Unidad 15

Creación de interfaces gráficas

Programación

Índice

Introducción

Todas las aplicaciones deben de disponer de una interfaz para que el usuario pueda interactuar con la aplicación.

Hasta ahora se ha interactuado con las aplicaciones mediante una consola de comandos y haciendo uso de cadenas de texto, esta forma es valida para ciertos sistemas que no requieran una parte gráfica o por temas de rendimiento. Lo cierto es que la gran mayoría de aplicaciones tienen una interfaz gráfica, aportando facilidad y sencillez a la comunicación entra el usuario y la aplicación.

Java dispone de dos tecnologías, AWT y Swing que permiten hacer diseños de interfaces gráficas para nuestras aplicaciones.

Al finalizar esta unidad

- Aprenderemos conceptos básicos del desarrollo de interfaces.
- Conoceremos las librerías gráficas para el diseño.
- Aprenderemos el uso de los distintos elementos gráficos en una interfaz.
- Desarrollaremos conceptos de eventualidad y accionadores.
- Usaremos herramientas gráficas para el desarrollo de interfaces.

15.1. Librería AWT

Java AWT, Abstract Window Toolkit, es la librería más nativa que permite la creación de interfaces gráficas con el paquete java.awt, al darnos acceso a la clase común Component, las cuales definen los controles y la pantalla de la aplicación en desarrollo respectivamente.

15.1.1. Contenedores y controles

La clase raíz Component contienen todos los elementos (a partir de ahora componentes) que intervienen en una interfaz gráfica. Los componentes pueden ser contenedores o controles.

Los contenedores hacen uso de la clase principal Container, son componentes que definen la estructura de ventanas de una aplicación y pueden albergar otros componentes o controles en su interior. A continuación, se detallan los principales:

- **Windows:** Contenedor en forma de ventana sin bordes, título ni barras de menú.
- **Frame:** Ventana que hace uso del contenedor Windows y tiene barra de título y menú.
- **Dialog:** Ventana que hace uso del contenedor Windows y tiene borde y título.
- **FileDialog:** Ventana de diálogo que permite al usuario seleccionar archivos.
- **Panel:** Contenedor que hace uso de la Ventana Frame y permite agrupar controles dentro de él.

Los controles son componentes contenidos en un contenedor que permiten definir acciones o visualizar contenido en una interfaz. Los más destacados:

- **Button:** Botón con título, permite realizar ciertas acciones, como hacer click encima de él.
- **Checkbox:** Botón de selección. Permite seleccionar varias opciones.
- **Choice:** Desplegable con diferentes opciones.
- **Label:** Etiqueta de información.
- **TextField:** Campo para introducir texto.

Tanto los contenedores como los controles disponen de métodos para definir aspectos visuales o comportamientos, como, por ejemplo, las dimensiones, posición, si esta visible o no, etc.

```
public static void main(String[] args) {

    Frame marco = new Frame();
    marco.setSize(400,400);
    marco.setLayout(null);
    marco.setVisible(true);

    Label label = new Label("Etiqueta");
    label.setBounds(100, 100, 100, 20);

    TextField campo = new TextField();
    campo.setBounds(200, 100, 100, 20);

    Button boton = new Button("Botón");
    boton.setBounds(175, 200, 60, 40);

    marco.add(label);
    marco.add(campo);
    marco.add(boton);

}
```

Imagen 1. Contenedor Frame y controles.

15.1.2. Eventos

Los eventos son acciones que se producen cuando un usuario interactúa con un componente. La principal clase que contienen la funcionalidad de los eventos es EventObject y a partir de ella existen varias subclases dependiendo de la acción, ActionEvent, ComponentEvent, KeyEvent, MouseEvent, InputEvent, ItemEvent, TextEvent, WindowsEvent.

El procedimiento para desarrollar eventos es el siguiente:

1. El usuario realiza una acción concreta sobre el componente
2. Se captura la acción concreta
3. Se realiza la funcionalidad de la acción
4. Se devuelve los resultados

Para añadir un capturador de eventos al componente se realiza mediante el método addXListener(new xListener) donde X es la acción concreta Action, Windows, etc.

```
public static void main(String[] args) {

    Frame marco = new Frame();
    marco.setSize(400,400);
    marco.setLayout(null);
    marco.setVisible(true);

    marco.addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e) {
            marco.dispose();
        }
    });

    Label label = new Label("Etiqueta");
    label.setBounds(100, 100, 100, 20);

    TextField campo = new TextField();
    campo.setBounds(200, 100, 100, 20);

    Button boton = new Button("Botón");

    boton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            campo.setText("Acción del botón");
        }
    });
    boton.setBounds(175, 200, 60, 40);

    marco.add(label);
    marco.add(campo);
    marco.add(boton);

}
```

Imagen 2. Eventos de cierre de frame y acción de botón

15.1.3. Layouts

Los contenedores pueden tener asociado un Layout, traducido como manejador de componentes, los LayoutManager permiten modificar la localización de los componentes y su tamaño, con el fin de que estos se coloquen donde son necesarios, ya que en caso de no especificarse los componentes ocuparán una posición absoluta.

Existen distintos layout de los cuales podemos hacer uso en función de nuestras necesidades.

BorderLayout

Permite colocar elementos en el centro de la ventana, así como en los cuatro bordes, Empezando por el superior y girando en el sentido de las agujas del reloj se denominan NORTH, SOUTH, EAST y WEST, con CENTER como el que se encuentra en el centro.

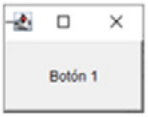
```
public static void main(String[] args) {

    Frame frame = new Frame();
    frame.setSize(400,400);

    BorderLayout borderLayout = new BorderLayout();
    frame.setLayout(borderLayout);

    frame.add(new Button("NORTH"),BorderLayout.NORTH);
    frame.add(new Button("SOUTH"),BorderLayout.SOUTH);
    frame.add(new Button("EAST"),BorderLayout.EAST);
    frame.add(new Button("WEST"),BorderLayout.WEST);
    frame.add(new Button("CENTER"),BorderLayout.CENTER);

    frame.setVisible(true);

}
```

Imagen 3. BorderLayout

CardLayout

Gestiona los componentes en el mismo espacio. Se añaden en una lista y se pueden ir mostrando haciendo uso del método .next().

```
public static void main(String[] args) {

    Frame frame = new Frame();
    frame.setSize(100,100);

    CardLayout cardLayout = new CardLayout();
    frame.setLayout(cardLayout);

    Button button = new Button("Botón 1");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            cardLayout.next(frame);
        }
    });
    frame.add(button);
    Button button2 = new Button("Botón 2");

    button2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            cardLayout.next(frame);
        }
    });
    frame.add(button2);

    frame.setVisible(true);

}
```



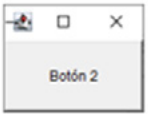


Imagen 4. CardLayout

FlowLayout

Coloca cada componente horizontalmente de izquierda a derecha. Si sobrepasa la fila comienza en la siguiente.

```
public static void main(String[] args) {

    Frame frame = new Frame();
    frame.setSize(100,100);

    FlowLayout flowLayout = new FlowLayout();
    frame.setLayout(flowLayout);

    frame.add(new Button("1"));
    frame.add(new Button("2"));
    frame.add(new Button("3"));
    frame.add(new Button("4"));
    frame.add(new Button("5"));


    frame.setVisible(true);

}
```

Imagen 5. FlowLayout

GridLayout

Similar a FlowLayout pero esta vez sigue el patrón de una tabla, yendo de izquierda a derecha y de arriba hacia abajo, distribuyendo cada elemento en filas y columnas.

```
public static void main(String[] args) {

    Frame frame = new Frame();
    frame.setSize(100,100);

    GridLayout GridLayout = new GridLayout();
    GridLayout.setColumns(2);
    GridLayout.setRows(3);
    frame.setLayout(GridLayout);

    frame.add(new Button("1"));
    frame.add(new Button("2"));
    frame.add(new Button("3"));
    frame.add(new Button("4"));
    frame.add(new Button("5"));

    frame.setVisible(true);

}
```

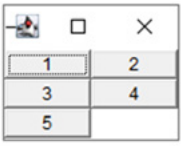


Imagen 6. GridLayout

15.1.4. Menús

Otro componente disponible son los menús. Los menús se situación en la parte de arriba de la ventana y permite organizar acciones de la aplicación. La clase principal para desarrollar un menú es MenuComponent y sus posibles controles son: MenuBar, Menu, MenuItem, MenuShortcut, PopupMenu, CheckboxMenuItem.

```
public static void main(String[] args) {

    Frame frame = new Frame();
    frame.setSize(220,200);
    frame.setVisible(true);

    MenuBar menuBar = new MenuBar();

    Menu menuFile = new Menu("File");
    MenuItem itemNew = new MenuItem("New");
    MenuItem itemOpen = new MenuItem("Open");
    MenuItem itemSave = new MenuItem("Save");
    menuFile.add(itemNew);
    menuFile.add(itemOpen);
    menuFile.add(itemSave);

    Menu menuEdit = new Menu("Edit");
    MenuItem itemSelectA = new MenuItem("Select All");
    menuEdit.add(itemSelectA);

    Menu menuExit = new Menu("Exit");

    menuBar.add(menuFile);
    menuBar.add(menuEdit);
    menuBar.add(menuExit);

    frame.setMenuBar(menuBar);

}
```

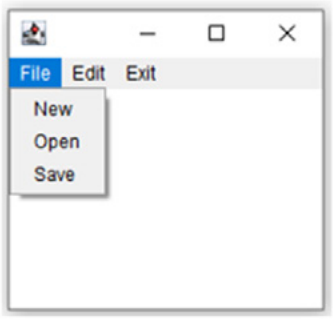


Imagen 7. Ejemplo de menú

15.2. Librería Swing

Swing es una evolución de AWT que incorpora nuevas herramientas al tiempo que elimina algunas limitaciones, dándonos una mayor libertad de trabajo. La principal diferencia es que Swing dispone de todos los controles e incluso puede definir nuevos permitiendo desde Java que los pueda manejar, en cambio, AWT usa los controles nativos del sistema operativo sin poder hacerlos propios.

La gran mayoría de compontes descritos para AWT existen para Swing y se pueden diferenciar al apreciar la letra J delante del nombre de la clase del componente. Por ejemplo, JButton es de swing y Button de AWT.

El paquete para el uso de Swing es javax.swing.

15.2.1. Contenedores y controles

En Swing también se distinguen entre contenedores y controles. La gran mayoría son los mismos que AWT solo añadiendo la letra J delante.

Los principales contenedores:

- **JFrame:** Ventana que hace uso del contenedor Windows y tiene barra de título y menú.
- **JDialog:** Ventana que hace uso del contenedor Windows y tiene borde y título.
- **JPanel:** Contenedor que hace uso de la Ventana Frame y permite agrupar controles dentro de él.

A parte de los controles de AWT se amplía con los siguientes:

- **JComboBox:** Lista de opciones seleccionables.
- **JColorChooser:** Permite seleccionar una paleta de colores.
- **JPaswordFiel:** Es un campo de texto, pero lo que se escriba se oculta con "*"
- **JProgressBar:** Una barra de progreso.
- **JTable:** Representa una tabla para mostrar datos.
- **JTabbedPane:** Agrupación de pestañas y por cada pestaña una agrupación de controles.
- **JTree:** Representa una estructura en jerarquía.

```
public static void main(String[] args) {

    JFrame jFrame =  new JFrame();
    jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jFrame.setSize(200,200);
    jFrame.setVisible(true);

    JTabbedPane panelSolapas = new JTabbedPane();
    JPanel jPanel1 = new JPanel();

    JLabel jLabel = new JLabel("JLabel1");
    JTextField campo = new JTextField("JTextField");
    JLabel jLabel1 = new JLabel("JPasswordField");
    JPasswordField passwordField = new JPasswordField("JPasswordField");
    JButton boton = new JButton("Botón");

    jPanel1.add(jLabel);
    jPanel1.add(campo);
    jPanel1.add(jLabel1);
    jPanel1.add(passwordField);
    jPanel1.add(boton);

    panelSolapas.add("Pestaña1", jPanel1);

    JPanel jPanel2 = new JPanel();
    DefaultMutableTreeNode raiz = new DefaultMutableTreeNode("JTree");
    DefaultMutableTreeNode n1 = new DefaultMutableTreeNode("Node 1");
    DefaultMutableTreeNode ns1 = new DefaultMutableTreeNode("Node 1.1");
    raiz.add(n1);
    n1.add(ns1);
    JTree arbol = new JTree(raiz);
    jPanel2.add(arbol);

    JProgressBar progressBar = new JProgressBar(0,100);
    progressBar.setValue(25);
    jPanel2.add(progressBar);

    panelSolapas.add("Pestaña2",jPanel2);

    jFrame.getContentPane().add(panelSolapas);

}
```

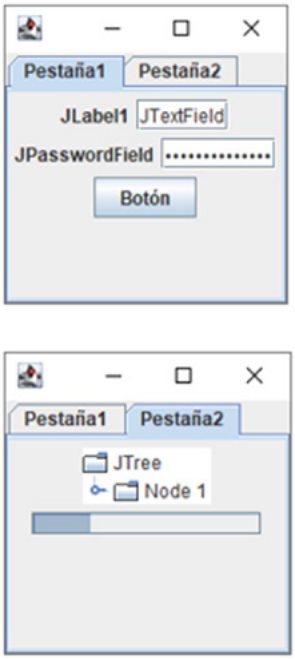


Imagen 8. JFrame con diferentes controladores

15.2.2. Eventos

El procedimiento de eventos es idéntico a AWT. Mirar el punto 15.1.2.

15.2.3. Layouts

Igual que sucedía en AWT los layouts cumplen la misma funcionalidad de organización de los controles de un contenedor. Existen los mismos Layouts que en AWT y para Swing vamos a explicar los nuevos layouts más usados.

GridBagLayout

En mucho más flexible que GridLayout, permitiendo colocar cada componente en la celda deseada de una cuadricula ya hecha, al tiempo que se le asocia otro componente de tipo GridBagConstraints. Las columnas y filas de la cuadrícula se encuentran numeradas comenzando desde cero.

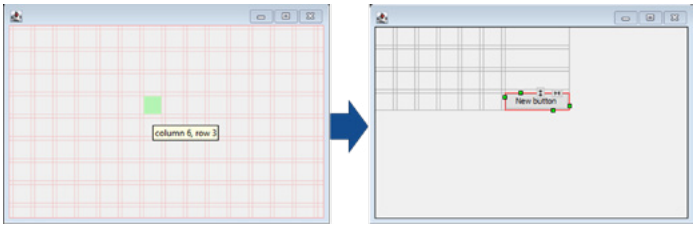


Imagen 9. GridBagLayout

15.2.4. Modificando la apariencia

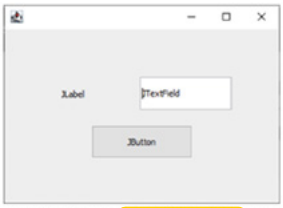
Swing permite cambiar la apariencia con que se muestran los controles. Esta característica se denomina Look & Feel y es posible cambiarla a 5 estilos predefinidos:

- MetalLookAndFeel
- NimbusLookAndFeel
- MotifLookAndFeel
- WindowsLookAndFeel
- WindowsClassicLookAndFeel

El estilo por defecto es el MetalLookAndFeel y es posible cambiarlo con la siguiente línea UIManager.setLookAndFeel(estilo)

```
public static void main(String[] args) {

    JFrame frame = new JFrame();
    frame.setBounds(100, 100, 450, 300);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    try {
        UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");

    } catch (ClassNotFoundException | InstantiationException | IllegalAccessException
            | UnsupportedLookAndFeelException e2) {
        e2.printStackTrace();
    }

    JLabel lblNewLabel = new JLabel("JLabel");
    lblNewLabel.setBounds(74, 64, 81, 39);

    JTextField txtJtextfield = new JTextField();
    txtJtextfield.setText("JTextField");
    txtJtextfield.setBounds(176, 61, 119, 42);
    txtJtextfield.setColumns(10);

    JButton btnNewButton = new JButton("JButton");
    btnNewButton.setBounds(113, 123, 131, 42);

    frame.getContentPane().add(lblNewLabel);
    frame.getContentPane().add(txtJtextfield);
    frame.getContentPane().add(btnNewButton);

    frame.setVisible(true);

}
```

javax.swing.plaf.metal.MetalLookAndFeel



com.sun.java.swing.plaf.windows.WindowsLookAndFeel

com.sun.java.swing.plaf.motif.MotifLookAndFeel

Imagen 10. Aplicación de estilos

Otra posibilidad para cambiar el aspecto de algunos controles es incrustar código HTML embebido. De esta forma es posible aplicar los estilos procedentes de HTML a nuestros controles.

```
public static void main(String[] args) {

    JFrame frame = new JFrame();
    frame.setBounds(100, 100, 400, 400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    String html = "<html>"
                + "<p>"
                + "<font face=\"Comic Sans MS,arial,verdana\" size=\"5\" color=\"green\">Ejemplo de estilo HTML</font>"
                + "</p>"
                + "</html>";

    JLabel lblNewLabel = new JLabel(html);
    lblNewLabel.setBounds(74, 64, 350, 150);

    frame.getContentPane().add(lblNewLabel);

    frame.setVisible(true);

}
```

Imagen 11. Estilos con HTML

15.3. Herramientas gráficas

Trabajar directamente interfaces gráficas es costoso debido a la cantidad de código que se genera, la multitud de controles y propiedades que existen. Existen herramientas gráficas que nos facilitan esta tarea permitiendo hacer drag a drop de los controles y que se generé automáticamente el código.

Por defecto **NetBeans** ya tiene un módulo que nos ofrece tener la herramienta gráfica para el desarrollo de interfaces. Con **Eclipse** si no lleva el módulo instalado será necesario instalarnos la herramienta WindowBuilder desde el Marketplace ubicado en el menú de ayuda.

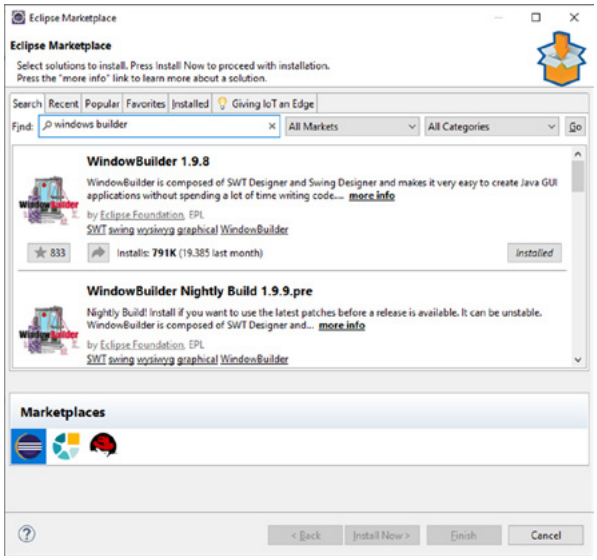


Imagen 12. Instalación WindowBuilder desde Eclipse

Independientemente se use NetBeans o Eclipse el funcionamiento de la herramienta gráfica es idéntico. Por cada clase nos permitirá tener dos pestañas activas, una vista de código (Source) y una vista gráfica (Design).

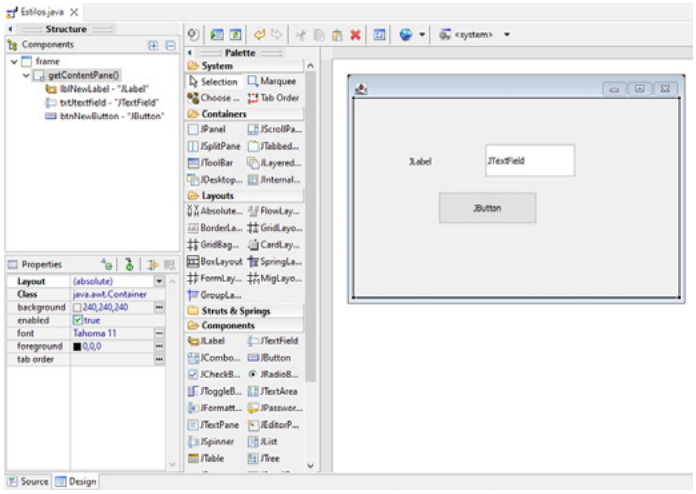


Imagen 13. Pestaña Design del modo gráfico para diseño de interfaces.

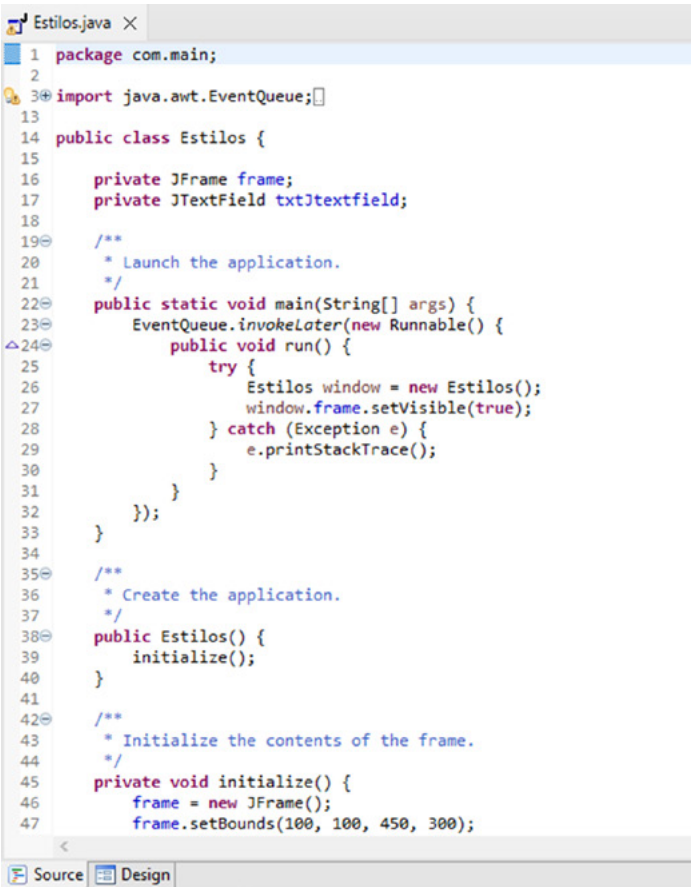


Imagen 14. Pestaña Source para diseño de interfaces.

Como se puede ver en las imágenes anteriores, en la pestaña Design existe tres partes de edición:

- Una paleta con los principales contenedores y controles, pudiendo hacer drag and drop de cada uno de ellos sobre la ventana.
- A la vez por el contenedor o control seleccionado podemos ver sus propiedades para poder aplicar estilos y los eventos que pueden producir.
- Y a modo resumen tenemos una jerarquía de todos los contenedores y controles que contiene la ventana activa.

Automáticamente cada acción que se realice en el modo gráfico se genera el código correspondiente en la pestaña Source.

Es posible trabajar directamente sobre la pestaña design como source, cualquier cambio que se realice se replica en ambas pestañas.

www.universae.com