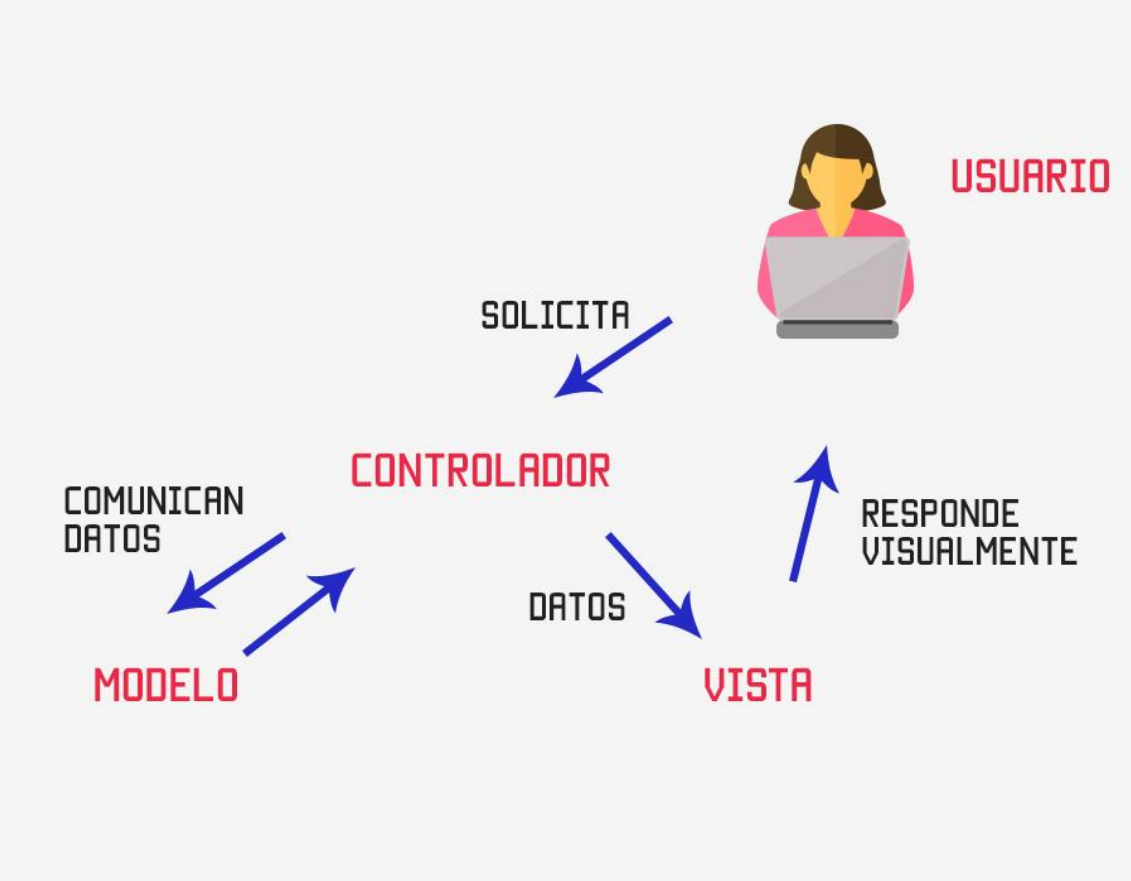
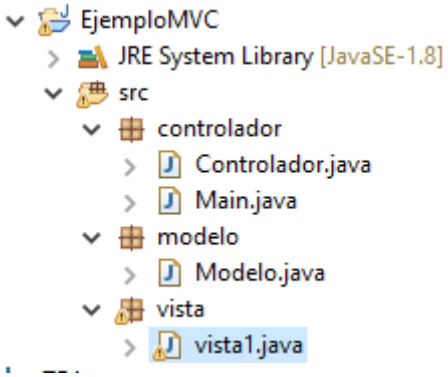
"Ejemplo MVC"

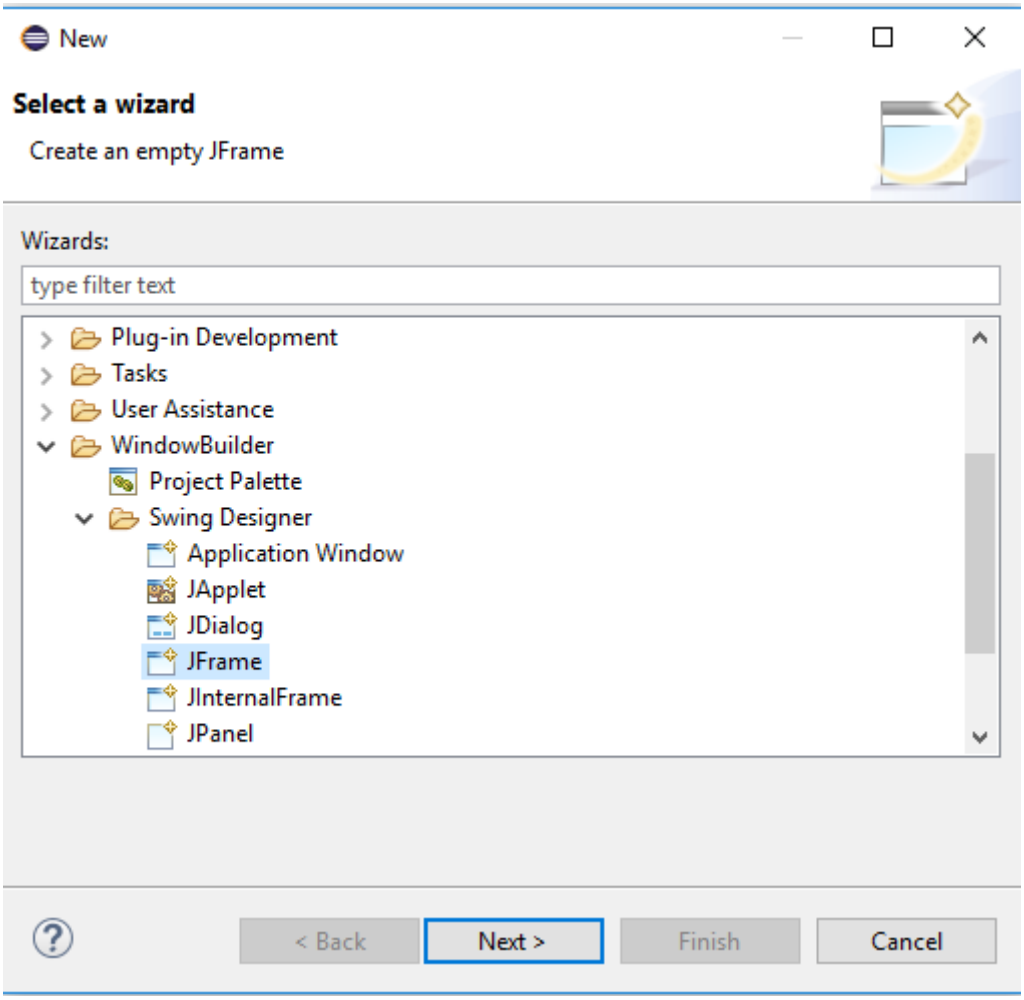


Siguiendo el ejemplo adjuntado a este archivo y tomando en cuenta que ya cuentan con el WindowsBuilder instalado en su Eclipse (ver instructivo) pasamos a la creación de la estructura de paquetes básica



Dividimos en tres paquetes las clases controlador, modelo y vista.

Creamos la vista haciendo click derecho al paquete vista y → new.. →others…
En la ventana seleccionar Jframe dentro de la carpeta WidowsBuilder



Por defecto se crea un código como el siguiente:

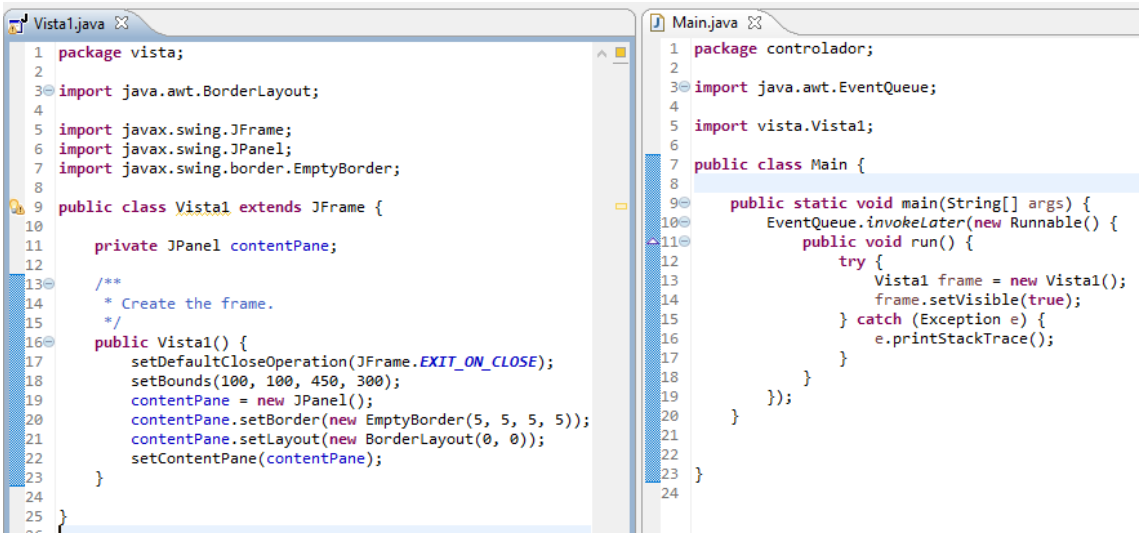
```
12  public class Vista1 extends JFrame {
13
14      private JPanel contentPane;
15
16      /**
17       * Launch the application.
18       */
19      public static void main(String[] args) {
20          EventQueue.invokeLater(new Runnable() {
21              public void run() {
22                  try {
23                      Vista1 frame = new Vista1();
24                      frame.setVisible(true);
25                  } catch (Exception e) {
26                      e.printStackTrace();
27                  }
28              }
29          });
30      }
31
32
33      /**
34       * Create the frame.
35       */
36      public Vista1() {
37          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38          setBounds(100, 100, 450, 300);
39          contentPane = new JPanel();
40          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
41          contentPane.setLayout(new BorderLayout(0, 0));
42          setContentPane(contentPane);
43      }
44
45  }
46
```

Source | Design

Vean que la clase Vista1 hereda de la clase Jframe, debajo autogenero el método main estático para que el sistema pueda ejecutarse y por último se tiene el constructor de la clase.

Nosotros no queremos que el sistema inicie desde una vista, ya que necesitamos armar la estructura que se presenta en la 1era imagen de este tutorial. Por lo cual, lo que hacemos a continuación es cortar ese método main (desde la línea de código 17 al 28) y pegarla en una nueva clase dentro del

paquete controlador que se encargara de crear el controlador de esta vista y el modelo necesario.



Vista1.java

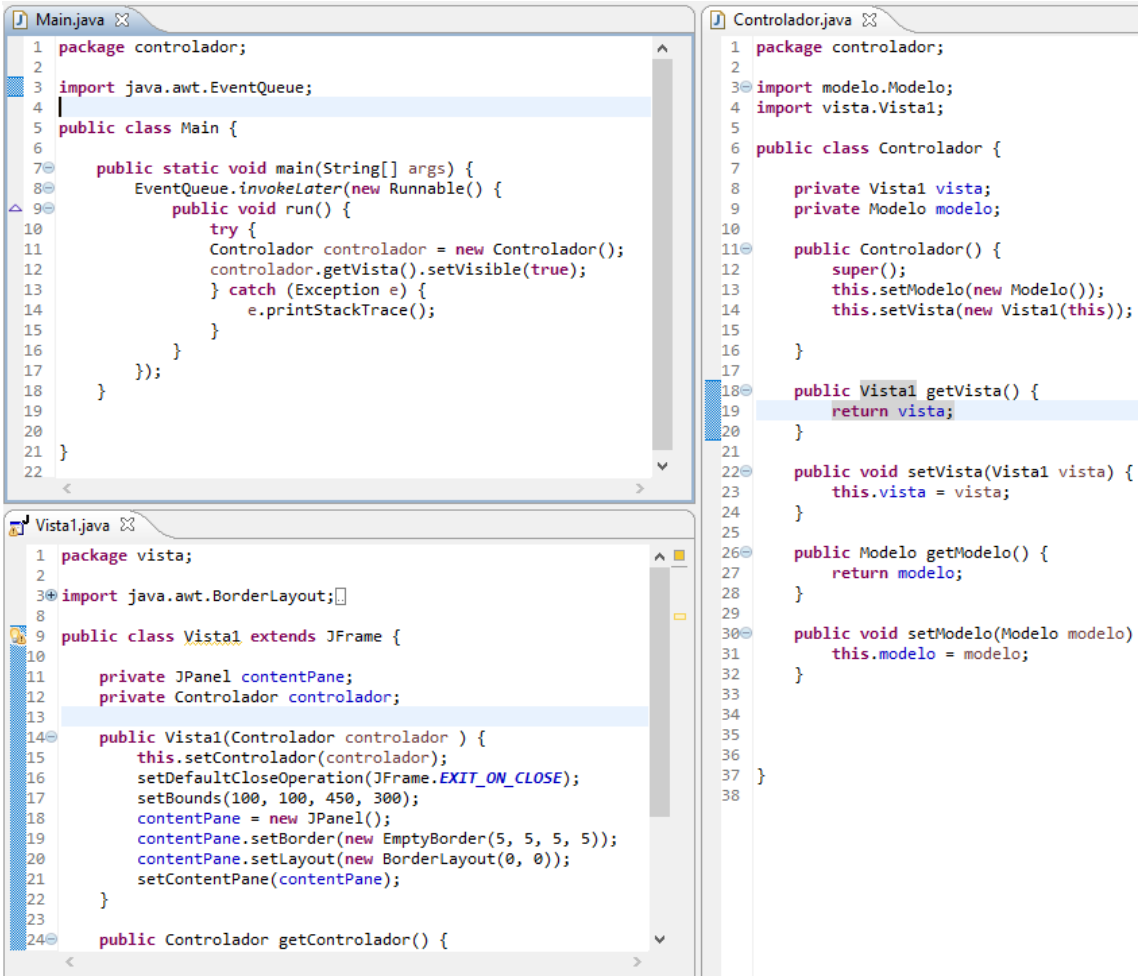
```
1  package vista;
2
3  import java.awt.BorderLayout;
4
5  import javax.swing.JFrame;
6  import javax.swing.JPanel;
7  import javax.swing.border.EmptyBorder;
8
9  public class Vista1 extends JFrame {
10
11     private JPanel contentPane;
12
13     /**
14      * Create the frame.
15      */
16     public Vista1() {
17         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18         setBounds(100, 100, 450, 300);
19         contentPane = new JPanel();
20         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
21         contentPane.setLayout(new BorderLayout(0, 0));
22         setContentPane(contentPane);
23     }
24
25 }
```

Main.java

```
1  package controlador;
2
3  import java.awt.EventQueue;
4
5  import vista.Vista1;
6
7  public class Main {
8
9      public static void main(String[] args) {
10         EventQueue.invokeLater(new Runnable() {
11             public void run() {
12                 try {
13                     Vista1 frame = new Vista1();
14                     frame.setVisible(true);
15                 } catch (Exception e) {
16                     e.printStackTrace();
17                 }
18             }
19         });
20     }
21
22
23 }
24
```

Ahora lo que tenemos que hacer es:

- Main debe de crear un nuevo objeto de Controlador
- Este controlador debe de crear la vista y luego hacerla visible una vez que todos los componentes visuales estén creados.



Main.java

```
1  package controlador;
2
3  import java.awt.EventQueue;
4
5  public class Main {
6
7      public static void main(String[] args) {
8          EventQueue.invokeLater(new Runnable() {
9              public void run() {
10                 try {
11                 Controlador controlador = new Controlador();
12                 controlador.getVista().setVisible(true);
13                 } catch (Exception e) {
14                     e.printStackTrace();
15                 }
16             }
17         });
18     }
19
20
21 }
22
```

Controlador.java

```
1  package controlador;
2
3  import modelo.Modelo;
4  import vista.Vista1;
5
6  public class Controlador {
7
8      private Vista1 vista;
9      private Modelo modelo;
10
11     public Controlador() {
12         super();
13         this.setModelo(new Modelo());
14         this.setVista(new Vista1(this));
15
16     }
17
18     public Vista1 getVista() {
19         return vista;
20     }
21
22     public void setVista(Vista1 vista) {
23         this.vista = vista;
24     }
25
26     public Modelo getModelo() {
27         return modelo;
28     }
29
30     public void setModelo(Modelo modelo)
31         this.modelo = modelo;
32     }
33
34
35
36
37 }
38
```

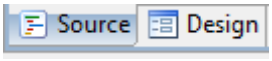
Vista1.java

```
1  package vista;
2
3  import java.awt.BorderLayout;
8
9  public class Vista1 extends JFrame {
10
11     private JPanel contentPane;
12     private Controlador controlador;
13
14     public Vista1(Controlador controlador ) {
15         this.setControlador(controlador);
16         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17         setBounds(100, 100, 450, 300);
18         contentPane = new JPanel();
19         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
20         contentPane.setLayout(new BorderLayout(0, 0));
21         setContentPane(contentPane);
22     }
23
24     public Controlador getControlador() {
```

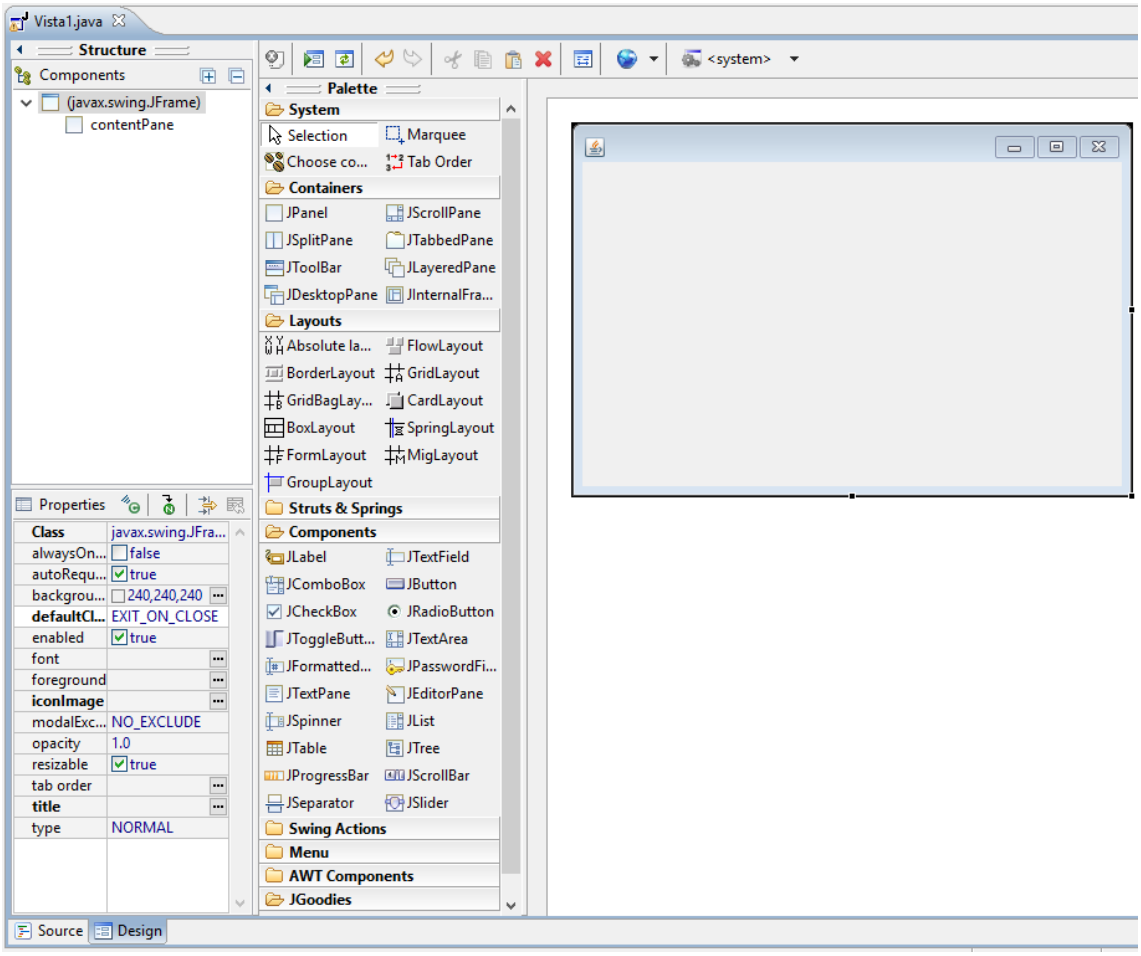
Entonces observen como las líneas 11 y 12 de la clase Main cambiaron, ahora en la línea 11 se crea el controlador y en la 12, luego de que se crean todos los componentes visuales, se hace visible la ventana.
En la clase del Controlador, en su constructor se puede ver como crea el modelo y luego la vista, pasándose a sí mismo (this) como parámetro del constructor de la vista. Luego en la clase Vista1 agregamos el atributo controlador, creamos getter y setters y por ultimo agregamos la línea 15 que setea el atributo controlador con lo que viene como parámetro.

A este punto la clase Main no es necesario modificarla más, hay que aclarar que lo que se está haciendo es encolar el código de los elementos visuales a lo último del todo, para que se puede ver reflejado toda la lógica ejecutada previamente.

Veamos en más detalle la vista, si vamos a la pestaña Design

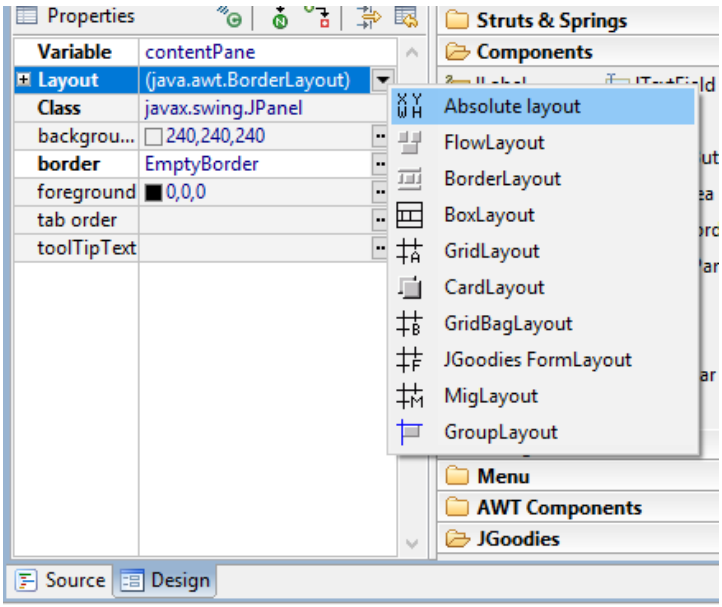


debajo del editor de la clase podemos ver la siguiente pantalla.



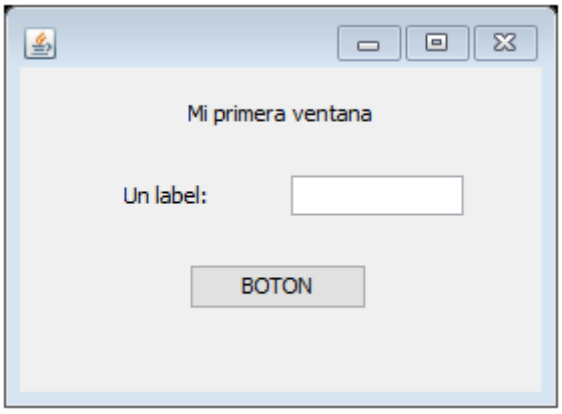
En el box de Componentes tenemos todos los elementos visuales representados en un árbol. Debajo, en Properties, si seleleccionamos un elemento desde el box de Componentes o directamente en el editor (ventana de la derecha) nos lista las propiedades del mismo.
En el medio vemos la Palette con todos los componentes visuales que se pueden arrastrar directamente al editor a su derecha.

Por defecto cada Jframe creado viene con un panel dentro, estos tienen la propiedad Layout, que son una especie de plantillas para la ubicación de los componentes visuales que están dentro, si lo deseamos cambiar se puede hacer de la siguiente forma:

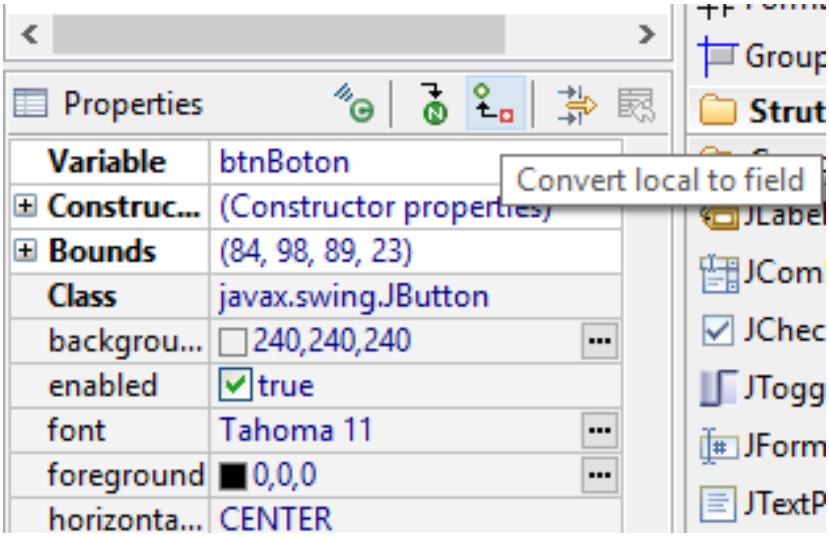


Absolute layout permite colocar elementos según un X e Y especificado por el programador.
Dentro de este panel se pueden agregar más paneles con diferentes Layout y así ir adaptando la organización de los componentes visuales dentro junto con cómo estos se comportan al agrandar o achicar la ventana.

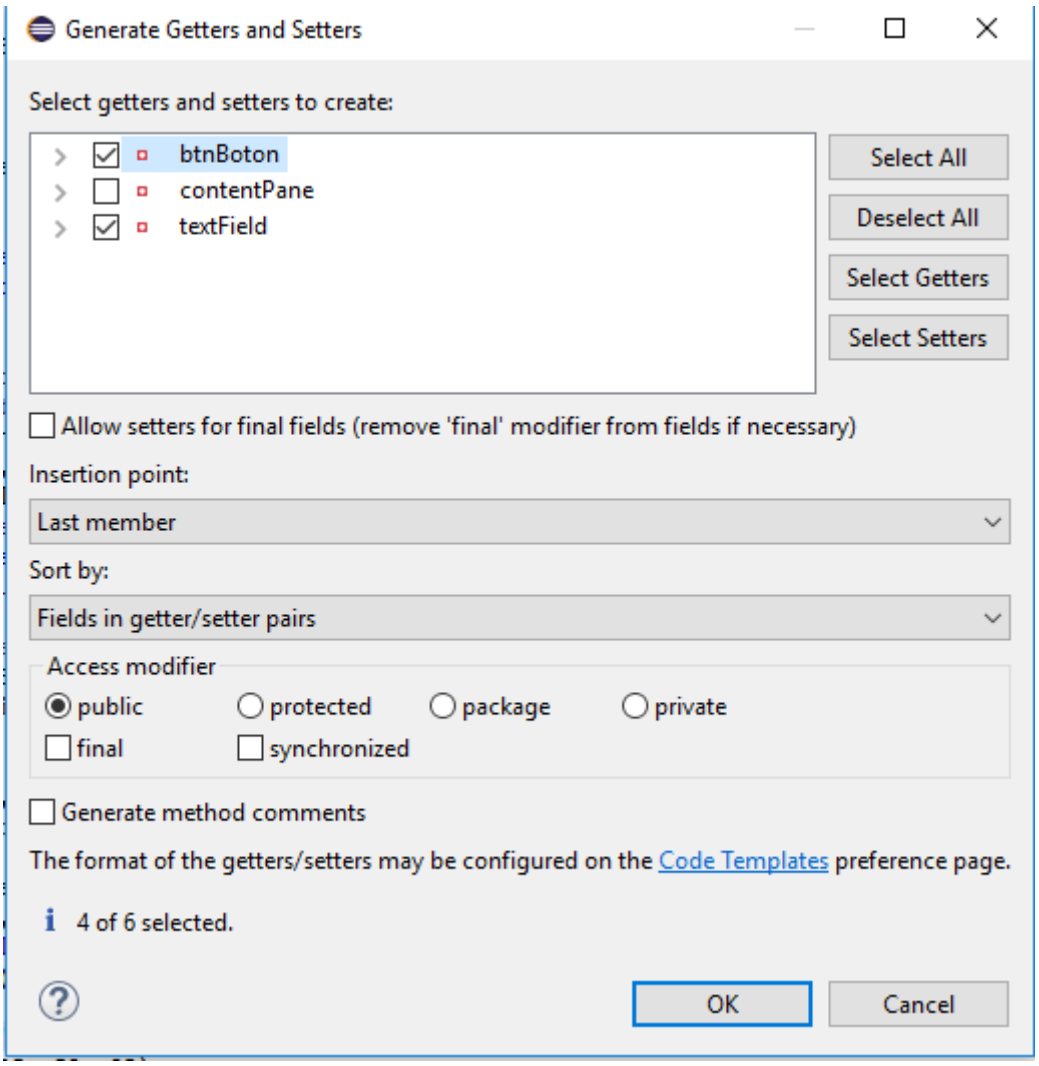
Haciendo uso de la paleta creamos la siguiente ventana:



Donde los textos son Jlabel la caja blanca donde se inserta el texto es un JtextField y el Botón es un Jbutton.
Ya teniendo todos los componentes visuales nos aseguramos de que todos los que necesitaremos luego tengan un metodo getter y setter asociados, para eso seleccionamos el elemento en el editor y hacemos clic en el siguiente boton del box de Properties, lo que va a suceder es que el objeto pasa de ser local al constructor para ser un atributo de la clase.



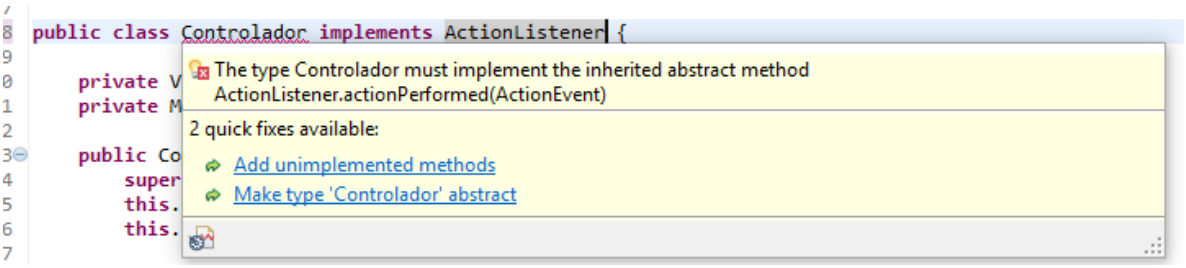
Desde la pestaña Source al autogenerar los getter y setters seleccionamos los elementos visuales botón y el text field y continuamos. Es importante **no** crear un get y set del panel ya que esto causa problemas en la visualización de la pantalla en tiempo de ejecución.



Para asociar una acción al botón hacemos doble click en el estando en el editor y esto autogenera el siguiente código.

```
43          btnBoton = new JButton("BOTON");
44          btnBoton.addActionListener(new ActionListener() {
45              public void actionPerformed(ActionEvent arg0) {
46              }
47          });
48          btnBoton.setBounds(84, 98, 89, 23);
49          contentPane.add(btnBoton);
```

Como se puede ver esto crea una nueva clase de un objeto ActionListener con un método actionPerformed dentro, esto no es lo que nosotros estamos queriendo hacer, nosotros queremos que los eventos sean capturados por el controlador, para esto lo que hacemos es que la clase Controlador implemente la interfaz ActionListener de la siguiente forma:



Noten que al implementar la interfaz ActionListener nos obliga a implementar el método actionPerformed(ActionEvent), cada elemento visual que asocie su comportamiento a este controlador pasara por este método, la implementación básica seria la siguiente:

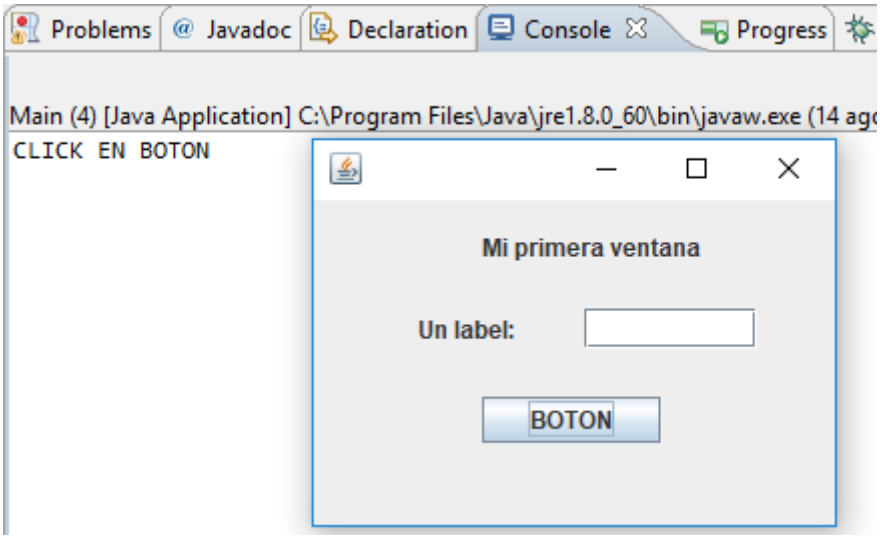
```
    @Override
    public void actionPerformed(ActionEvent arg0) {
        JButton btn = (JButton) arg0.getSource();
        if (btn.getText().equals("BOTON")) {
            System.out.println("CLICK EN BOTON");
        }

    }
```

El parámetro arg0 es necesario castearlo a un Jbutton para poder obtener el texto que tiene dentro y así identificarlo de los demás elementos visuales, así como se obtiene esa propiedad para identificar el origen del evento se puede utilizar cualquier otro atributo que se desee.
Por último, en la vista en vez de dejar el código auto generado dentro del método addActionListener(ActionPerform) del botón agregamos el controlador, quedando del a siguiente manera:

```
        btnBoton = new JButton("BOTON");
        btnBoton.addActionListener(this.getControlador());
        btnBoton.setBounds(84, 98, 89, 23);
        contentPane.add(btnBoton);
    }
```

Al ejecutar el programa y hacer click en el botón tenemos:

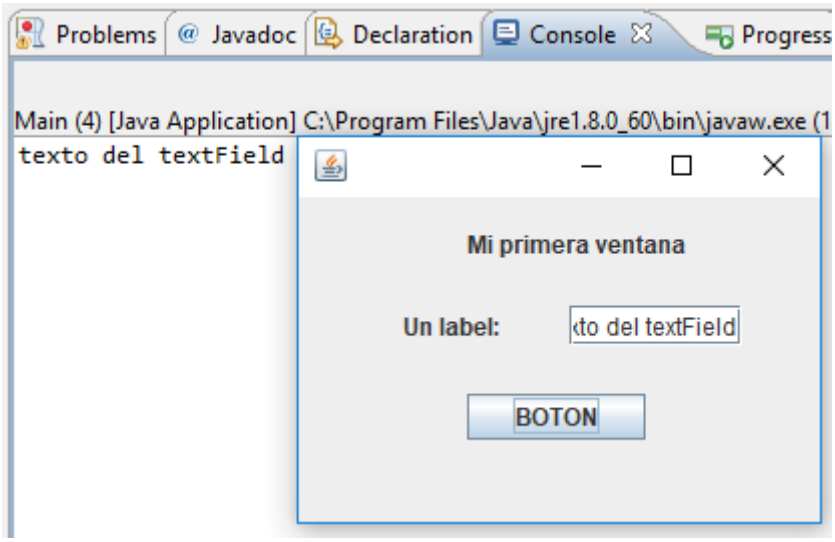


¿Cómo hacer para obtener lo que ingreso por teclado en el input?
Suponiendo que queremos obtener el texto al hacer click en el botón, nos quedaría un código así:

```
24      @Override
25      public void actionPerformed(ActionEvent arg0) {
26          JButton btn = (JButton) arg0.getSource();
27          if (btn.getText().equals("BOTON")) {
28              String texto = this.getVista().getTextField().getText();
29              System.out.println(texto);
30          }
31
32      }
```

El controlador conoce a la vista y lo obtiene a partir de un método geter (getVista()) previamente dentro de la vista creamos el getter del textField y al componente visual le pedimos el texto que contiene dentro.

Resultado



¿Cómo hacer para abrir una segunda ventana?

La acción del botón tiene que tener crear una nueva instancia de un controlador que maneje la nueva ventana, en este ejemplo Controlador2.

```
23      @Override
24      public void actionPerformed(ActionEvent arg0) {
25          JButton btn = (JButton) arg0.getSource();
26          if (btn.getText().equals("BOTON")) {
27              String texto = this.getVista().getTextField().getText();
28              System.out.println(texto);
29
30              new Controlador2();
31          }
32
33      }
```

Y el nuevo controlador tiene la siguiente forma:

```java
public class Controlador2 implements ActionListener {

    private Vista2 vista;

    public Controlador2() {
        super();
        this.setVista(new Vista2(this));
        this.getVista().setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent arg0) {
    }

    public Vista2 getVista() {
        return vista;
    }

    public void setVista(Vista2 vista2) {
        this.vista = vista2;
    }

}
```

Donde al pasar por el constructor crea la vista y la hace visible, recordar que de ser necesario se crearan nuevos objetos del modelo o bien serán enviados por parámetro por el otro controlador.

Por último, la Vista2 tiene la siguiente forma similar a la Vista1.

```
10  public class Vista2 extends JFrame {
11
12      private JPanel contentPane;
13      private Controlador2 controlador;
14
15
16      public Vista2(Controlador2 controlador) {
17          this.setControlador(controlador);
18          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19          setBounds(100, 100, 275, 169);
20          contentPane = new JPanel();
21          contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
22          setContentPane(contentPane);
23          contentPane.setLayout(null);
24
25          JLabel lblVista = new JLabel("VISTA 2");
26          lblVista.setBounds(98, 46, 121, 14);
27          contentPane.add(lblVista);
28      }
29
30
31      public Controlador2 getControlador() {
32          return controlador;
33      }
34
35
36      public void setControlador(Controlador2 controlador) {
37          this.controlador = controlador;
38      }
39
40  }
```

Donde se cuenta con el Controlador2 como atributo y se setea cuando viene la instancia por parámetro.

Resultado: