# Construcción de interfaces gráficos de usuario con Netbeans

9 de enero de 2004

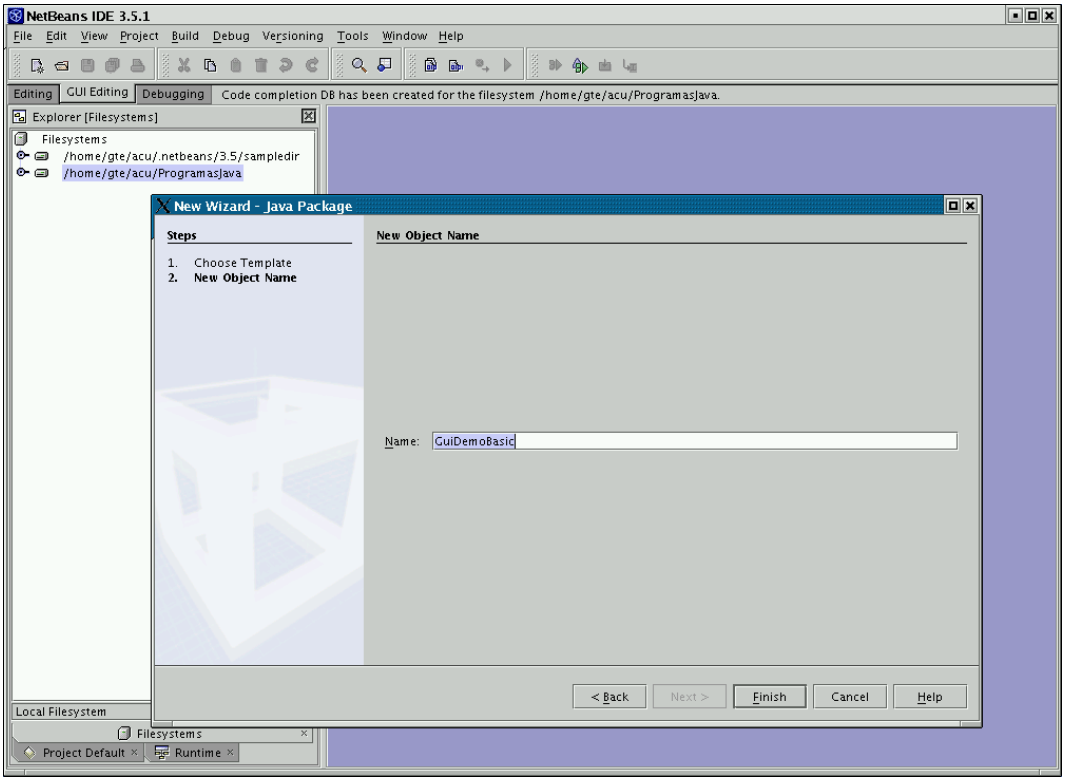
## Índice



## 1. Creación del contenedor de objetos

La creación del interfaz gráfico de usuario (GUI) con Netbeans es muy sencilla. Se comienza creando un *contenedor* mediante una determinada plantilla, y luego se arrastran a él, los componentes visuales, ajustándoles las propiedades que se necesiten.
Vamos a crear un GUI simple. El GUI concatenará los strings de dos campos de texto y luego mostrará el resultado en un tercer campo de texto.
Ejecuta el entorno Netbeans, abre un directorio de trabajo, y crea un paquete para el programa que vamos a crear:

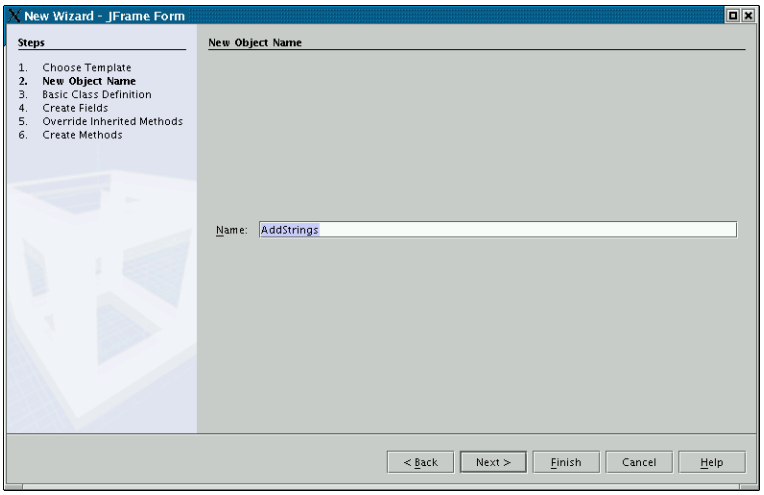
1. Ejecuta el IDE.
2. Pincha en la solapa **GUI Editing** de la ventana principal para cambiar al modo GUI.
3. Decide qué directorio quieres usar para colocar el código fuente. Si el directorio ya está montado, abre el directorio en la ventana Explorer. Si no está montado, pulsa con el botón derecho sobre el nodo de más alto nivel de *Filesystems*, selecciona **Mount Directory** del menú contextual, y monta el directorio.
4. Pincha con el botón derecho el directorio montado, selecciona **New** → **Java Package**. Llama *GuiDemoBasic* al nuevo paquete. Esta operación se muestra en la siguiente figura.



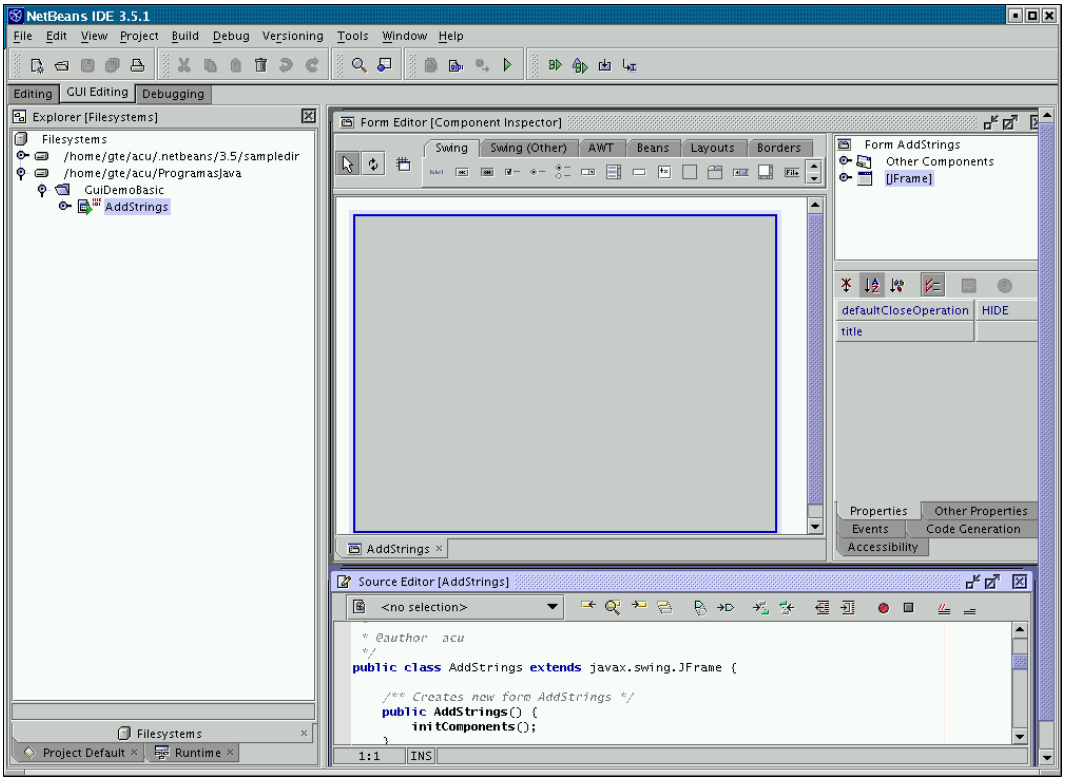

A continuación, crea el contenedor a partir de una plantilla. Nosotros elegiremos `JFrame` para este programa. Otros posibles contenedores son `Applet`, `Dialog`, `Frame` o `Panel` de AWT, y `JApplet`, `JDialog`, `JInternalFrame`, `JFrame` o `JPanel` de Swing.

1. Pincha con el botón derecho en el nuevo paquete **GuioDemoBasic**, y selecciona **New** → **JFrame Form** del menú contextual.

2. Nombra `AddStrings` al nuevo `JFrame`, y pincha en **Finish**:



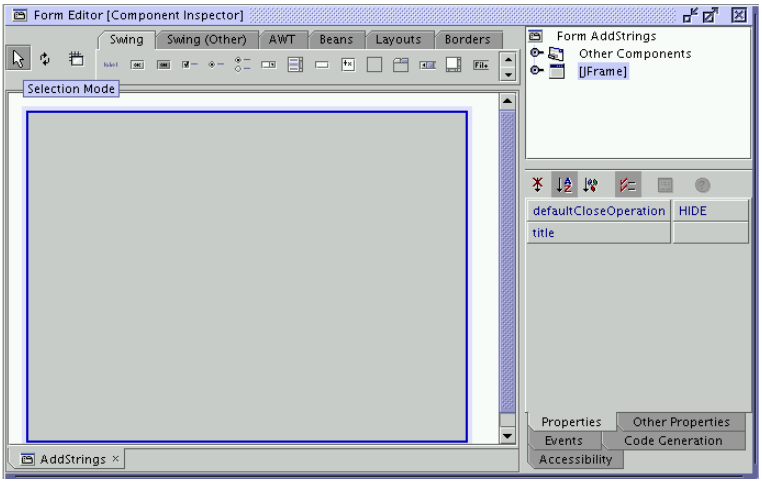

3. El IDE abre una ventana *Source Editor* con el código Java para la nueva clase del `JFrame`. Además abre una ventana *Form Editor*, que incluye la *paleta de componentes* en la parte superior, el *inspector de componentes* en la parte derecha y el panel *editor del form* bajo la paleta de componentes.




Si observamos con algún navegador de directorios lo que ha ocurrido tras los pasos anteriores, podemos comprobar que se ha creado un directorio llamado `GuiDemoBasic` que contiene el fichero `AddStrings.java` que contiene el código Java que aparece en el Source Editor. Además aparece el fichero `AddStrings.form` que es usado por el Form Editor para leer y guardar el form que estamos creando. Este último fichero no es necesario para compilar ni ejecutar el programa.
El código Java creado con Netbeans puede ser exportado, compilado y ejecutado fuera del IDE en cualquier entorno Java. También el código Java creado en otros entornos puede ser importado, modificado, compilado y ejecutado en Netbeans. Pero,actualmente no hay ninguna forma de generar un fichero `.form` a partir del código importado. Eso significa que el código importado debe modificarse a mano, en lugar de con el Form Editor.

## 2. Adición de componentes al contenedor

La paleta de componentes nos permite seleccionar qué componente queremos añadir al form. En la parte superior derecha de la paleta aparecen varias solapas, con las distintas categorías de componentes que podemos usar. Los tres botones de la parte izquierda de la paleta se usan para definir el modo del Form Editor. Por ahora, aseguraos que el botón *Selection Mode* está pulsado.

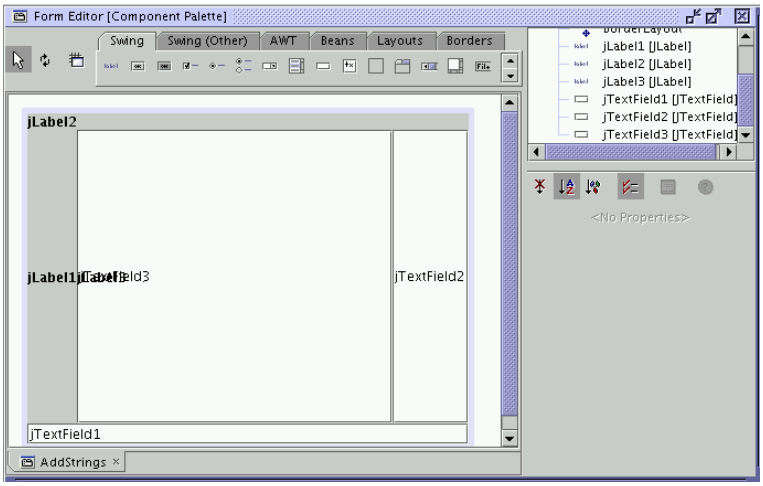


Vamos a añadir tres componentes `JLabel`. Para ello:

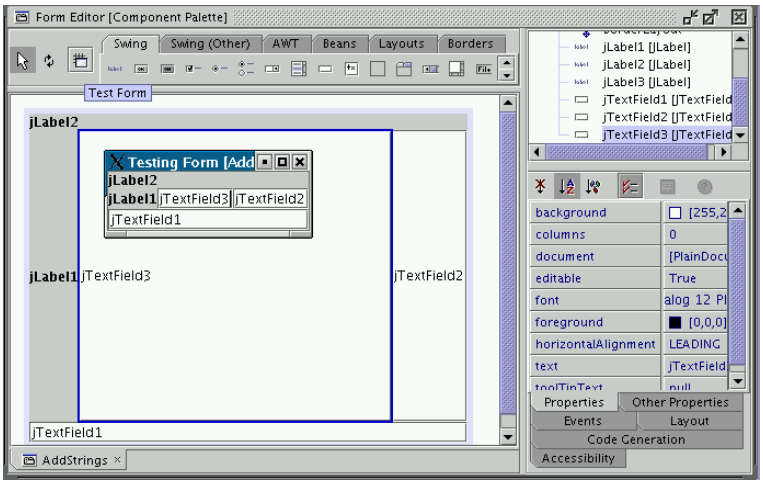
1. Pinchar en la solapa **Swing** de la paleta de componentes.
2. Pinchar en el botón `JLabel`.
3. Mantener pulsada la tecla **shift** y pinchar tres veces en el panel del editor de forms. Esto hace que aparezcan tres `JLabel`. Si no hubiesemos mantenido pulsada la tecla **shift**, sólo se hubiese añadido un `JLabel`.

En Java, la colocación y tamaño de los componentes dentro del contenedor está determinada por el gestor de posicionamiento (Layout manager) que tenga definido el contenedor. En este caso el gestor de posicionamiento que tiene definido el `JFrame` es por ahora `BorderLayout`.
Para añadir los componentes `JTextField` usaremos un procedimiento algo distinto.

1. Pincha con el botón derecho en el nodo **JFrame** del inspector de componentes.
2. Selecciona **Add From Palette** en el menú contextual y elige **Swing** → **JTextField**.
3. Repite el proceso tres veces.
4. Notar que los componentes aparecen tanto en el editor de forms como en el inspector de componentes.




Para ver la apariencia del GUI que hemos construidos podemos pulsar el botón **Test Form** (colocado en la parte superior izquierda del Form Editor). En modo Test el gestor de posicionamiento funciona del mismo modo que en la aplicación compilada, y también los componentes del GUI responden a eventos de ratón y teclado, aunque los componentes no están conectados a manejadores de eventos.




Modifiquemos el gestor de posicionamiento del `JFrame` para conseguir un aspecto mejor. Intentemos el gestor `FlowLayout`.

1. Pincha la solapa Layouts en la paleta de componentes.
2. Selecciona el botón **FlowLayout**.
3. Pincha en cualquier lugar dentro del `JFrame` en el editor del form.

Si testeamos ahora el form, obtendremos lo siguiente:


## 3. Configuración de componentes

Si miramos en el Source Editor el método `InitComponents()`, podemos ver que aparece sombreado en color azul claro. Esto nos indica que es código *reservado*, o sea que se genera automáticamente y no debe modificarse a mano.

1. Modificación del texto

   *a*) Seleccionar `jLabel1` en el inspector de componentes.
   *b*) Pulsar la solapa **Properties** y localizar la propiedad **text**.
   *c*) Cambiar su valor de `jLabel1` a `String A`.
   *d*) Notar que el valor también cambia en el editor de forms y en el código reservado del Source Editor.
   *e*) Modifica ahora el valor de la propiedad texto de los otros componentes como muestra la siguiente tabla. Para el componente `jTextField3` desactivaremos la propiedad **Editable**.

| **Componente** | **Text** | **Editable** |
|---|---|---|
| jLabel1 | String A | N/A |
| jLabel2 | String B | N/A |
| jLabel3 | String A+B | N/A |
| jTextField1 | A default | True |
| jTextField2 | B default | True |
| jTextField3 | result | False |

2. Cambiar el tamaño de componentes
   Los componentes tienen ahora mismo el tamaño que se adapta al texto que contienen.

   *a*) Pincha la solapa **Other Properties** y localiza la propiedad **preferredSize**.
   *b*) Cambiar el valor de todos los componentes a $[80, 30]$.
   *c*) Testea de nuevo el form.

3. Renombrar componentes.
   Demos nombres más significativos a los componentes.

   *a*) Pinchar con el botón derecho cada uno de los componentes en el inspector de componentes, seleccionando **Rename** del menú contextual.
   *b*) Renombra los componentes según muestra la siguiente tabla.

| **Nombre original** | **Nuevo nombre** |
|---|---|
| jLabel1 | lblA |
| jLabel2 | lblB |
| jLabel3 | lblSum |
| jTextField1 | tfA |
| jTextField2 | tfB |
| jTextField3 | tfSum |

4. Modificación del título del Frame

   *a*) Selecciona el node `JFrame` en el inspector de componentes.
   *b*) Selecciona la solapa **Properties**.
   *c*) Pon en el **title** el valor `AddString 1.0`.

## 4. Construcción de menús

Ahora añadiremos una barra de menús a nuestro form.

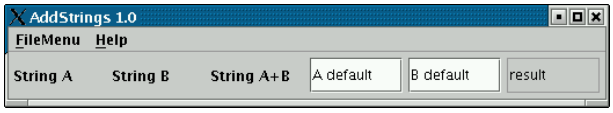
1. Seleccionar **JMenuBar** en la solapa Swing de la paleta de componentes.
2. Pinchar con el ratón en cualquier parte del panel del editor del form.

Inicialmente la barra de menús tendrá sólo un menú, sin ningún item. Vamos a añadir otro menú, y le pondremos a cada uno sus opciones (items).

1. Pinchar con el botón derecho del ratón en el nuevo **JMenuBar** en el inspector de componentes.
2. Seleccionar **Add JMenu** del menú contextual.
3. Pinchar con el botón derecho del ratón en **JMenu1**.
4. Seleccionar **Add JMenuItem** en el menú contextual.
   Además de **JMenuItem**, es posible elegir `JCheckBoxMenuItem`, `JRadioButtonMenuItem`, `JMenu` (para submenús), y `JSeparator`.
5. En **JMenu2** añadir otro `JMenuItem`, un `JSeparator` y un `JMenuItem`.
6. Renombrar los `JMenu` y `JMenuItem`, modificando además algunas propiedades según muestra la siguiente tabla:

| Nombre original | Nuevo nombre | Text | Tooltip text | Mnemonic |
|---|---|---|---|---|
| jMenu1 | FileMenu | File | File | F |
| jMenuItem1 | ExitItem | Exit | Exit | X |
| jMenu2 | HelpMenu | Help | Help | H |
| jMenuItem2 | ContetsItem | Contents | Contents | C |
| jMenuItem3 | AboutItem | About | About | A |

El nuevo aspecto del programa es el siguiente:




## 5. Modificando el gestor de posicionamiento

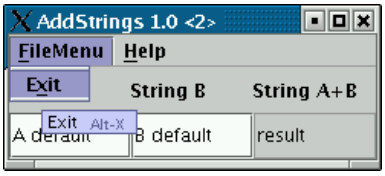
El gestor de posicionamiento que hemos usado (`FlowLayout`) nos ha permitido cambiar el tamaño de los componentes. Pero ahora queremos que las tres etiquetas queden alineadas juntos con los tres campos de texto. Para conseguir esto cambiaremos de nuevo el gestor de posicionamiento al tipo `GridLayout`.

1. Pincha con el botón derecho en el nodo **JFrame** del inspector de componentes.
2. Selecciona **GridLayout** en el menú contextual.

3. Selecciona el nodo **GridLayout** en el inspector de componentes.

4. Cambia la propiedad **Columns** al valor 3 y la propiedad **Rows** al valor 2.

Ahora todo queda bien alineado, aunque hemos perdido en parte el control sobre el tamaño de los componentes, ya que `GridLayout` hace que todas las celdas tengan el mismo tamaño, suficientemente grande para que quepa el más grande de ellos. El resultado ahora es:




## 6. Copiando objetos

Es posible copiar y pegar objetos en la ventana Explorer con facilidad. Pueden copiarse miembros de clases (campos, constructores, métodos, patrones bean) entre clases, y también componentes GUI entre forms. Copiemos la clase que hemos construido hasta el momento en un nuevo programa:

1. Crear un nuevo paquete para la copia. Pincha con el botón derecho en el sistema de ficheros montado, y selecciona **New Java Package**, y nombra *GuiDemoAdvanced* al nuevo paquete.

2. Pincha con el botón derecho en el Explorer, en el nodo **AddStrings** del antiguo paquete *GuiDemoBasic* .

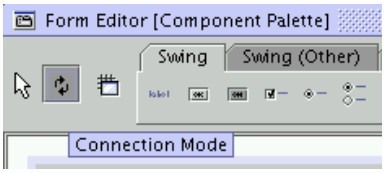
3. Pincha con el botón derecho el nodo *GuiDemoAdvanced* y seleccionar **Paste Copy** en el menú contextual.

4. Haz doble click en el nuevo nodo `AddStrings` para abrirlo en el las ventanas Source Editor y Form Editor. Notar que el nombre del paquete ha sido cambiado automáticamente.

5. En el inspector de componentes define la propiedad *título* (title) con *AddStrings 2.0*.

## 7. Asistente de conexión

Nuestro objetivo en el programa que estamos construyendo es que el tercer `JTextField` (`tfSum`) muestre la concatenación de los otros dos. Si `tfA` o `tfB` recibe un `ActionEvent`, la propiedad text de `tfSum` debe cambiar para mostrar el resultado.
El asistente de conexión nos ayuda a establecer enlaces entre los eventos de un componente y las acciones de otro.
El asistente de conexión se activa pinchando en el botón **Connection Mode** de la paleta de componentes (botón que está junto al de **Selection Mode**).

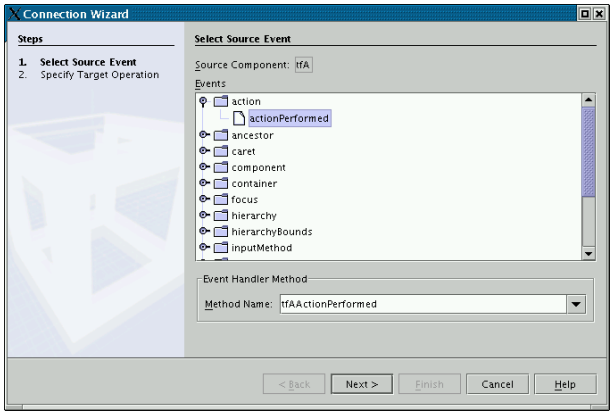



1. Selecciona el modo Conexión.

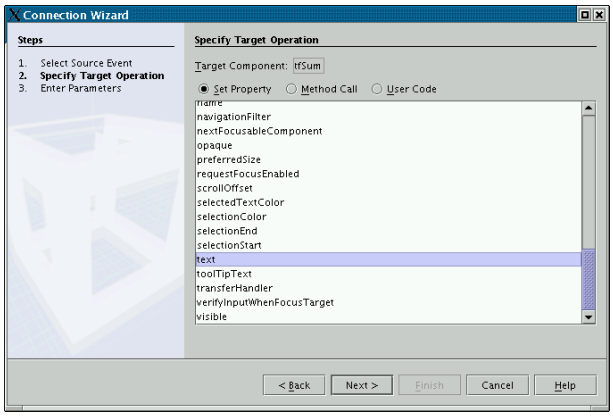
2. Selecciona el componente que enviará el `ActionEvent` en Form Editor. O sea, el `JTextField` llamado `tfA`.

3. Selecciona ahora el componente destino. O sea, el `JTextField` llamado `tfSum`.

4. El asistente de conexión se lanza. Abre el nodo action, y selecciona **actionPerformed**. Acepta el nombre por defecto del método `tfAActionPerfomed`, y pulsa **Next**.

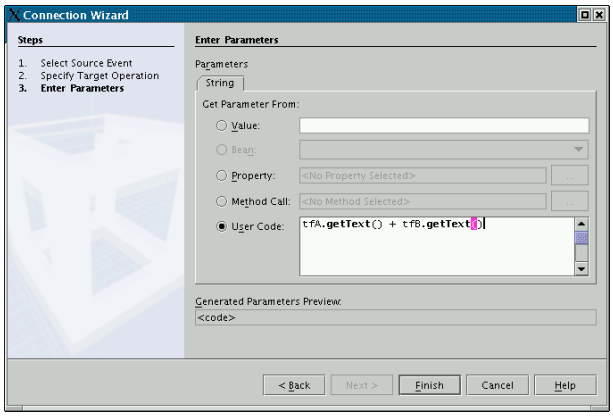



5. En el siguiente panel del asistente mantener la operación por defecto *Set Property*, seleccionar la propiedad **text**, y pinchar en **Next**.




6. En el último panel especificar *Get Parameter From* pulsando el botón **User Code**. Introducir luego el siguiente código y pulsar en **Finish**:

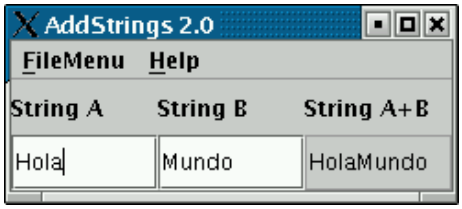
```
tfA.getText() + tfB.getText()
```




7. Si miramos en la ventana Source Editor podemos ver se ha añadido un nuevo método `tfAActionPerformed`, para manejar el `ActionEvent` que lanza el componente `tfA`.

8. Repetir la operación para el otro campo de texto, `tfB`.

9. Compilar y ejecutar. Cambiar el campo de texto para comprobar que funciona. El campo de texto resultado, `tfSum`, muestra `tfA` concatenado con `tfB`, como ilustra la siguiente figura.

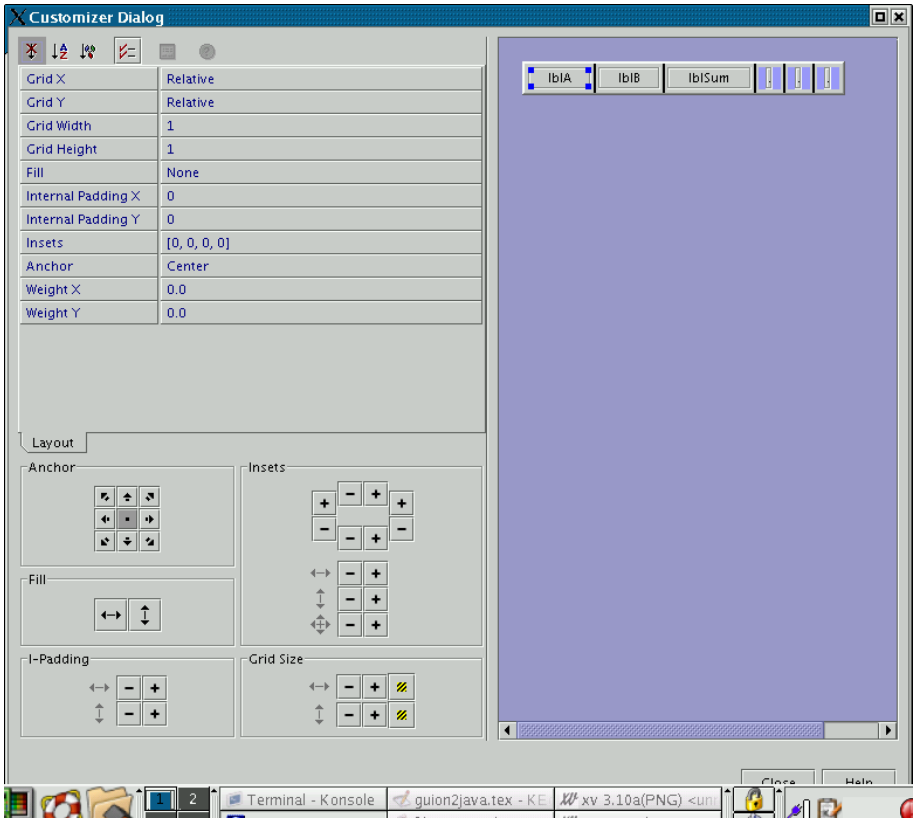



## 8. El gestor de posicionamiento GridBagLayout

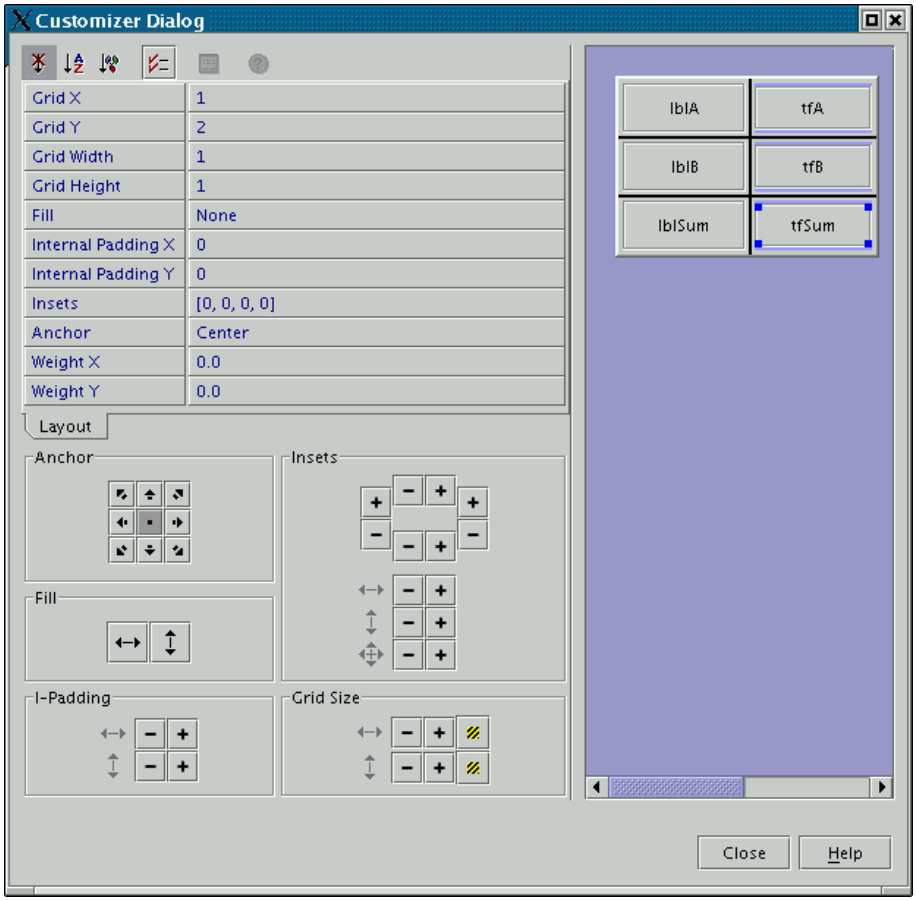
Cuando se necesita más control sobre el tamaño y posición de los componentes GUI, puede usarse `GridBagLayout`. También podriamos utilizar `NullLayout`, pero esto reduciría la portabilidad entre distintas plataformas. `GridBagLayout` es un gestor de posicionamiento estándar de Java que fue diseñado para control preciso de los componentes GUI. El inconveniente que presenta es que es complejo de configurar y tedioso de modificar. Sin embargo usando el *Optimizador de GridBagLayout* de Netbeans se facilita bastante la tarea. Con él, podemos ajustar visualmente el tamaño y posición de los componentes y luego ajustar numéricamente para obtener una precisión más exacta.
`GridBagLayout` también divide el contenedor en una rejilla de varias filas y columnas (al igual que `GridLayout`). Pero ahora, las filas y columnas no necesitan ser todas de la misma altura o anchura.
Vamos a utilizar el *Optimizador de GridBagLayout* para nuestro programa `AddStrings`:

1. Establecer `GridBagLayout` como gestor de posicionamiento. Puede hacerse arrastrando `GridBagLayout` desde la paleta de componentes hasta el editor del form, o pinchando con el botón derecho del ratón en el nodo `JFrame` del inspector de componentes y seleccionar **Set Layout** en el menú contextual.

2. Iniciar el optimizador. En el inspector de componentes pinchar con el botón derecho el nodo `GridBagLayout`, y luego seleccionar **Customize** del menú contextual para abrir el optimizador (**Customizer Dialog**).




3. Recoloca los componentes `JLabel` y `JTextField` según la siguiente figura:



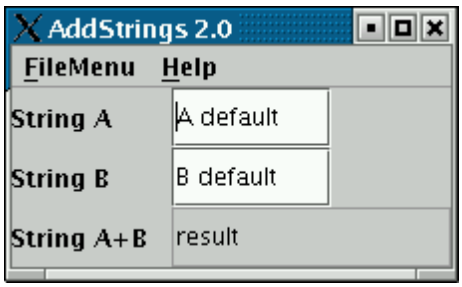

4. Ajustar algunas propiedades de los componentes. Seleccionar el componente en el panel derecho, y luego ajusta los valores con los paneles de la izquierda. Puede hacerse ajustando las propiedades de la parte superior, o bien con los controles visuales de la parte inferior.

   Nosotros sólo ajustaremos el campo de texto `tfSum`. Debería ser del doble de tamaño de los campos de entrada de texto. Ajusta las propiedades según la siguiente tabla, luego cierra el optimizador, compila y ejecuta.

| **Propiedad** | **Nuevo valor** |
|---|---|
| Fill | Horizontal |
| Grid width | 4 |
| Internal padding X | 60 |

5. Las propiedades del gestor de posicionamiento pueden también ajustarse sin abrir el optimizador. Basta con seleccionar el componente en el inspector de componentes y luego hacer pinchar la solapa **Layout** de la parte de abajo.

Al ejecutar ahora el programa obtenemos:



## 9. Añadir manejadores de eventos

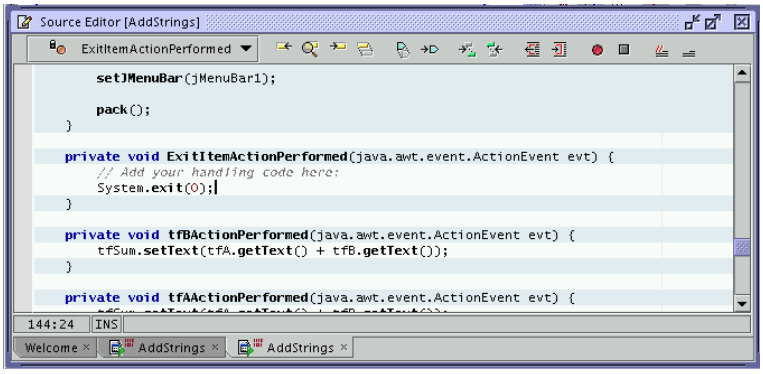
Miremos detallamente el código que genera Netbeans para manejar eventos del GUI. Automáticamente crea un método para manejar el evento `windowClosing`. Localizar el método `exitForm` en el código fuente. Este método es el código por defecto, que se ejecutará con el evento `windowClosing`, y que fue creado cuando se creo el contenedor `JFrame`. La primera y última líneas de `exitForm` son código reservado (sombreadas en azul claro), pero el cuerpo del método no lo es. Podemos modificar libremente el cuerpo de este método. Este es el patrón común para métodos manejadores de eventos que genera Netbeans. Por defecto el método `exitForm` para un `JFrame` contiene sólo la siguiente línea:

```
System.exit(0);
```

Además del evento `windowClosing`, podemos manejar los demás eventos de los componentes GUI, creando un método por cada evento que seleccionemos manualmente. El *asistente de conexión* incluía una forma para seleccionar un evento y luego crear un método para manejarlo. Veamos otra forma de generar un método para manejar un evento.

1. Abrir el nodo **ExitItem** en el Explorer: **AddStrings** → **Form AddStrings** → [**JFrame**] → **jMenuBar1** → **FileMenu** → **ExitItem**

2. Pinchar con el botón derecho este nodo y seleccionar en el menú contextual **Events** → **Action** → **actionPerformed**. Como resultado se añade el método `ExitItemActionPerformed()` al código fuente.

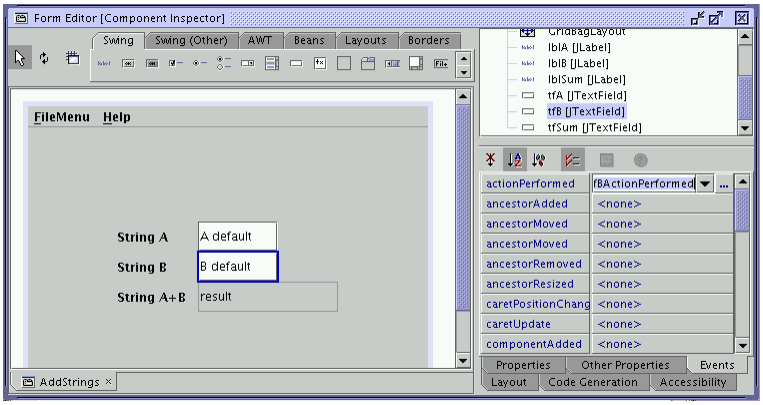



3. Localizar este método en el código fuente. Es un método vacío. Copiar o escribir la sentencia `System.exit(0);` del método `exitForm` en el método `ExitItemActionPerformed()`.

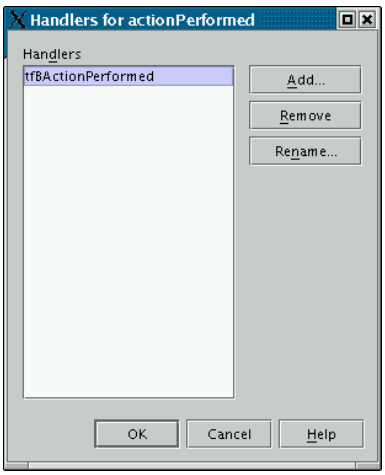
Ahora la opción **Exit** del menú **FileMenu** hace que termine el programa.
Veamos otra forma para asociar un método al evento de un componente. En la versión actual del programa tenemos dos métodos `tfAActionPerformed` y `tfBActionPerformed` que fueron creados con el *asistente de conexión* para conectar las acciones en los campos de texto `tfA` y `tfB`, con el resultado en `tfSum`. El asistente de conexión creo estos métodos vacíos, que luego nosotros rellenamos con código para que apareciese el resultado en `tfSum`. Sin embargo los dos métodos contienen exactamente el mismo código. Esto es redundante, por lo que vamos a solucionarlo.

1. Seleccionar `tfB` en el inspector de componentes, y pinchar en la solapa **Event** de la parte inferior de la ventana.

2. Pinchar `tfBActionPerformed` en la parte superior de la columna derecha, y luego pinchar el botón ....




3. En la ventana de diálogo que aparece seleccionar **Remove** para eliminar el método `tfBActionPerformed`.



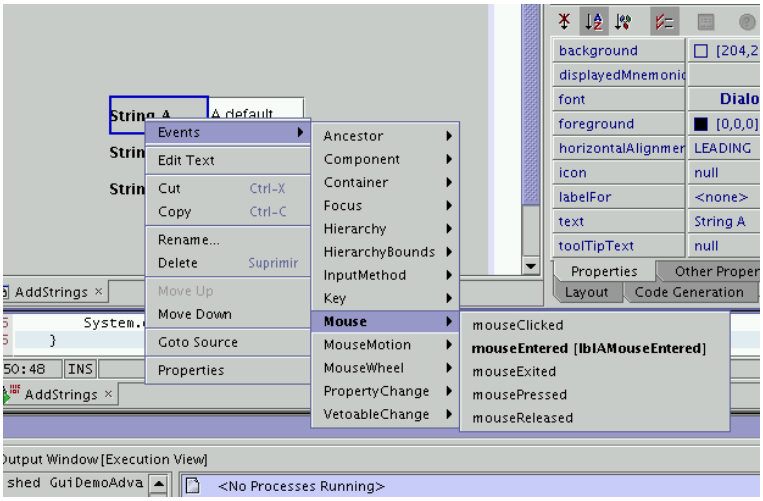

4. Añadir ahora el método `tfAActionPerformed`, y pinchar en **OK** para terminar.

Si miramos el código fuente podemos ver que el método `tfBActionPerformed` ha sido eliminado. Ahora tenemos un programa algo más simple. Esta aproximación permite asociar fácilmente el mismo método para manejar varios eventos.
Veamos dos últimas formas de crear un método manejador de eventos.

1. En el inspector de componentes, pinchar con el botón derecho en el nodo `lblA`.
2. En el menú contextual seleccionar **Events** → **Mouse** → **mouseEntered**. Esto crea el método `lblAMouseEntered()`.
3. Insertar la siguiente línea en este método:

   ```
   lblA.setForeground(java.awt.Color.red);
   ```

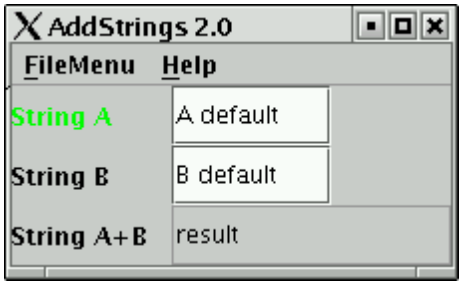
4. Ahora en el editor de forms, pincha con el botón derecho del ratón en la etiqueta `lblA` (con texto **String A**).
5. Selecciona **Events** → **Mouse** → **mouseExited** en el menú contextual.




6. Inserta el siguiente método en el método `lblAMouseExited()`:

   ```
   lblA.setForeground(java.awt.Color.green);
   ```

Compila el programa y ejecútalo. Ahora la etiqueta **String A** cambia de color cuando el cursor del ratón entra y sale el rectángulo que la encuadra.




## 10. Contenedores dentro de contenedores

Un `JPanel` puede contener cualquier número de componentes. Puesto que un `JPanel` es él mismo un componente, puede colocarse dentro de otro `JPanel`, un `JFrame`, u otro contenedor GUI. Esto permite la construcción de complejos GUIs.
Si todos los componentes de todos los contenedores de un programa, fuesen visibles al mismo tiempo en Netbeans, resultaría vialmente confuso e inmanejable. El editor de forms evita esta confusión mostrando sólo un contenedor con sus componentes al mismo tiempo. Veremos que el desarrollador tiene una forma sencilla para elegir el contenedor a mostrar.
Vamos a rellenar el espacio libre que queda en la parte superior derecha de `AddStrings` con un `JPanel`. En este `JPanel` colocaremos un par de componentes.

1. Seleccionar el icono **JPanel** de la paleta de componentes y luego pincha la esquina vacía de `AddStrings` en el editor de form para añadir el `JPanel`. Esto hace que aparezca un pequeño cuadrado azul en esta esquina del editor de forms.
   Usemos el optimizador del *GridBagLayout* para ajustar el espacio que ocupa.

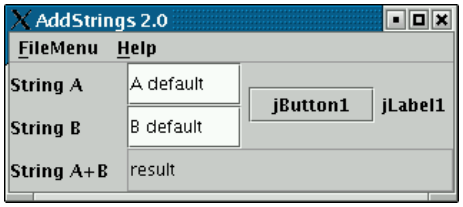
2. Pincha con el botón derecho del ratón el nodo **GridBagLayout** en el inspector de componentes y selecciona **Customize** en el menú contextual.

3. Selecciona el `JPanel` en la ventana de diálogo del optimizador y luego ajustar *GridWidth* al valor 3 y *GridHeight* al valor 2.

4. Cerrar el optimizador.

Si miramos el editor de forms podemos comprobar que el `JPanel` parece no estar en el form, ya que el pequeño cuadrado azul ha desaparecido. No hay problema, el `JPanel` está todavía en el inspector de componentes.

1. Selecciona el `JPanel` en el inspector de componentes, lo que hará que aparezca de nuevo el cuadrado azul.
2. Pincha con el botón derecho del ratón el `JPanel` en el inspector de componentes y selecciona **Design This Container** del menú contextual. Esto hará que el `JPanel` ocupe todo el espacio y el `JFrame` desaparezca.
3. Insertar un `JButton` y un `JLabel` desde la paleta de componentes en el `JPanel`.
4. Compila y ejecuta para comprobar el resultado.

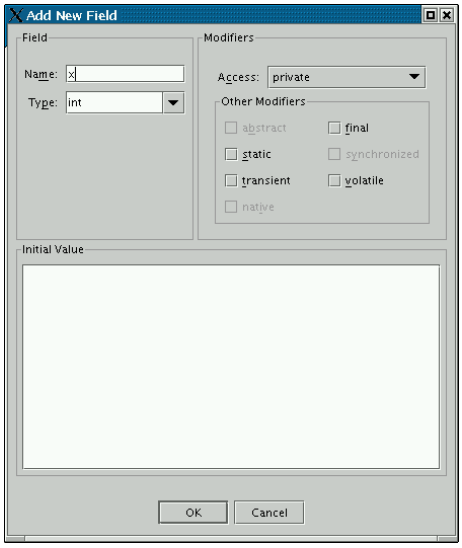



En este momento el editor de forms muestra el `JPanel`. Para volver al `JFrame` pincha con el botón derecho del ratón y comprueba que ahora las aparecen las opciones **Design This Container** y **Design Top Container**. Esto nos da una forma sencilla para movernos entre el `JFrame` y el `JPanel`.

## 11. Creación de un panel para dibujar y gestionar sus eventos

Vamos a modificar el programa *AddStrings* para añadir un panel en el que dibujaremos una línea desde el vértice superior izquierda, hasta el último punto en el que hicimos click con el ratón. Inicialmente la línea estará dibujada desde el vertice superior izquierda hasta las coordenadas $50, 50$. El nuevo panel sustituirá al antiguo panel `jPanel1`. Este nuevo panel debe hacerse creando una nueva clase que sea subclase de `JPanel`, en la que sobreescribiremos el método `paint()`.

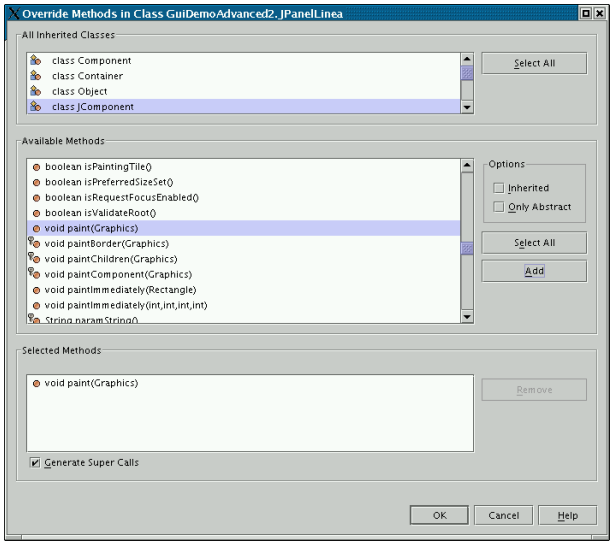
1. En primer lugar crearemos un nuevo paquete llamado **GuiDemoAdvanced2**.
2. Usando el Explorer, copia el nodo **AddStrings** del antiguo paquete **GuiDemoAdvanced** en el nuevo paquete **GuiDemoAdvanced2**.
3. Haz doble click en el nuevo nodo **AddStrings** para abrirlo en el editor de forms y editor de código fuente.
4. En el inspector de componentes selecciona el nodo `jPanel1` y elimínalo mediante el menú contextual.
5. Selecciona el nodo GuiDemoAdvanced2 en el Explorer, y crea un nuevo **JPanel**: **Menú File** → **New** → **Java GUI Forms** → **JPanel Form**. Nombraremos *JPanelLinea* a esta nueva clase.
6. Añade dos campos `x` e `y` de tipo `int` a la clase `JPanelLinea` que nos servirán para guardar la última posición donde hemos pinchado con el ratón dentro del panel. Para añadir estos campos podemos hacerlo usando el editor del código fuente, o bien pinchando con el botón derecho el nodo **Fields** en la clase `JPanelLinea` del Explorer, y seleccionando **AddField** en el menú contextual.



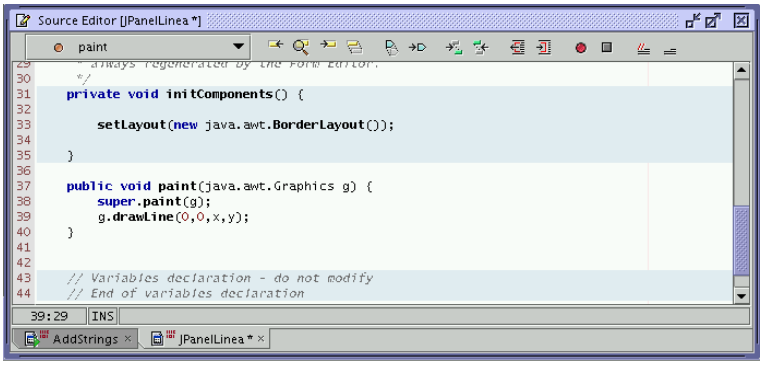

7. Usando el editor del código fuente incluye las sentencias de inicialización de `x` e `y` en el constructor de `JPanelLinea`, después de la llamada a `initComponents();`.

```
x=50;
y=50;
```

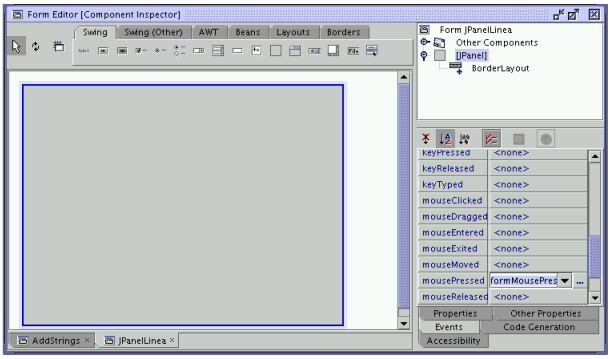
8. Sobreescribe el método `paint(Graphics)` en la clase `JPanelLinea` (método heredado de `JComponents`). Para ello selecciona el nodo **class JPanelLinea** en el Explorer, pincha con el botón derecho del ratón y selecciona **Override Methods...**. En el diálogo que aparece selecciona la clase **JComponent** y el método **void paint(Graphics)**, pincha en el botón **Add** y luego el botón **OK**.




9. Añade la sentencia `g.drawLine(0,0,x,y);` al final del método `paint()`.



10. Manejar el evento `mousePressed`. Puede hacerse seleccionando el nodo **JPanel** en el inspector de componentes, luego la solapa **Events** y pinchando finalmente en el evento `mousePressed`. Este paso hará que se añada el método `formMousePressed(java.awt.event.MouseEvent evt)` a la clase `JPanelLinea`.




11. Añade las sentencias siguientes en el método `formMousePressed(java.awt.event.MouseEvent evt)`:

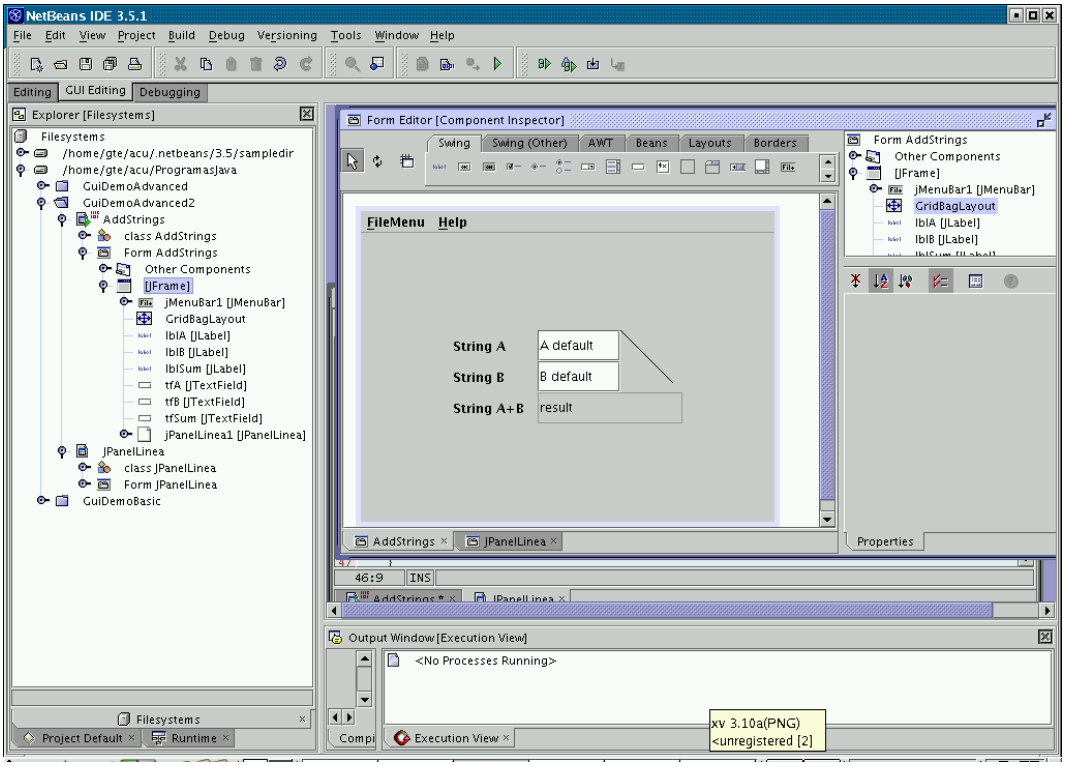
```
x=evt.getX();
y=evt.getY();
repaint();
```

12. Compila la clase `JPanelLinea`. Este paso es muy importante, ya que de no hacerlo el siguiente paso no se podrá hacer.

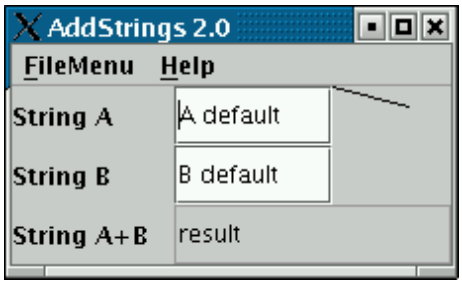
13. Añade el panel `JPanelLinea` como componente del `JFrame` de *AddStrings*. Para ello utilizando el Explorer:

    - Abre nodos de *AddStrings* hasta que sea visible el nodo [**JFrame**].
    - Copia `JPanelLinea` al portapapeles (con el botón derecho del ratón pincha el nodo `JPanelLinea` y selecciona *Copy* en el menú contextual).
    - Pega el panel en el `JFrame`: Pincha con el botón derecho del ratón el nodo [**JFrame**] de **AddStrings**, y selecciona **Paste** en el menú contextual. Este paso hará que se añada al `JFrame` de *AddStrings* un `JPanelLinea` llamado `jPanelLinea1`.
    - Abre el optimizador del `GridBagLayout` del `JFrame` de *AddStrings*.
    - Modifica las siguientes propiedades del `JPanelLinea`:

| **Propiedad** | **Valor** |
|---|---|
| Grid Width | 3 |
| Grid Height | 2 |
| Fill | Both |




Compila y ejecuta el programa. Puedes comprobar que al pinchar con el ratón dentro del panel, aparece una línea desde el vértice superior izquierda del panel hasta el punto donde hemos pinchado.




## 12. Localización de los programas ya terminados

En este guión se han desarrollado tres programas. Podéis encontrar los ficheros .java y .form de estos tres programas en la dirección `http://decsai.ugr.es/~acu/NTP/archivos/ProgramasGuion2Java`.