Técnicas de programación avanzada

Curso 2023-2024

Tema 4.2 Gestión de eventos

Tema 1: Objetos y memoria.

1.1 Características básicas del lenguaje. Primer programa. Compilación y Ejecución. IDE.

1.2 Sentencias de control. Secuencia, selección e iteración.

1.3 Abstracción. Clases, objetos, métodos y atributos.

1.4 Sobrecarga de métodos y encapsulamiento.

Tema 2. Otros conceptos fundamentales de la Programación Orientada a Objetos

2.1 Herencia. Interfaces y clases abstractas. Agregación.

2.2 Polimorfismo.

2.3 Gestión de Excepciones.

2.4 Genericidad y plantillas.

2.5 Utilidades. Entrada y Salida.

2.6 Anotaciones.

Tema 3. Patrones de Diseño.

3.1 Concepto de Patrones de Diseño.

3.2 Patrones de creación.

3.3 Patrones estructurales.

3.4 Patrones de comportamiento.

Tema 4. Programación de Interfaces.

4.1 Interfaces Gráficas de Usuario.

4.2 Gestión de eventos.

Tema 5. Temas Avanzados.

5.1 Concurrencia.

5.2 Inversión de Control. Definición y ejemplos. Inyección de dependencias.

5.3 Expresiones avanzadas del lenguaje.

4.2 Gestión de eventos

Gestionar eventos al interaccionar con los objetos

1. **Componentes de Interfaz de Usuario**: Primero, creas componentes de la interfaz de usuario como botones, campos de texto, etc. Estos componentes pueden generar eventos como clics de botón o entradas de texto.

2. **Eventos**: Un evento es una acción que ocurre debido a la interacción del usuario con los componentes de la interfaz. Por ejemplo, hacer clic en un botón o escribir en un campo de texto.

3. **Listeners** (Escuchas): Para manejar estos eventos, necesitas algo conocido como "listener" o escucha. Un listener es un objeto que está atento a un evento específico. Java proporciona varias interfaces de listener para diferentes tipos de eventos, como ActionListener para acciones de botón, MouseListener para eventos del mouse, etc.

4. **Registrar el Listener con el Componente**: Después de implementar la interfaz del listener, debes registrar este listener con el componente que deseas monitorear. Por ejemplo, si tienes un botón y quieres manejar clics en él, debes registrar tu listener con ese botón usando el método addActionListener.

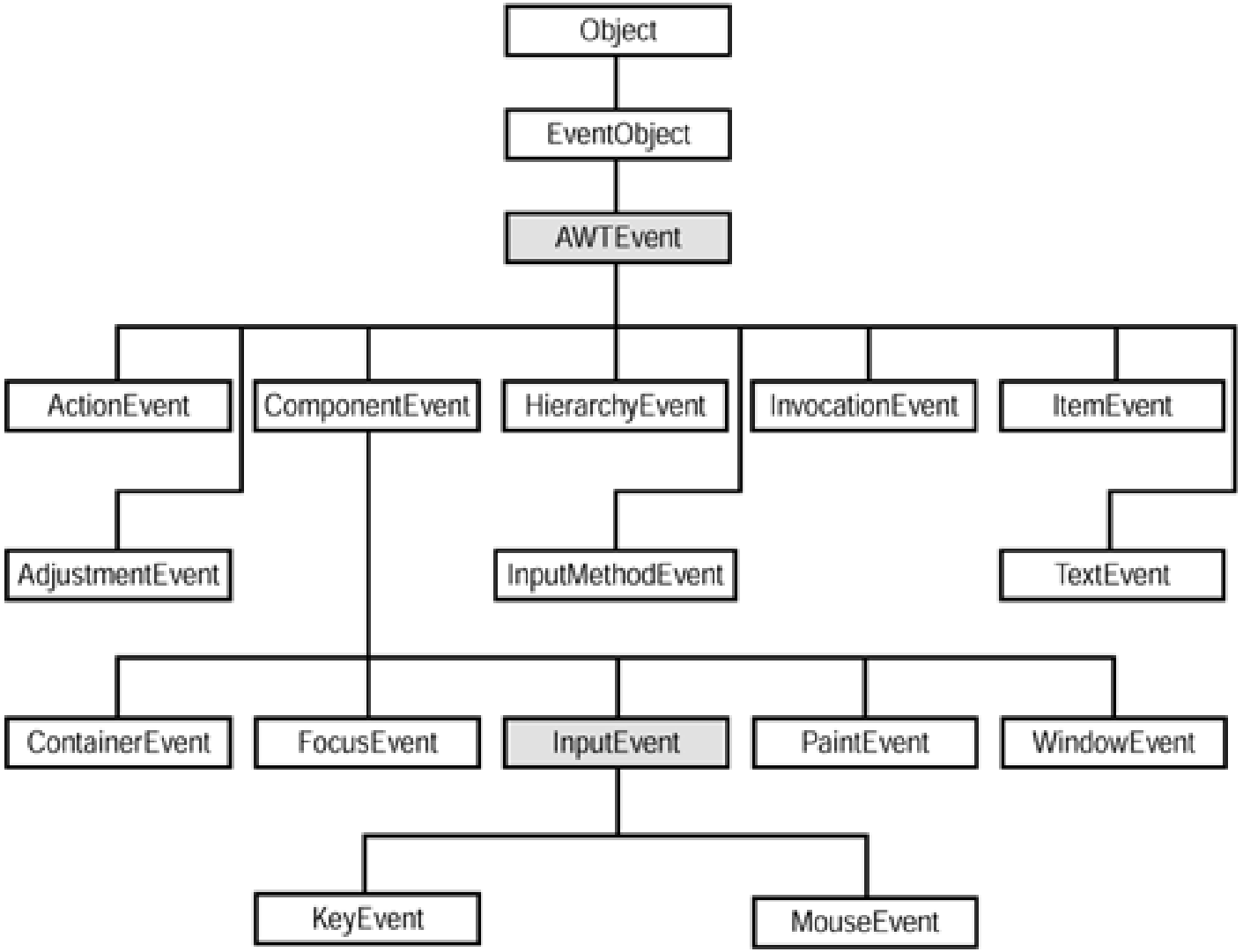
5. **Implementar la Interfaz del Listener**: Debes implementar la interfaz del listener correspondiente en tu clase. Por ejemplo, si quieres manejar clics de botón, tu clase debe implementar ActionListener. Esta interfaz requiere que implementes el método actionPerformed(ActionEvent e).

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

public class MiVentana extends JFrame implements ActionListener
{
    private JButton miBoton;

    public MiVentana() {
        miBoton = new JButton("Haz clic aquí");
        miBoton.addActionListener(this); // listener
        this.add(miBoton);
        this.setSize(300, 200);
        this.setVisible(true);
    }

    // Implementación del método de ActionListener
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == miBoton) {
            System.out.println("Botón pulsado!");
        }
    }

    public static void main(String[] args) {
        new MiVentana();
    }
}
```

4.2 Gestión de eventos

Gestionar eventos al interaccionar con los objetos

Cada componente puede aplicar un tipo determinado de Listeners (Tabla)



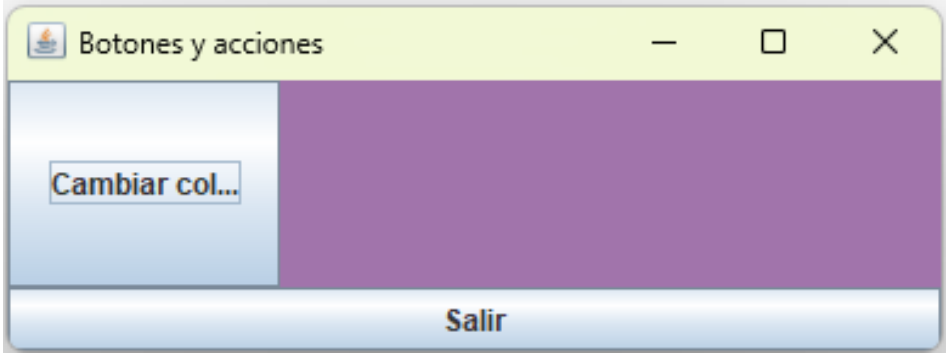
Ejercicios

4.2 Gestión de eventos

Ejercicios

Escriba un programa utilizando Java Swing con las siguientes características:

1. Muestre una ventana gráfica (*Jframe*) que tenga un título
2. En dicho *Jframe* habrá un panel (*JPanel*)
3. Dicho panel estará configurado con *BorderLayout*
4. Dicho panel contendrá dos botones: “Salir” y “Cambiar color”
5. Al pulsar el botón “Salir”, se cerrará el programa
6. Al pulsar el botón “Cambiar color”, se cambiará el color del fondo del panel (método *setBackground(Color c)*)



4.2 Gestión de eventos

Listeners: métodos y eventos

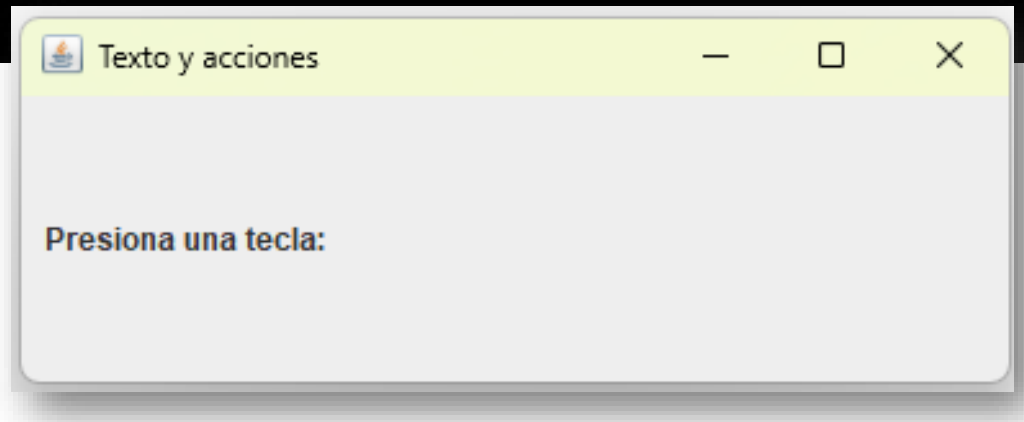
NOMBRE LISTENER	DESCRIPCIÓN	MÉTODOS	EVENTOS
ActionListener	Se produce al hacer click en un componente, también si se pulsa Enter teniendo el foco en el componente.	public void actionPerformed(ActionEvent e)	•**JButton**: click o pulsar Enter con el foco activado en él. •**JList**: doble click en un elemento de la lista. •**JMenuItem**: selecciona una opción del menú. •**JTextField**: al pulsar Enter con el foco activado.
KeyListener	Se produce al pulsar una tecla. según el método cambiara la forma de pulsar la tecla.	public void keyTyped(KeyEvent e) public void keyPressed(KeyEvent e) public void keyReleased(KeyEvent e)	Cuando pulsamos una tecla, segun el Listener: •**keyTyped**:al pulsar y soltar la tecla. •**keyPressed**: al pulsar la tecla. •**keyReleased**: al soltar la tecla.
FocusListener	Se produce cuando un componente gana o pierde el foco, es decir, que esta seleccionado.	public void focusGained(FocusEvent e) public void focusLost(FocusEvent e)	Recibir o perder el foco.
MouseListener	Se produce cuando realizamos una acción con el ratón.	public void mouseClicked(MouseEvent e) public void mouseEntered(MouseEvent e) public void mouseExited(MouseEvent e) public void mousePressed(MouseEvent e) public void mouseReleased(MouseEvent e)	Según el Listener: •**mouseCliked**: pinchar y soltar. •**mouseEntered**: entrar en un componente con el puntero. •**mouseExited**: salir de un componente con el puntero •**mousePressed**: presionar el botón. •**mouseReleased**: soltar el botón.
MouseMotionListener	Se produce con el movimiento del mouse.	public void mouseDragged(MouseEvent e) public void mouseMoved(MouseEvent e)	Según el Listener: •**mouseDragged**: click y arrastrar un componente. •**mouseMoved**: al mover el puntero sobre un elemento

https://www.discoduroderoer.es/eventos-y-listeners-en-java/

4.2 Gestión de eventos

KeyListener – Entradas por teclado

```
public class ExPanelEvents extends JPanel implements KeyListener{

  JLabel result;

  ExPanelEvents(){
  super(new BorderLayout());
  result = new JLabel();
  result.setText("   Presiona una tecla: ");
  add(result,BorderLayout.CENTER);

  addKeyListener(this);

}
```




```
 //KeyListener events
@Override
public void keyTyped(KeyEvent e) {
  result.setText("    Presionado: "
                    + e.getKeyChar());
}

@Override
public void keyPressed(KeyEvent e) { }

@Override
public void keyReleased(KeyEvent e) { }
```

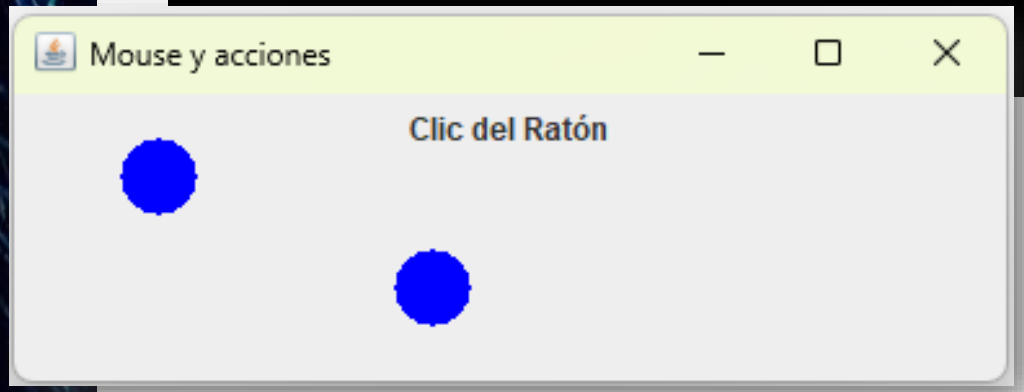
https://github.com/aalonsopuig/Java_Ejemplos_4/tree/main/src/j_eventos_teclado

4.2 Gestión de eventos

MouseListener y MouseMotionListener – Eventos del ratón

```
public class ExPanelMouseListener extends JPanel implements MouseListener, MouseMotionListener{

  Label l;

  ExPanelMouseListener(){
  addMouseListener(this);
  setLayout(null);

  l=new Label();
  l.setBounds(20,50,100,20);
  add(l);
  setSize(300,300);
  setVisible(true);
}
```

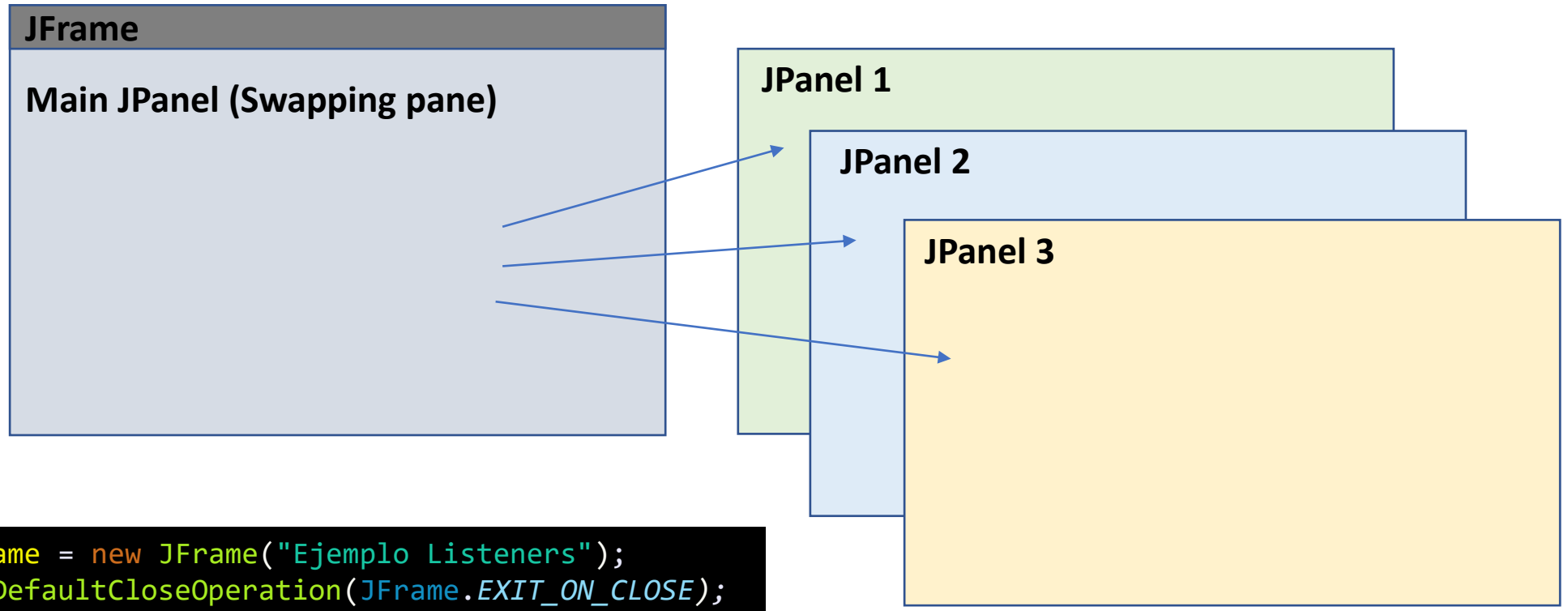
```
public void mouseClicked(MouseEvent e) {
  l.setText("Mouse Clicked");
  Graphics g=getGraphics();
  g.setColor(Color.BLUE);
  g.fillOval(e.getX()-15,e.getY()-15,30,30);}
//MouseListener events
public void mouseEntered(MouseEvent e) {
  l.setText("Mouse Entered");}
public void mouseExited(MouseEvent e) {
  l.setText("Mouse Exited");}
public void mousePressed(MouseEvent e) {
  l.setText("Mouse Pressed");}
public void mouseReleased(MouseEvent e) {
  l.setText("Mouse Released"); }
//MouseMotionListener events
public void mouseDragged(MouseEvent e) {}
public void mouseMoved(MouseEvent e) {}
```



https://github.com/aalonsopuig/Java_Ejemplos_4/tree/main/src/k_eventos_mouse

4.2 Gestión de eventos

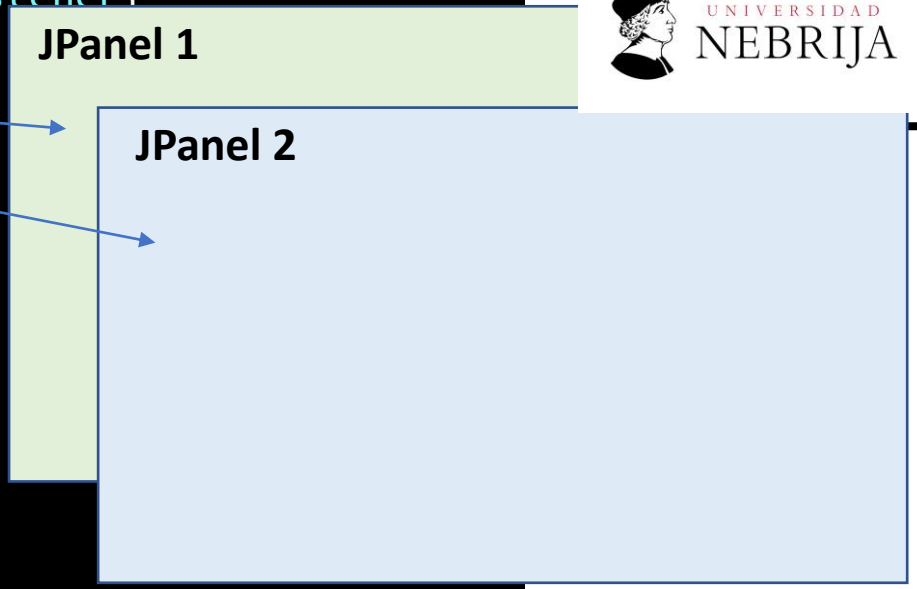
Cambio del contenido de la ventana




```java
JFrame frame = new JFrame("Ejemplo Listeners");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

frame.setBounds(100, 100, 400, 400);

JComponent newContentPane = new SwapingPanel();
frame.setContentPane(newContentPane);

frame.setVisible(true);
```

https://github.com/aalonsopuig/Java_Ejemplos_4/tree/main/src/1_swap_panel

```java
public class SwapingPanel extends JPanel implements ActionListener{

    JPanel firstPanel = new ExPanelEvents();
    JPanel secondPanel = new ExPanelMouseListener();

    public SwapingPanel() {
        super(new BorderLayout());

        JButton swap1 = new JButton("Cambiar ventana");
        swap1.addActionListener(this);
        firstPanel.add(swap1,BorderLayout.NORTH);
        add(firstPanel);
    }

    public void actionPerformed(ActionEvent e) {
        for (Component component : getComponents())
            if (firstPanel == component) {
                remove(firstPanel);
                add(secondPanel);
            } else {
                remove(secondPanel);
                add(firstPanel);
            }
        repaint();
        revalidate();
     }
}
```



https://github.com/aalonsopuig/Java_Ejemplos_4/tree/main/src/1_swap_panel

Ejercicios

4.2 Gestión de eventos

Ejercicios

Escriba un programa utilizando Java Swing con las siguientes características:

1. Muestre una ventana gráfica (*Jframe*) que tenga un título
2. En dicho *Jframe* habrá un panel (*JPanel*)
3. Dicho panel estará configurado con *BorderLayout*
4. Dicho panel contendrá dos botones: “Salir” y “Cambiar color”
5. Al pulsar el botón “Salir”, se cerrará el programa
6. Al pulsar el botón “Cambiar color”, se cambiará el color del fondo del panel (método *setBackground(Color c)*) a un color aleatorio
7. Al hacer clic con el ratón en la zona de color, dibujará un pequeño cuadrado (método *fillRect(int x, int y, int width, int height)*) de colores aleatorios.

Bibliografía

- Deitel, H. M. & Deitel, P. J.(2008). *Java: como programar*. Pearson education. Séptima edición.
- The Java tutorials. https://docs.oracle.com/javase/tutorial/
- Páginas de bibliotecas, tutoriales, etc.:
  - https://docs.oracle.com/javase/tutorial/uiswing/learn/index.html
  - https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.wb.doc.user%2Fhtml%2Findex.html
  - https://www.w3schools.com/java/default.asp
  - https://docstore.mik.ua/orelly/java-ent/jnut/index.htm

Técnicas de Programación Avanzada

```
exit(); //Gracias!
```