PROGRAMACIÓN EN JAVA

TEMA 2: Mi primera aplicación

Bibliografía

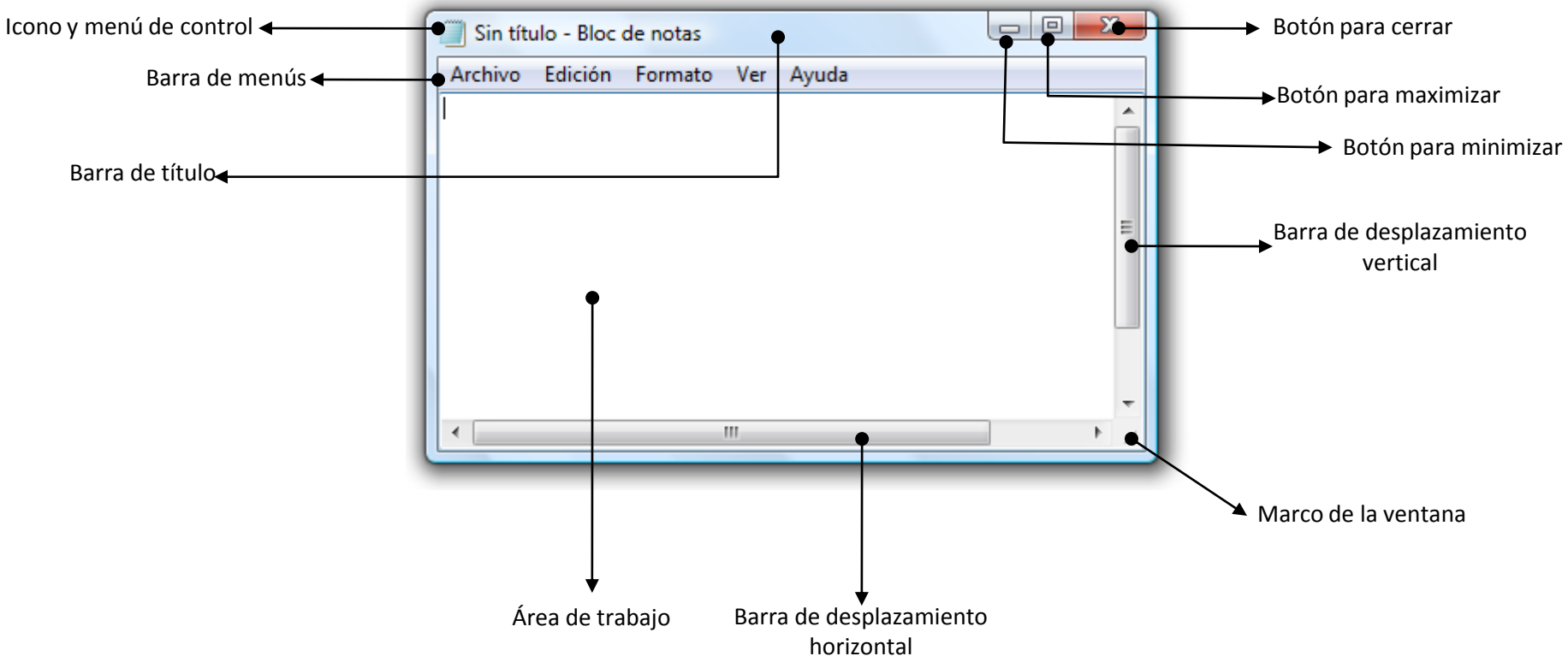
- “JAVA 2 Interfaces gráficas y aplicaciones para Internet – 2ª Edición”. Fco. Javier Ceballos. Ed. Ra-Ma, 2006. Capítulo: 1

Contenido

- Mi primera aplicación
- Formularios
- Biblioteca JFC (JAVA Foundation Classes)
- ¿Swing o AWT?
- Estructura de una aplicación
- Diseño de la interfaz gráfica
- Crear un componente Swing
- Componentes Swing más comunes
- Contenedores
- Administradores de diseño
- Añadir los componentes al contenedor
- Manejo de eventos
- Asignar manejadores de eventos a un objeto
- Responder a los eventos

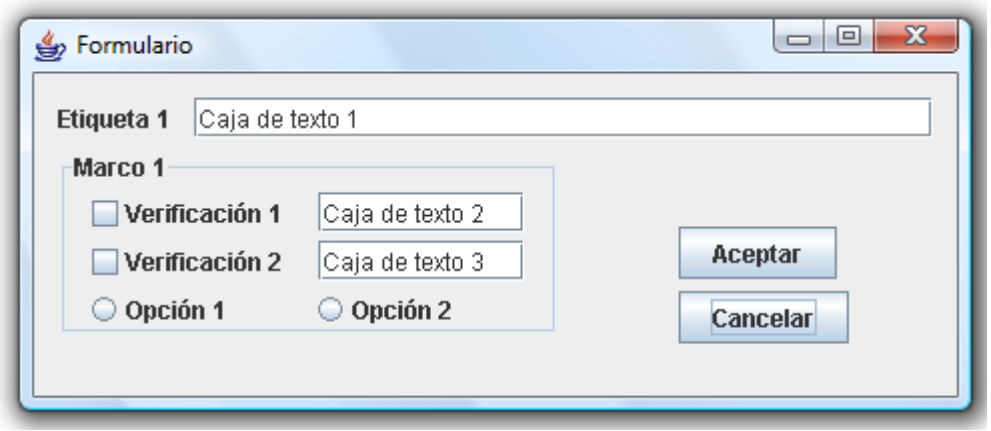
Mi primera aplicación

- La ventaja de trabajar con ventanas es que todas se comportan de la misma forma independientemente del S.O.
- Además, utilizan los mismos componentes básicos para introducir órdenes (menús, botones, etc.)
- Por ejemplo, una ventana típica de Windows tiene las siguientes partes:



Formularios

- Es la unidad fundamental de una aplicación que visualice una interfaz gráfica
- Es realmente una ventana sobre la que se dibujan otros objetos llamados componentes o controles (etiquetas, cajas de texto, casillas de verificación, botones de opción, botones de pulsación, etc.), con el fin de aceptar, procesar o visualizar datos

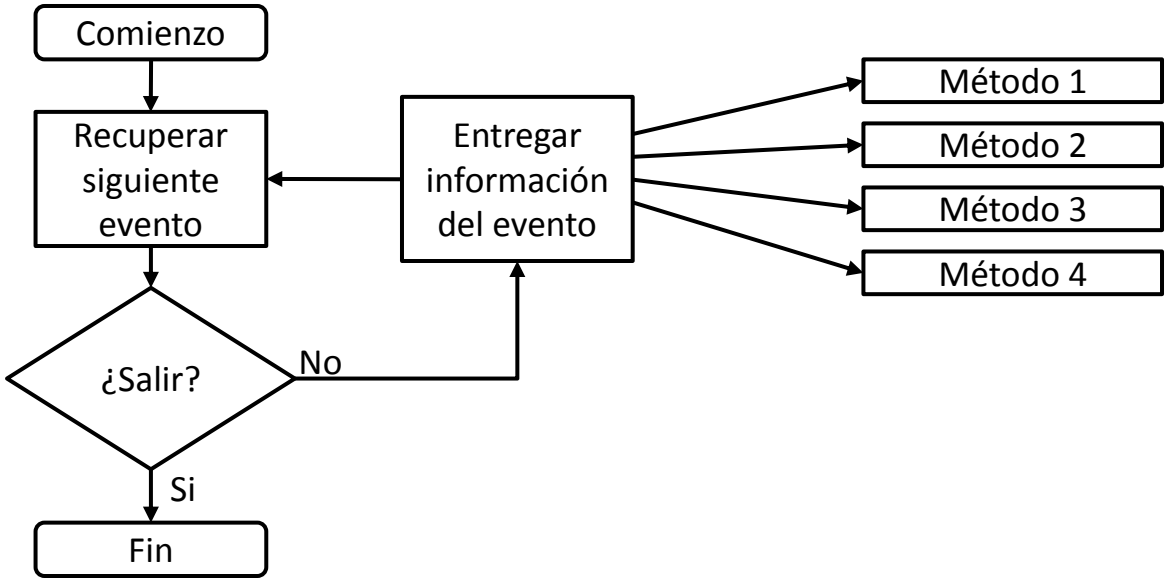


Formularios (cont.)

- Una vez finalizado el diseño de la interfaz gráfica, se debe escribir el código fuente relacionado con cada objeto.
- Cada objeto está ligado a un código que permanece inactivo hasta que se produzca el evento que lo activa.
- Es por esto que una aplicación con interfaz gráfica trabaja estableciendo entre los objetos que la componen, una comunicación mediante mensajes, los cuales son producidos por eventos. Cuando un mensaje llega a un objeto, éste tiene que procesarlo.

Formularios (cont.)

- Evento: mecanismo mediante el cual el objeto puede notificar de la ocurrencia de un suceso.
- Puede ser producido por:
  1. El usuario
  2. El sistema
  3. Indirectamente por el código



- A esta forma de programar se le denomina programación conducida por eventos y orientada a objetos.

Biblioteca JFC (JAVA Foundation Classes)

- Es una biblioteca de clases utilizada para diseñar aplicaciones que utilicen interfaces gráficas.
- Aquí se agrupan las siguientes APIs:

  1. **Swing:** Conjuntos de componentes escritos en JAVA que se ejecutan uniformemente en cualquier plataforma nativa que soporta la VM de JAVA.

  2. **AWT:** Grupo de componentes común a todas las plataformas, pero escritos específicamente para cada plataforma en código nativo (no escritos en JAVA).

  3. **Accesibilidad:** Soporte para usuarios con limitaciones.

  4. **JAVA 2D:** Incorporación de gráficos 2D de alta calidad, texto e imágenes.

  5. **Soporte para arrastrar y colocar (drag and drop):** Transferencia de datos entre aplicaciones mediante la simple operación de arrastrarlos hasta el lugar de destino.

¿Swing o AWT?

- Los componentes *Swing* están implementados absolutamente con código no nativo.
- Los componentes *Swing* son independientes de la plataforma. Razón que justifica su utilización.
- Los componentes *Swing* proporcionan más capacidades que los *AWT.*
- Los nombres de los componentes *Swing* empiezan por J

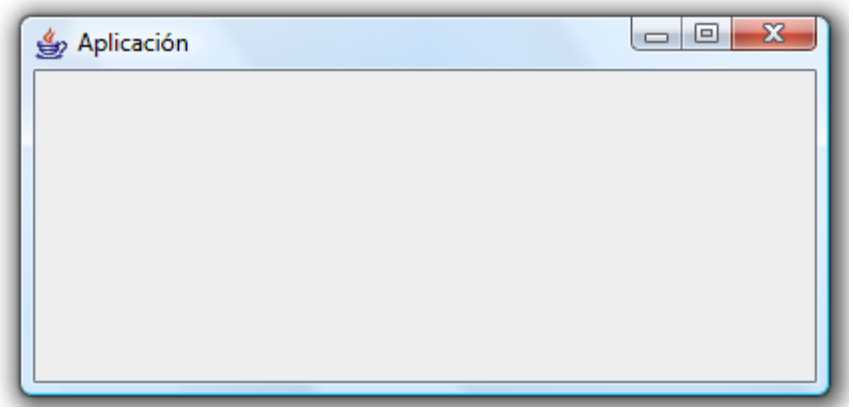
  Ejemplo: AWT Button Ξ Swing JButton
- Los componentes *AWT* se localizan en el paquete *java.awt*
- Los componentes *Swing* se localizan en el paquete *javax.swing*
- Todos los componentes *Swing* son sub-clases de la clase *Jcomponent.*

Estructura de una aplicación

- ¿Cómo construir una aplicación mínima que presente una interfaz gráfica?

  R= A través del estudio, desde un punto de vista práctico, sobre cuáles son y cómo interaccionan entre sí los elementos que configuran una aplicación.

- Para ello utilizaremos la siguiente aplicación, la cual denominaremos *Caplicacion.java*

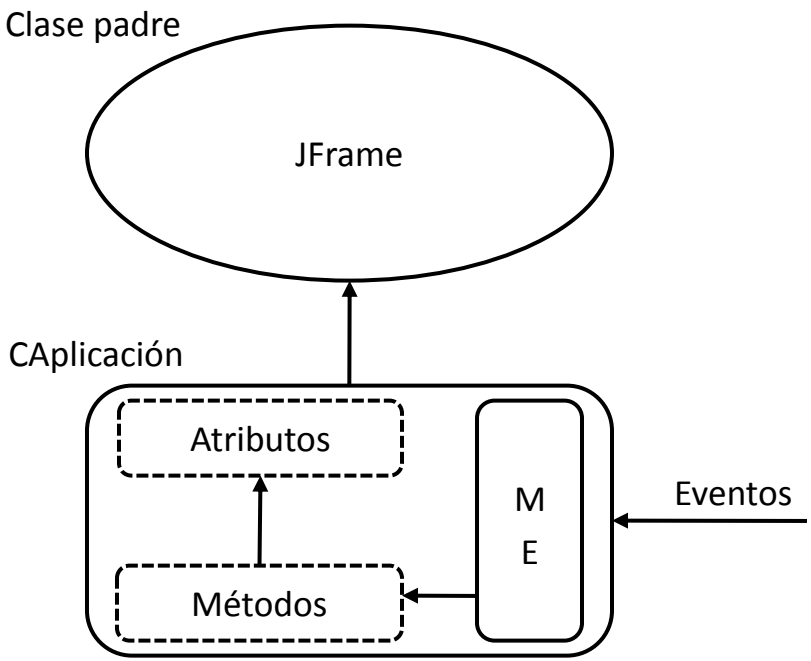


- Una aplicación que muestra una interfaz gráfica cuando se ejecuta, no es más que un objeto de una clase derivada de *JFrame* (perteneciente al paquete *javax.swing*)

Estructura de una aplicación (cont.)

```
public class CAplicacion extends javax.swing.Jframe {
   /*Crear un nuevo formulario de la clase CAplicacion*/
   public CAplicacion() //Constructor
   {
      setSize(300,200);  //Tamaño del formulario
      setTitle("Aplicación");  //Título del formulario
      initComponents();  //Iniciar los controles o componentes
   }

   /*Este método es llamado desde el constructor CAplicacion*/
   private void initComponents()
   {
      addWindowListener(new java.awt.event.WindowAdapter()
      {
         public void windowClosing(java.awt.event.WindowEvent evt)
         {
            exitForm(evt);
         }
      }
      );
   }

   /*Salir de la aplicación*/
   private void exitForm(java.awt.event.WindowEvent evt) {
      System.exit(0);  //Salir de la aplicación
   }

   public static void main(String[] args) {
      new CAplicacion().setVisible(true);
   }

   //Declaración de variables

}
```

Estructura de una aplicación (cont.)



- La clase *CAplicacion* hereda todos los atributos y métodos de su clase padre *JFrame*. Además, esta clase encapsula tres métodos más: el *constructor*, *initComponents* y *exitForm*
- El método *initComponents* añade al objeto *CAplicacion* un objeto de la clase *WindowAdapter* (el manejador de eventos de ventana - *ME*)

Diseño de la interfaz gráfica

- Añadir a la ventana de la aplicación los componentes *Swing* necesarios.
- Este proceso requiere:
  1. Crear los componentes
  2. Establecer las propiedades de c/u de los componentes
  3. Añadir los componentes al contenedor aportado por el propio formulario (supeditada a un administrador de diseño)

Crear un componente Swing

- No difiere en nada de cómo lo hacemos con un objeto de cualquier otras clase.
- Se crea el componente (invocando al constructor) y se inician las propiedades del mismo (invocando a los métodos correspondientes)

Ejemplo:

```
//Crear un botón
JButton jBtSaludo = new JButton("Haga clic aquí");
//Establecer como tecla de acceso la c
jBtSaludo.setMnemonic('c');
//Asignar una descripción abreviada
jBtSaludo.setToolTipText("Botón de pulsación");
```

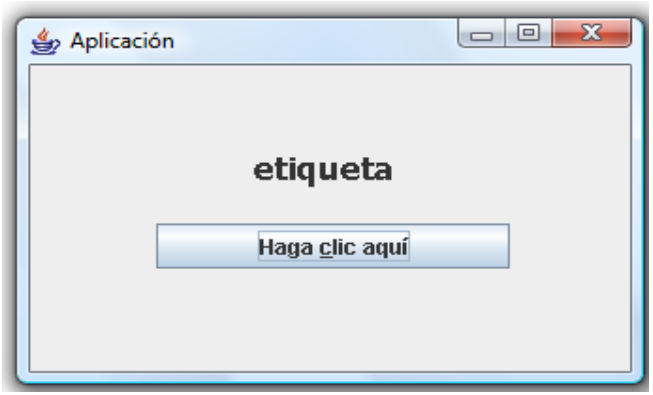
Componentes Swing más comunes

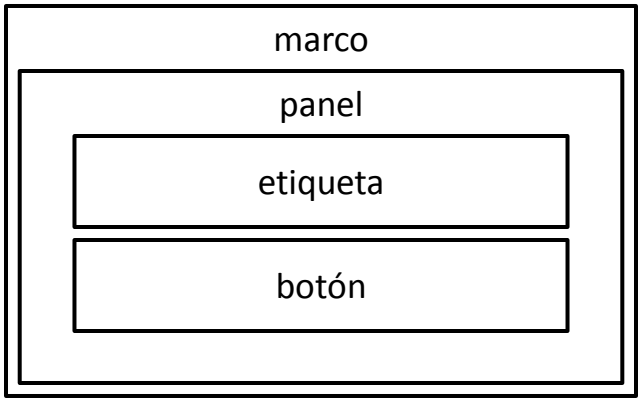
- **Etiquetas:** Se implementan a partir de la clase *JLabel*
- **Botones:** Se implementan a partir de la clase *JButton*
- **Cajas de texto:** Se implementan a partir de la clase *JTextField* las de una sola línea, y a partir de *JTextArea* las de varia líneas
- **Casillas de verificación:** Se implementan a partir de la clase *JCheckBox*
- **Botones de opción:** Se implementan a partir de la clase *JRadioButton*
- **Listas:** Se implementan a partir de la clase *JList*
- **Barras de desplazamiento:** Se implementan a partir de la clase *JScrollBar*
- **Cuadros de diálogo estándar:** Se implementan a partir de la clase *JOptionPane*

NOTA: Los componentes *Swing* se localizan en el paquete *javax.swing* y todos los componentes son subclases de la clase *JComponent*

Contenedores

- Son componentes *Swing* utilizados para ubicar otros componentes





- Para ayudarnos a ubicar adecuadamente los componentes, es posible utilizar una jerarquía de contenedores
- La raíz de la jerarquía siempre será el marco de la ventana (*JFrame*, *JDialog* y *JApplet*)
- Cada contenedor define un componente denominado panel (*JPanel*)

Ejemplo:

```
JPanel jPnPanel = new JPanel();  //Creamos un objeto JPanel
jPnPanel.add(etiqueta);  //Añadimos una etiqueta al panel
jPnPanel.add(boton);  //Añadimos un botón al panel
```

Administradores de diseño

- Está pensado para mostrar varios componentes a la vez en un orden preestablecido
- De forma predeterminada cada contenedor tiene asignado un administrador de diseño (Ejemplo: los contenedores de nivel superior tienen asignado *BorderLayout*)
- *Swing* proporciona 6 administradores de diseño
  1. **BorderLayout:** Divide un contenedor en 5 secciones denominadas: *norte*, *sur*, *este*, *oeste* y *centro*
  2. **GridLayout:** Coloca los componentes en el contenedor en filas y columnas
  3. **GridBagLayout:** Coloca los componentes en el contenedor en filas y columnas (permite a un componente ocupar más de una fila o columna)
  4. **CardLayout:** Utiliza una filosofía de trabajo con pestañas
  5. **BoxLayout:** Coloca los componentes en el contenedor en una única fila o columna, ajustándose al espacio que haya
  6. **FlowLayout:** Coloca los componentes en el contenedor de izq. a der.
- También podemos utilizar el administrador de diseño absoluto o no utilizar ninguno (administrador de diseño nulo: *null*), estos permiten colocar los componentes donde se quiera

Ejemplos de asignaciones

```
jPnPanel.setLayout(new GridLayout(0,1));        getContentPane().setLayout(null);
```

Añadir los componentes al contenedor

- Los pasos a seguir son:
  1. Asignar un administrador de diseño

     Ejemplo:

     /*Asignamos al contenedor *JFrame*, un administrador de diseño *null**/
     getContentPane().setLayout(null);

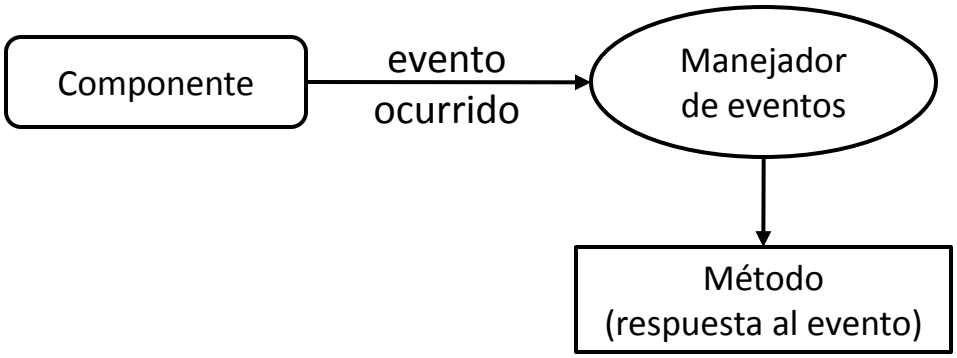
  2. Añadir el componente y editar sus propiedades

     Ejemplo:

     /*Añadimos a la clase una variable de tipo *JLabel**/
     private javax.swing.JLabel jEtSaludo = new javax.swing.JLabel();
     /*Modificamos algunas propiedades de la etiqueta*/
     jEtSaludo.setText("Etiqueta");
     jEtSaludo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
     jEtSaludo.setFont(new java.awt.Font("Dialog",1,18));
     /*Añadimos la etiqueta al panel raíz del objeto*/
     getContentPane().add(jEtSaludo);
     jEtSaludo.setBounds(42,36,204,30); //Posición y tamaño dentro del contenedor

Manejo de eventos

- Cuando sobre un componente ocurra algún evento se espera que suceda algo (ese algo hay que programarlo)
- Pero, ¿Cómo manejamos ese evento?
- A través de objetos denominados manejadores o escuchadores de eventos
- Es por esto, que un manejador de eventos es un objeto en el que un componente delega la tarea de manipular un tipo particular de eventos



- Un componente tendrá asociados tantos manejadores de eventos como tipos de eventos tenga que manejar

Asignar manejadores de eventos a un objeto

- Pasos para vincular un manejador de eventos con un componente:

  1. Creamos una clase que implemente la interfaz que aporta los métodos que permiten responder al tipo de eventos que tratamos de manejar

     Ejemplo:

```
class ManejadorEvtjBt implements java.awt.event.ActionListener
{
   public void actionPerformed(java.awt.event.ActionEvent evt)
   {
      //Código que responde al evento evt
   }
}
```

  2. Construir un objeto de esta clase (corresponderá con el manejador de eventos)

     Ejemplo:

```
java.awt.event.ActionListener al = new ManejadorEvtjBt();
```

  3. Vinculamos el manejador de eventos con el componente

     Ejemplo:

```
jBtSaludo.addActionListener(al);
```

Responder a los eventos

- Asignar a cada uno de los componentes la tarea que debe desempeñar
- Pasos a seguir:

1. Añadir un manejador que manipule el evento

Ejemplo:

```
private void initComponents()
{
   //…
   jBtSaludo.setToolTipText(“Botón de pulsación”);
   jBtSaludo.setText(“Haga clic aquí”);
   jBtSaludo.addActionListener(new java.awt.event.ActionListener()
   {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
         jBtSaludoActionPerformed(evt);
      }
   } );
   //…
}
```

2. Escribir el método (código a ejecutarse para responder al evento)

Ejemplo:

```
private void jBtSaludoActionPerformed(java.awt.event.ActionEvent evt)
{
   //Código a ejecutar
}
```