

# Ambientes gráficos I

## (Swing)

Jonatan Gómez Perdomo, Ph.D.

[jgomezpe@unal.edu.co](mailto:jgomezpe@unal.edu.co)

Arles Rodríguez, Ph.D.

[aerodriguezp@unal.edu.co](mailto:aerodriguezp@unal.edu.co)

Camilo Cubides, Ph.D.(c)

[eccubidesg@unal.edu.co](mailto:eccubidesg@unal.edu.co)

Grupo de investigación en vida artificial – Research Group on Artificial Life – (Alife)

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Universidad Nacional de Colombia

# Outline

1 Algunas componentes de Swing

2 Práctica de clase



# Outline

## 1 Algunas componentes de Swing

## 2 Práctica de clase



# Marcos en Swing (JFrame) I

## Definición (JFrame)

Es un contenedor donde colocar componentes. La clase `JFrame` se encuentra en el paquete `javax.swing`. La mayoría de las ventanas son instancias de la clase `JFrame` o subclases de `JFrame`. `JFrame` proporciona los atributos y comportamientos básicos de una ventana

Constructores y métodos:

`JFrame()`: Es el constructor de nuestra ventana, aparece sin título y no puede verse en pantalla hasta que se llame el método para hacerlo visible.

`setVisible(boolean b)`: Define si nuestro `JFrame` puede verse o no y únicamente acepta valores booleanos como `true` o `false`.



# Marcos en Swing (JFrame) II

`setTitle(String b)`: Define el nombre del JFrame (el título), y lo que acepta es una cadena para nombrarla.

`setSize(int ancho, int largo)`: Es la definición del ancho y largo del JFrame que vamos a crear.

`pack()`: Redefine la ventana para que entre en un espacio determinado.

`setLocation(int horizontal, int vertical)`: Definición de la posición en pantalla donde va a estar nuestro JFrame una vez que lo creamos.

`setDefaultCloseOperation(int operation)`: Define la operación que va a realizar la ventana cuando se oprime el botón de cerrar.

Véase la clase Ventana



# Paneles en Swing (JPanel) I

## Definición (JPanel)

Los paneles en Java son objetos contenedores, la finalidad de estos objetos es la agrupación de otros objetos tales como botones, campos de texto, etiquetas, selectores, etc; una gran ventaja de usar `JPanel` en Java es que podemos manejar la agrupación de una mejor forma, supongamos que tenemos una serie de botones en un panel, y deseamos desactivarlos todos a la vez, en lugar de hacerlo individualmente con los botones, podemos desactivar el panel y con esto los botones.

Constructores y métodos:

`JPanel()`: Crea un panel vacío.



# Paneles en Swing (JPanel) II

`JPanel(boolean doblebuffered)`: dependiendo del valor del booleano se habilitará o no el doble buffer al momento de crear el panel. El doble Buffer se usa para que al momento de dibujar en un panel este no parpadee.

`JPanel(Layout gestor)`: Crea un JPanel con el gestor de organización proporcionado. Layout es un gestor de organización, es decir, es el que determina como van a estar distribuidos los componentes en el panel, dentro de estos distribuidores esta el FlowLayout, el BorderLayout, el BoxLayout, el GridLayout, etc.

`add(Component componente)` Adiciona un componente al panel.

*Véanse las clases PanelCiudad y PanelControles*



# Etiquetas en Swing (JLabel) I

## Definición

Las etiquetas en Java permiten agregar texto o imágenes en una ventana que sólo pueden ser editados por funciones del programa.

Constructores y métodos:

`JLabel()`: Crea una etiqueta vacía.

`JLabel(String text)`: Crea una etiqueta con el String text justificado a la izquierda.

`JLabel(String text, int alignment)`: Crea una etiqueta con el String text, justificado con alignment. Los valores de alignment son: `JLabel.LEFT`, `JLabel.RIGHT` y `JLabel.CENTER`.





# Etiquetas en Swing (JLabel) II

`void setText(String text)`: Cambia el texto de la etiqueta por el `String text`.

`String getText()`: Retorna el texto en la etiqueta.

`void setAlignment(int alignment)`: cambia la justificación de la etiqueta según el valor de `alignment`.

`int getAlignment()`: Retorna la justificación actual del objeto.



# Botones en Swing (JButton) I

## Definición

Los botones en Java sirven como mecanismo de interacción entre el usuario y el programa por medio de la interfaz gráfica que representan, ya que pueden realizar tareas programadas, verificar condiciones o seleccionar elementos en grupos específicos.



# Botones en Swing (JButton) II

Constructores y métodos:

`JButton()`: Crea un botón sin etiqueta.

`JButton(String text)`: Crea un botón con la etiqueta `text`.

`void setLabel(String label)`: Modifica la etiqueta del botón.

`String getLabel()`: Retorna la etiqueta asociada al botón.

`addActionListener(ActionListener listener)`: Añade un auditor al botón, es decir, `ActionListener`, escucha los eventos. Cuando se da clic al botón, el suceso será gestionado por el `ActionListener listener`.



# Campos de texto en Swing (JTextField) I

## Definición

Permite usar un campo de texto de una sola línea, que son editables. Los campos de texto permiten al usuario introducir cadenas y editar texto utilizando los cursores, las teclas de cortar y pegar, y las selecciones que se hacen con el ratón.

Constructores y métodos:

`JTextField()`: Crea un campo de texto por omisión.

`JTextField(int numChars)`: Crea un campo de texto con tamaño definido por `n` caracteres.

`JTextField(String str)`: Crea un campo de texto inicializado con la cadena de caracteres dada `String str`.



# Campos de texto en Swing (JTextField) II

`JTextField(String str, int numChars)`: Crea un campo de texto inicializado con la cadena de caracteres y además especifica su tamaño.

`void setText (String str)`: Este método permite establecer un determinado texto.

`String getText()`: Con este método se puede obtener la cadena del campo de texto.

`String getSelectedText()`: Con este método puede obtenerse el texto actualmente seleccionado.

`void select(int startIndex, int endIndex)`: Con este método se puede seleccionar una parte del texto del campo de texto creado.



# Campos de texto en Swing (JTextField) III

`void setEditable(boolean editable)`: Este método permite controlar si el contenido de un campo de texto puede ser modificado por el usuario, `true` para que el texto se pueda cambiar y `false` para que no se pueda editar.

`boolean isEditable()`: Este método permite determinar si un campo de texto es editable o no, retorna `true` si es editable y `false` si no lo es.

`void setEchoChar(char ch)`: Este método permite que el texto que se introduce no sea visible, como cuando se introduce una clave de acceso.

`boolean echoCharIsSet()`: Este método permite revisar si el campo de texto está en modo ocultar caracteres.

`char getEchoChar()`: Este método permite obtener los caracteres que están ocultos.



# Áreas de texto en Swing (JTextArea) I

## Definición

Ala áreas de texto son un sencillo editor multilíneas que utiliza una sólo tipo de fuente.

Constructores y métodos:

`JTextArea()`: Crea un campo de texto por defecto.

`JTextArea(int numLines, int numChars)`: Permite crear un área de texto con el alto y ancho definidos.

`JTextArea(String str)`: Permite crea un área de texto con un texto inicial especificado.

`JTextArea(String str, int numLines, int numChars)`: Este método permite crear un área de texto con el alto y ancho definidos y además con un texto inicial específico.



# Áreas de texto en Swing (JTextArea) II

`void setText (String str)`: Este método permite establecer un determinado texto.

`String getText()`: Con este método se puede obtener la cadena del área de texto.

`String getSelectedText()`: Con este método puede obtenerse el texto actualmente seleccionado.

`void select(int startIndex, int endIndex)`: Con este método se puede seleccionar una parte del texto del área de texto creado.

`void setEditable(boolean editable)`: Este método permite controlar si el contenido de área de texto puede ser modificado por el usuario, `true` para que el texto se pueda cambiar y `false` para que no se pueda editar.





# Áreas de texto en Swing (JTextArea) III

- `boolean isEditable()`: Este método permite determinar si un área de texto es editable o no, retorna `true` si es editable y `false` si no lo es.
- `void append(String str)`: Este método permite añadir una cadena específica al final del texto actual.
- `void insert(String str, int index)`: Este método permite insertar una cadena de caracteres en un punto especificado.
- `void replaceRange(String str, int startIdx, int endIdx)`: Este método permite reemplazar los caracteres dados en `str` desde `startIdx` hasta `endIdx`.



# Outline

1 Algunas componentes de Swing

2 Práctica de clase



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (I)

- 1 Declarar un atributo botón en la clase `PanelControles` mediante la siguiente instrucción

```
protected JButton jButtonParar = null;
```

- 2 Definir y agregar un botón en el método `initComponents()` de la clase `PanelControles`

```
jButtonParar = new JButton("Parar animación");  
add(jButtonParar);
```

- 3 Agregar al botón `jButtonParar` el auditor eventoClic después de haber definido éste

```
jButtonParar.addActionListener(eventoClic);
```



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (II)

- 4 Declarar un atributo botón en la clase EventoClic mediante la siguiente instrucción

```
protected JButton jButtonParar = null;
```

- 5 Agregar el botón jButtonParar al constructor de la clase EventoClic así

```
EventoClic(JTextArea jTextArea, JButton jButton,  
           JButton jButtonParar, JTextField jTextField,  
           JCheckBox jCheckBox, ...)
```

- 6 Asignar al atributo this.jButtonParar el parámetro jButtonParar del constructor de la clase EventoClic así

```
this.jButtonParar = jButtonParar;
```



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (III)

- 7 Modificar la definición de la variable eventoClic de la clase PanelControles así

```
EventoClic eventoClic = new EventoClic(jTextArea,  
    jButton, jButtonParar, jTextField, jCheckBox,  
    jRadioButtonAdelante, jSpinner);
```

- 8 Declarar un atributo botón en la clase Hilo mediante la siguiente instrucción

```
protected JButton jButtonParar = null;
```

- 9 Agregar el botón jButtonParar al constructor de la clase Hilo así

```
Hilo(JTextArea jTextArea, JButton jButton,  
    JButton jButtonParar, JTextField jTextField,  
    JCheckBox jCheckBox, ...)
```



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (IV)

- 10 Asignar al atributo `this.jButtonParar` el parámetro `jButtonParar` del constructor de la clase `Hilo` así

```
this.jButtonParar = jButtonParar;
```

- 11 Modificar la definición de la variable `hilo` de la clase `EventoClic` así

```
hilo = new Hilo(jTextArea, jButton, jButtonParar,  
               jTextField, jCheckBox, jRadioAdelante, jSpinner);
```

- 12 Declarar un atributo booleano publico en la clase `Hilo` mediante la siguiente instrucción

```
public boolean parar = false;
```



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (VI)

### 13 Agregar la instrucción

```
parar = false;
```

antes del último ciclo for del método run() de la clase Hilo así

```
parar = false;
for (int i = 0;
    i < ParametrosDibujo.ITERACIONES_CIUADAD; i++) {
    if (jRadioAdelante.isSelected()) {
        ciudad.mover();
    }
    ...
}
```



# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (VII)

### 14 Agregar el condicional

```
if(parar){  
    i = ParametrosDibujo.ITERACIONES_CIUADAD;  
}
```

al final del último ciclo for del método run() de la clase Hilo así

```
parar = false;  
for (int i = 0;  
    i < ParametrosDibujo.ITERACIONES_CIUADAD; i++) {  
    ...  
    ...  
    if(parar){  
        i = ParametrosDibujo.ITERACIONES_CIUADAD;  
    }  
}
```





# Trabajo en clase

## Agregar un botón para detener la ejecución de la animación (VII)

- 15 Agregar un condicional alternativo al condicional del método `actionPerformed(ActionEvent ae)` de la clase `EventoClic`

```
public void actionPerformed(ActionEvent ae) {  
    if (ae.getSource() == jButton) {  
        ...  
        thread.start();  
    } else if (ae.getSource() == jButtonParar){  
        if(hilo != null){  
            hilo.parar = true;  
        }  
    }  
}
```

- 16 Ejecutar el programa y probar.

