Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Unidad 5

Patrón MVC
Introducción a GUI en Java

Hechos QUE CONECTAN

Patrón MVC - Modelo Vista Controlador

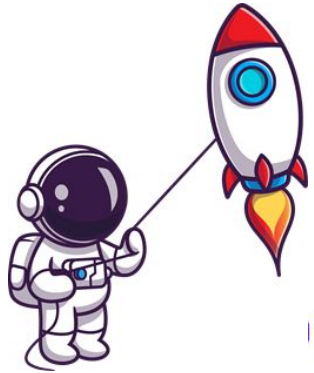
Es un **patrón de arquitectura** de las aplicaciones software.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman **Modelos**, **Vistas** y **Controladores**.

MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

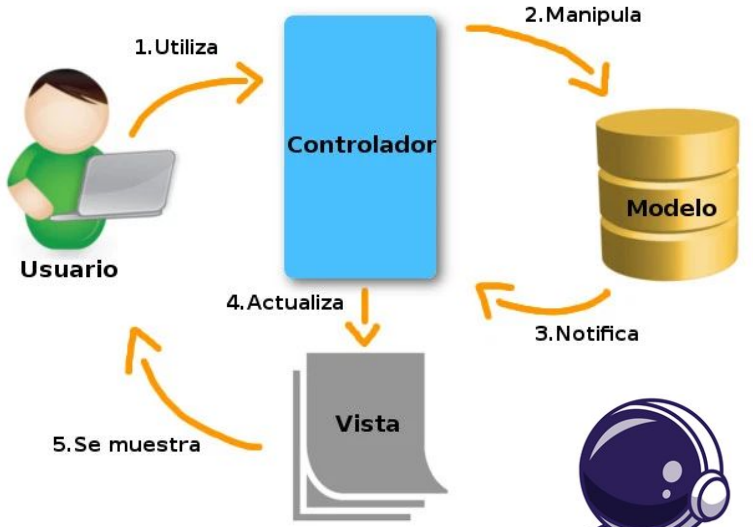
Historia MVC

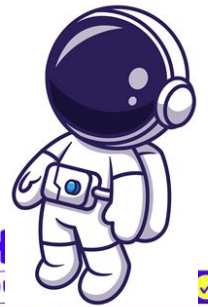
- Descrito por primera vez en 1979 para Smalltalk (Link)
- Utilizado en múltiples frameworks
  - Java Swing
  - Java Enterprise Edition (J2EE)
  - XForms (Formato XML estándar del W3C para la especificación de un modelo de proceso de datos XML e interfaces de usuario como formularios web)
  - GTK+ (escrito en C, toolkit creado por Gnome para construir aplicaciones gráficas, inicialmente para el sistema X Window)
  - ASP.NET MVC Framework (Microsoft)
  - Apache Struts (framework para aplicaciones web J2EE)
  - Ruby on Rails (framework para aplicaciones web con Ruby)
  - Etc.



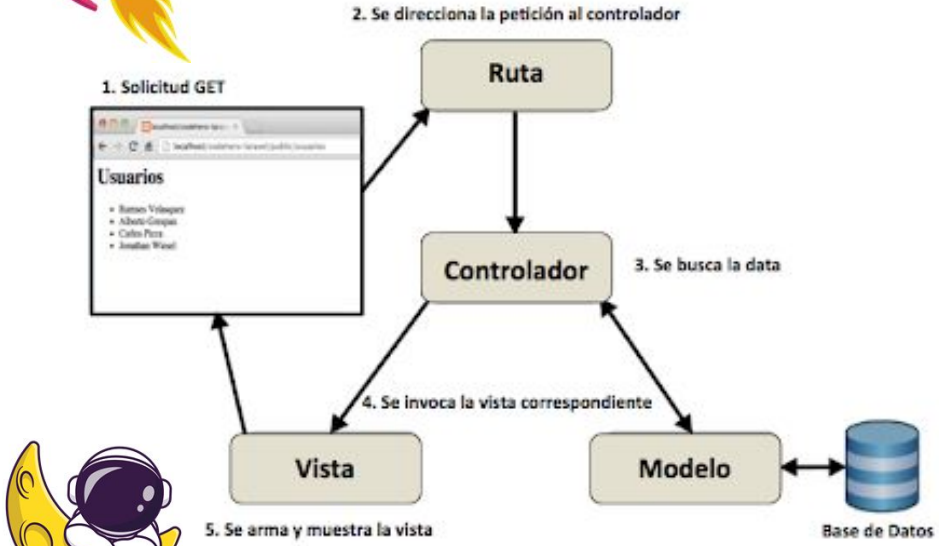
El patrón MVC

- El **Modelo**: Es la representación de la información, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).
- El **Controlador**: Responde a eventos (usualmente del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo'.
- La **Vista**: Presenta el 'modelo' (información) en un formato adecuado para interactuar usualmente con el usuario.





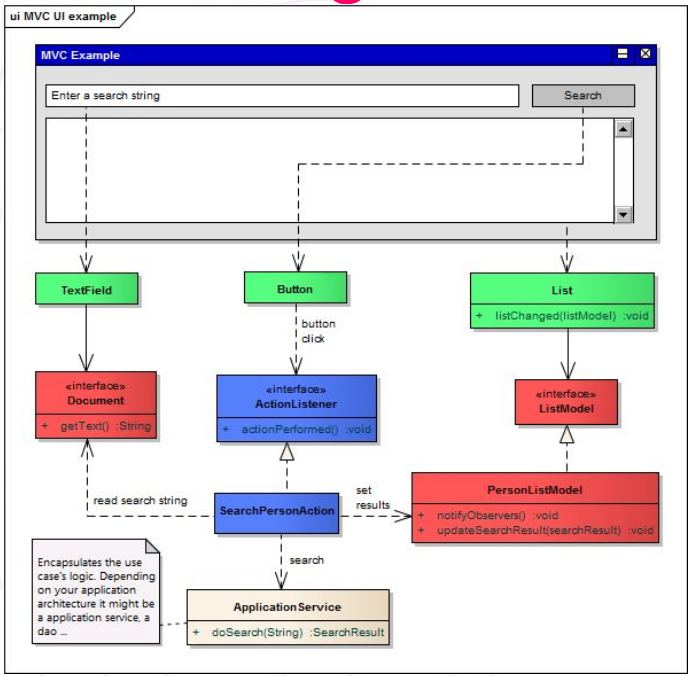
MVC en aplicaciones web



- **Vista**: la página HTML
- **Controlador**: código que obtiene datos dinámicamente y genera el contenido HTML
- **Modelo**: la información almacenada en una base de datos o en XML, junto con las reglas de negocio que transforman esa información (teniendo en cuenta las acciones de los usuarios)

MVC en Java Swing

- **Modelo**: El modelo lo realiza el desarrollador.
- **Vista**: Conjunto de objetos de clases que heredan de java.awt.Component.
- **Controlador**: El controlador es el thread de tratamiento de eventos, que captura y propaga los eventos a la vista y al modelo. Clases de tratamiento de los eventos (a veces como clases anónimas) que implementan interfaces de tipo EventListener (ActionListener, MouseListener, WindowListener, etc.)



Ventajas del MVC

- Fácil organización del código en tres componentes diferentes.
- Crea independencia del funcionamiento.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de otras capas.
- Si trabaja con un equipo de programadores entonces les da una mayor facilidad para poder seguir el trabajo entre varios integrantes.
- Facilita el mantenimiento en caso de errores.
- Hacen que las aplicaciones sean fácilmente extensibles.
- Poder adaptarse a los frameworks de hoy en día.

Desventajas del MVC

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos sencillos.

Vamos al código

- Descarguemos el proyecto Calculadora-MVC
  https://github.com/cesardiaz-utp/MisionTIC2022-Ciclo2-Unidad5-MVC
- Revisar el funcionamiento de la aplicación cambiando en el método **App.main()** la variable **tipo** entre:
  - TipoVista.CONSOLA
  - TipoVista.SUMA_GUI
  - TipoVista.RESTA_GUI

Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Introducción a GUI en Java

Conceptos básicos

Hechos QUE CONECTAN

Interfaces Gráficas de Usuario

Hasta ahora hemos desarrollado programas que usan la consola para interactuar con el usuario.

Esa forma de interfaz de usuario es muy simple y nos ha permitido centrarnos en todo aquello que tiene que ver tan sólo con la programación orientada a objetos con el lenguaje Java, sin tener que tratar al mismo tiempo con ventanas, botones y otros elementos similares.

Las interfaces gráficas de usuario (GUI) ofrecen al usuario ventanas, cuadros de diálogo, barras de herramientas, botones, listas desplegables y muchos otros elementos con los que ya estamos muy acostumbrados a tratar.

Las aplicaciones son conducidas por eventos y se desarrollan haciendo uso de las clases que para ello nos ofrece la API de Java

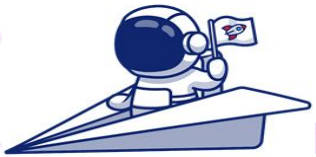
Interfaces Gráficas de Usuario

La interfaz de usuario es la parte del programa que permite al usuario interaccionar con él.

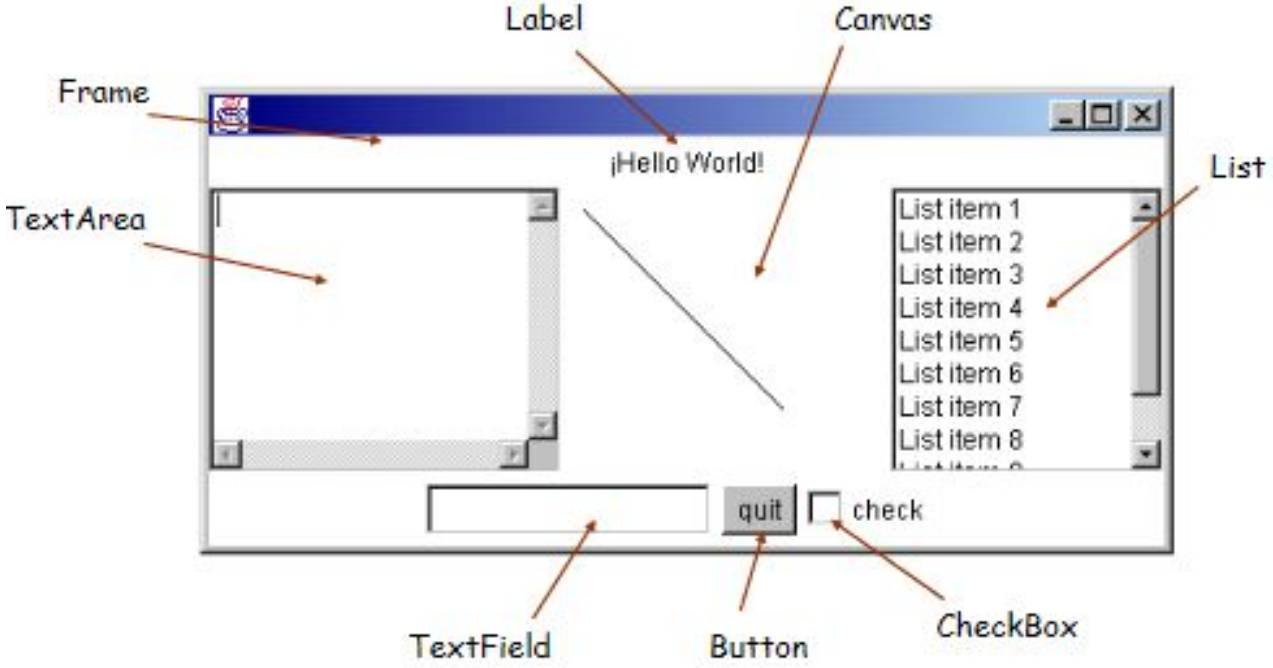
La API de Java proporciona una biblioteca de clases para el desarrollo de Interfaces gráficas de usuario (en realidad son dos: AWT y Swing).

La biblioteca proporciona un conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en las que se ejecuten.

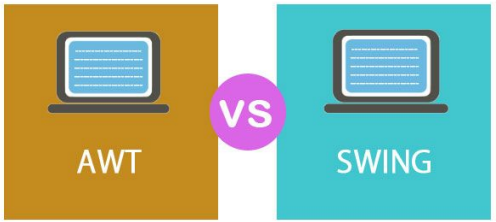
La estructura básica de la biblioteca gira en torno a **componentes** y **contenedores**. Los contenedores contienen componentes y son componentes a su vez, de forma que los eventos pueden tratarse tanto en contenedores como en componentes.



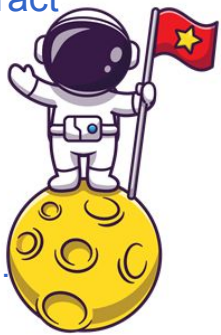
AWT - Abstract Window Toolkit



AWT



En Java, existen dos APIs para la programación de interfaces gráficos de usuario AWT (Abstract Window Toolkit) y Swing.

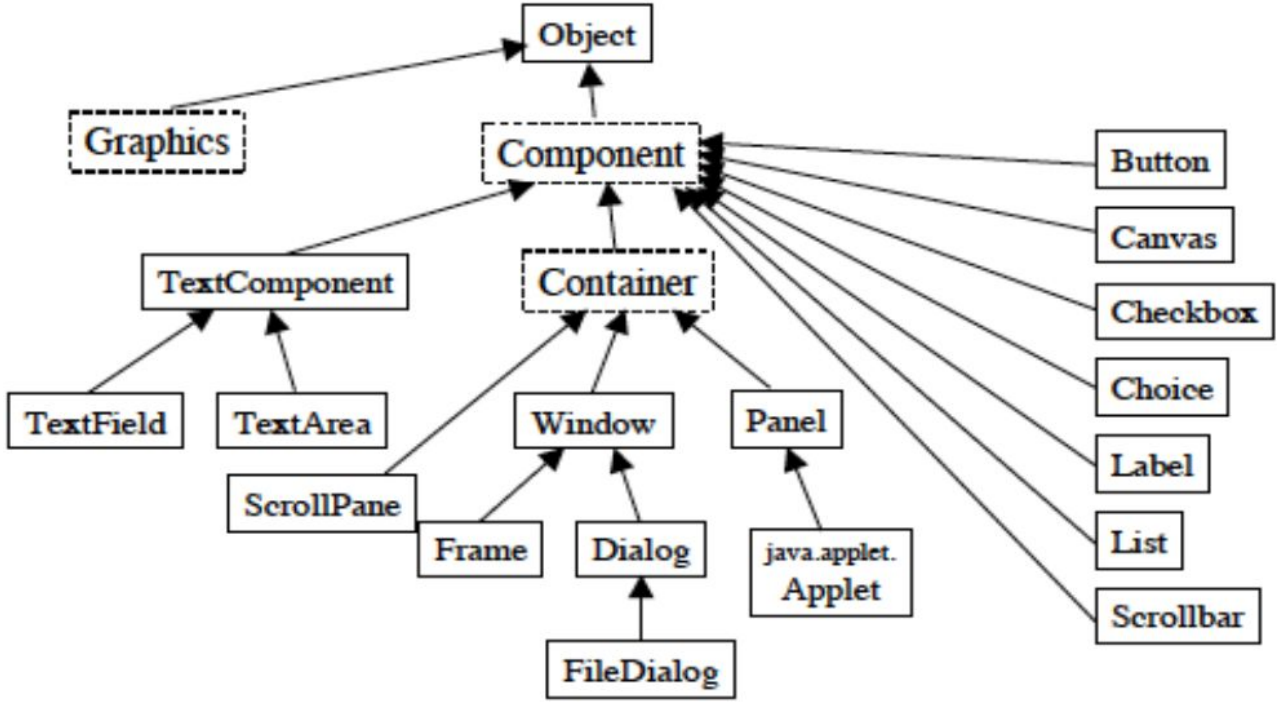


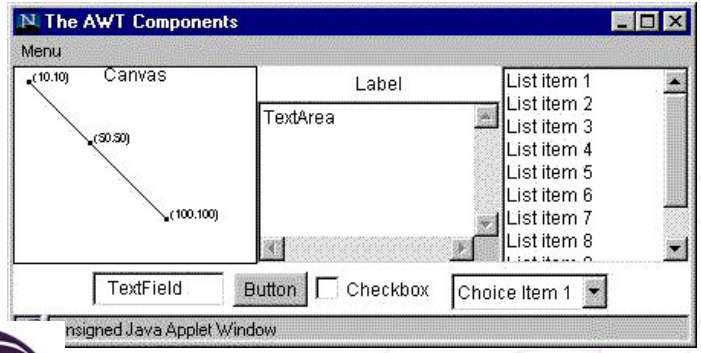
AWT fue la primera API disponible en Java y sus principales características son:

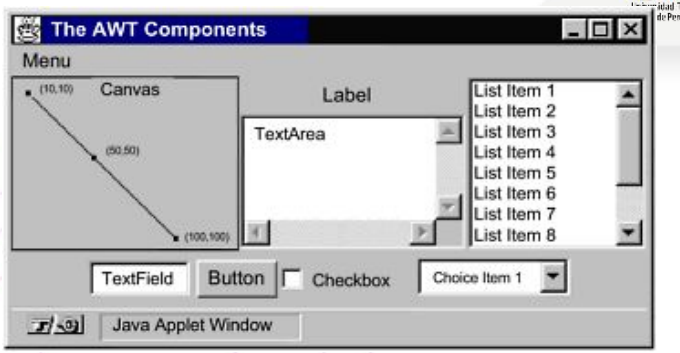
- La creación de componentes gráficos se delega al Sistema Operativo.
- El Sistema Operativo se encarga de dibujar los componentes gráficos y de la detección de eventos.
- El aspecto de la aplicación es el nativo del Sistema Operativo.

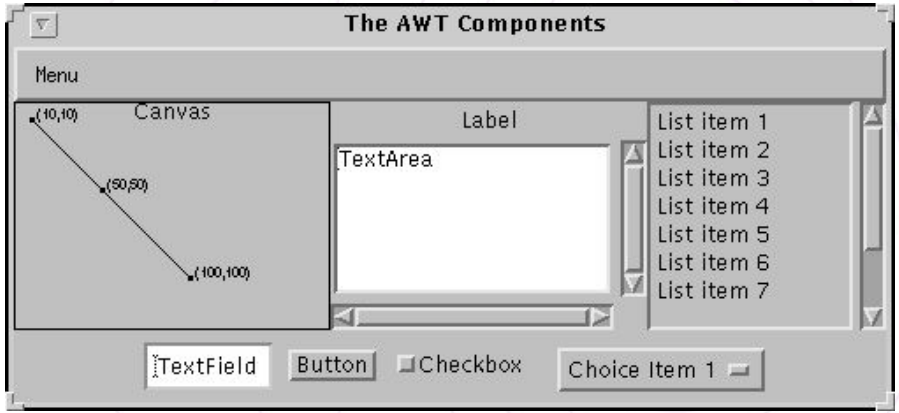
La principal desventaja de AWT es que descansa directamente sobre el Sistema Operativo quien interviene tanto en la creación de componentes gráficos como en la detección de eventos, de modo que la aplicación se puede ralentizar si la interfaz contiene muchos elementos gráficos, y por otro lado no se pueden introducir cambios en el aspecto de los componentes.

Jerarquía de Clases AWT



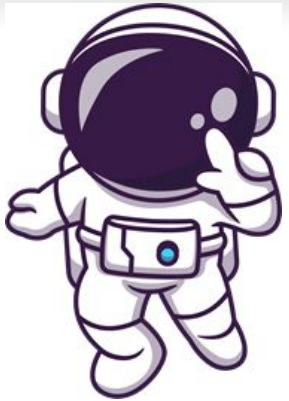






Ejemplo de ventanas en AWT

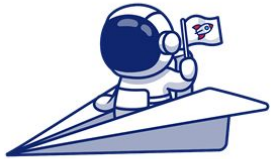
Swing




AppletViewer: MiApplet10.class
Subprograma
ToggleButton
RadioButton
CheckBox
Grupo de botones
1
2
3
Boton
Esto es un botón con imagen incrustada
Subprograma iniciado.

Swing

El API Swing viene a liberar la creación de interfaces gráficos de la carga que supone la dependencia con respecto al Sistema Operativo.



Las principales características de este API son:

- Swing se encarga de dibujar los componentes y de detectar la interacción sobre ellos.
- El conjunto de componentes es más grande que el que proporciona el Sistema Operativo.
- Se tiene control absoluto sobre el aspecto de los componentes.

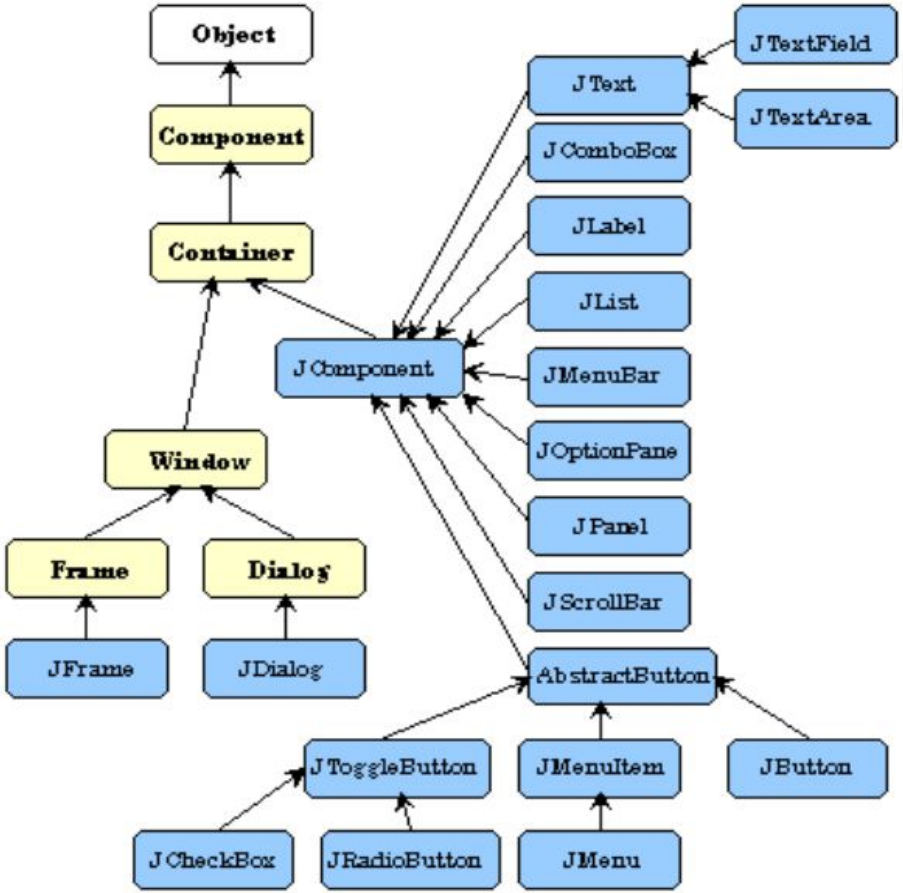
Por todo ellos, Swing ha ido desplazando a AWT en la creación de interfaces gráficos de usuario en Java.

https://docs.oracle.com/javase/tutorial/uiswing/index.html

Jerarquía de Clases Swing

La biblioteca Swing está construida sobre el conjunto de herramientas de widgets abstractos de Java (AWT), un kit de herramientas GUI más antiguo que depende de la plataforma.

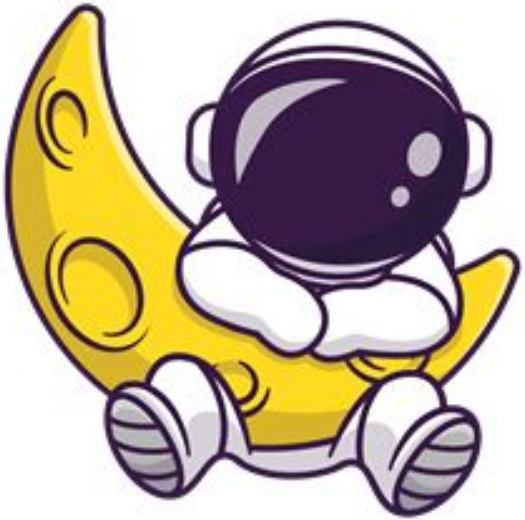
Puede utilizar los componentes de la GUI de Java como el botón, el cuadro de texto, etc. de la biblioteca y no tiene que crear los componentes desde cero.

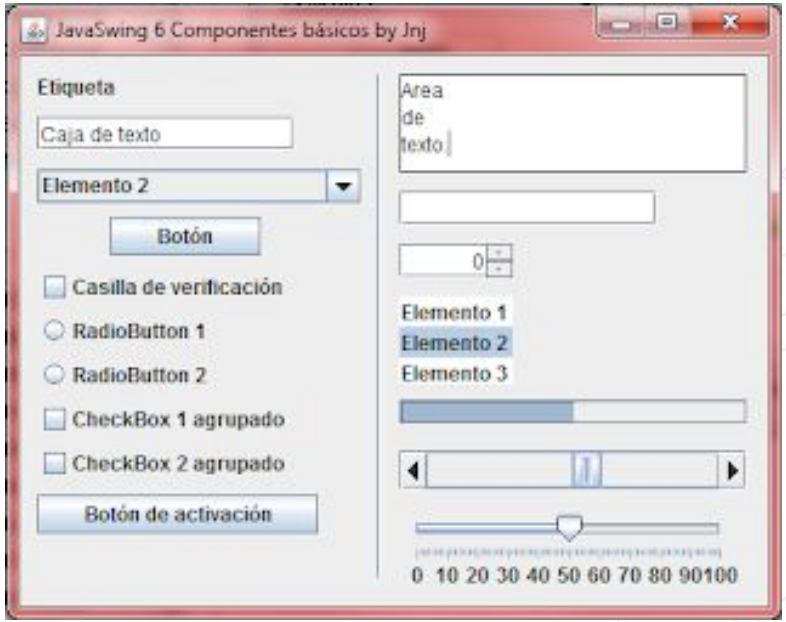


Jerarquía de Clases - Swing

Categorías de clases:

- Contenedores:
  - JFrame, JApplet, JWindow, JDialog
- Componentes intermedios:
  - JPanel, JScrollPane
- Componentes:
  - JLabel, JBbutton, JTextField, JTextArea, ...
- Clases de soporte:
  - Graphics, Color, Font, ...





Ejemplo de ventanas en Swing

Ejemplo de GUI

```
import javax.swing.*;

public class PrimeraGUI {

    public static void main(String args[]) {
        JFrame frame = new JFrame("Mi primera GUI");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);
        JButton button1 = new JButton("Presionar");
        frame.getContentPane().add(button1);
        frame.setVisible(true);
    }
}
```

¿Qué es una clase de contenedor?

Las clases de contenedor son clases que pueden tener otros componentes. Entonces, para crear una GUI, necesitamos al menos un objeto contenedor.

Hay 3 tipos de contenedores.

- **Panel**: es un contenedor puro y no es una ventana en sí misma. El único propósito de un Panel es organizar los componentes en una ventana.
- **Marco**: es una ventana en pleno funcionamiento con su título e iconos.
- **Diálogo**: se puede considerar como una ventana emergente que aparece cuando se debe mostrar un mensaje. No es una ventana completamente funcional como el Marco.

Contenedores de alto nivel:

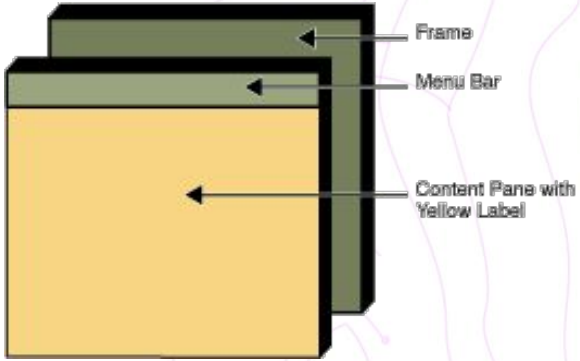
- **JFrame**: Habitualmente la clase JFrame se emplea para crear la ventana principal de una aplicación en Swing.
- **JDialog**: Ventanas de interacción con el usuario.

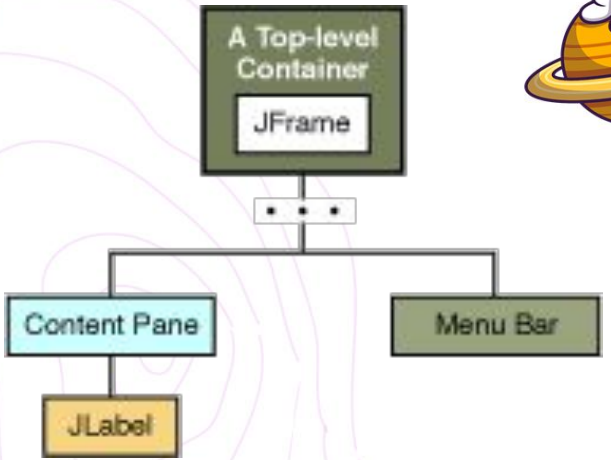
Contenedores intermedios:

- **JPanel**: Agrupa a otros componentes.
- **JScrollPanel**: Incluye barras de desplazamiento.

JFrame

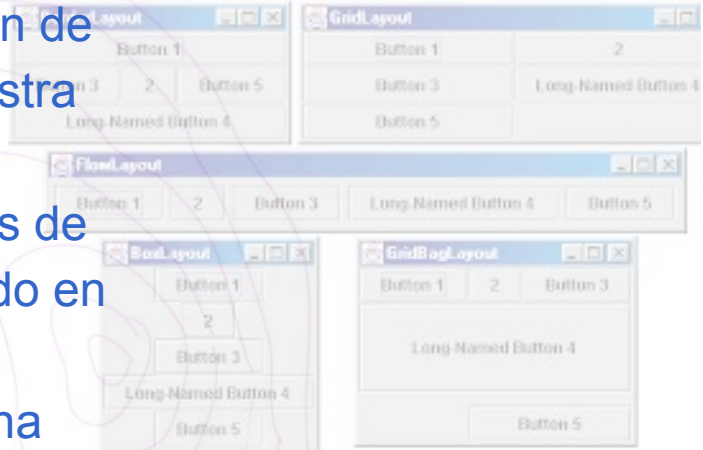

TopLevelDemo

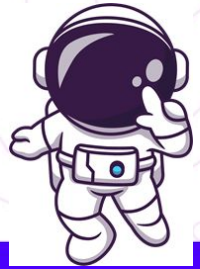

Frame
Menu Bar
Content Pane with
Yellow Label


A Top-level
Container
JFrame
Content Pane
Menu Bar
JLabel

Gestores de aspecto (Layout Manager)

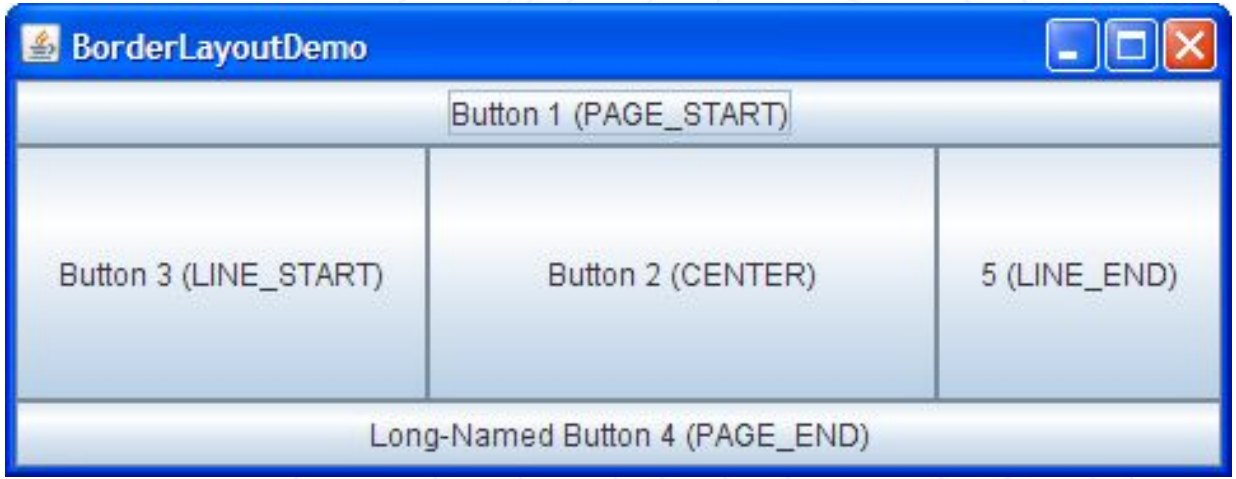
- Cuando se programan interfaces gráficos de usuario, un aspecto importante es la colocación de los Componentes dentro de la ventana de nuestra aplicación.
- Estos Gestores de Aspecto son los encargados de colocar los Componentes que vamos añadiendo en los Contenedores.
- Cada uno de los Gestores de Aspecto sigue una política de colocación de los componentes.





java.awt.BorderLayout

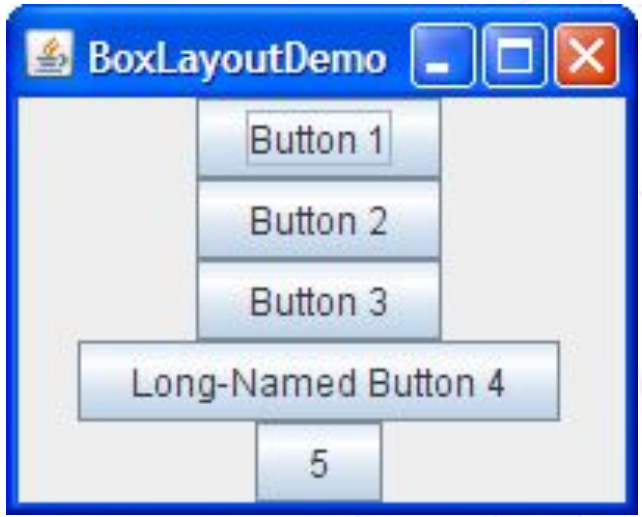
Coloca componentes en hasta cinco áreas: **PAGE_START**(arriba), **PAGE_END** (abajo), **LINE_START** (izquierda), **LINE_END** (derecha) y **CENTER** (centro).
Es el administrador de diseño predeterminado para cada java JFrame

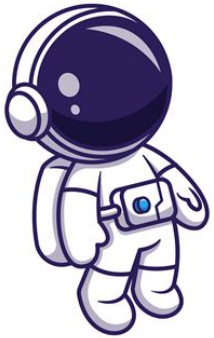


https://docs.oracle.com/javase/tutorial/uiswing/layout/border.html

javax.swing.BoxLayout

BoxLayout apila sus componentes uno encima del otro o los coloca en una fila, su elección

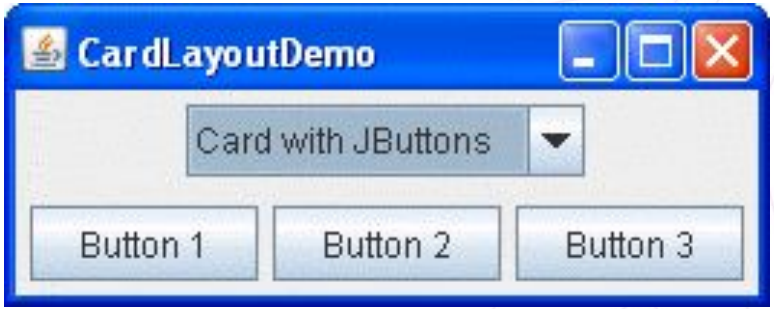


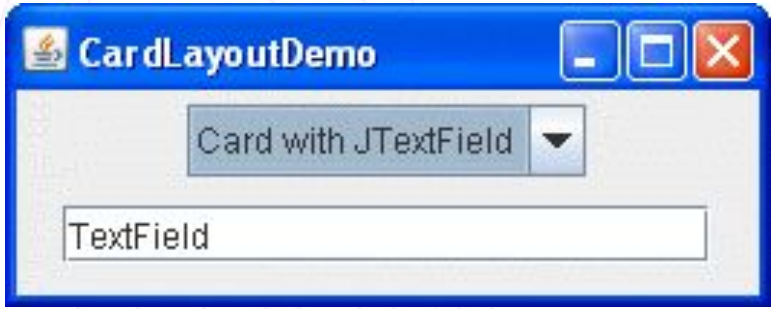


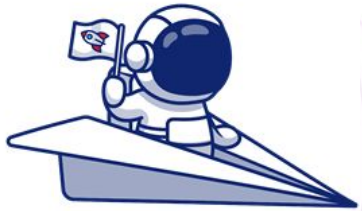
https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html

java.awt.CardLayout

Administra dos o más componentes (generalmente instancias de JPanel) que comparten el mismo espacio de visualización. Cuando utilice la clase CardLayout, deje que el usuario elija entre los componentes mediante un cuadro combinado.







https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html

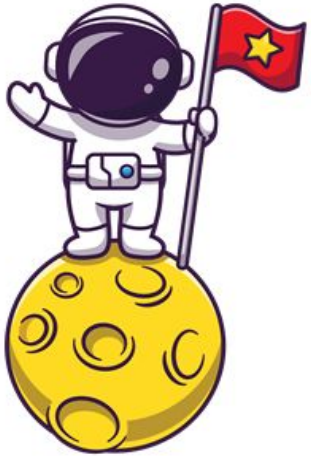
java.awt.FlowLayout

Coloca los componentes en una fila, dimensionados según su tamaño preferido. Si el espacio horizontal en el contenedor es demasiado pequeño para poner todos los componentes en una fila, usa varias filas. Si el contenedor es más ancho de lo necesario para una fila de componentes, la fila está, por defecto, centra horizontalmente dentro del contenedor (puede modificarse este comportamiento).

https://docs.oracle.com/javase/tutorial/uiswing/layout/flow.html

java.awt.GridLayout

Coloca componentes en una cuadrícula de celdas. Cada componente ocupa todo el espacio disponible dentro de su celda y cada celda tiene exactamente el mismo tamaño. Si se cambia el tamaño de la ventana, el objeto el tamaño de la celda para que las celdas sean lo más grandes posible, dado el espacio disponible para el contenedor.



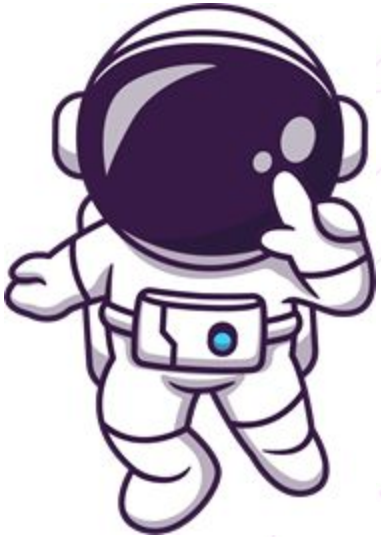
https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html

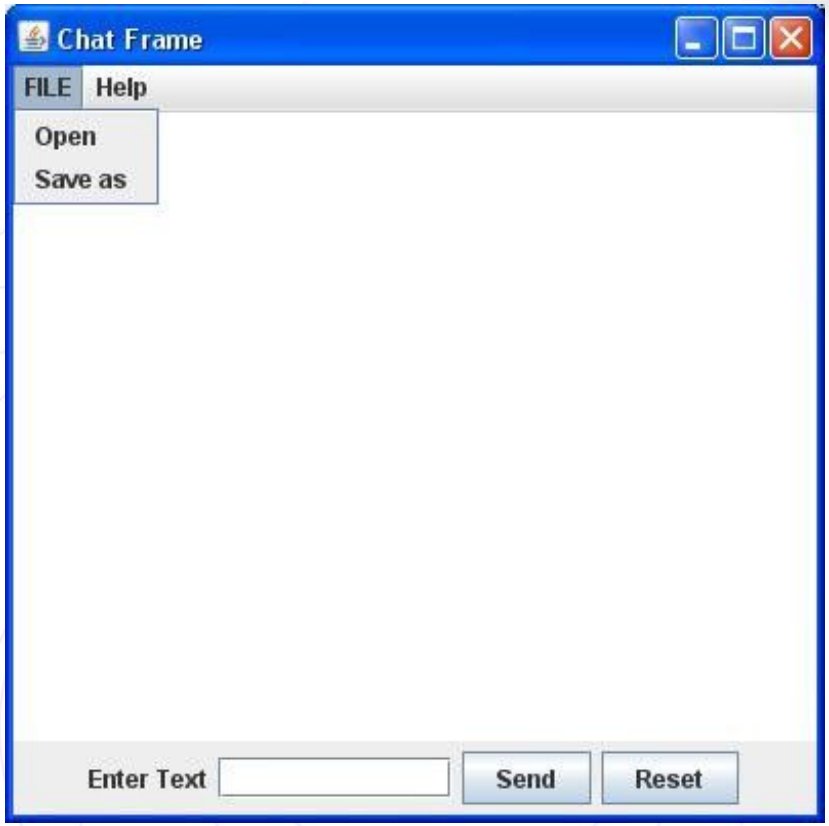
java.awt.GridBagLayout

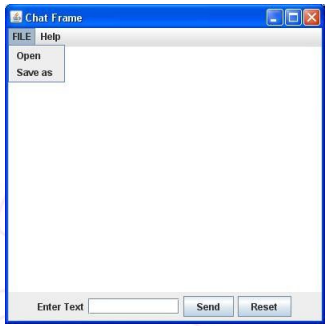
GridBagLayout es uno de los administradores de diseño más flexibles y complejos que ofrece la plataforma Java. Coloca los componentes en una cuadrícula de filas y columnas, lo que permite que los componentes especificados abarquen varias filas o columnas. No todas las filas tienen necesariamente la misma altura.

https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html

¿Cómo diseñaremos esto?



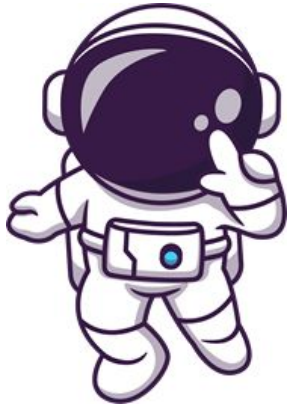





```
import javax.swing.*;
import java.awt.*;
public class SegundaGui {

  public static void main(String args[]) {
    // Creando el Marco
    JFrame frame = new JFrame("Chat Frame");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 400);

    // Creando MenuBar y agregando componentes
    JMenuBar mb = new JMenuBar();
    JMenu m1 = new JMenu("ARCHIVO");
    JMenu m2 = new JMenu("Ayuda");
    mb.add(m1);
    mb.add(m2);
    JMenuItem m11 = new JMenuItem("Abrir");
    JMenuItem m12 = new JMenuItem("Guardar como");
    m1.add(m11);
    m1.add(m12);

    ...
```

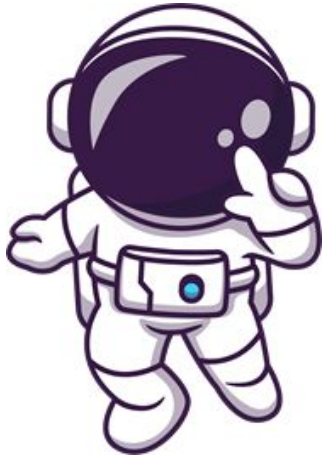
```
    ...
    // Creando el panel en la parte inferior
    JPanel panel = new JPanel();
    JLabel label = new JLabel("Introducir texto");
    JTextField tf = new JTextField(10);
    JButton send = new JButton("Enviar");
    JButton reset = new JButton("Restablecer");

    panel.add(label);
    panel.add(tf);
    panel.add(send);
    panel.add(reset);

    // Área de texto en el centro
    JTextArea ta = new JTextArea();

    // Agregar componentes al marco.
    frame.getContentPane().add(BorderLayout.SOUTH,
panel);
    frame.getContentPane().add(BorderLayout.NORTH, mb);
    frame.getContentPane().add(BorderLayout.CENTER, ta);
    frame.setVisible(true);
  }
}
```

¿y ahora esto?




CoDejaVu : JFrame Componentes Atomicos
Componentes Atomicos
JLabel : Esto es un Label o Etiqueta
JButton : Boton
JCheckBox : Check1 Check2
JRadioButton : Radio1 Radio2
JToogleButton : Activar
JComboBox : Opciones
JSeparator :
JSpinner : 0
JSlider :
JProgressBar :

¿y qué hacemos con esto?



Para la próxima sesión...

- Terminar los ejercicios que no se terminaron… (si aplica)
- Revisar la solución de los ejercicios en el repositorio https://github.com/cesardiaz-utp/MisionTIC2022-Ciclo2-Unidad5-IntroGUI
- Continuar con el ejemplo de MVC y complete las vistas para los tipos:
  - TipoVista.MULTIPLICACION_GUI
  - TipoVista.DIVISION_GUI
  - TipoVista.MODULO_GUI