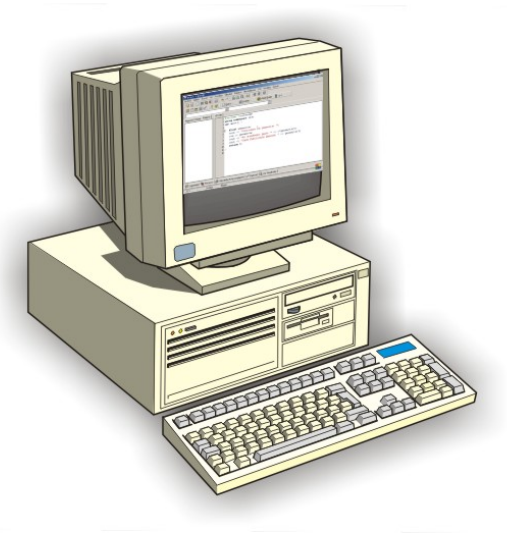


Guión 4

Construcción de interfaces gráficos de usuario con Netbeans 4.1

Noviembre de 2005

DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Nuevas Tecnologías de la Programación

Curso 2005/2006

Índice

1. Pasos iniciales

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. Comienza ejecutando el entorno Netbeans.

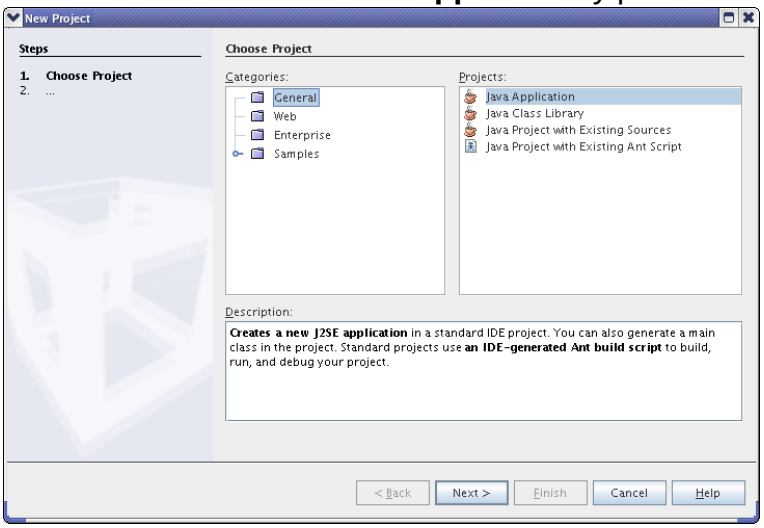
En el IDE de Netbeans, todos los desarrollos Java necesitan la creación de un proyecto. Un proyecto es un grupo de ficheros fuente Java más sus *metadatos* asociados, incluyendo ficheros de propiedades, un script Ant que controla parámetros de compilación y ejecución, etc.

1.1. Creación del proyecto

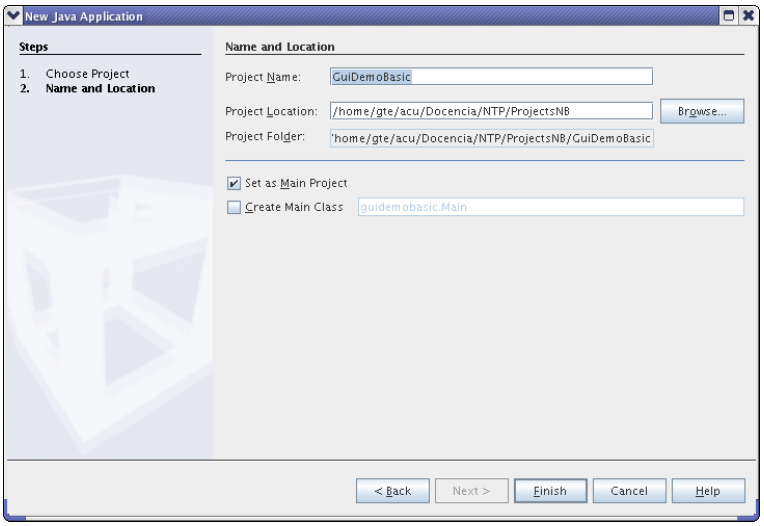
Para crear el proyecto de nuestra aplicación seguiremos los siguientes pasos:

1. Selecciona **Menú File** → **New Project (Ctrl + Shift + N)**, o bien pulsa en el icono **New Project** de la *barra de utilidades* del IDE.

2. Seleccionamos **General** → **Java Application** y pulsa el botón **Next**.



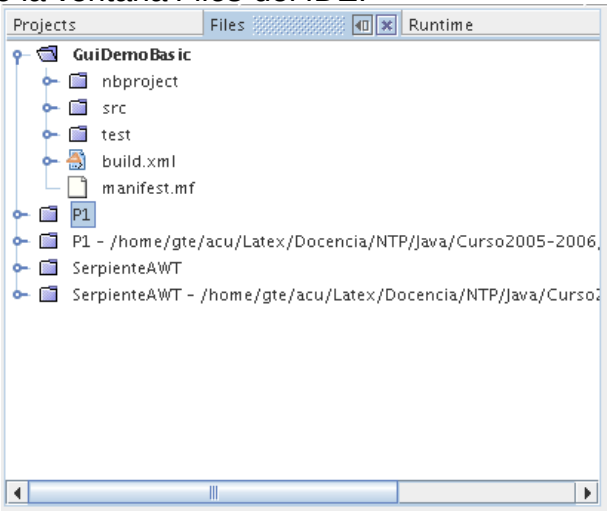
3. Como nombre del proyecto introducimos `GuiDemoBasic`. Como carpeta (directorio) donde colocar el proyecto usaremos el directorio `ProjectsNB/GuiDemoBasic`. La carpeta `ProjectsNB` debe estar previamente creada dentro de tu *home*. Para ayudarte a seleccionar esta carpeta puedes pinchar en el botón **Browse**.



4. Asegúrate que está seleccionada la opción *Set as Main Project* y no seleccionada la opción *Create Main Class*.

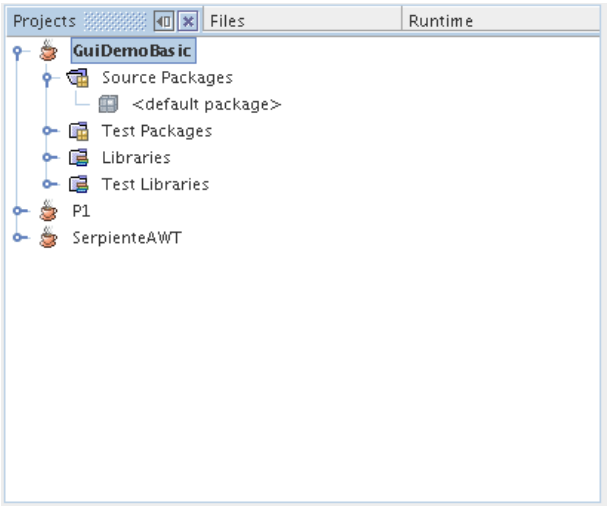
5. Pulsa el botón **Finish**.

Los pasos anteriores han creado una carpeta *GuiDemoBasic* en nuestro sistema de ficheros en el sitio deseado. Esta carpeta contiene todos los ficheros asociados al proyecto. Puede verse la estructura de carpetas creadas usando la ventana *Files* del IDE.



1.2. Creación del contenedor de objetos

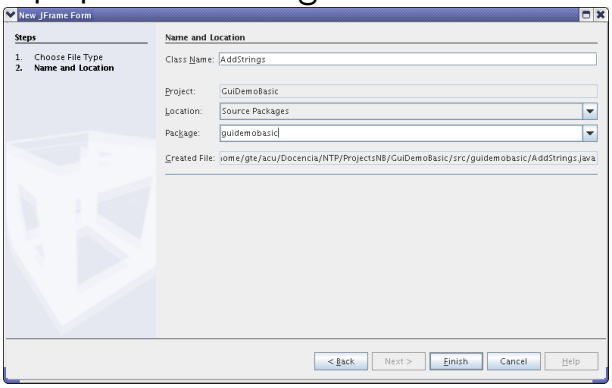
Por ahora, podemos observar que la carpeta *Source Packages* en la ventana *Projects* contiene un paquete `<default package>` que está vacío por ahora:



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. Nosotros crearemos un contenedor *JFrame* y lo colocaremos en un nuevo *paquete*. Otros posibles contenedores son `Applet`, `Dialog`, `Frame` o `Panel` de AWT, y `JApplet`, `JDialog`, `JInternalFrame`, `JFrame` o `JPanel` de Swing, que pueden seleccionarse con **Menú File** → **New File** → **Java Gui Forms**.

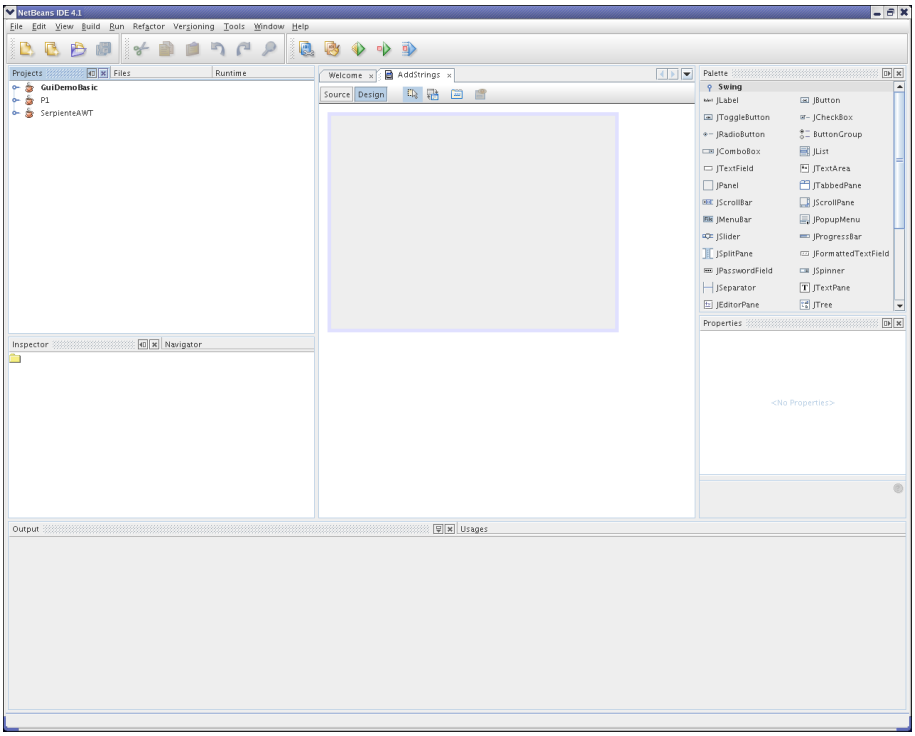
1. Pinchar con el botón derecho del ratón sobre el nodo *GuiDemoBasic* de la ventana de proyectos, y elegir **New** → **JFrame Form**

2. Introduce `AddStrings` como nombre para el nuevo *JFrame*. Como nombre del paquete introduce `guidemobasic`.



3. Pulsa el botón **Finish**.

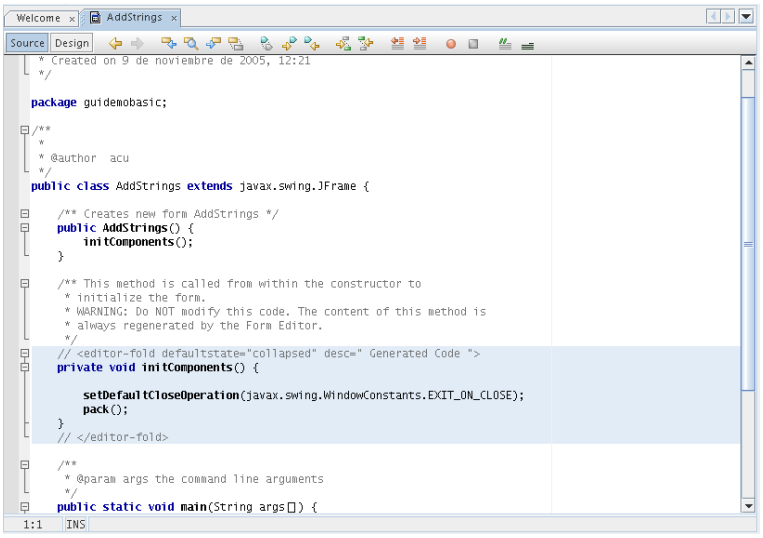
El IDE crea el form *AddStrings* y la clase *AddStrings* dentro del fichero `AddString.java`. El paquete `guidemobasic` sustituye ahora al paquete `<default package>`, y el form *AddStrings* se abre en la ventana *Editor* en la vista de diseño (*Design*) que muestra una vista gráfica de los componentes GUI. Además se abre una ventana para la *paleta de componentes* en la parte superior derecha del IDE, el *inspector de componentes* aparece en la parte izquierda debajo de la ventana de proyectos, y la *ventana de propiedades* aparece en la parte derecha debajo del *inspector de componentes*.



Una breve descripción de las ventanas que aparecen ahora en el IDE, mencionadas anteriormente es:

- Inspector: Proporciona una representación de todos los componentes, visuales y no visuales, en la aplicación como una estructura jerárquica. Además nos muestra cual componente en el árbol está siendo editado actualmente en el *Editor*.
- Paleta de componentes (*Palette*: Contiene solapas para varios componentes JFC/Swing, AWT y JavaBeans, junto con una selección de *layout managers*.
- Propiedades (*Properties*): Muestra las propiedades de los componentes que está actualmente seleccionado en el Inspector, Editor, ventana de Proyectos o ventana *Files*.
- Editor: Aparece en la parte central del IDE. En la figura anterior el Editor en modo de diseño (*Design*) el cual muestra una vista visual del form `JFrame`. También puede seleccionarse el modo de código fuente (*Source*) que mostrará el código fuente asociado a este `JFrame`.

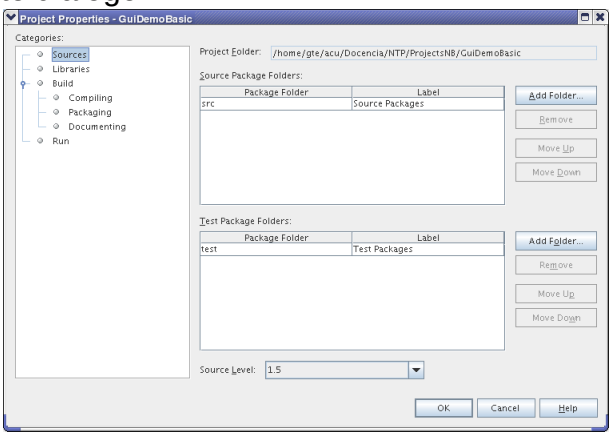
Si pinchamos en la solapa *Source* dentro de la ventana *Editor*, podemos ver el código fuente del *JFrame* que acabamos de crear. Las partes sombreadas en azul corresponden a código generado automáticamente por el IDE (*Guarded Blocks*) que no puede editarse directamente en la vista *Source*. Sólo podemos modificar las partes sombreadas en blanco.



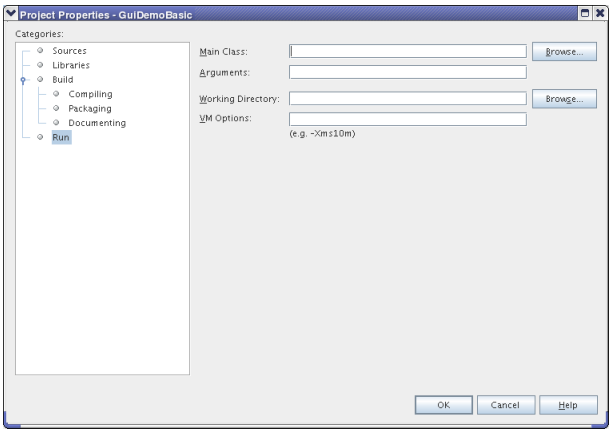
1.3. Definición de la clase principal del proyecto

Debemos ahora definir la clase principal (*Main Class*) de forma que los comandos de construcción y ejecución (*Build* y *Run*) funcionen de forma correcta. Podemos poner como clase principal, cualquier clase con un método *main*, pero en este caso lo que queremos es poner como clase principal, la clase *AddStrings* que hemos creado en los pasos anteriores.

1. En la ventana de proyectos, pincha con el botón derecho en el nodo del proyecto *GuiDemoBasic* y elige *Properties*. Aparecerá entonces el siguiente diálogo:

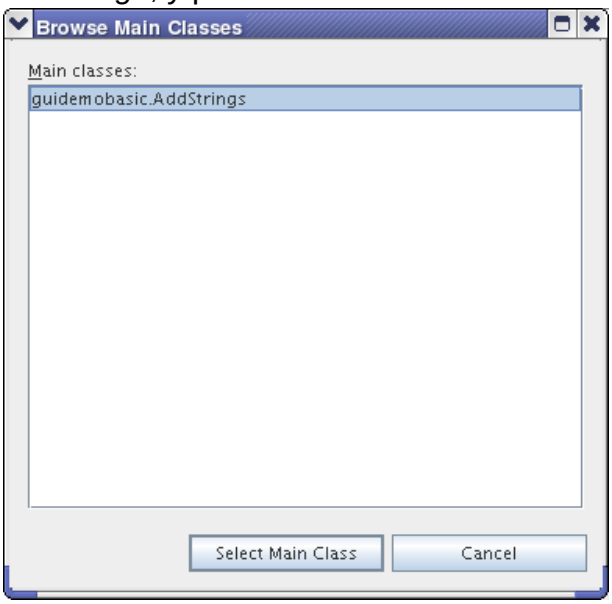


2. En el panel de categorías (*Categories*) del diálogo de propiedades del proyecto, selecciona el nodo *Run*.



3. En el panel derecho, pincha el botón *Browse* que está a la izquierda del campo *Main Class*.

4. En el diálogo *Browse Main Classes* que aparece, selecciona *guidemobasic.AddStrings*, y pulsa *Select Main Class*.



5. Pincha *Ok* para salir del diálogo de propiedades del proyecto.

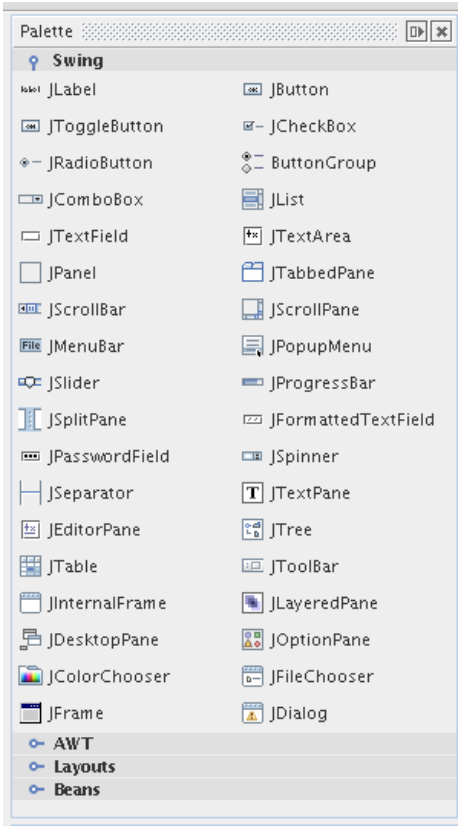
Si observamos con algún navegador de directorios lo que ha ocurrido tras los pasos anteriores, podemos comprobar que se ha creado un directorio llamado `GuiDemoBasic` que contiene varios directorios. Uno de ellos es el directorio `src` que contendrá todo el código fuente del proyecto. Por ahora contiene el directorio `guidemobasic` que corresponde al paquete del mismo nombre. Este directorio a su vez aloja el fichero `AddStrings.java` que contiene el código Java de la clase correspondiente al `JFrame` de nuestro proyecto. 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

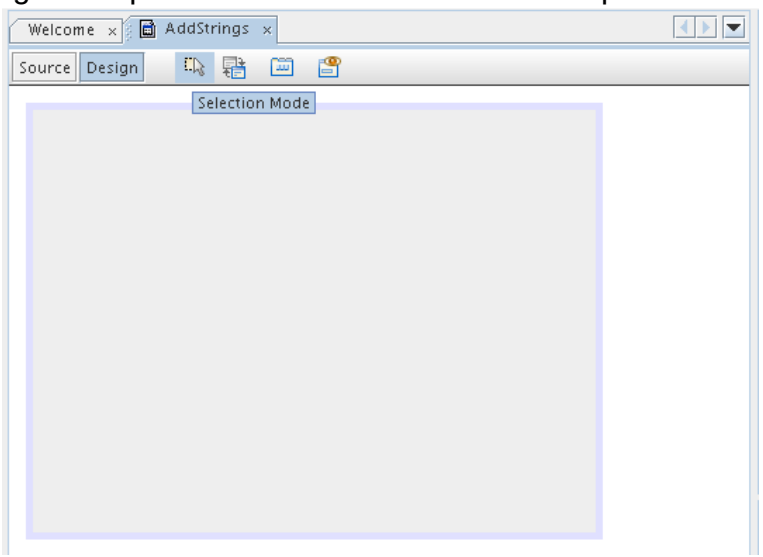
generar un fichero `.form` a partir de código importado. Eso significa que el código importado debe modificarse a mano, en lugar de con el Form Editor.

2. Adición de componentes al contenedor

La paleta de componentes nos permite seleccionar qué componentes queremos añadir al form. Los componentes están clasificados en distintas categorías: Swing, AWT, Layout Managers y JavaBeans.



Los cuatro botones que hay a la derecha de la solapa *AddStrings* en la ventana *Form Editor* permiten seleccionar el modo del Form Editor. Por ahora, aseguraos que el botón *Selection Mode* está pulsado.



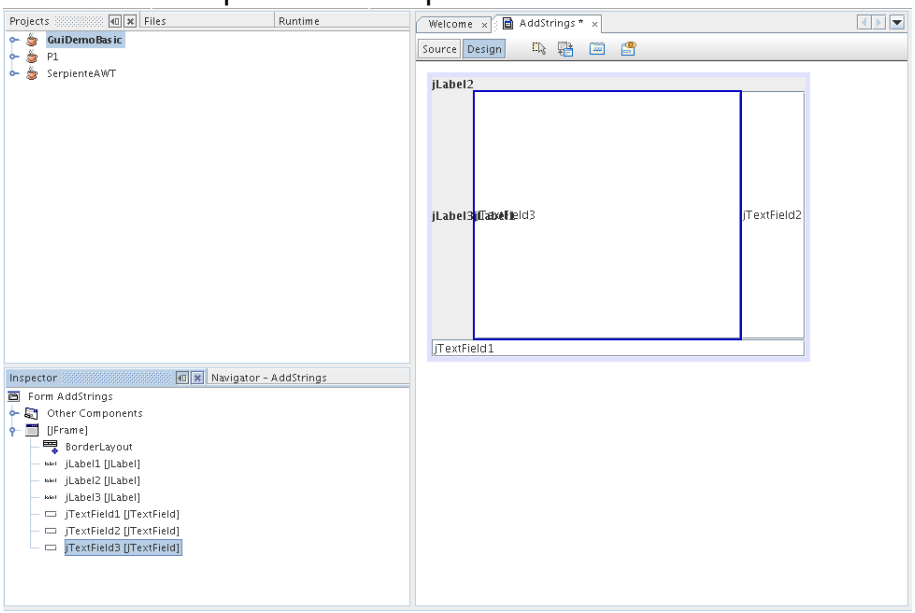
Vamos a añadir tres componentes `JLabel`. Para ello:

1. Desplegar la categoría **Swing** de la paleta de componentes si no lo estaba ya.
2. Seleccionar el componente `JLabel`.
3. Mantener pulsada la tecla **Shift** y pinchar tres veces en el panel del editor de forms con el botón izquierdo del ratón. Esto hace que aparezcan tres `JLabel` etiquetados con *jLabel1*, *jLabel2* y *jLabel3*. Si no hubiesemos mantenido pulsada la tecla **Shift**, sólo se hubiese añadido un `JLabel`.

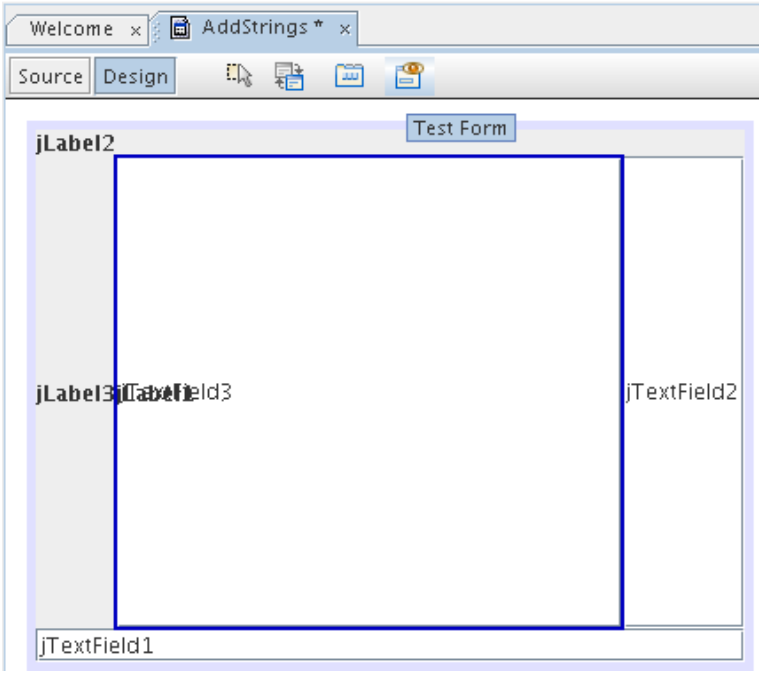
En Java, la colocación y tamaño de los componentes dentro del contenedor está determinada por el gestor de posicionamiento (*Layout manager*) que tenga definido. En este caso el gestor de posicionamiento que tiene definido el `JFrame` es por ahora `BorderLayout`.

Para añadir los componentes `JTextField` usaremos un procedimiento algo distinto.

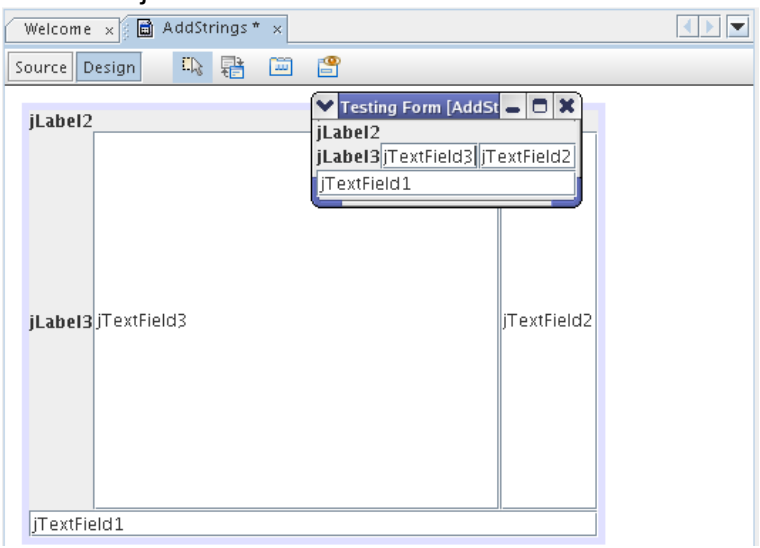
1. Pincha con el botón derecho en el nodo **JFrame** del inspector de componentes.
2. Selecciona **Add From Palette** en el menú contextual y elige **Swing → JTextField**.
3. Repite el proceso tres veces.
4. Notar que los componentes aparecen tanto en el editor de forms como en el inspector de componentes.



Para ver la apariencia del GUI que hemos construido podemos pulsar el botón **Test Form** (colocado en la parte superior derecha 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.

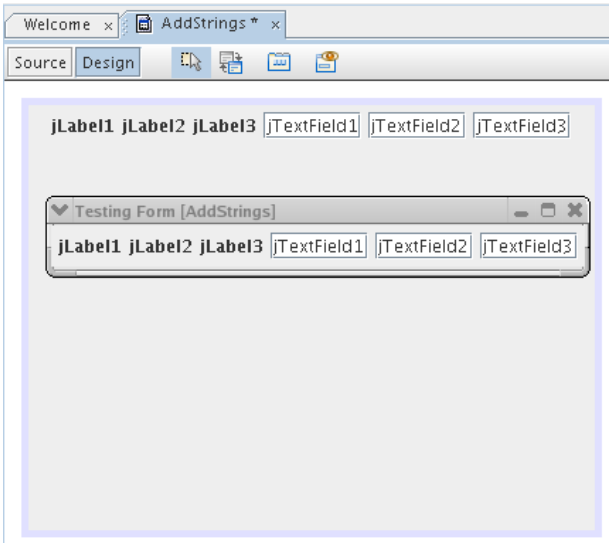


3. Selección del Layout Manager

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

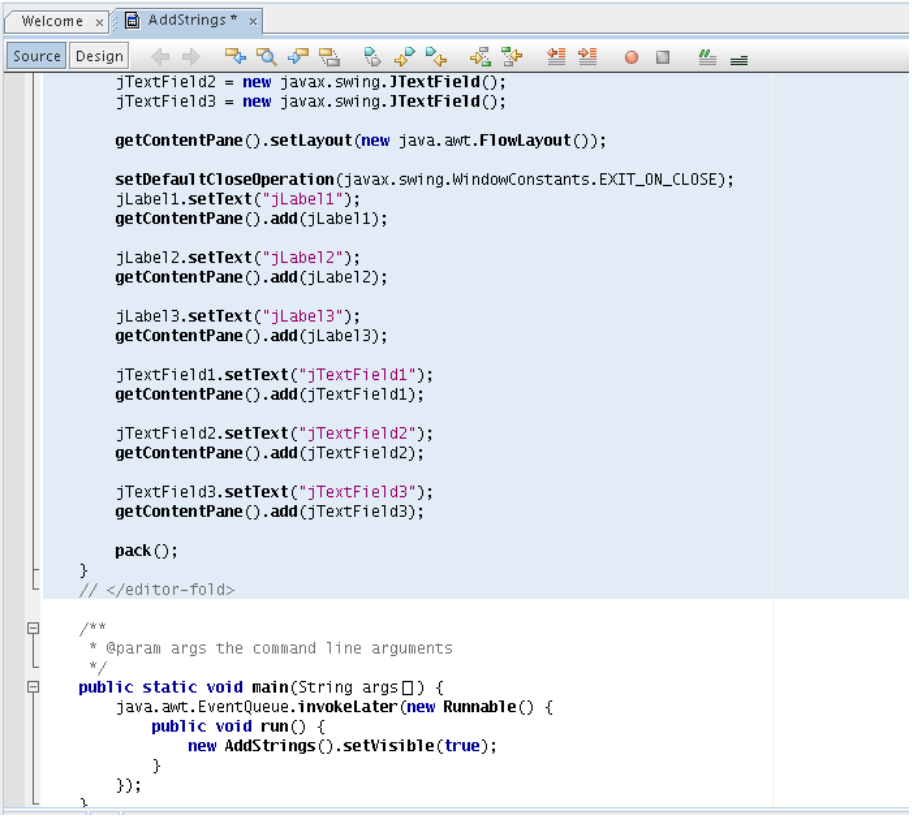
1. Despliega la categoría *Layouts* en la paleta de componentes.
2. Selecciona el botón **FlowLayout**.
3. Pincha en cualquier lugar dentro del `JFrame` en el editor del form.

Si testeamos ahora el form, obtendremos lo siguiente:



4. 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 tocando en el código fuente.

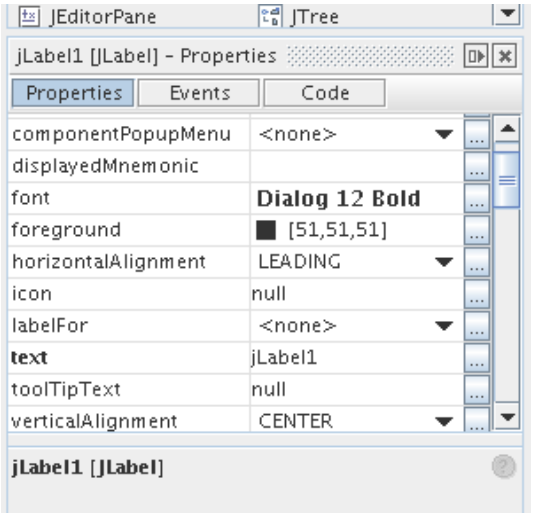


Modifiquemos algunas propiedades de los componentes de nuestro interfaz:

1. Modificación del texto

   a) Seleccionar `jLabel1` en el inspector de componentes.

b) Selecciona la solapa **Properties** (si no lo estaba ya) 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) Dentro de **Properties** despliega la categoría **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 en 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`.

5. Construcción de menús

Ahora añadiremos una barra de menús a nuestro form.

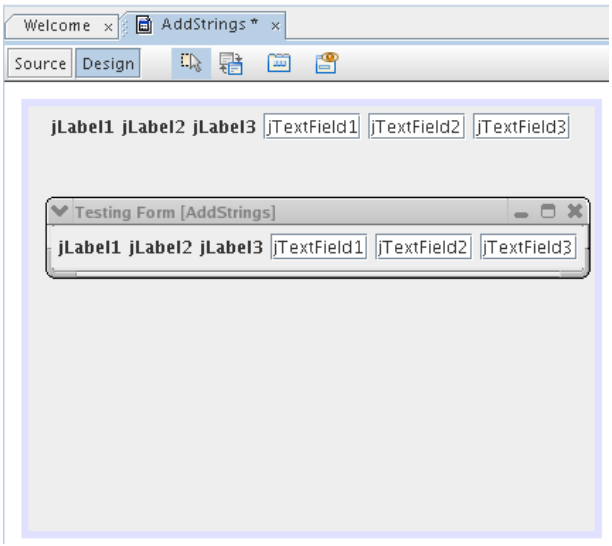
1. Seleccionar **JMenuBar** en la solapa Swing de la paleta de componentes.
2. Pinchar con el ratón en cualquier parte del panel del editor del form.

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 al ejecutarlo con *Test form* es el siguiente:

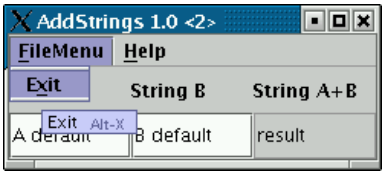


6. Modificando el gestor de posicionamiento

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 juntas 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 **Set Layout** → **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:



7. Copiando objetos

Es posible copiar y pegar objetos en la ventana *Projects* 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 proyecto para la copia. Nombra *GuiDemoAdvanced* al nuevo proyecto. Asegurate que no están seleccionadas las opciones *Set as Main Project* y la *Create Main Class*.

2. Crea el paquete *guidemoadvanced* en el nuevo proyecto *GuiDemoAdvanced*. Puedes hacer esto, pulsando con el botón derecho del ratón sobre el nodo *Source Packages* del proyecto *GuiDemoAdvanced* en la ventana *Projects*, y luego seleccionando **New** $rightarrow$ **Java Package**.

3. En la ventana *Projects*, pincha con el botón derecho en el nodo **AddStrings** del antiguo proyecto *GuiDemoBasic* y selecciona **Copy**.

4. Pincha con el botón derecho el nodo del paquete *guidemoadvanced* en la ventana *Projects* y selecciona **Paste** en el menú contextual.

5. Haz doble click en el nuevo nodo `AddStrings` para abrirlo en las ventanas Source Editor y Form Editor. Notar que el nombre del paquete ha sido cambiado automáticamente si vemos el código fuente de esta nueva clase.

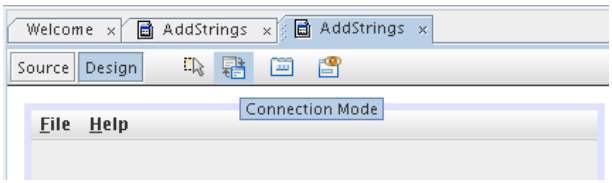
6. En el inspector de componentes define la propiedad *título* (title) con *AddStrings 2.0*.

8. 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** del editor de Forms (botón que está junto al de **Selection Mode**).

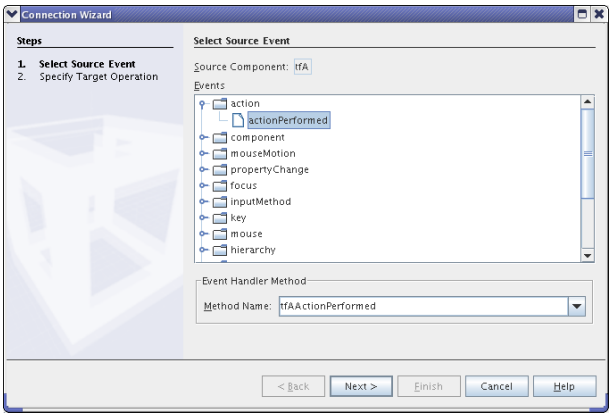


1. Selecciona el modo Conexión.

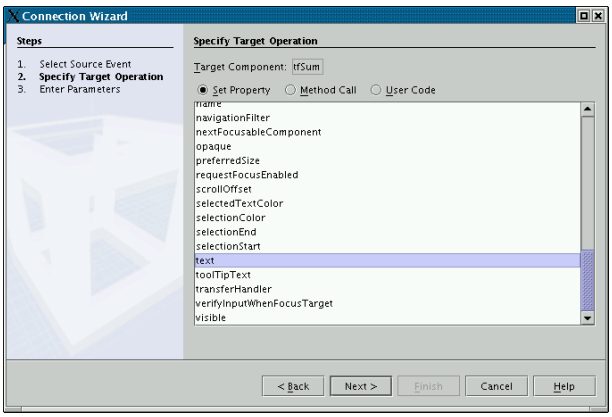
2. Selecciona el componente que enviará el `ActionEvent` en Form Editor. O sea, el `JTextField` llamado `tfA`.

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**.

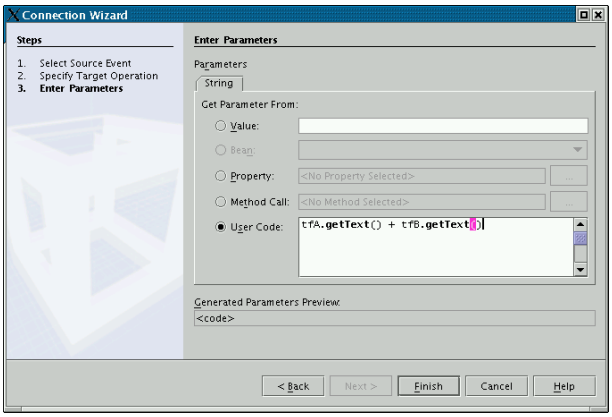


5. En el siguiente panel del asistente, mantener la operación por defecto *Set Property*, seleccionar la propiedad **text**, y pinchar en **Next**.



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

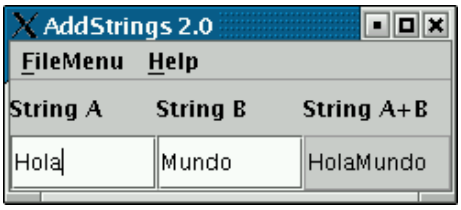
```
tfA.getText() + tfB.getText()
```



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.



9. 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.

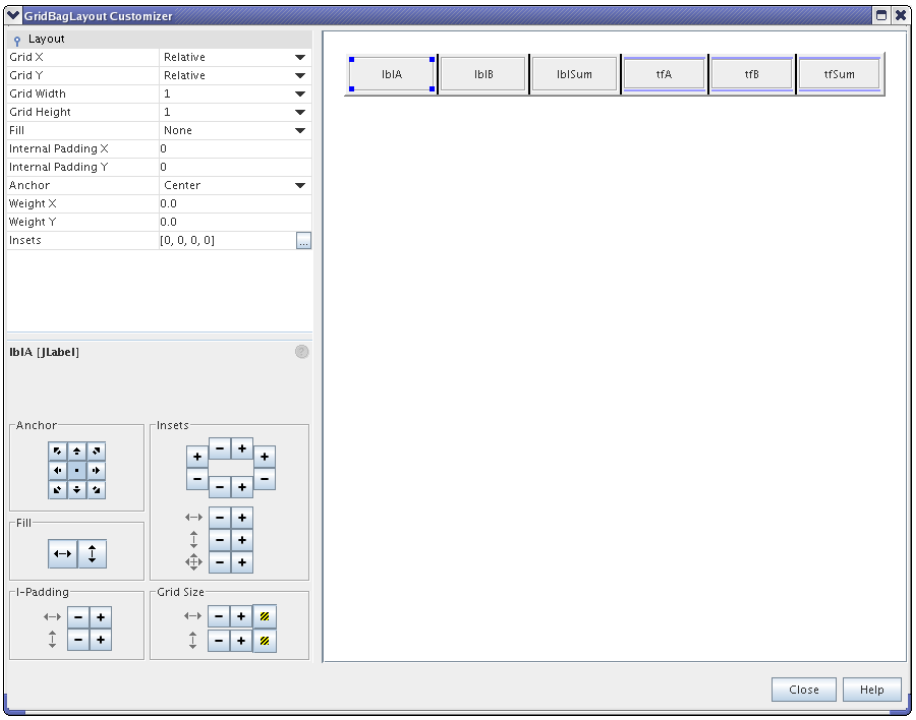
`GridBagLayout` también divide el contenedor en una rejilla de varias filas y columnas (al igual que `GridLayout`). Pero ahora, las filas y columnas no necesitan ser todas de la misma altura o anchura.

Vamos a utilizar el *Optimizador de GridBagLayout* para nuestro programa `AddStrings`:

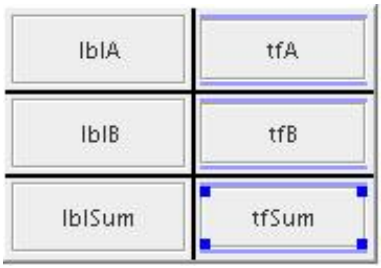
1. Establecer `GridBagLayout` como gestor de posicionamiento. Puede hacerse arrastrando `GridBagLayout` desde la paleta de componentes hasta el editor del form, o pinchando con el botón derecho

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 (**GridBagLayout Customizer**).



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



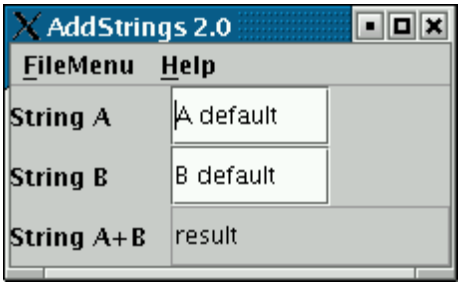
4. Ajustar algunas propiedades de los componentes. Seleccionar el componente en el panel derecho, y luego ajusta los valores con los paneles de la izquierda. Puede hacerse ajustando las propiedades de la parte superior, o bien con los controles visuales de la parte inferior.

   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 de un componente como el texto `tfSum`, pueden también ajustarse sin abrir el optimizador. Basta con seleccionar el componente en el inspector de componentes y luego cambiar las propiedades de la categoría *Layout* en la ventana **Properties**.

Al ejecutar ahora el programa obtenemos:

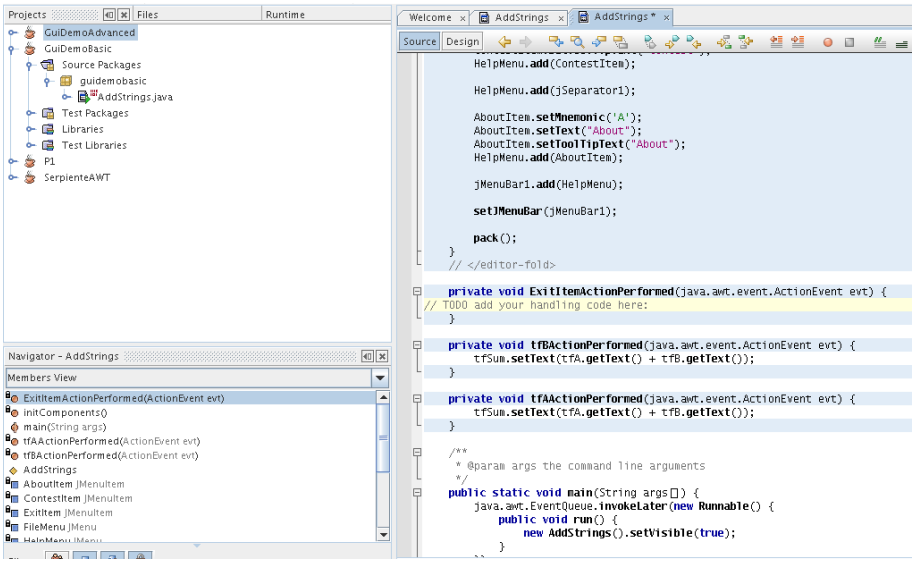


10. Añadir manejadores de eventos

Podemos manejar los 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 inspector de componentes: **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. Escribir la sentencia

   `System.exit(0);`

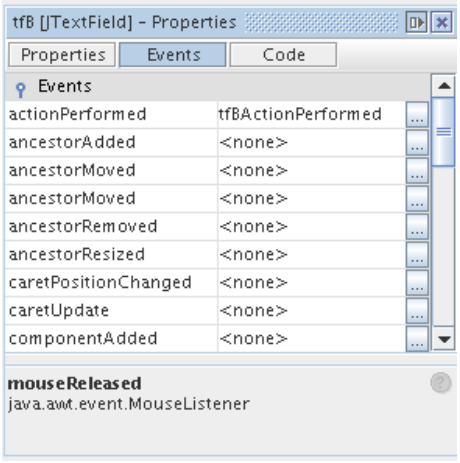
   en el método `ExitItemActionPerformed()`.

   La primera y última líneas de `ExitItemActionPerformed()` 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.

Ahora la opción **Exit** del menú **FileMenu** hace que termine el programa.

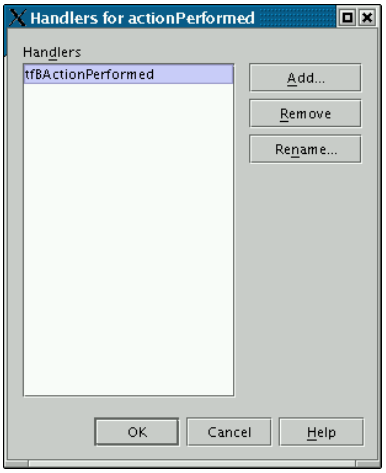
Veamos otra forma para asociar un método al evento de un componente. En la versión actual del programa tenemos dos métodos `tfAActionPerformed` y `tfBActionPerformed` que fueron creados con el *asistente de conexión* para conectar las acciones en los campos de texto `tfA` y `tfB`, con el resultado en `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** que hay al lado de la solapa **Properties** (parte inferior derecha del IDE).



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.

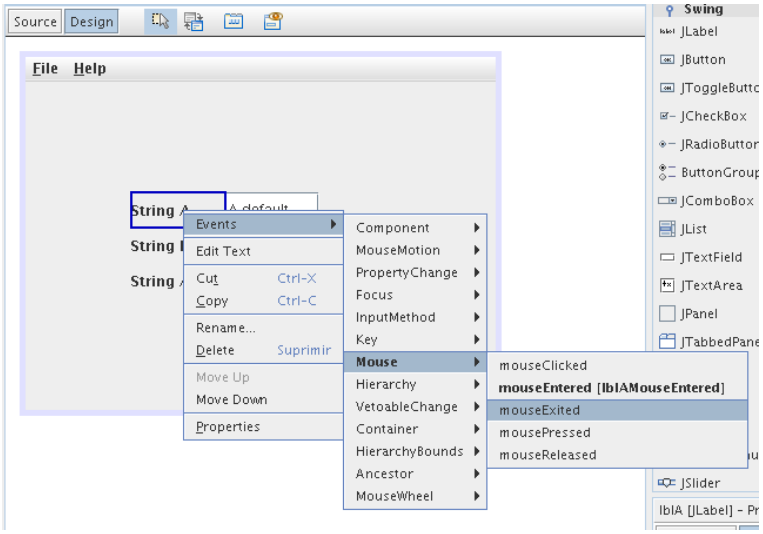
1. En el inspector de componentes, pinchar con el botón derecho en el nodo `lblA`.

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);
```

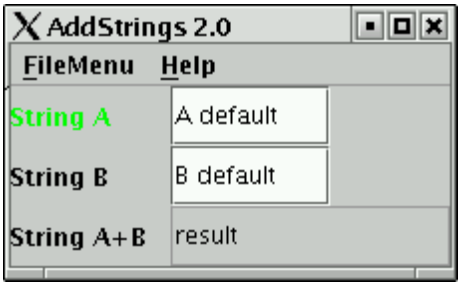
4. Ahora en el editor de forms, pincha con el botón derecho del ratón en la etiqueta `lblA` (con texto **String A**) y selecciona **Events** → **Mouse** → **mouseExited** en el menú contextual.



5. 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.



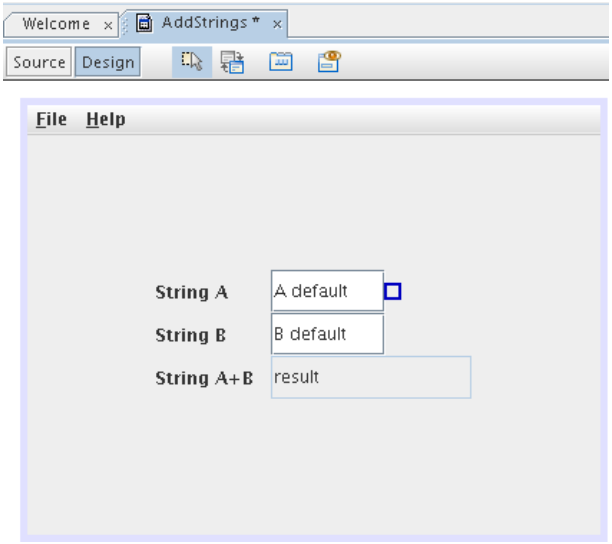
11. Contenedores dentro de contenedores

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.



   Usaremos el optimizador del *GridBagLayout* para ajustar el espacio que ocupa este `JPanel`.

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.

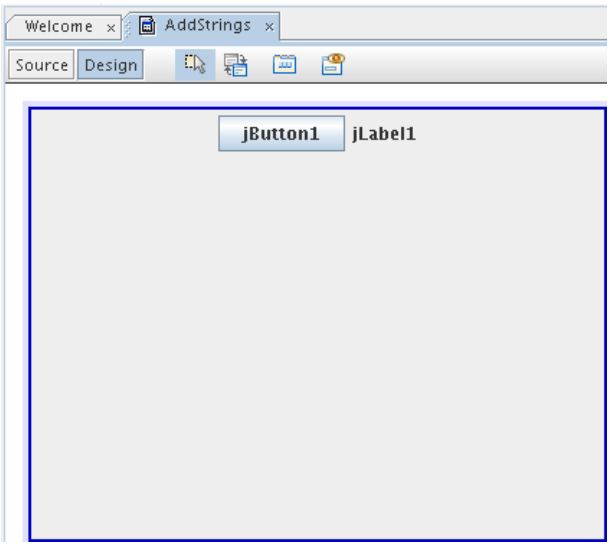
4. Cierra el optimizador pulsando en el botón **Close**.

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.

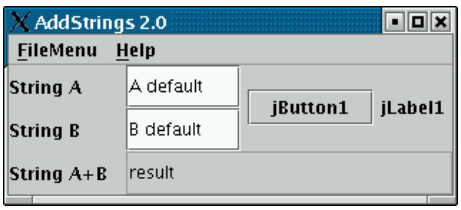
1. Selecciona el `JPanel` en el inspector de componentes, lo que hará que aparezca de nuevo el cuadrado azul.

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 en el editor de forms, y el `JFrame` desaparezca.

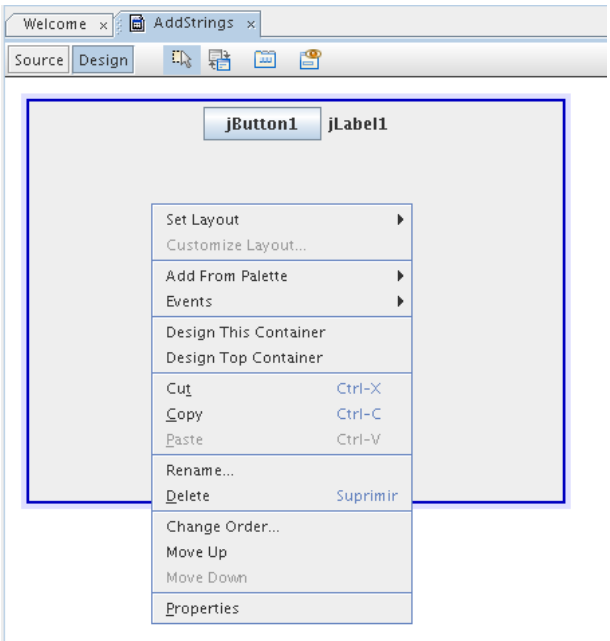
3. Inserta un `JButton` y un `JLabel` desde la paleta de componentes en el `JPanel`.



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`.



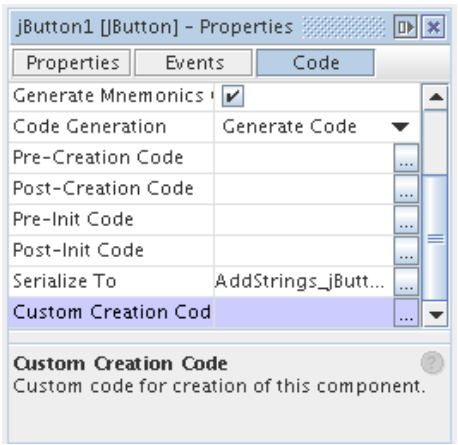
12. Modificación de propiedades de generación de código

Como hemos visto en secciones anteriores las zonas de código sombreadas en azul (*guarded code*) son generadas automáticamente por Netbeans y no pueden modificarse con el editor de código fuente. Podriamos modificar este código indirectamente con un editor de código externo, pero no es recomendable, ya que Netbeans regenera automáticamente el *código reservado* (*guarded code*) cada vez que se abre de nuevo el objeto fuente, con lo que se perderían los cambios realizados fuera de Netbeans. El código reservado puede identificarse con un editor de texto externo a Netbeans, buscando comentarios que comiencen con `//GEN-`.

Pero a veces necesitamos modificar código reservado por alguna razón. Por ejemplo, queremos pasar uno o varios parámetros de instanciación al constructor de la clase de un componente del GUI (el IDE utiliza normalmente el *constructor por defecto*) o bien queremos que una determinada propiedad de un componente se defina con el valor que devuelve un determinado método. La forma recomendada de modificar código reservado es a través de las utilidades que proporciona Netbeans para ello.

- Modificación de código generado para **componentes del Form**.

1. En el inspector de componentes, selecciona el componente al que queremos cambiar su código de inicialización. Por ejemplo, selecciona el componente `jButton1`.
2. Pincha el botón **Code** que hay en la parte derecha de la ventana de propiedades.



3. Selecciona la propiedad que deseas editar, e introduce el valor deseado.
   Por ejemplo, para incluir un argumento `String` en el constructor que crea el componente `jButton1`, pincharemos en el botón etiquetado con `...` de la propiedad *Custom Creation Code*. Aparecerá el siguiente diálogo:

Incluye la siguiente línea de código y pulsa **Ok**.

```
new javax.swing.JButton("Hola");
```

Elimina ahora el texto en la propiedad *text* de este componente `jButton1` en la ventana de propieades, para que así el `JButton` aparezca con el texto que se ha incluido en el constructor.

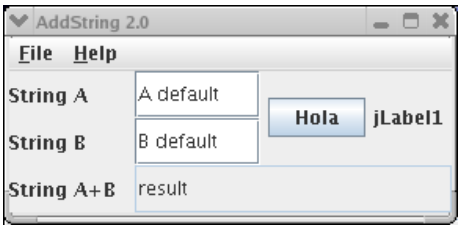
Visualiza el código fuente, y comprueba como ahora el componente `jButton1` se crea con:

```
jButton1 = new javax.swing.JButton("Hola");
```

```
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        lblA = new javax.swing.JLabel();
        lblB = new javax.swing.JLabel();
        lblSum = new javax.swing.JLabel();
        tfA = new javax.swing.JTextField();
        tfB = new javax.swing.JTextField();
        tfSum = new javax.swing.JTextField();
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton("Hola");
        jLabel1 = new javax.swing.JLabel();
```

Ejecuta el programa y podrás ver como `jButton1` está etiquetado con *Hola*.



- Modificación del código generado para una **propiedad de un componente**.

1. Selecciona el componente en el inspector.
2. Despliega la ventana de propiedades.
3. Selecciona la propiedad a la que queremos modificar el código de inicialización.
4. Pincha el botón `...` para abrir el diálogo *Editor de Propiedades* (*Editor Property*).
5. Selecciona *Form Connection* en el combo box de *Select Mode*.

6. En el editor de propiedades, selecciona el tipo de inicialización de código que se quiere utilizar (Value, Bean, Property, Method Call o User Code).

13. 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 `paintComponent(Graphics g)`.

1. Crear un nuevo proyecto. Nombra *GuiDemoAdvanced2* al nuevo proyecto. Asegurate que no están seleccionadas las opciones *Set as Main Project* y la *Create Main Class*.

2. Copia el paquete completo `guidemoadvanced` desde el proyecto *GuiDemoAdvanced* hasta este nuevo proyecto *GuiDemoAdvanced2*, haciendo uso de **Copy** y **Paste**.

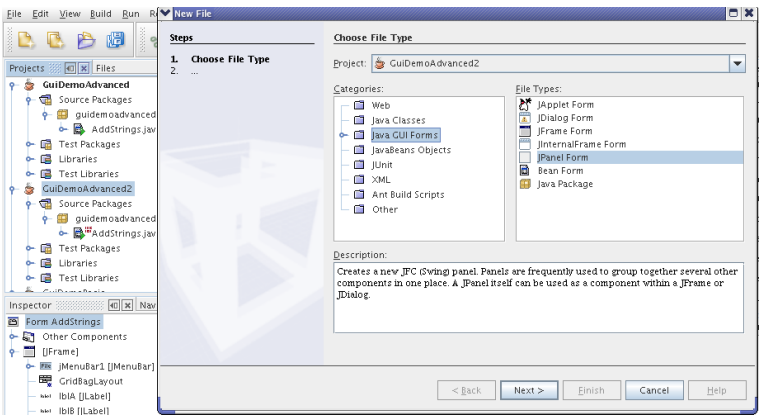
3. Haz doble click en el nuevo nodo `AddStrings` del proyecto *GuiDemoAdvanced2* para abrirlo en las ventanas Source Editor y Form Editor.

4. En el inspector de componentes define la propiedad *título* (title) con *AddStrings 3.0*.

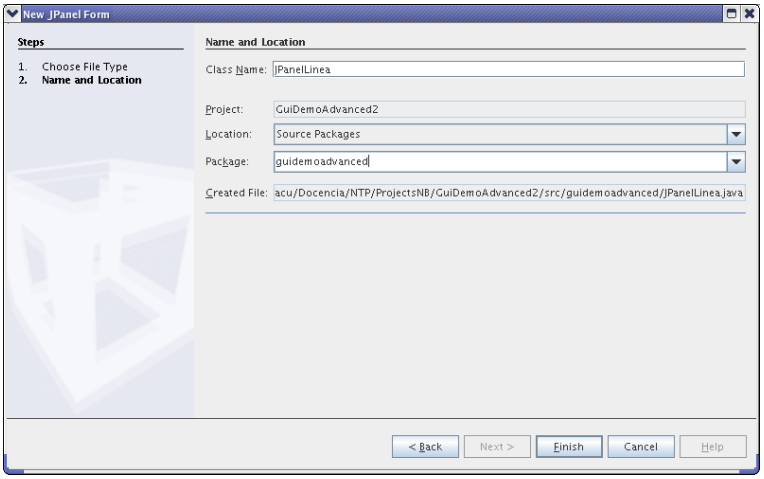
A continuación vamos a dar los pasos necesarios para incluir el panel de dibujo de líneas en nuestro nuevo proyecto.

1. En el inspector de componentes selecciona el nodo `jPanel1` y elimínalo mediante el menú contextual.

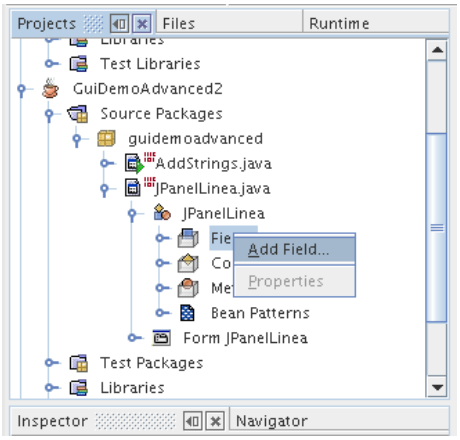
2. Selecciona el nodo *GuiDemoAdvanced2* en la ventana de proyectos, y crea un nuevo **JPanel**: **Menú File** → **New File** → **Java GUI Forms** → **JPanel Form**.



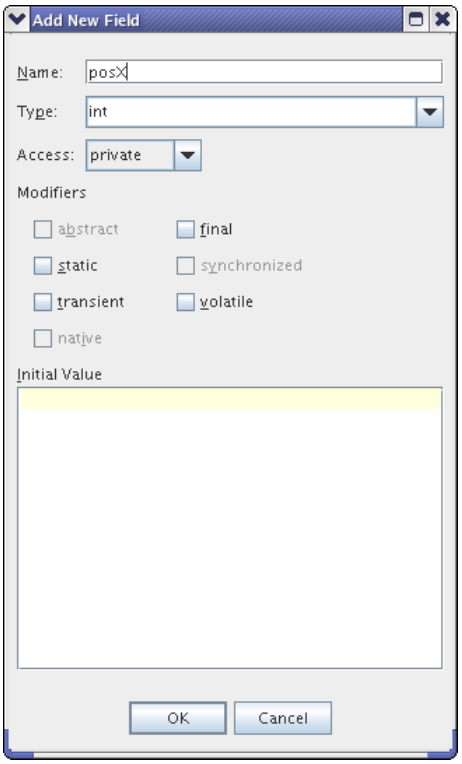
3. Nombraremos *JPanelLinea* a esta nueva clase y la incluiremos en el paquete *guidemoadvanced*.



4. Añade dos campos `posX` y `posY` 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.



Si usamos este último método aparecerá el siguiente diálogo:

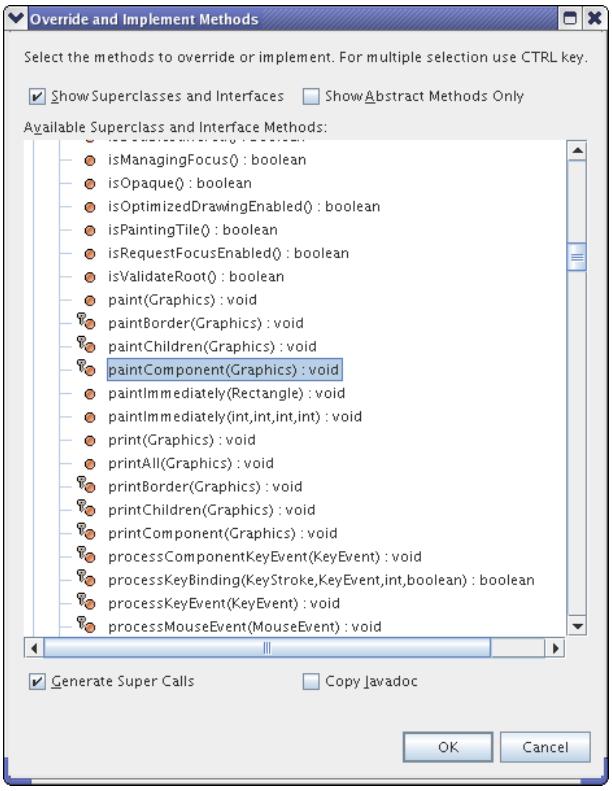


5. Usando el editor del código fuente incluye las sentencias de inicialización de `posX` e `posY` en el constructor de `JPanelLinea`, después de la llamada a `initComponents();`.

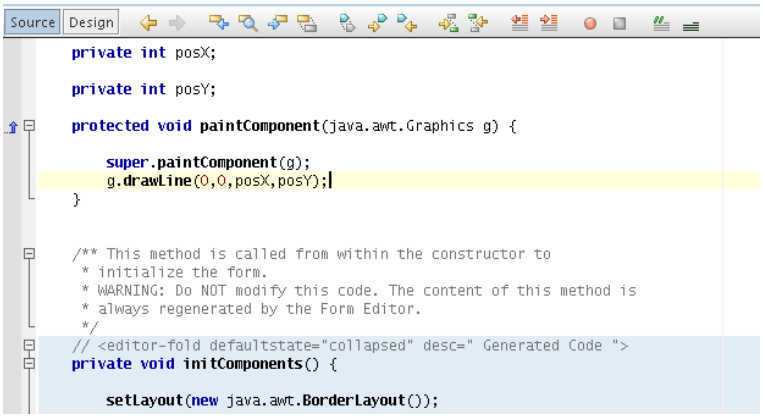
```
posX=50;
posY=50;
```

6. Sobreescribe el método `paintComponent(Graphics)` en la clase `JPanelLinea` (método heredado de `JComponents`). Para ello selecciona el nodo **class JPanelLinea** en la ventana de proyectos, pincha con el botón derecho del ratón sobre él y selecciona **Tools → Override Methods...**.

7. En el diálogo que aparece selecciona *Show Superclasses and interfaces* y selecciona el método **void paintComponent(Graphics)** de la clase **JComponent** y pulsa en el botón **Ok**.

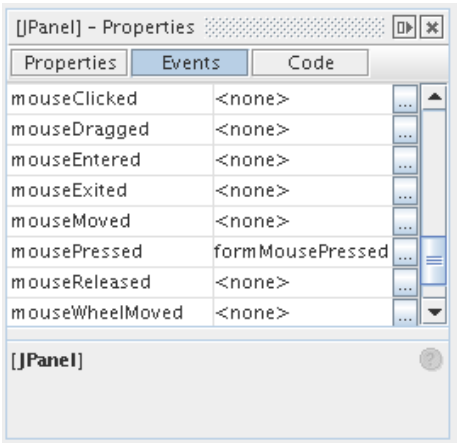


8. Añade la sentencia `g.drawLine(0,0,posX,posY);` al final del método `paintComponent()`.



9. Controlar el evento `mousePressed`. Puede hacerse seleccionando el nodo **JPanel** del form `JPanelLinea` en el inspector de componentes, seleccionando luego la solapa **Events** de la ventana de propiedades del `JPanel` y pinchando finalmente en el evento `mousePressed`. Este paso hará que se añada a la clase `JPanelLinea` el método:

   `formMousePressed(java.awt.event.MouseEvent evt)`



10. Añade al método `formMousePressed()` lo siguiente:

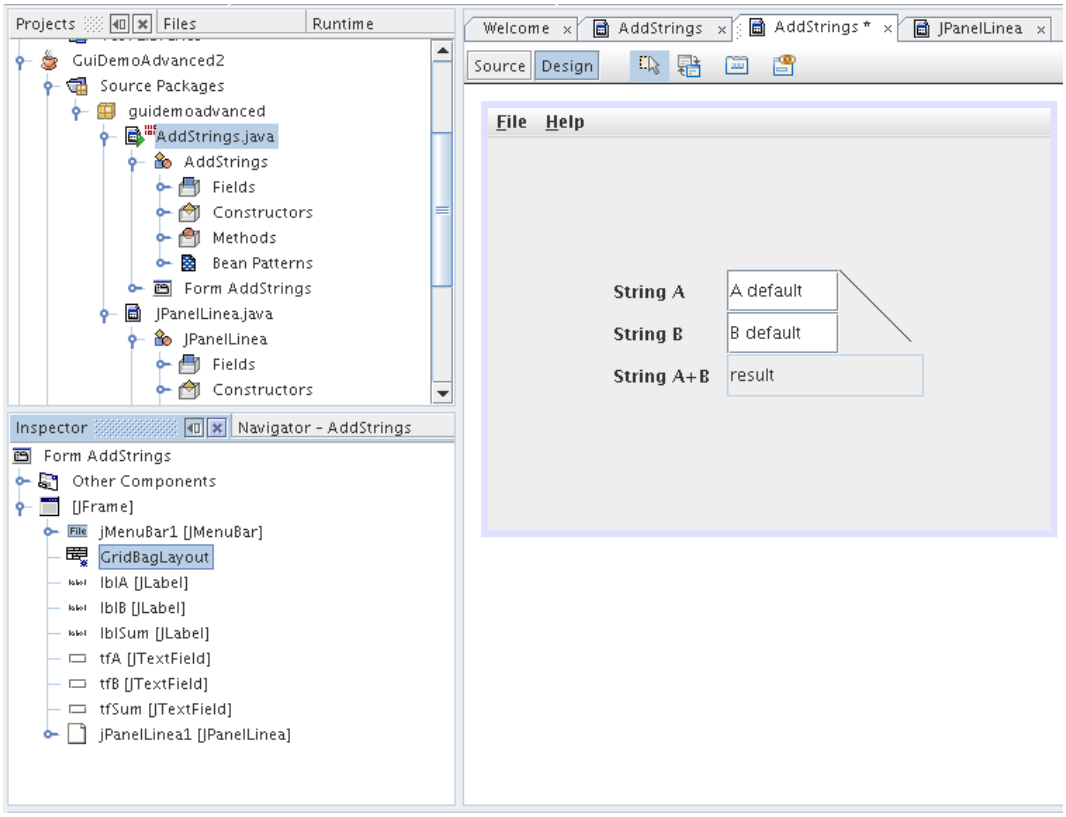
```
posX=evt.getX();
posY=evt.getY();
repaint();
```

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

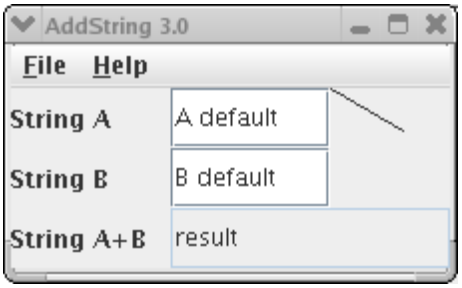
12. Añade el panel `JPanelLinea` como componente del `JFrame` de *AddStrings*. Para ello realiza los siguientes pasos:

    - Abre nodos del form *AddStrings* en el inspector de componentes hasta que sea visible el nodo `[JFrame]`.
    - Copia `JPanelLinea` al portapapeles (con el botón derecho del ratón pincha el nodo `JPanelLinea` de la ventana de proyectos 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` en el inspector de componentes, 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.



14. Localización de los programas ya terminados

En este guión se han desarrollado tres programas. Podéis encontrar los proyectos de estos tres programas en la dirección http://decsai.ugr.es/~acu/NTP/archivos/programasguion2java.tgz.