

Programación de Interfaces Gráficas en Java

Objetivo: Programar aplicaciones con interfaces gráficas usando objetos gráficos de Swing y definidos por el programador

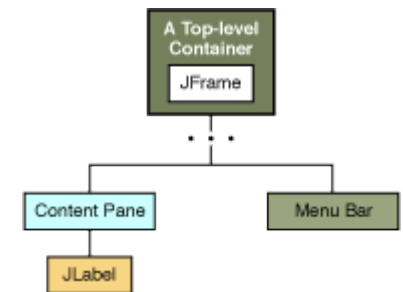
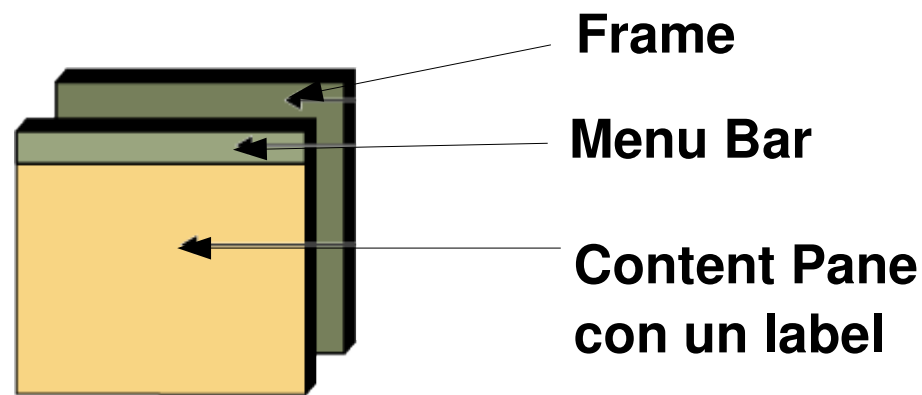
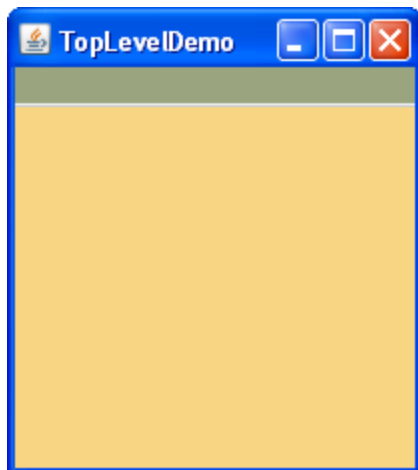
Agustín J. González
ELO329

AWT y Swing

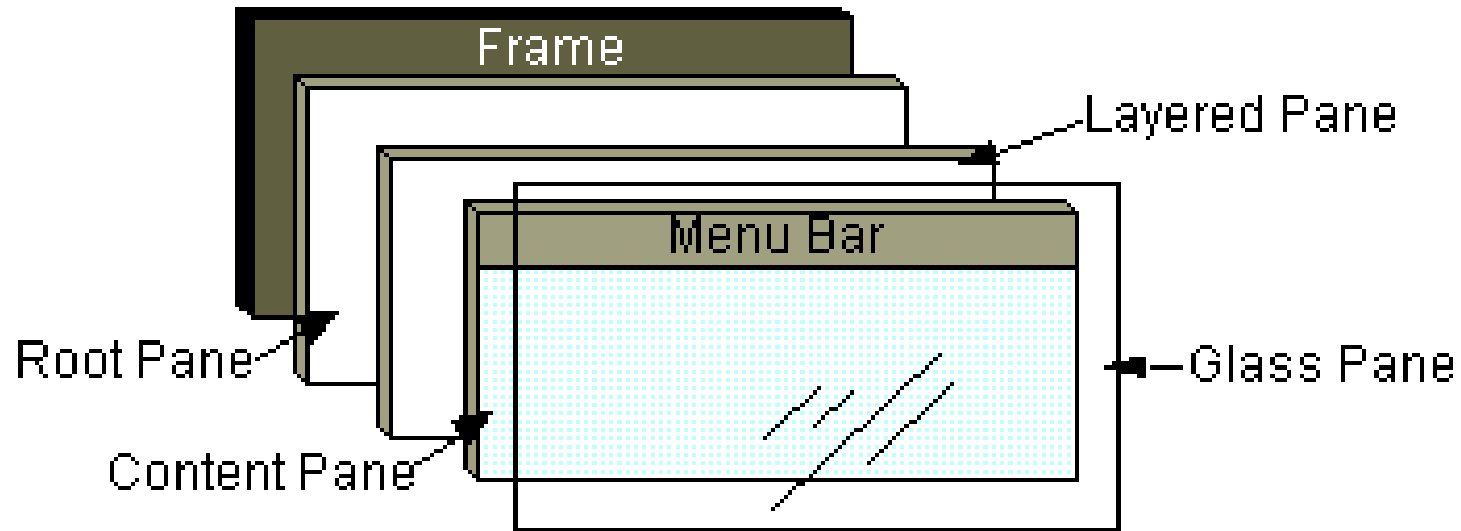
- En sus orígenes Java introdujo la AWT (Abstract Window Toolkit). Ésta “creaba” los objetos delegando su creación y comportamiento a herramientas nativas de la plataforma donde corre la Máquina Virtual Java.
- Este esquema condujo a problemas por diferencias en distintas plataformas y S.O.
- La solución fue desarrollar todos los objetos de la GUI basados sólo en elementos muy básicos y comunes en todas las plataformas. Así surge Swing.
- Junto a la JDK de Oracle es posible bajar códigos de ejemplos de Swing. Ver la carpeta que contiene el ejecutable javac y podrá encontrar el directorio demo un nivel más arriba.

Desplegando información

- Todos los objetos gráficos de una aplicación Java forman una jerarquía. En lo más alto de la jerarquía está el JFrame, JDialog, o un JApplet.
- Ahora veremos la estructura de los JFrame.
- Ejemplo:



Estructura de un JFrame



- El RootPane está contenido en el JFrame. También lo traen los JFrame y los otros contenedores de ventanas superiores (autónomas): JDialog, JApplet, JFrame.
- El root pane tiene 4 partes: panel de capas, panel de contenido, una barra de menú opcional y un panel transparente.

Layered Pane (panel de capas múltiples)

- Contiene la barra de menú opcional y el panel para poner contenidos.
- Puede también contener otras componentes en orden especificado por eje Z (profundidad).
- Ver más detalles en curso tutorial de Swing
- Ver LayeredDemo.java (*)

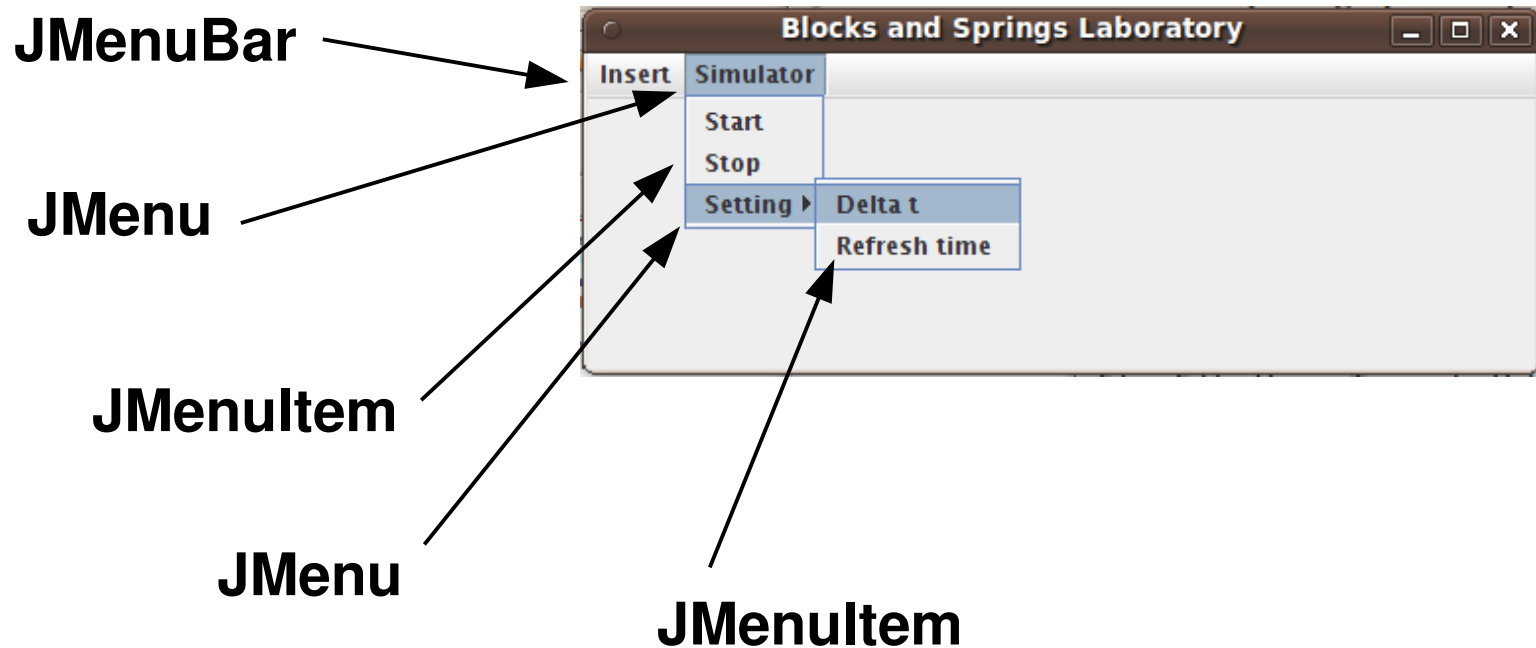
* de referencia, no se requiere estudio exhaustivo

Panel transparente (de vidrio)

- Oculto por omisión (default).
- Si se hace visible, es como una hoja de vidrio sobre todos las partes del panel raíz.
- Es transparente, a menos que se implemente un método para pintarlo.
- Puede interceptar los eventos de la ventana panel de contenido y menú.
- Ver `GlassPaneDemo.java` (*)
- Revisar guía visual de componentes Swing

Menús (así es en plural)

- Algunos elementos de un menú



Menús: Ejemplo

- Crear un frame
 - Crear un menubar
 - Crear un menu
 - Crear algunos ítemes del menu
 - Capturar eventos
 - Agregar ítem al menu
 - Agregar el menu al menubar
 - Incorporar el menubar al frame
 - Por ejemplo de menú más completo ver: MenuDemo.java (*)
- ```
JFrame f = new JFrame("MenuT");
JMenuBar mb = new JMenuBar();
JMenu menu = new JMenu("Choose");
JMenuItem item1, item2;
item1 = new JMenuItem("Data 1");
item2 = new JMenuItem("Data 2");
// Action listeners!!
menu.add(item1);
menu.add(item2);
mb.add(menu);
f.setJMenuBar(mb);
```



# Modelos y Vistas de Objetos

- Asociado a cada objeto gráfico debemos distinguir el modelo de un objeto de la vista del mismo.
- El **modelo es el conjunto de atributos de un objeto**, corresponde a la representación en memoria de un objeto. Por ejemplo, para un termómetro, basta el atributo double temperatura.
- La **vista es la apariencia visual** que decidimos dar al objeto. Por ejemplo, un termómetro puede ser digital, columna de mercurio, la intensidad de un color, etc.
- Por un lado tenemos el cambio del estado de un objeto y por otro el cambio en la vista correspondiente a ese nuevo estado.
- Objetos Swing (JButton, JFrame, etc.) mantienen esa consistencia, pero debemos ocuparnos de ello en objetos gráficos fuera de Swing (círculo, triángulo, rectángulo, etc).

# Creación de objetos gráficos no presentes en Swing

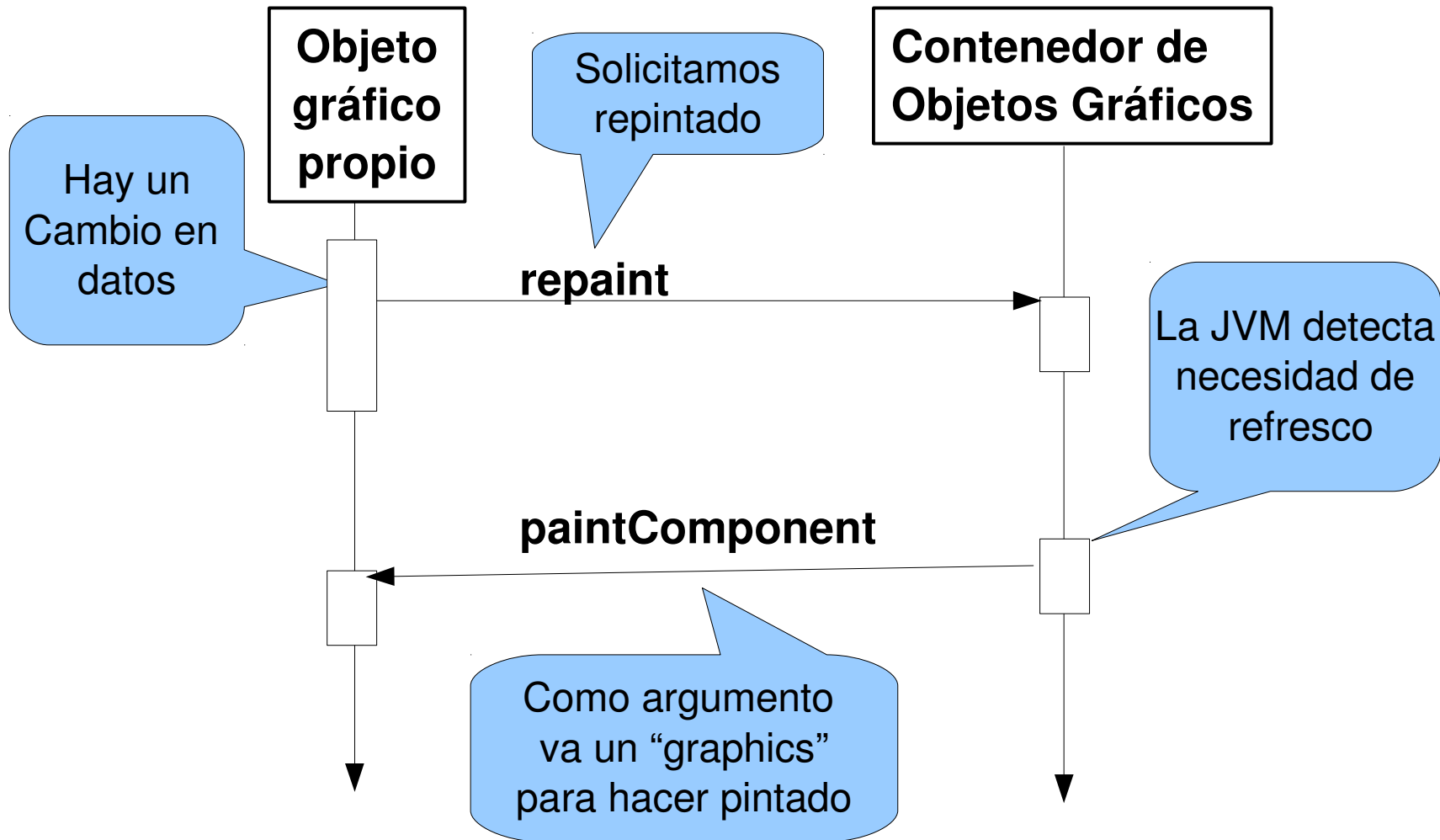
- En general hay que tratar de usar componentes estándares de Swing. Ellas se encargan de hacer su (re)pintado en pantalla cuando corresponda.
- Este es el caso de JLabels, JButtons, componentes de texto, icons, borders.
- Objetos gráficos nuevos se pueden crear heredando de JPanel (heredar para luego poder redefinir el método paintComponent).
- Java tiene clases para representar líneas, círculos, etc. con ellos podemos crear objetos gráficos propios.

# Pintado de objetos propios: método `Component::repaint()`

- Cuando una componente cambia alguno de sus atributos, por ejemplo un `JLabel` cambia su texto, el método `repaint` es invocado. Swing consigue así itinerar el repintado de la componente gráficas.
- Si nosotros hemos construido líneas, círculos, etc. y cambiamos algunos de sus atributos, debemos llamar a `repaint()`, del `JPanel`. Así logramos que luego se invoque el método `paintComponent(Graphics)` de `JPanel` (el cual hemos redefinido)
- `Graphics` tiene métodos para pintar líneas, círculos, etc.
- Ver ejemplo: `Sketch.java` y `MouseTest.java`

# Petición de repintado: repaint

- Diagrama de secuencia para repintado.



# Java 2D: Clases para crear objetos gráficos propios

- Java 2D provee gráficos, texto e imágenes de dos dimensiones a través de extensiones de Abstract Windowing Toolkit (AWT)
- Incluye clases para Rectángulos, Líneas, Elipses.
- La clase Graphics2D, a través de su método draw, permite dibujar estos objetos debido a que todos ellos implementan la interfaz shape.
- Ver demo: ShapesDemo2D.java (\*)

# Cambio del número de componentes gráficas de un Panel

- Cuando agregamos un objeto Swing a un JFrame éste no gatilla su repintado inmediatamente sino cuando el programador lo solicita o cuando ocurre algún cambio del JFrame (como cambio de tamaño, etc)
- En estos casos podemos invocar el método `validate()` del componente gráfico que lo contiene.
- Así logramos actualizar su despliegue.
- Ver `CreaBotones.java`

