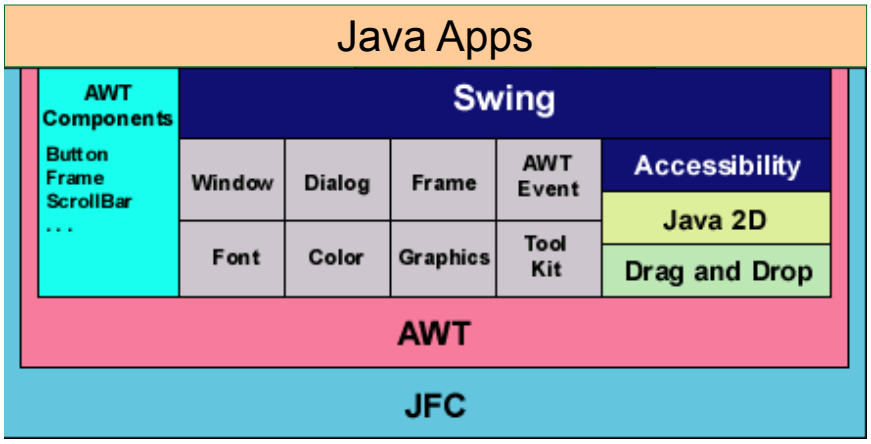
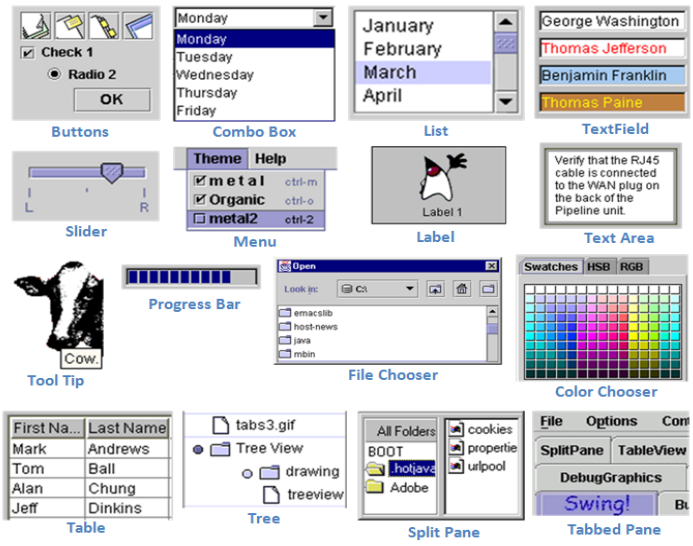
Plataforma Java



Ing. César Omar Aranda 3

Componentes GUI



Ing. César Omar Aranda 4

Swing

- Disponible desde la versión 1.2 de Java
- Implementada sin usar código nativo (100% Java), con lo cual la GUI se verá de la misma forma en distintas plataformas.
- Basada en la arquitectura MVC.
- Admiten configuraciones de aspecto visual (Look & Feel).
- Los componentes están en Swing en javax.swing
- Ejemplo JApplet:
  - Añadir los componentes gráficos (JComponent) al Panel raíz (JRootPane).
  - Igual mecanismo para administradores de diseño, paneles, etc.
  - Al añadir o quitar componentes, **los mensajes se mandan al panel raíz, no al applet**
  - **this** es el JApplet: **this.getContentPane().add(mi_boton)**

Ing. César Omar Aranda 5

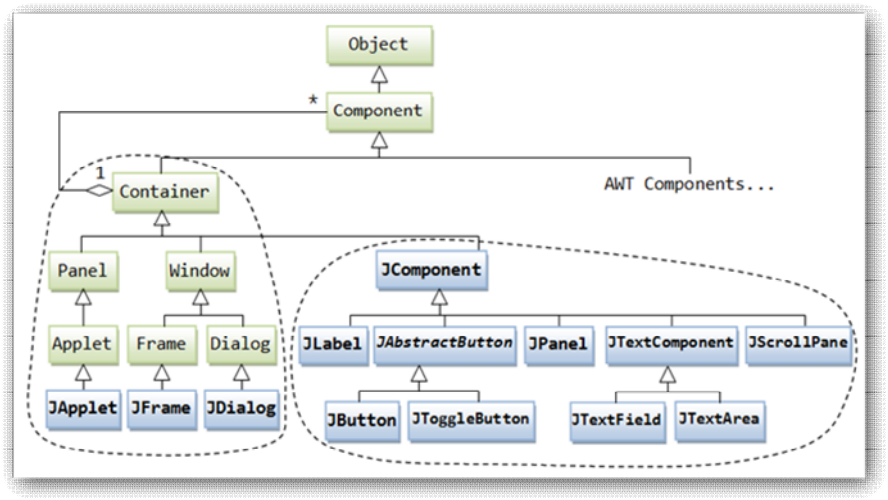
Aplicaciones basadas en GUI

El desarrollo de una aplicación basada en Interfaces Gráficas de Usuario en Java requiere considerar:

- La Estructura de la jerarquía de herencia
  - Que define el comportamiento y atributos de los componentes en la GUI de la aplicación.
- La Estructura de la jerarquía de contenedores
  - Que define cómo se disponen todos los componentes en la GUI de la aplicación.
- El Manejo de eventos.
  - Que define cómo reacciona la aplicación ante los eventos de la GUI.

Ing. César Omar Aranda 6

Jerarquía de Herencia



Ing. César Omar Aranda 7

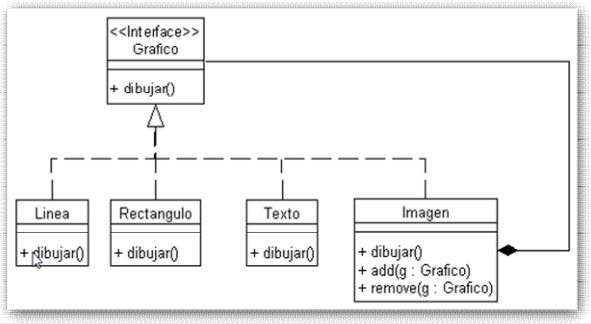
Comprendiendo a Composite

- El patron Composite es también conocido como *composición recursiva*.
- Permite construir objetos complejos mediante composición recursiva de objetos más simples o similares, bajo una estructura de árbol.
- Permite manipular consistentemente los objetos en el árbol, pidiendo que todos los objetos en el árbol tengan una superclase o interfaz común.

Ing. César Omar Aranda 8

Problema y Solución

- Problema
  - Necesitamos crear una serie de clases para guardar información acerca de una serie de figuras que serán círculos, cuadrados y triángulos. Además necesitamos poder tratar también grupos de imágenes porque nuestro programa permite seleccionar varias de estas figuras a la vez para moverlas por la pantalla.
- Solución



Ing. César Omar Aranda 9

Ejemplo de Implementación

```
public abstract class Componente
{
        protected String nombre;
        public Componente (String nombre)
        {
                this.nombre = nombre;
        }
        abstract public void agregar(Componente c);
        abstract public void eliminar(Componente c);
        abstract public void mostrar(int profundidad);
}
```

```
class Hoja extends Componente
{
        public Hoja (String nombre)
        {
                super(nombre);
        }
        public void agregar(Componente c)
        {
                System.out.println("no se puede agregar la hoja");
        }
        public void eliminar(Componente c)
        {
                System.out.println("no se puede quitar la hoja");
        }
        public void mostrar(int depth)
        {
                System.out.println('-' + "" + nombre);
        }
}
```

```
class Compuesto extends Componente
{
        private ArrayList<Componente> hijo = new ArrayList<Componente>();
        public Compuesto (String name)
        {
                super(name);
        }
        @Override
        public void agregar(Componente componente)
        {
                hijo.add(componente);
        }
        @Override
        public void eliminar(Componente componente)
        {
                hijo.remove(componente);
        }
        @Override
        public void mostrar(int profundidad)
        {
                System.out.println(nombre + " nivel: " + profundidad);
                for (int i = 0; i < hijo.size(); i++)
                        hijo.get(i).mostrar(profundidad + 1);
        }
}
```

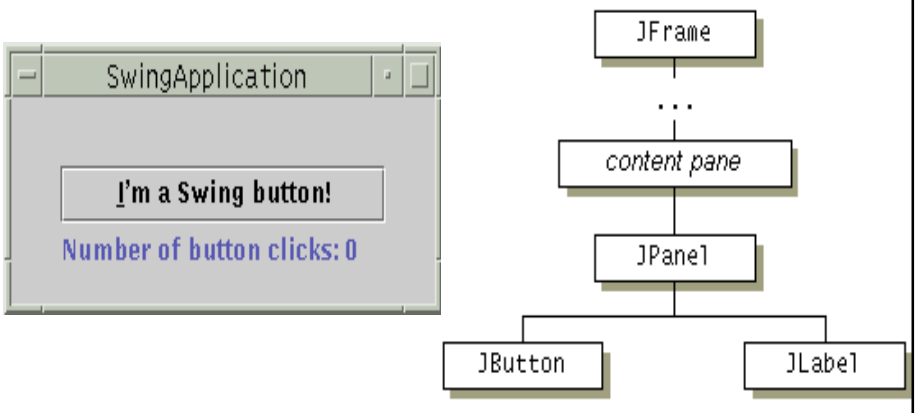
```
public class Client
{
        public static void main(String[] args)
        {
                Compuesto raiz = new Compuesto("root");
                raiz.agregar(new Hoja("hoja A"));
                raiz.agregar(new Hoja("hoja B"));
                Compuesto comp = new Compuesto("Compuesto X");
                comp.agregar(new Hoja("hoja XA"));
                comp.agregar(new Hoja("hoja XB"));
                raiz.agregar(comp);
                raiz.agregar(new Hoja("hoja C"));
                Hoja l = new Hoja("hoja D");
                raiz.agregar(l);
                raiz.eliminar(l);
                raiz.mostrar(1);
        }
}
```

Jerarquía de Contenedores

- Anidamiento (jerarquía) de componentes contenedores. Cada programa Swing contiene al menos una.
- Usan un Layout Manager para determinar cómo se disponen los componentes en los contenedores.
- Swing provee 4 contenedores de alto nivel (ventana base de la GUI):
  - JWindow
  - JFrame
  - JDialog
  - JApplet
- La jerarquía está compuesta de diferentes capas. Cada contenedor de alto nivel contiene un contenedor intermedio conocido como "content pane ". En general, no es necesario conocer qué hay entre el contenedor de alto nivel y el content pane.

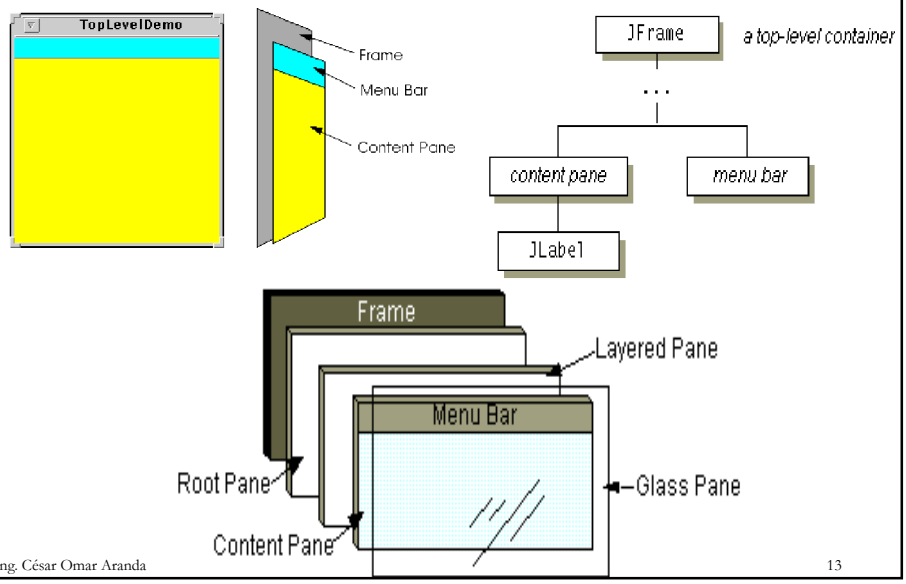
Ing. César Omar Aranda 11

Ejemplo: Jerarquía de Contenedores



Ing. César Omar Aranda 12

Estructura de un JFrame



Ing. César Omar Aranda 13

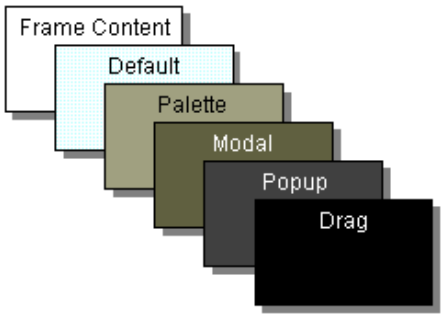
Root Pane

- "Añadido" en forma invisible al contenedor de alto nivel.
- Creado por Swing cuando instancia un contenedor de alto nivel.
- Es una instancia de JLayeredPane la que contiene la barra de menús y el content pane.
- Maneja prácticamente todo entre el contenedor de alto nivel y los componentes atómicos.
- A tener en cuenta si se requiere interceptar los clicks del mouse o 'marcar' varios componentes.

Ing. César Omar Aranda 14

Layered pane

- Provisto por root pane pero también puede crearse.
- Contenedor con profundidad, tal que componentes que se superponen (ej: popup menus) pueden aparecer unos encima de otros (z-buffering).



Ing. César Omar Aranda 15

Content Pane

- Usualmente es un JPanel.
- En la mayoría de las aplicaciones Swing contiene casi todo, excepto la barra de menú.
- Debe ser creado explícitamente.

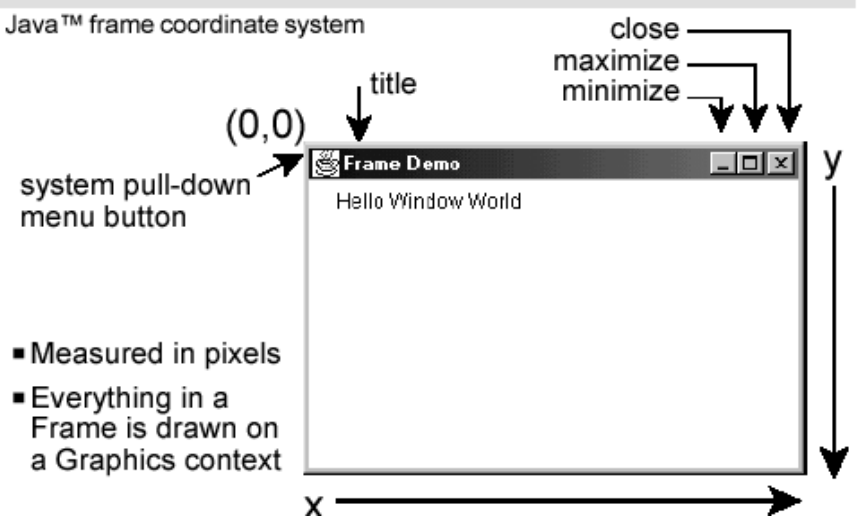
Ing. César Omar Aranda 16

Glass Pane

- Útil para 'marcar' objetos o interceptar eventos
- Por ejemplo: bloquear todos los eventos del mouse en un área que contenga uno o más componentes

Ing. César Omar Aranda 17

Jframe: coordenadas



Ing. César Omar Aranda 18

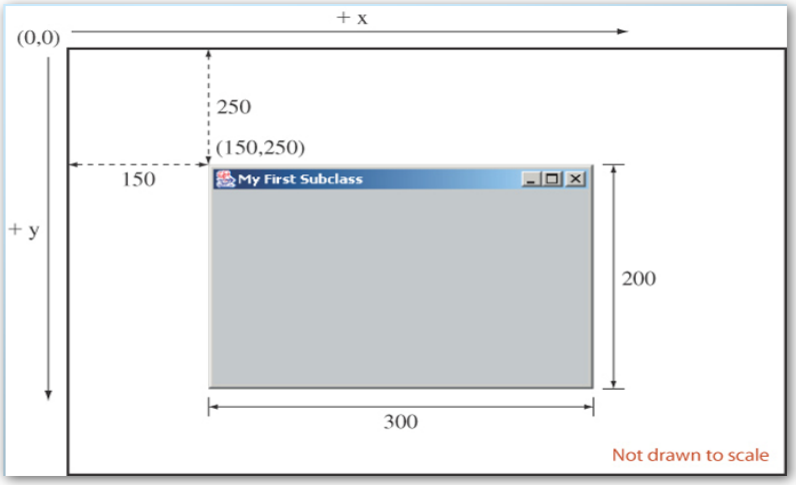
Ejemplo de Posicionamiento

```
1.  import javax.swing.*;
2.  class JFrameSubclass1 extends JFrame{
3.          private static final int ANCHO= 300;
4.          private static final int ALTO= 200;
5.          private static final int ORIGEN_X= 150;
6.          private static final int ORIGEN_Y= 250;
7.          public JFrameSubclass1 (){
8.                  setTitle ("My First Subclass");
9.                  setSize (ANCHO, ALTO);
10.                 setLocation (ORIGEN_X, ORIGEN_Y);
11.                 setDefaultCloseOperation(EXIT_ON_CLOSE);
12. }
```

Ing. César Omar Aranda 19

Mostrando JFrameSubclass1

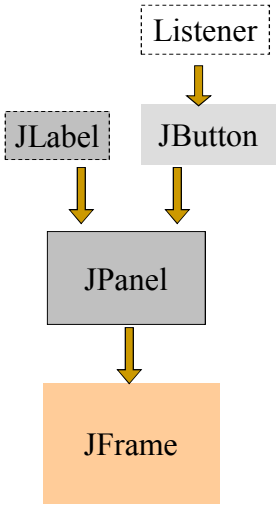
- La ventana marco JFrameSubclass1 sobre la pantalla



Ing. César Omar Aranda 20

Construcción Bottom Up

- Crear (libremente):
  - Frame
  - Panel
  - Componentes
  - Listener
- Añadir (bottom up)
  - Listeners a los componentes
  - Componentes al panel
  - Panel al Frame



Ing. César Omar Aranda 21

Pasos en la Construcción de GUI

- Para crear una aplicación con Swing se debe:
  - Crear un JFrame
  - Llenarlo de components según los requerimientos de la aplicación
  - Mostrar el JFrame en pantalla invocando el método `setVisible(true)`
  - Ejemplo

```
1. public static void main(String[] args)
2. {
3.     JFrame frame = new JFrame();
4.     frame.setVisible(true);
5. }
```

Ing. César Omar Aranda 22

Contenedor Base

- Se acostumbra (pero no es obligatorio) declarar una subclase de JFrame y en el constructor llenar el Frame de componentes

```
1. public class FrameAlumnos extends JFrame {
2.    JTextField nombre;          JTextField fechaNac;
3.    FrameAlumnos() {
4.       JPanel contentPane = (JPanel) getContentPane();
5.       nombre = new JTextField();
6.       contentPane.add(nombre);
7.       FechaNac = new JtextField();
8.       contentPane.add(fechaNac);
9.    }  ...
```

Ing. César Omar Aranda 23

Panel base

- Ubicación de componentes en un Frame (Layout)
- La clase JPanel es un contenedor de objetos
- Un JFrame tiene un panel principal que se obtiene invocando getContentPane()

```
1. JFrame frame = new JFrame();
2. JPanel contentPane = (JPanel) frame.getContentPane();
3. . . .
```

- Un panel puede contener componentes finales (JLabel, JTextField, etc.) u otros paneles (Jpanel)
- Esto permite acomodar las cosas en el Frame
- Se puede utilizar posicionamiento absoluto (x, y) pero esto no es recomendable

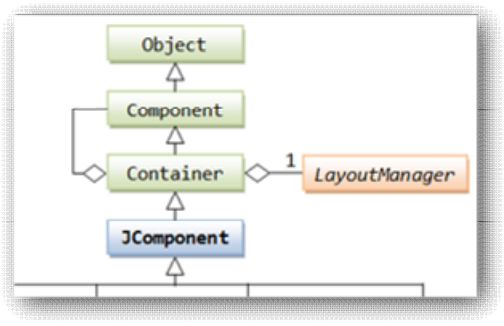
Ing. César Omar Aranda 24

Layout

- Los Layouts son clases que determinan la forma como se acomodan los componentes en un panel
- A cada JPanel se le puede asignar un Layout
  - `JPanel panel = new Jpanel()`
  - `panel.setLayout(new FlowLayout());`
- Algunos Layouts comunmente usados
  - FlowLayout. De izquierda a derecha y de arriba abajo.
  - GridLayout. Una tabla o cuadrícula (todas las celdas del mismo tamaño).
  - BoxLayout. De arriba abajo o de izquierda a derecha. Tamaño variable (muy importante).

Ing. César Omar Aranda 25

Relación entre Container y Layout



Ing. César Omar Aranda 26

Diálogos

- Un diálogo es un frame que permite recolectar datos para realizar algún procesamiento
- En Java existe una clase JDialog para este fin.
- JDialog es subclase de JFrame y permite definir diálogos modales y no modales
- Si un diálogo es modal cuando se activa no se puede acceder a ningún otro elemento del programa
- Si el diálog es modal se abre la ventana (window) del diálogo pero el usuario puede seleccionar y trabajar con otras ventanas de la aplicación
- En Swing si el diálogo es modal el hilo que abre el diálogo se bloquea hasta que el diálogo sea cerrado.

Ing. César Omar Aranda 27

Diálogos modales

- Para crear un diálogo modal se debe especificar en el constructor
  - `JDialog dlg = new Jdialog(frame, "Titulo", true);`
- El tercer parámetros es booleano e indica si el diálogo es modal
- Generalmente se crea una subclase de JDialog:

```
public class DialogoDatos extends Jdialog {
    JTextField nombre;
    ...
    public DialogoDatos(JFrame frame) {
        super(frame, "Título", true);
        ContentPane cp = (ContentPane) getContentPane();
        ...
    }
```

Ing. César Omar Aranda 28

Diálogos modales

- Ejemplo de uso
  - `DialogoDatos dlg = new DialogoDatos(this);`
  - `dlg.setVisible(true);`
  - `String nombre = dlg.getNombre();`
  - `...`
- Si el diálogo no es modal el código que sigue a la instrucción continúa ejecutándose en un hilo paralelo.
- Por lo tanto la lógica a ejecutar debe colocarse en los métodos de los ActionListener definidos en el diálogo

Ing. César Omar Aranda 29

Codificar y analizar

```
1.  public class VerParametros {
2.    public static void main(String[] args) {
3.      JFrame frame = new JFrame("Saludo sencillo");
4.      Container cp = frame.getContentPane();
5.      cp.setLayout(new FlowLayout());
6.      JButton btnHola = new JButton("Güenas!");
7.      btnHola.setPreferredSize(new Dimension(100,
80));
8.      cp.add(btnHola);
9.
frame.setDefaultCloseOperation(JFrame.EXIT_ON_C
LOSE);
10.     frame.setSize(300, 150);
11.     frame.setLocationRelativeTo(null);
12.     frame.setVisible(true);
13.
14.     System.out.println(btnHola.getSize());
15.     System.out.println(btnHola.getLocation());
16.
System.out.println(btnHola.getLocationOnScreen());
17.
18.     System.out.println(cp.getSize());
19.     System.out.println(cp.getLocation());
20.     System.out.println(cp.getLocationOnScreen());
21.
22.     System.out.println(frame.getSize());
23.     System.out.println(frame.getLocation());
24.     System.out.println(frame.getLocationOnScreen());
25.   }
26. }
```

27. **Analiza:**
28. ¿Qué sentencia indica que la aplicación aparezca en el centro de la pantalla?
29. ¿Cuánto ocupa la barra de título?
30. ¿Dónde aparece escrito el saludo?
31. ¿Cómo se podría modificar la justificación del texto?
32. ¿Se puede escribir texto fuera de un componente?

Ing. César Omar Aranda 30

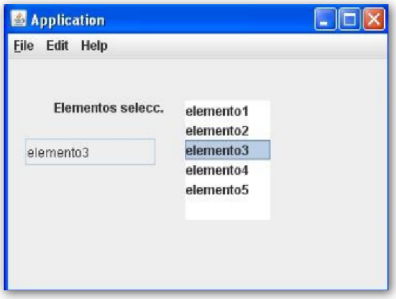
Ejemplo JComboBox

```
private String[] contenidos ={"Manzana","Naranja","Platano","Tiburon","Tomate","Trucha"};

private JComboBox jComboBox1 = new JComboBox(contenidos);
```


```
String nombre= (String) jComboBox1.getSelectedItem();

jTextField1.setText("Esto es un " + nombre);

jLabel2.setIcon(new ImageIcon("imagenes/"+nombre+".gif"));
```

Ejemplo JList

```
private String[] contenidos={"elemento1",
   "elemento2","elemento3","elemento4","elemento5"};

private JList=new JList(contenidos);
```




```
jTextField1.setText((String)jList1.getSelectedValue());
```

Bibliografía

- Lectura complementaria
  - DEITEL, F. y DEITEL, H. (2008). Cómo programar en Java. 7ma edición, Capítulo 11. Pearson Education.
  - FROUFE QUINTAS, A. (2003). Java 2: Manual de usuario y tutorial. 3ra edición. México: Alfaomega.
  - GAMMA, E; HELM, R. y otros (2003). Patrones de Diseño. Pearson Education.
- Referencias
  - http://pervasive2.morselli.unimo.it/~nicola/courses/IngegneriaDelSoftware/index.html
  - http://www.javaya.com.ar/index.php?inicio=20 (itemes 30 a 39)