Austral

Próxima clase

- Repaso
- Administradores de diseño avanzados: ventajas, tipos, como definirlos
- Gestión de eventos
- Ejemplos
- Práctica