

# DESARROLLO DE INTERFACES



 JUNTA DE  
EXTREMADURA



@vanza  


## Tema 1.

# Desarrollo de Interfaces

## 1.- Elaboración de interfaces de usuario.



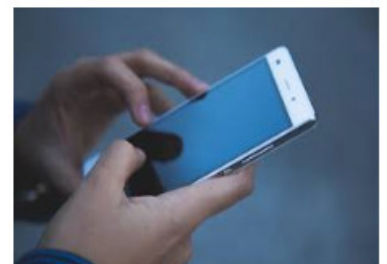
### Caso práctico

Lo primero que el equipo tiene que buscar es una definición del objeto de su investigación y los tipos que hay, para ello se hacen las siguientes preguntas ¿cómo puede una persona interactuar con una aplicación informática? o ¿de qué elementos dispone para hacerlo? Ada recuerda sus comienzos en este mundo cuando tenía que vérselas con el sistema Unix, en el que tenía que escribir órdenes para realizar tareas, **María** y **Juan** están más acostumbrados a usar entornos gráficos y el ratón. Mientras discuten todo esto **Ana** contesta a su móvil de última generación simplemente deslizando el dedo sobre la pantalla.



**La interfaz de usuario es el elemento de una aplicación informática que permite al usuario comunicarse con ella.**

El desarrollo de aplicaciones hoy día no puede ser entendido sin dedicar un porcentaje bastante importante de tiempo a planificar, analizar, diseñar, implementar y probar sus interfaces de usuario ya que son el medio fundamental por el cual el usuario puede comunicarse con la aplicación y realizar las operaciones para las que ha sido diseñada. Existen diferentes tipos de interfaces de usuario:



- ❑ **Textuales.** La comunicación se produce por medio de la inserción de órdenes escritas en un [intérprete de órdenes](#).
- ❑ **Gráficas.** La [interfaz](#) consta de un conjunto de elementos visuales como [iconos](#) o [menús](#) con los que se interacciona, normalmente, mediante un [elemento apuntador](#) (el ratón por ejemplo). Popularizaron el mundo de la informática para usuarios noveles.
- ❑ **Táctiles.** La comunicación se produce a través de un dispositivo táctil, generalmente una pantalla que puede reaccionar ante la presión táctil de elementos apuntadores, como por ejemplo los dedos. Se usan habitualmente en dispositivos móviles, terminales de puntos de venta y para el diseño de gráficos por ordenador. Existen dos tipos, resistivas y capacitivas (las más utilizadas y presentes en todos los smartphones actuales).

A lo largo de esta unidad nos centraremos en la creación de interfaces gráficas. Veremos que una interfaz gráfica está formada por un conjunto de ventanas, llamadas formularios, y que dentro de ellos podemos colocar diferentes elementos visuales, que se denominan controles o componentes con los que al interactuar daremos órdenes o recibiremos información..



### Para saber más

En el siguiente enlace encontrarás una web sobre elementos que pueden aparecer en interfaces gráficas de usuario y maneras de diseñarlas:

[Interfaces gráficas de usuario.](#)

También te recomendamos los siguientes enlaces:

- 10 Principios de usabilidad para diseño de interfaces de usuario: [Enlace](#)
- Diseño de apps: usabilidad y experiencia de usuario: [Enlace](#)



### Autoevaluación

¿En que orden crees que surgieron los diferentes tipos de interfaces?

- ☐ Táctiles, gráficas, textuales.
- ☐ Gráficas, textuales, táctiles.
- ☐ Textuales, táctiles, gráficas.
- ☐ Textuales, gráficas, táctiles.

## 2.- Componentes.



### Caso práctico

Parece claro que serán interfaces gráficas las que desarrolle BK Programación. Este tipo de interfaces se componen de una serie de elementos gráficos que permiten al usuario enviar sus peticiones a una aplicación y recibir respuestas. Pero ¿cuáles son sus bases?, ¿dónde podemos encontrar estos componentes? y ¿cómo se usan para construir una interfaz gráfica?



Una interfaz gráfica se comporta como un todo para proporcionar un servicio al usuario, permitiendo que éste realice peticiones y mostrando el resultado de las acciones realizadas. Sin embargo, una interfaz se compone de una serie de elementos gráficos atómicos que tienen sus propias características y funciones, de modo que la combinación de todos estos elementos forman la interfaz de una aplicación. A estos elementos se les llama componentes o controles.



Algunos de los componente más típicos son:

- ❑ **Etiquetas:** Permiten situar un texto en la interfaz. No son interactivos y puede utilizarse para escribir texto en varias líneas.
- ❑ **Campos de texto:** cuadros de una sola línea en los que podemos escribir algún dato.
- ❑ **Áreas de texto:** cuadros de varias líneas en los que podemos escribir párrafos.
- ❑ **Botones:** áreas rectangulares que se pueden pulsar para llevar a cabo alguna acción.
- ❑ **Botones de radio:** botones circulares que se presentan agrupados para realizar una selección de un único elemento entre ellos. Se usan para preguntar un dato dentro de un conjunto. El botón marcado se representa mediante un círculo.
- ❑ **Cuadros de verificación:** botones en forma de rectángulo. Se usan para marcar una opción. Cuando está marcada aparece con un tic dentro de ella. Se pueden seleccionar varios en un conjunto.
- ❑ **Imágenes:** se usan para añadir información gráfica a la interfaz.
- ❑ **Password:** es un cuadro de texto en el que los caracteres aparecen ocultos. Se usa para escribir contraseñas que no deben ser vistas por otros usuarios.
- ❑ **Listas:** conjunto de datos que se presentan en un cuadro entre los que es posible elegir uno o varios.
- ❑ **Listas desplegables:** combinación de cuadro de texto y lista, permites escribir un dato o seleccionarlo de la lista que aparece oculta y puede ser desplegada.

## 2.1.- Bibliotecas de componentes.



### Caso práctico

Ya sabemos que los componentes son elementos básicos con una función concreta que al agruparse de una u otra forma generan una interfaz con un cometido más amplio, pero ¿cómo podemos añadirlos a la aplicación que estamos desarrollando?



Los componentes que pueden formar parte de una interfaz gráfica se suelen presentar agrupados en bibliotecas con la posibilidad de que el usuario pueda generar sus propios componentes y añadirlos o crear sus propias bibliotecas. Se componen de un conjunto de clases que se pueden incluir en proyectos software para crear interfaces gráficas. El uso de unas bibliotecas u otras depende de varios factores, uno de los más importantes, por supuesto, es el lenguaje de programación o el entorno de desarrollo que se vaya a usar. Dependiendo de esto podemos destacar:

#### JAVA Foundation Classes (JFC):

Las JFC incluyen las bibliotecas para crear las interfaces gráficas de las aplicaciones Java y applets de Java.

- ❑ **AWT**: Primera biblioteca de Java para la creación de interfaces gráficas. Es común a todas las plataformas pero cada una tiene sus propios componentes, escritos en **código nativo** para ellos. Prácticamente en desuso, exclusivamente en proyectos antiguos en mantenimiento.
- ❑ **Swing**: Surgida con posterioridad, sus componentes son totalmente **multiplataforma** porque no tienen nada de código nativo. Su precursor es AWT, de hecho muchos componentes Swing derivan de AWT, basta con añadir una j al principio del nombre AWT para tener el nombre Swing, por ejemplo el elemento Button de AWT tiene su correspondencia Swing en Jbutton aunque se han añadido gran cantidad de componentes nuevos. Es el estándar actual para el desarrollo de interfaces gráficas en Java.
- ❑ Además existen bibliotecas para desarrollo gráfico en 2D y 3D, que permiten realizar tareas de arrastrar y soltar (drag and drop).

#### Bibliotecas **MSDN** de Microsoft (C#, ASP, ...):

- ❑ **.NET framework**: hace alusión tanto al componente integral que permite la compilación y ejecución de aplicaciones y webs como a la propia biblioteca de componentes que permite su creación. Para el

desarrollo de interfaces gráficas la biblioteca incluye ADO.NET, ASP.NET, formularios Windows Forms y la WPF (👉 [Windows Presentation Foundation](#)).



### Para saber más

Es interesante darse una vuelta por Wikipedia y profundizar algo más en los conceptos más importantes. Sin duda estos conceptos son la base sobre la que asentar el resto de conocimientos:

- biblioteca: [https://es.wikipedia.org/wiki/Biblioteca\\_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica))
- AWT : [https://es.wikipedia.org/wiki/Abstract\\_Window\\_Toolkit](https://es.wikipedia.org/wiki/Abstract_Window_Toolkit)
- código nativo: [https://es.wikipedia.org/wiki/C%C3%B3digo\\_nativo](https://es.wikipedia.org/wiki/C%C3%B3digo_nativo)
- Swing: [https://es.wikipedia.org/wiki/Swing\\_\(biblioteca\\_gr%C3%A1fica\)](https://es.wikipedia.org/wiki/Swing_(biblioteca_gr%C3%A1fica))
- Multiplataforma: <https://es.wikipedia.org/wiki/Multiplataforma>
- .NET framework: [https://es.wikipedia.org/wiki/Microsoft\\_.NET](https://es.wikipedia.org/wiki/Microsoft_.NET)
- Windows presentation foundation: [https://es.wikipedia.org/wiki/Windows\\_Presentation\\_Foundation](https://es.wikipedia.org/wiki/Windows_Presentation_Foundation)

### Bibliotecas basadas en XML:

- ☑ También existen bibliotecas implementadas en lenguajes intermedios basados en tecnologías XML (que se verán en la siguiente unidad temática). Normalmente disponen de mecanismos para elaborar las interfaces y traducirlas a diferentes lenguajes de programación, para después ser integradas en la aplicación final. Cabe destacar las librerías QT, desarrollada como un proyecto de software libre en el que participan importantes empresas en el sector de telefonía, como Nokia.

### 3.- Herramientas para la elaboración de interfaces.



#### Caso práctico

El equipo lleva algún tiempo analizando las características de las interfaces gráficas, saben que se componen de un conjunto de elementos básicos, los controles que se suelen almacenar en bibliotecas. Ana está algo preocupada ya que le parece que no va a ser capaz de aprenderse tantos nombres nuevos, y menos para añadirlos a mano al código de la aplicación, por lo que se pregunta si no existirá una manera más sencilla de hacer todo esto.



Con las bibliotecas tienes la base para crear tus aplicaciones, están compuestas de código que puedes escribir sin más, aunque lo habitual es valerse de algún entorno integrado de desarrollo de software que facilite esta labor mediante algún sistema de ventanas, en el que se incluyen diferentes regiones rectangulares, llamadas "ventanas" cuya parte central es un conjunto de herramientas (toolkit). A este tipo de herramientas se le llama IDE (Integrated Development Environment) y provee de procedimientos que permiten incluir los componentes de las bibliotecas de componentes gráficos y añadirlos de un modo sencillo e intuitivo en la aplicación que estés desarrollando. También sirven como medio de entrada de las acciones del usuario.

A continuación se listan varios de los IDE más utilizados en la actualidad:

- ❑ **Microsoft Visual Studio:** Es un entorno para el desarrollo de aplicaciones en entornos de escritorio, web y dispositivos móviles con la biblioteca .NET framework de Microsoft. Permite el uso de diferentes lenguajes de programación como C++, C# o ASP. En la actualidad se pueden crear aplicaciones para Windows 7, aplicaciones web y RIA (Rich Internet Applications).
- ❑ **NetBeans:** IDE distribuido por Oracle bajo licencia GNU GPL. Está desarrollado en Java, aunque permite crear aplicaciones en diferentes lenguajes, Java, C++, PHP, Ajax, Python y otras.
- ❑ **Eclipse:** IDE creado inicialmente por IBM ahora es desarrollado por la fundación Eclipse. Se distribuye bajo la Licencia Pública Eclipse, que, aunque libre, es incompatible con la licencia GNU GPL. Tiene como característica más destacable su ligereza ya que se basa en módulos, partiendo de una base muy sencilla con un editor de texto con resaltado de sintaxis, compilador, un módulo de pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc. y refactorización. Si se precisan funcionalidades más allá, éstas se incluirán como módulos aparte que van completando el IDE.
- ❑ **JDeveloper:** Es un entorno de desarrollo integrado desarrollado por Oracle Corporation para los lenguajes Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML y otros. Las primeras

versiones de 1998 estaban basadas en el entorno JBuilder de Borland, pero desde la versión 9i de 2001 está basado en Java, no estando ya relacionado con el código anterior de JBuilder. Tras la adquisición de Sun Microsystem por parte de Oracle, está cada vez en más desuso, ya que NetBeans ofrece mayores opciones.

- ▣ **Android Studio:** Entorno de desarrollo oficial para la plataforma Android. Diseñado específicamente para el desarrollo de Android.

Además de los IDE, existen editores de texto que sin ser un IDE, siguen siendo herramientas a tener muy en cuenta como Sublime Text, Visual Studio Code, Atom... Con la ayuda de complementos o plugins estos editores de texto llegan a hacer las mismas funciones que los propios IDE.



### Autoevaluación

¿En qué componente se pueden seleccionar varias opciones a la vez?

- ☐ Botones de radio.
- ☐ Texto.
- ☐ Botón.
- ☐ Cuadros de verificación.



## 3.1.- NetBeans.



### Caso práctico

Por fin el equipo ha tomado una decisión en cuanto a la tecnología, el lenguaje de programación y la herramienta. **Ana** está mucho más tranquila pues parece que no necesitará conocer tantos nombres de memoria, ha estado investigando NetBeans por su cuenta y es muy intuitivo, para la creación de las interfaces, para la edición de código ya que cuenta con abundantes ayudas gráficas, y en la edición del texto. Además empieza creando él mismo el esqueleto de la aplicación, e incluso una de las ventanas más comúnmente utilizada: el cuadro de diálogo Acerca de...



Para desarrollar los contenidos de esta unidad te proponemos NetBeans como entorno de desarrollo integrado utilizando la biblioteca swing para la generación de interfaces. Se ha seleccionado este entorno por varios motivos, en primer lugar permite crear aplicaciones tanto de escritorio, como web como para dispositivos móviles y además se distribuye bajo licencia GNU GPL. Es multiplataforma e incluye varios lenguajes de programación.

Una de sus principales ventajas es que resuelve por sí mismo el tema de la colocación de los componentes en una interfaz gráfica, aspecto de cierta complejidad a la hora de programar, de forma que el desarrollador o desarrolladora sólo tiene que colocar los controles usando el ratón y el IDE se encarga de programarlo.

También permite la inclusión de componentes creados por otros desarrolladores que completan la paleta swing/AWT.



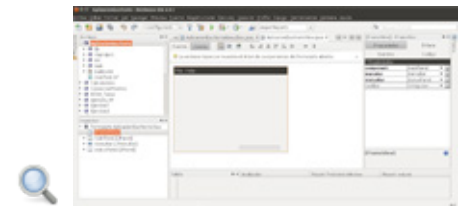
### Para saber más

Seguramente ya has instalado y probado este software. Si no lo has hecho todavía, desde el siguiente enlace puedes descargar la última versión de NetBeans junto con el compilador Java (JDK).

[Descarga e instalación de la última versión de NetBeans + JDK](#)

Si sólo quieres instalar NetBeans, te lo puedes descargar desde su propia [web oficial](#) (elegir la versión completa para no tener problemas a lo largo del curso).

Una vez que tengas el entorno instalado y funcionando debes comenzar por crear un proyecto que pueda utilizar las clases incluidas en la librería swing, para ello sólo debes seleccionar como tipo de proyecto **Java Desktop Application** dentro de la categoría Java. A continuación rellena los datos principales del proyecto, como su nombre, la localización de los archivos en disco y si es una aplicación básica o de base de datos (de momento elegiremos la primera opción). Cuando lo hayas hecho te aparecerá una pantalla como la de la imagen.



A continuación tienes una presentación con los principales elementos de esta interfaz para que te vayas familiarizando con ellos.



## 4.- Contenedores.



### Caso práctico



El siguiente paso es diseñar la interfaz en si misma y cuando sepas qué componentes vas a necesitar, donde colocarlos y con qué tamaño tendrás que crearla usando el editor del interfaces del IDE. Pero antes de empezar a añadir elementos a la interfaz necesitarás un sitio donde ponerlos por lo que lo primero que tendrás que hacer es seleccionar un contenedor y luego empezar a diseñar.





Un **formulario** es una ventana que dispone de tres botones para minimizarse, maximizarse o cerrarse, una barra de título y está delimitado por unos bordes. Es la base para crear una aplicación de escritorio. Sobre él se añadirán controles o componentes, sencillos como botones o cajas de texto o más complejos, como barras de menú o rejillas de datos, que le dan funcionalidad.



Una aplicación de escritorio de NetBeans se compone de una serie de formularios. Para crear un formulario tendrás que usar un **contenedor** Java, que es un componente que permite incluir a su vez otros componentes incluidos otros contenedores, que se usarán para distribuir los controles. Por eso se dice que los contenedores forman una **estructura jerárquica**.

Un formulario está formado por un contenedor especial que se llama  **contenedor de nivel superior**. Este tipo incluye un  **panel de contenido** (contentpane) que permite añadir otros componentes, incluidos otros contenedores que se utilicen facilitar la distribución de elementos.

Como contenedor de nivel superior de un formulario, puedes elegir entre una ventana (JFrame), un diálogo (JDialog) o un  [applet](#) (JApplet), según la necesidad. Todos estos componentes derivan, en la jerarquía de clases de java, de Windows, que representa una ventana típica.

- ❑ Ventana (JFrame): es un formulario con título, los botones para maximizar, minimizar o cerrar y borde. Aparece un icono por defecto en forma de taza de café que puedes modificar, y puede contener una barra de menú.
- ❑ Diálogo (JDialog): formularios que se suelen usar para solicitar información al usuario. Su principal característica es que pueden ser  [modales](#) o no. Una ventana modal recibe todas las entradas de usuario e impide que se active otra ventana.
- ❑ Applet (JApplet): ventana que ejecuta una aplicación Java en el contexto de una página web.

**En una aplicación de escritorio se suele crear una ventana principal que sea de tipo JFrame y, si necesitamos ventanas secundarias utilizaremos otras ventanas o diálogos.**



### Debes conocer

Una vez que conoces qué es un contenedor y los distintos tipos que hay, debes saber también cómo incluirlos en tu aplicación usando NetBeans y cómo influye eso en el código. El siguiente enlace abre un documento en el que tienes algunos ejemplos y un ejercicio que debes completar para continuar con estos contenidos.



[Añadir contenedores a una aplicación Java.](#) (0.15 MB)



### Para saber más

Si quieres saber un poco más acerca de los contenedores de primer nivel puedes consultar este apartado de la página web de Oracle.

[Contenedores de primer nivel.](#)

A su vez, en la siguiente presentación se muestra un poco más a fondo en qué consisten las componentes y los contenedores (con las flechas del teclado se puede avanzar y retroceder en la presentación):

[Swing: Contenedores y componentes](#)

## 4.1.- Contenedores secundarios.



### Caso práctico

Ana está inmersa en la creación de una interfaz, a pesar de que NetBeans le facilita mucho el proceso con las ayudas visuales, no puede terminar su interfaz puesto que el formulario que está construyendo es algo más complejo ya que necesitaría añadir algún tipo de panel con pestañas para mostrar diferentes controles en función de la selección del usuario, y esto no lo puede conseguir sólo con una ventana y los controles más sencillos. Decide investigar si puede usar otro tipo de contenedores que le resuelvan su problema.



También se interpretan como diálogos los siguientes componentes:

- ❑ **Panel de opciones (JOptionPane):** genera ventanas con botones para responder cuestiones con respuestas del tipo si-no, aceptar-cancelar, aceptar, etc.
- ❑ **Selector de archivos (JFileChooser):** permite seleccionar un archivo del sistema de archivos del equipo donde se ejecuta la aplicación.
- ❑ **Selector de colores (JColorChooser):** permite seleccionar entre un conjunto de colores y devolverlo usando el código adecuado.

Puedes usar otro tipo de contenedores para distribuir el resto de los controles que se incluyen en la ventana principales, entre los más habituales tienes:

- ❑ **Paneles (JPanel):** representa un contenedor intermedio, cuya función principal es la de colocar controles.
- ❑ **Barra de menús (JMenu):** permite la creación de menús complejos de opciones.
- ❑ **Barra de herramientas (JToolBar):** se utiliza para contener iconos de acceso a las opciones de la aplicación.
- ❑ **Pestañas (JTabbedPane):** tipo particular de panel que permite la distribución de elementos en pestañas o tarjetas.
- ❑ **Paneles deslizables (JScrollPane):** tipo especial de panel que permite desplazar sus contenidos de manera automática.
- ❑ **Ventanas internas (JInternalFrame):** ventanas hijas que no pueden rebasar los límites de la ventana padre donde se han creado. Se usan en aplicaciones que tienes varios documentos abiertos simultáneamente.

- ❑ **Paneles divididos (JSplitPane):** permite visualizar dos componentes, uno a cada lado, asignando espacio dinámicamente a cada uno.

## 5.- Componentes de la interfaz.



### Caso práctico

**María** ha echado una mano a **Ana** y al final han conseguido elaborar la base de la interfaz con pestañas que necesitaba. Ahora sólo tiene que añadir el resto de controles que componen su interfaz.

Cuando **María** va colocando componentes en el formulario aparecen las formas correctas, pero con los nombres y tamaño por defecto. Tiene especial cuidado con la posición y el tamaño y procura que la composición quede armónica, aunque le gustaría poder tener un poco más de control sobre estos aspectos.



Los componentes o controles gráficos de un formulario son elementos gráficos que se anidan en los contenedores para formar aplicaciones. Se utiliza para mostrar información, como etiquetas o imágenes, listas (👉 **componentes pasivos**) o árboles, pero, sobre todo, para recabar información del usuario, como cuadros de texto, botones, o listas de selección (👉 **componentes activos**).

Un componente se reconoce por su clase, que define su aspecto y funcionalidad y por su **nombre** que lo identifica dentro de la aplicación. Puesto que es un objeto, según la clase a la que pertenezca tendrá, una serie de propiedades que podremos modificar para adaptar el componente, por ejemplo, el texto mostrado, o el color.

La **colocación de componentes** en el formulario se rige por unas reglas, denominadas 👉 **Layout**, que establecen el orden y la posición en la que deben ser mostrados, pudiendo ser en torno a los límites del formulario (norte, sur, este y oeste), en forma de rejilla, o fluidos, uno tras otro, en una o varias filas.

Cuando un componente es susceptible de interactuar con el usuario se gestiona mediante lo que se conoce como **manejo de eventos**, una acción sobre el componente que debe provocar una respuesta se conoce como 👉 **evento**, la gestión de la respuesta se realiza a través de los 👉 **manejadores de eventos**, que son unas funciones específicas que se asocian a un elemento del componente denominado 👉 **escuchador**, en las que se programa la acción a realizar.

Veamos todas estas cosas con un poco más de detenimiento.



## Para saber más

Si quieres ampliar un poco más tus conocimientos sobre swing aquí tienes el enlace a la página principal de la documentación de esta biblioteca en el sitio de Oracle.

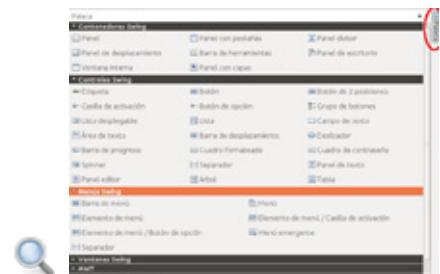
[Swing en Oracle.](#)



## 5.1.- Añadir y eliminar componentes de la interfaz.

Los componentes se pueden añadir desde la paleta, que, si recordamos suele anclarse a la derecha de la interfaz del IDE NetBeans, y mientras no se use queda replegada (en la imagen puedes ver rodeado de rojo el botón que la hace aparecer).

A la derecha tienes la lista de controles para añadir a una interfaz en NetBeans. Están organizados en las siguientes categorías:



- ❑ **Contenedores swing**: son secundarios en la jerarquía de contenedores y se usan para distribuir y organizar el resto de controles.
- ❑ **Controles swing**: básicos para crear una interfaz útil para comunicarse con el usuario y mostrar o solicitar información.
- ❑ **Menús swing**: incluyen los controles necesarios para crear menús de aplicación complejos, con varias bloques, elementos activos e inactivos, etc. y menús contextuales (Popup Menu)
- ❑ **Ventanas swing**: permiten añadir a la aplicación ventanas (JFrame), diálogos (JDialog), selectores de ficheros y colores (JFileChooser y JColorChooser) y paneles de opciones (JOptionPane) para crear diálogos que se contestan con Sí/No.

La **adición** de componentes a la interfaz se realiza seleccionando el control en la paleta y pinchando sobre la interfaz que se está construyendo. El control aparece en la interfaz con su aspecto por defecto que puedes modificar. Si arrastras el control se moverá sobre la superficie de su contenedor y si haces clic sobre una esquina y desplazas el ratón lo cambiarás de tamaño.

Al colocar un control sobre un formulario aparecen unas guías que te permiten colocarlo con más facilidad, esto será así mientras tengas activa la opción **diseño libre**, lo puedes comprobar en el inspector haciendo clic con el botón secundario en el nodo raíz de la interfaz y seleccionando activar gestor de distribución. Si **mueves** un control estas guías te permitirán relacionarlo con otros componentes para que lo puedas alinear mejor.

Conforme vas colocando controles en el panel del formulario éstos aparecen reflejados en el **Inspector**, de forma que los seleccionas tanto haciendo clic sobre ellos como sobre su nombre. Trabajar con el inspector facilita colocar controles dentro de contenedores porque admite operaciones de arrastrar y soltar.

Para **eliminar** un control basta con seleccionarlo (de cualquiera de las formas descritas) y pulsar la tecla **Supr.**, o bien seleccionar la opción Suprimir del **menú contextual**.



### Ejercicio resuelto

Prueba a hacer prácticas añadiendo y eliminando componentes a tu interfaz. Trabaja con los contenedores, para añadir dentro de ellos controles y observa como aparecen creando una jerarquía en el Inspector. Si los colocas en la ventana principal y ejecutas la aplicación pulsando la vista diseño podrás comprobar el funcionamiento de los controles que hayas añadido. Añade un panel con pestañas como han hecho María y Ana en su aplicación.



### Para saber más

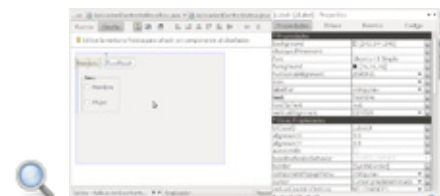
Si tienes interés en conocer un poco más acerca de los componentes gráficos de swing y verlos en acción a través de ejemplos, puedes visitar estos enlaces a la página oficial de Java de Oracle y a [jvaswing.org](http://jvaswing.org).

[Ejemplos de uso de los componentes swing.](#)

[Descripción de componentes swing.](#)

## 5.2.- Modificación de propiedades.

Una vez que has colocado un control en el formulario puedes modificar sus propiedades para adaptarlo a tu interfaz. Con el control seleccionado accedes a sus propiedades en el panel propiedades, que lógicamente será diferente para cada tipo de control.



Se suele comenzar por modificar el **nombre** del control para localizarlo con más facilidad en el código de la clase que genera el formulario. Para hacerlo, puedes sacar el menú contextual del control en el Inspector y seleccionar cambiar nombre de la variable...

Otra propiedad muy común es el **ToolTipText**. Es un pequeño recuadro de color amarillo, normalmente, en el que aparece una breve descripción de para qué sirve el control. Lo puedes establecer en la propiedad **ToolTipText**.

En la imagen puedes ver las propiedades de una etiqueta que hemos añadido al principio del formulario.



### Ejercicio resuelto

Añade al diálogo que has creado en un punto anterior los controles necesarios para que un usuario pueda rellenar una pequeña encuesta. Al final debe quedarte una interfaz semejante a la de la imagen. La interfaz que debes crear se emplea para hacer una pequeña encuesta en la que se preguntan al usuario algunos datos personales, como la profesión, edad, o número de hermanos, y otros datos relacionados con sus gustos sobre los deportes o en qué medida le gusta al usuario ir de compras, ver la televisión o ir al cine. Mientras que vas haciendo el interfaz, puedes usar el botón Diseño previo que se encuentra sobre la zona de diseño del formulario.



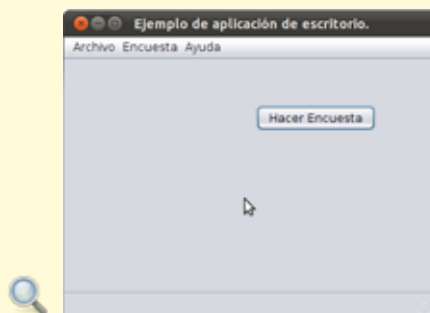
## 5.3.- Añadir funcionalidad desde NetBeans.

Creamos interfaces para permitir que el usuario pueda interactuar con la aplicación pero siempre será necesario añadir código para darles la funcionalidad para la que ha sido creadas. Cuando usamos un entorno integrado como NetBeans parte de este código se genera de forma automática facilitando en gran medida el trabajo del desarrollador o desarrolladora, pero tendrás que abrir este código y modificar algunas cosas para que el formulario realice las tareas para las que ha sido diseñado.



### Ejercicio resuelto

Recupera la aplicación a la que has añadido el diálogo en el punto anterior. Abre el formulario principal y añade un menú llamado encuesta con un submenú que se llame realizar encuesta. Al hacer clic sobre hacer encuesta se debe abrir el formulario para realizar la encuesta. Debes abrir el formulario en modo modal. Añade un botón al formulario principal, ponle el texto Hacer encuesta y modifícalo para que al hacer clic sobre el botón también se abra el formulario del mismo modo.



## 5.4.- Ubicación y alineamiento de los componentes.

Internamente la herramienta emplea el mecanismo de Java para disponer los elementos llamado Layout, distribución o diseño. Swing dispone de ocho tipos de distribuciones:

- ❑ **BoderLayout**: aloja los componentes en los límites del formulario, por lo que cuando los colocamos debemos indicar si van al norte, sur, este u oeste.
- ❑ **GridLayout**: Diseña mediante una rejilla, en la que los componentes se organizan por filas y columnas.
- ❑ **GridBagLayout**: semejante a GridLayout, pero permite a un componente que ocupe más de una celda.
- ❑ **CardLayout**: diseño por paneles. Permite la colocación de distintos componentes en momentos distintos de la ejecución.
- ❑ **BoxLayout**: diseño en caja. Coloca los componentes en una fila o columna ajustándose al espacio que haya.
- ❑ **FlowLayout**: diseña alojando los componentes de izquierda a derecha mientras quede espacio, si no queda pasa a la fila siguiente.
- ❑ **GroupLayout**: se creó para ser utilizado en herramientas de diseño gráfico de interfaces. Trabaja por separado la distribución vertical y horizontal para definir exactamente el posicionamiento de los componentes. Se utiliza en NetBeans.
- ❑ **SpringLayout**: es muy flexible y se usa también para herramientas de diseño gráfico de interfaces. En este caso se especifican las relaciones entre los límites de los componentes bajo su control.


Programar el diseño de un formulario es una de las tareas más arduas en Java, si bien está ampliamente superado gracias al uso de IDEs que facilitan la colocación de componentes a golpe de ratón y sin necesidad de escribir código. Por ejemplo NetBeans, usa el diseño **GroupLayout**.



### Ejercicio resuelto

Las guías que aparecen cuando se añaden elementos a un formulario facilitan sobremanera la colocación de los elementos, aunque se puede hacer necesaria algo más de precisión, para ello podemos usar los botones de alineamiento que encontramos sobre la zona de diseño. Utiliza los iconos de alineación de componentes para alinear horizontalmente los componentes para especificar la profesión, y también los componentes que preguntan por el número de hermanos y la edad. Para alinear las aficiones utiliza la alineación en columna a la derecha para las etiquetas y a la izquierda para los deslizadores.

Modifica el ancho de aquellos controles que consideres conveniente para armonizar el formulario.



### Para saber más

Si quieres ver un ejemplo de como se distribuyen objetos con cada una de las distribuciones que hemos visto puedes visitar esta página:

[Guía visual de los gestores de diseño.](#)



### Debes conocer

En el siguiente enlace puedes encontrar los archivos del proyecto de NetBeans con la aplicación que hemos estado haciendo. Te recomiendo que le eches un vistazo para comprobar los nombres que se le han puesto a las variables, los `ToolTipText`, etc.



[Enlace al ejemplo.](#) (1.09 MB)

## 5.5.- Enlace de componentes a orígenes de datos.



### Caso práctico

**Ana y María** están muy satisfechas del trabajo realizado, sin embargo se han dado cuenta de que éste no ha echo más que empezar. El hotel para el que están creando la aplicación es una empresa que maneja grandes cantidades de información sobre sus empleados, proveedores y clientes y si quieren crear una aplicación de calidad será imprescindible que las interfaces que están creando puedan conectarse a la base de datos del hotel para recuperar, añadir y eliminar información. Así que se ponen manos a la obra para aprender a conectar los componentes a una base de datos.



Uno de los aspecto que aporta mayor flexibilidad y potencia a una aplicación es que pueda acceder a la información contenida en una base de datos.



### Debes conocer

Para poder seguir los contenidos de este punto es imprescindible que tengas conocimiento básicos de **MySQL**, puesto que es necesario que lo tengas instalado y que crees una base de datos para hacer el ejemplo. En estos enlaces tienes toda la información acerca de MySQL y de la herramienta **XAMPP**, que además de MySQL instala el servidor web Apache, **PHP** y la herramienta phpmyadmin para gestionar MySQL en modo gráfico.

[Página oficial de MySql.](#)

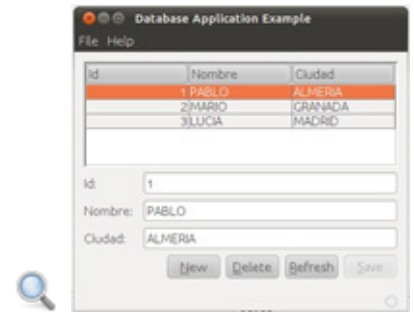
[Página oficial de XAMPP.](#)

Los pasos a dar para conseguir que una aplicación java realizada con NetBeans son:

1. Primero debemos crear la base de datos, para realizar el ejemplo de esta materia, tendrás que crear una base de datos en MySQL y una conexión en NetBeans. Puedes encontrar cómo se hace en el siguiente documento.

### Creación de la base de datos.

2. Crearemos un nuevo proyecto NetBeans de tipo Java Desktop Application que se llame Agenda. Esta vez al seleccionar el intérprete de órdenes de la aplicación seleccionar Aplicación de base de datos.
3. Ahora te pedirá los datos para la tabla maestra, debes seleccionar la cadena de conexión a la base de datos agenda y como tabla de base de datos: contactos.
4. En opciones de detalle selecciona el apartado cuadros de texto.
5. Finaliza el asistente, en el formulario Agenda aparecerán una tabla para mostrar el resultado de la consulta.



Si seleccionas la tabla que ha aparecido en el formulario y observas el Inspector verás que se llama **masterTable** (Jtable). Esta tabla en concreto tiene tres columnas y se le asigna contenido a partir de un elemento, que también puedes ver más abajo en el Inspector llamado **list** (JList) que se rellena a partir de una consulta llamada Query (Jquery), en cuyas propiedades aparece la consulta ejecutada que es `Select c from contactos c`. Haz clic con el botón secundario sobre la tabla >> Enlazar >> Elements, verás como enlaza con la lista y como se obtienen de ella los tres campos de la tabla.

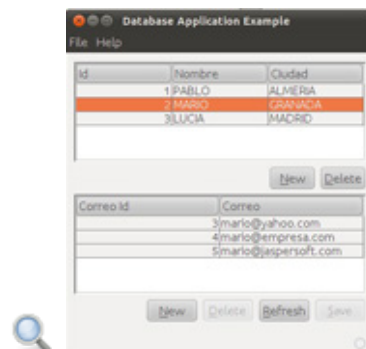
Por otra parte se han añadido tres etiquetas y tres campos de texto al formulario, los campos de texto enlazan (botón secundario >> Enlazar >> Text ) con la tabla maestra en el enlazado de fuente de donde se saca el valor del campo del registro actual mediante la expresión `${SelectedElement.nombre_campo}` de esta manera cada vez que se cambia el registro activo de la tabla se modifica el contenido de los campos de texto.



### 5.5.1.- Formularios maestro-detalle.

Un formulario maestro detalles es aquel en que participan dos o más tablas, entre las que existe una relación de tipo clave externa, por ejemplo, entre los contactos de la agenda y sus correos electrónicos. Está formado por un formulario principal, que lleva asociado otro formulario más pequeño a continuación, de forma que si en el principal aparecen los datos de los contactos, en el secundario aparecerán los datos de los correos del registro activo del formulario principal.

Para hacer un formulario de este tipo en NetBeans debes seguir los pasos dados en el punto anterior hasta el paso 3, a continuación haz lo siguiente:



4. En opciones de detalle selecciona el apartado Tabla CORREOS (ID\_CONTACTO referencia CONTACTOS.ID).
5. Finaliza el asistente, en el formulario Agenda aparecerán dos tablas para mostrar el resultado de la consulta. La de arriba será la tabla principal y la de abajo será la secundaria. Cuando seleccionas un contacto en la tabla superior aparecen sus correos electrónicos (si los tiene) en el inferior.

El funcionamiento de la tabla principal es semejante al caso anterior, la tabla secundaria obtiene sus datos de una lista denominada correosList que se genera a través de código.



#### Debes conocer

En el siguiente enlace tienes el proyecto NetBeans para la consulta sencilla que sólo muestra los datos de los contactos. Recuerda que para que funcione el nombre de tu base de datos debe ser agenda.



[Enlace al ejemplo de base de datos.](#) (12.01 MB)



[Enlace al ejemplo del formulario maestro-detalle.](#) (12.02 MB)


## 6.- Asociación de acciones a eventos.



### Caso práctico

La interfaz gráfica ha sido diseñada con éxito por **María** y por **Ana**, Ahora queda una labor fundamental, qué va a hacer la aplicación cuando el usuario interactúe con ella. **Ada** quiere que la aplicación tenga completa funcionalidad cuando el usuario interactúe con ella con el ratón, teclado, cuando habrá una ventana, etc.



En las aplicaciones que utilizan una  interfaz gráfica de usuario (GUI), a diferencia de la ejecución de un programa de consola, donde el orden de ejecución de las instrucciones dependerá del método Main, el orden de ejecución dependerá de la interacción que realiza el usuario o usuaria sobre la aplicación.

Durante el diseño de la aplicación se deberá considerar qué acciones deseamos que realice nuestra aplicación, cuándo debe realizarlas, y definir los manejadores de eventos que serán invocados de manera automática cuando sobre la aplicación se produzca el evento.

En el lenguaje Java se gestiona el modelo de eventos de la siguiente forma: los objetos sobre los que se producen los eventos (event sources) "registran" los objetos que habrán de gestionarlos (event listeners), para lo cual los event listeners dispondrán de los métodos adecuados. Estos métodos se llamarán automáticamente cuando se produzca el evento. La forma de garantizar que los event listeners disponen de los métodos apropiados para gestionar los eventos es obligarles a implementar una determinada interfaz Listener.

Las interfaces Listener se corresponden con los tipos de eventos que se pueden producir. En los apartados siguientes se verán con más detalle los componentes que pueden recibir eventos, los distintos tipos de eventos y los métodos de las interfaces Listener que hay que definir para gestionarlos. En este punto es muy importante ser capaz de buscar la información correspondiente en la documentación de Java.

Cuando se está diseñando una aplicación, una vez que se ha añadido los diferentes componentes de la interfaz, se procede a definir los objetos Listener, que son los objetos que se encargan de responder a los eventos, para cada evento que se quiera soportar. Una vez definidos los objetos Listener, se procede a implementar los métodos de las interfaces Listener que se vayan a hacer cargo de la gestión de eventos.



## Para saber más

En el siguiente enlace verás una tabla explicativa de los distintos eventos básicos que puedes encontrarte en Java, además incluye un pequeño ejemplo.

[Eventos y Listeners en Java](#)

## 7.- Diálogos modales y no modales.



### Caso práctico

**María** ha diseñado junto con **Ana**, una aplicación de interfaz gráfica que puede tener abierta varias ventanas de manera simultánea, sin embargo, hay ventanas que van a mostrar mensajes de advertencia y que no pueden perder el foco hasta que el usuario pulse el botón aceptar. Necesita conocer la forma de controlar la posibilidad de abrir varias ventanas, y poder pasar el foco de una a otra.



Cuando se diseña una aplicación de interfaz gráfica, el elemento básico es el diálogo o ventana. El diálogo es un área visual que contiene los elementos de interfaz de usuario, tales como menús, botones, listas, etc. mostrando la aplicación y permitiendo la entrada de datos.

Los diálogos se representan casi siempre como objetos de dos dimensiones colocados en el escritorio. La mayoría de ellos pueden ser redimensionados, movidos, ocultos, restaurados, etc.

Existen dos modalidades de diseñar diálogos. La modalidad se refiere a la forma de mantener el foco que va a tener el diálogo con respecto a los demás diálogos.

Un diálogo será no modal, si una vez que se encuentra activo permite alternar el foco a cualquier otro diálogo que se encuentre abierto en el sistema o dentro de la propia aplicación. Normalmente, la mayoría de los diálogos son de este tipo. Dentro de los diálogos modales nos podemos encontrar los modales respecto a una aplicación o los modales respecto al sistema.

Los modales respecto a una aplicación permiten alternar el foco a otros diálogos del sistema, pero no al diálogo que le da origen (diálogo padre) hasta que se produzca una determinada acción sobre ella. Típicamente, son diálogos modales de aplicación, los que se implementan para confirmar una acción del usuario.

Un diálogo modal respecto al sistema no va ceder el foco a ninguna otra aplicación hasta que se produzca una determinada acción sobre él. No suelen ser muy habituales, salvo para cuando se quiere gestionar un evento a nivel de sistema, que requiera la atención inmediata del usuario. Un ejemplo podría ser un diálogo que permita apagar el equipo.

**Para abrir un diálogo modal debes pasar como segundo parámetro del constructor el valor true. Puedes comprobarlo en el ejemplo realizado del formulario para la encuesta.**



## Autoevaluación

### Un escuchador (listener).

- ☐ Es el objeto que produce los eventos.
- ☐ Es cualquier suceso que ocurra en la aplicación, como un clic de ratón.
- ☐ Es un tipo de componente.
- ☐ Es el objeto que se encarga de responder al evento.

## 7.1.- Diálogos modales.

Los diálogos modales se utilizan de forma generalizada en aplicaciones y por el propio sistema operativo.

En una aplicación de interfaz gráfica en Java es necesario definir un objeto de la clase `Frame`. Este objeto va a ser el diálogo padre de toda la aplicación, derivando de él el resto de diálogos que queramos añadir a la aplicación. Para añadir un diálogo modal, Java ya permitía la creación de diálogos modales a través del constructor de la clase `JDialog` como hemos comentado en el punto anterior.

Para crear diálogos modales, en el constructor del diálogo establecemos el parámetro "modal" a `true`, de forma que este diálogo creado mantiene el foco, impidiendo que pueda ser tomado por cualquier otro, dentro de la aplicación, de forma que no podemos seguir ejecutando la aplicación hasta cerrarlo.



```
package mipaquete;
import javax.swing.*;
public class Ventana {
    public static void main(String[] args) {
        JFrame ventana = new JFrame("");
        ventana.setAlwaysOnTop(true);
        ventana.setSize(300,300);
        ventana.setVisible(true);
        JDialog dialogo_Modal = new JDialog(ventana, "Dialogo Modal", true);
        dialogo_Modal.setSize(300,300);
        dialogo_Modal.setLocationRelativeTo(null);
        dialogo_Modal.setVisible(true);
    }
}
```

A partir de Java 6, se puede definir la modalidad de un diálogo utilizando dos clases estáticas dentro de la clase `Dialog`. Las clases son `static class ModalityType` y `static class ModalExclusionType`. Las dos clases incorporan una enumeración que indica el nivel de bloqueo que pueden generar.

La clase `static class ModalityType` sirve para definir el tipo de bloqueo, o también llamado alcance del bloqueo en una aplicación, su enumeración contiene:

- ❑ **APPLICATION\_MODAL**: bloquea todas las ventanas de la aplicación, excepto las ventanas que se deriven de ella.
- ❑ **DOCUMENT\_MODAL**: bloquea la ventana padre de la ventana y su jerarquía superior.
- ❑ **MODELESS**: no bloquea ninguna ventana.
- ❑ **TOOLKIT\_MODAL**: bloquea las ventanas de nivel superior que corren en el mismo `TOOLKIT`.

Podemos excluir del bloqueo a una ventana o diálogo utilizando la clase `ModalExclusionType`, esta puede excluir del bloqueo según alguna de estas opciones de su enumeración:

- ❑ **APPLICATION\_EXCLUDE**: La ventana que utiliza este valor de enumeración no será bloqueada por ningún diálogo de la aplicación.
- ❑ **NO\_EXCLUDE**: Indica que la ventana no estará excluida del bloqueo si este ocurriera.
- ❑ **TOOLKIT\_EXCLUDE**: Indica que no se bloqueará esta ventana si se llamase a `APPLICATION_MODAL` o `TOOLKIT_MODAL`.



### Para saber más

En el siguiente enlace se presentan todos los diálogos modales estándar en Java.

[Creación de diálogo modales estándar.](#)

## 7.2.- Diálogos no modales.

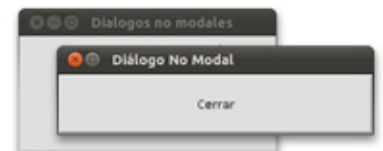


### Caso práctico

**María** cree útil que si en la aplicación de Gestión Hotelera un cliente quiere hacer una reserva, se pueda tener activa al mismo tiempo la ventana con los datos del cliente y la ventana de gestión de reservas, de forma que se pueda pasar el foco de una a otra y facilitar la labor del usuario de la aplicación.



Dentro de una aplicación de interfaz gráfico, nos podemos encontrar con aplicaciones que presentan varias ventanas o diálogos abiertos de manera simultánea. Diremos que los diálogos son no modales, si podemos pasar el foco de un diálogo a otro sin ningún tipo de restricción.



Un ejemplo de aplicación GUI que presenta formularios no modales es el propio entorno de desarrollo NetBeans, donde podemos pasar de una ventana a otra sin ningún problema.

Para definir diálogo no modales en Java, el código que se utiliza es el siguiente:

```
package mipaquete;
import javax.swing.*;
public class Ventana {
    public static void main(String[] args) {
        JFrame ventana = new JFrame("");
        ventana.setAlwaysOnTop(true);
        ventana.setSize(300,300);
        ventana.setVisible(true);
        JDialog dialogoNoModal = new JDialog(ventana, "Dialogo No Modal");
        dialogoNoModal.setSize(300,300);
        dialogoNoModal.setLocationRelativeTo(null);
        dialogoNoModal.setVisible(true);
    }
}
```



Un diálogo no modal permite cambiar el foco a cualquier otro diálogo o ventana abiertos en el sistema.



### Autoevaluación

Si queremos mostrar un diálogo de advertencia dentro de una aplicación de interfaz, el diálogo ¿será no modal?

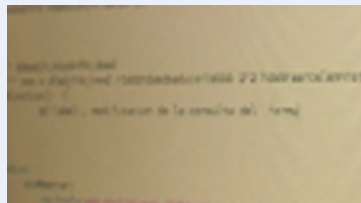
- ☐ Sí.
- ☐ No.

## 8.- Edición de código generado por herramientas de diseño.

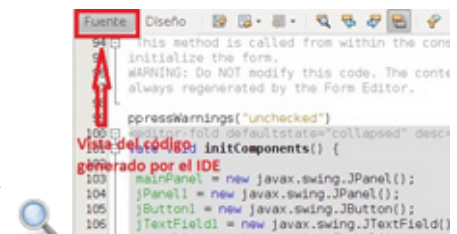


### Caso práctico

**Ana** ha diseñado una interfaz gráfica de mucha calidad, sin embargo le gustaría cambiar algunos detalles de algún componente, pero no encuentra en el IDE la forma de hacerlo. **Ada** le indica que la única manera es modificando directamente el código Java creado por NetBeans, y que puede hacerlo sin ningún problema.



La implementación de una aplicación de interfaz gráfica (GUI) requiere la definición de muchos objetos para la interfaz, establecer sus propiedades, conocer los eventos que se quieren controlar para poder crear los manejadores de eventos, etc. Si la aplicación es simple, puede resultar asumible la implementación de esa forma, pero si la aplicación tiene cierta envergadura, sería un trabajo arduo y lento.



En la actualidad no se concibe la implementación de una aplicación GUI sin el uso de un entorno de desarrollo, que de manera rápida y visual, nos permita crear los diálogos de la aplicación, añadir aquellos objetos que deseemos y establecer de forma gráfica y rápida el aspecto que nos interese.

Con los IDE, como por ejemplo NetBeans, podemos crear la interfaz de manera rápida, y sin tener que implementar una gran cantidad de líneas de código, que el entorno realiza por nosotros.

A pesar de todas las facilidades que nos ofrecen los IDE, siempre es necesario modificar determinados aspectos del código. Para ello podemos modificar con la ventana de código.

El IDE automáticamente genera bloques de código cuando realizas el diseño de una ventana con las herramientas gráficas en "Vista Diseño". Este código se muestra enmarcado dentro del fichero .java, pero puede ser modificado. Se puede modificar la manera de inicializar los componentes, los formularios, las propiedades de los componentes editándolos en la ventana "Fuente" que nos muestra el código fuente generado como respuesta a tu diseño gráfico. Además puedes escribir código personalizado y específico donde consideres necesario.

Para modificar un componente de la ventana, se puede actuar de la siguiente forma:

1. En la ventana Inspector, seleccionas el componente al que quieres editar su código de inicialización.

2. Haz clic en el botón Código en la parte de arriba de la ventana de Propiedades para ver el código de las propiedades.
3. Selecciona la propiedad que desees editar y añade el valor que quieras.

El IDE actualizará el componente seleccionado en el código que había generado con el nuevo valor.

Para modificar la inicialización de código generado para una propiedad de un componente se debe actuar de la siguiente forma:

1. Selecciona el componente en la ventana Inspector.
2. Haz clic en el botón Propiedades en la ventana de propiedades.
3. Selecciona la propiedad que quieras modificar en el código de inicialización.
4. Haz clic en el botón (...) para llegar a la ventana de diálogo Editor de Propiedades.
5. Establece el valor que desees que tenga la propiedad en cuestión.

Para modificar de manera libre el código generado por el IDE, simplemente puedes seleccionar el botón "Fuente" que aparece en la barra de herramientas del IDE, y podrás modificar directamente cualquier parte del código y adaptarlo a tus necesidades.

## 9.- Clases, propiedades, métodos.




### Caso práctico

El equipo de desarrollo de BK Programación, para poder dar funcionalidad a la aplicación de Gestión Hotelera, aparte de conocer los componentes que se pueden incorporar en una interfaz gráfica de usuario, necesitan conocer las clases, propiedades y métodos de Java, para que la aplicación tenga la mayor funcionalidad posible y se adapte a las necesidades del cliente.



Para la implementación de Interfaces Gráficas de Usuario en Java, JavaSoft ha creado un conjunto de clases que son agradables desde el punto de vista visual y fáciles de utilizar para el programador. Esta colección de clases son las Java Foundation Classes (JFC), que en la plataforma Java 2 están constituidas por cinco grupo de clases: AWT, Java 2D, accesibilidad, arrastrar y soltar y swing.

- ✓ AWT. Bibliotecas de clases Java para el desarrollo de Interfaces gráficas de usuario.
- ✓ Java 2D. Es un conjunto de clases gráficas bajo licencia IBM/Taligent.
- ✓ Accesibilidad, proporciona clases para facilitar el uso de ordenadores y tecnología informática a personas con deficiencias visuales como lupas de pantallas, etc.
- ✓ Arrastrar y soltar (Drag and Drop), son clases en la que se soporta los  [JavaBeans](#).
- ✓ Swing. Es la parte más importante para el desarrollo de aplicaciones de interfaz gráfica. Swing proporciona un conjunto de componentes muy bien descritos y especificados, de forma que su presentación visual es independiente de la plataforma donde se ejecuta la aplicación que utilice sus clases. Swing extiende AWT añadiendo un conjunto de componentes, `JComponents`, y sus clases de soporte.

Actualmente, swing a desplazado a AWT debido a que los componentes de swing, al estar escritos en Java, ofrecen mayor funcionalidad, e independiente de la plataforma.



### Autoevaluación

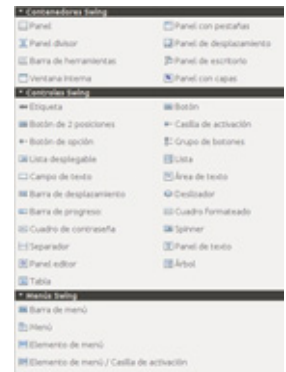
Si se diseña una interfaz gráfica con un Entorno Integrado de Desarrollo, solo se pueden cambiar los valores de las propiedades de un componente desde la ventana de propiedades.

- ☐ Verdadero.
- ☐ Falso.

## 9.1.- Clases.

En el desarrollo de interfaces gráficas de usuario basada en el lenguaje Java, actualmente se utilizan los componentes swing.

Los componentes swing son objetos de clases derivadas de la clase **JComponent** que deriva de la clase **java.awt.Component**. Para crear interfaces gráficas de usuario, swing combina los componentes de la clase **JComponent** en contenedores de nivel alto **JWindow**, **JFrame**, **JDialog** y **JApplet**.



- ❑ **JWindow** es una ventana sin barra de título y sin botones.
- ❑ **JFrame** es una ventana con barra de título y con los botones que permiten su manipulación.
- ❑ **JDialog** permite visualizar un cuadro de diálogo.
- ❑ **JApplet** permite crear un **applet swing**.

Para dar funcionalidad a las ventanas, la swing proporciona un conjunto de componentes que derivan de la clase **JComponent**, los más utilizados son:

- ❑ **JComponent**: Clase base para los componentes swing.
  - **AbstractButton**: Define el comportamiento común de los botones y los menús.
    - **JButton**. Botón.
    - **JMenuItem**. Elemento de un menú.
      - ✖ **JCheckBoxMenuItem**: Elemento del menú que se puede seleccionar o deseleccionar.
      - ✖ **JMenu**: Menú de una barra de menús.
      - ✖ **JRadioButtonMenuItem**: Elemento que forma parte de un grupo de elementos de menú.
    - **JToggleButton**: Botón de estados.
      - ✖ **JCheckBox**. Casilla de verificación.
      - ✖ **JRadioButton**: Botón de opción.
  - **JColorChooser**: Diálogo para seleccionar colores.
  - **JComboBox**: Combinación de caja de texto y lista desplegable.
  - **JLabel**: Etiqueta.
  - **JList**: Lista desplegable.
  - **JMenuBar**: Barra de menús.
  - **JOptionPane**: Componente que facilita la visualización de un cuadro de diálogo.
  - **JPanel**: Contenedor genérico.
  - **JPopupMenu**: Menú que aparece cuando se selecciona un elemento de una barra de menús.
  - **JProgressBar**: Barra de progreso.
  - **JScrollbar**: Barra de desplazamiento.
  - **JScrollPane**: Área de trabajo con barras de desplazamiento.
  - **JSeparator**: Separador para colocar entre dos elementos del menú.
  - **JSlider**: Permite seleccionar un valor dentro de un intervalo que define.
  - **JTableHeader**: Se utiliza para manejar la cabecera de una tabla.
  - **JTextComponent**: Clase base para los componentes de texto.

- **JEditorPane**: Edita diferentes tipos de contenido.
  - ✖ **JTextPane**: Edita texto con formato, incluyendo imágenes.
- **JTextArea**: Caja de texto multilínea.
- **TextField**: Caja de texto de una línea.
  - ✖ **JPasswordField**: Se usa para introducir contraseñas.
- ➡ **JToolBar**: Barra de herramientas.

## 9.2.- Propiedades.



### Caso práctico

Ya conocemos los componentes que podemos incorporar en una aplicación de interfaz gráfica. ¿Cómo se puede cambiar el texto de un botón? ¿Y el color de fondo de una ventana? El equipo de desarrollo de BK Programación debe conocer las propiedades más importantes de los componentes de la swing para poder responder a estas preguntas.



Las propiedades de los componentes nos van a permitir adaptar su comportamiento y apariencia.

Las propiedades más importantes que presentan la mayoría de los componentes swing son las siguientes:

- ❑ **background**: Determina el color de fondo del componente.
- ❑ **font**: Establece el tipo de fuente que va a mostrar el componente.
- ❑ **foreground**: Establece el color de la fuente que muestra el componente.
- ❑ **horizontalAlignment**: Establece la alineación del texto dentro del componente en el eje X.
- ❑ **icon**: Indica si el componente muestra o no un icono, y cuál sería.
- ❑ **labelFor**: Indicaría si el componente es etiquetable.
- ❑ **text**: Muestra el texto que indica la propiedad en componentes como caja de texto o etiquetas.
- ❑ **toolTipText**: Con esta propiedad definimos el texto que aparece como tool tip.
- ❑ **verticalAlignment**: Establece la alineación del texto dentro del componente en el eje Y.
- ❑ **alignmentX**: Alineamiento horizontal preestablecido para el componente.
- ❑ **alignmentY**: Alineamiento vertical preestablecido para el componente.
- ❑ **autoscrolls**: Determina si el componente puede realizar scroll de forma automática cuando se arrastra el ratón.
- ❑ **border**: Establece el tipo de borde que va a presentar el componente.
- ❑ **componentPopupMenu**: Establece el menú contextual que se muestra en este componente.
- ❑ **cursor**: Establece el tipo de cursor que se va a mostrar cuando el curso entre en el componente.
- ❑ **disableIcon**: Establece el icono a mostrar si el componente no está activo.
- ❑ **enabled**: Nos indica si el componente está o no activo.
- ❑ **focusable**: Establece si el componente puede o no recibir el foco.
- ❑ **horizontalTextPosition**: Establece la posición horizontal del texto del componente, en relación con su imagen.



- ❑ **iconTextGap**: Si el componente tiene activo el texto y el icono, con esta propiedad se establece la distancia entre ellos.
- ❑ **inheritsPopupMenu**: Indica si el menú contextual se hereda o no.
- ❑ **inputVerifier**: Establece el componente de verificación para este componente.
- ❑ **maximunsize**: Establece el tamaño máximo del componente.
- ❑ **minimunsize**: Establece el tamaño mínimo del componente.
- ❑ **name**: Establece el nombre del componente.
- ❑ **opaque**: Modifica la opacidad del componente.
- ❑ **preferredSize**: Tamaño predefinido para este componente.
- ❑ **verifyInputWhenFocusTarget**: Si el componente es verificado antes de recibir el foco.
- ❑ **verticalTextPosition**: Establece la posición vertical del texto del componente, en relación con su imagen.

Las propiedades que hemos enumerado anteriormente, son las más habituales que te puedes encontrar en los componentes de las swing de Java. Para modificar los valores de las propiedades dispones de la ventana de propiedades del IDE, o bien puedes hacerlo desde el código fuente Java. Si deseas modificar el valor de una propiedad desde el código fuente, el IDE te va a mostrar una lista de todas la propiedades disponibles para el componente en cuestión, una vez que escribas el nombre del objeto y un punto.

## 9.3.- Métodos.



### Caso práctico

Ana ha diseñado gran parte de la interfaz gráfica junto con María. Ese aspecto de la aplicación es muy profesional, sin embargo, ¿cómo puedo modificar su aspecto cuando se esté ejecutando? ¿Cómo puedo activar o desactivar un componente en ejecución? ¿Cómo puedo responder a eventos? Es necesario conocer los métodos asociados a los componentes, para poder dar funcionalidad completa a la aplicación.



Cada componente que forma parte de una aplicación GUI utilizando Java, dispone de un conjunto completo de métodos, que nos permite comunicarnos con el componente y modificar su aspecto o comportamiento. A continuación se va a mostrar una lista de los componentes más importantes de la swing Java, junto con sus métodos más destacados:

- ❑ **JFrame**. Los métodos más importantes son: `getTitle()`, `setBounds(x,yx,w,h)`, `setLocation`, `setMaximumSize(w,h)`, `setMinimumSize(w,h)`, `setPreferredSize(w,h)`, `setResizable(bool)`, `setSize(w,h)`, `setTitle(str)` y `setVisible(bool)`.
- ❑ **JPanel**. El panel de contenido dispone del método `add()`, para añadir componentes al panel.
- ❑ **JLabel**: Las etiquetas tienen como métodos más importantes: `setText()` que permite modificar el texto, `setVerticalAlignment()` para modificar la alineación vertical, etc.
- ❑  **JButton**: Los botones presentan entre otros métodos: `setEnabled(bool)` que permite activar o desactivar el botón, `isEnabled()` que permite comprobar si está o no activo, `setMnemonic(char)` que permite asociar una tecla al botón, etc.
- ❑ **JProgressBar, JScrollBar**. Estos componentes disponen de los siguientes métodos: `setExtent()`, `setMaximum()`, `setMinimum()`, `setValue()`, `getValueIsAdjusting()` y `setOrientation()`.
- ❑ **JSlider**: Este componente tiene como métodos más destacados: `setPaintTicks(bool)`, `setPaintLabels(bool)`, `setMajorTickSpacing(int)` y `setMinorTickSpacing(int)` para determinar los valores del intervalo, `setSnapToTicks(true)` y `setInverted(bool)`.
- ❑ **JRadioButton**. Este componente tiene como método principal `isSelected()` que devuelve el estado del mismo.
- ❑ **JList**. En las listas desplegables destacan los métodos siguientes para editar los elementos de la lista: `addElement(objName)`, `elementAt(index)`, `removeElement(objName)`, `removeAllElements()` y `removeElementAt(idk)`. Para comprobar el estado de la lista se usan los

métodos `contains(objName)`, `indexOf(name)` y `size()`. Para convertir la lista en array se usa `copyInto(array)`.

- ❑ **JComboBox**. Dispone de diferentes métodos, destacando `setEditable(true)` que permite editar los ítems del combobox. Para recuperar los ítems seleccionados se usa `getSelectedItem()` y `getSelectedIndex()`. Otros métodos útiles son `getItemAt(idx)`, `getItemCount()`, `setSelectedItem(obj)` y `setMaximumRowCount(x)`.
- ❑ **JTextPane** y **JTextArea**. Como métodos más útiles de estos componentes destacan `getText()`, `setText()`, `append(str)`, `insert(str,pos)`, `replace(str,beg,end)`, `setEditable(bool)`, `setLineWrap(bool)` y `setWrapStyleWord(bool)`.
- ❑ **JMenuItem**. Para añadir separadores se usa el método `addSeparator()` o `insertSeparator(posn)`. Los Menú ítems se pueden activar o desactivar con `setEnabled(bool)` y comprobar si están activos con `isEnabled()`. Para añadir teclas de acceso rápido se usa el método `setMnemonic(char)`.

Cada componente dispone de muchos más métodos que puedes utilizar en tus aplicaciones, utilizando un IDE como NetBeans, una vez que hayas definido un objeto, por ejemplo el objeto `etSaludo` de la clase `JLabel`, cuando invoques `etSaludo` y escribas a continuación un punto, automáticamente te aparecerá una lista con todos los métodos que puedes usar para ese objeto, junto con una breve explicación de su uso.

**Cada componente dispone de sus propios métodos, mediante los cuales podemos comunicarnos con él en tiempo de ejecución, permitiendo adaptar su apariencia y comportamiento.**


## 10.- Eventos, escuchadores.



### Caso práctico

Cuando se diseñan aplicaciones de interfaz gráfica, la aplicación tiene que responder a los eventos que se producen en cada momento. **Juan** se va a encargar de establecer a qué eventos va a responder la aplicación, programando el código que debe ejecutarse en cada momento, en función del componente que haya recibido el evento, y en función del tipo de evento recibido.



En un entorno GUI, cuando se produce un movimiento de ratón, se pulsa un tecla, se cierra una ventana, etc. se produce un evento. El evento es recibido por un objeto, como puede ser un botón, una caja de texto, etc. El objeto que recibe el evento, para poder responder a él, debe implementar la interfaz apropiada (event  handler) y registrarla como un escuchador (event listener) en el componente GUI (event source u objeto que va recibir el evento) apropiado.

Los eventos se agrupan en función al componente que lo produce o a la acción que lo provoca.

El evento tiene los siguientes elementos:

- ❑ La **fuentes del evento** (event source): Es el componente que origina el evento.
- ❑ El **escuchador** (event listener): es el encargado de atrapar o escuchar el evento.
- ❑ El **manejador del evento** (event handler), es el método que permite implementar la interfaz.

El manejador del evento recibe un objeto evento (ActionEvent) que contiene información sobre el evento que se ha disparado, cuando esto sucede, el manejador del evento descifra el evento con dicho objeto y procesa lo solicitado por el usuario o usuaria.

Un componente de una interfaz gráfica dispara o activa los manejadores (event handler) dependiendo del tipo de evento que ha ocurrido. El componente puede tener registrado más de un manejador que maneja el mismo tipo de evento. Un evento puede ser escuchado por más de un manejador de eventos.

El manejador del evento requiere que en la declaración de la clase que maneja el evento (event handler), se indique la interfaz correspondiente al evento (XListener, donde X es el tipo de evento a escuchar). La clase puede implementar más de un XListener. Si la clase no implementa la interfaz puede ser que extienda a una clase que si la implementa:

```
public class miClase implements ActionListener{...}
```

En los componentes que son la fuente del evento (event source) se registra una instancia del manejador del evento (event handler), como un observador o escucha del tipo de eventos que maneja (`XListener`). Es decir, se le dice al componente que va a escuchar los eventos del tipo del manejador.

```
componente.addActionListener( instancia_de_miClaseListener);
```

En la clase que maneja el evento (event handler) se deben implementar los métodos de la interfaz `XListener` que descifren el evento (`XEvent`) y lo procesen.

```
public void actionPerformed(ActionEvent e) {  
    ...//Código que reaccione a la acción  
}
```

## 10.1.- Escuchadores.



### Caso práctico

La interfaz gráfica ha sido diseñada, y la aplicación tiene ya su aspecto final. Sin embargo, al hacer clic sobre las opciones del menú, la aplicación no realiza nada. Ana quiere aprender los mecanismos que siguen las aplicaciones para poder capturar los eventos que ocurren en la aplicación y responder a esos eventos ejecutando el código adecuado en cada momento.



Un escuchador (listener) es el objeto de la clase que implementa la interfaz de escucha de un evento y que contiene el método de respuesta al propio evento. Cuando se produce un determinado evento dentro de la aplicación GUI, si se desea responder a esos eventos, la aplicación deberá implementar una serie de métodos que se ejecuten de forma automática cuando se produzca esos eventos. Los métodos que se van a implementar para responder a los diferentes eventos dentro de una aplicación de interfaz gráfica, se definen en unas interfaces denominadas interfaces de escucha.

Cada interfaz de escucha contiene los métodos para gestionar un determinado grupo de eventos. La interfaz de escucha `WindowListener` define el formato de los métodos para la gestión de los eventos de ventana, como pueden ser abrir la ventana, cerrarla, maximizarla, minimizarla etc. Algunos de los métodos de `WindowListener` son:

- ☑ `public void windowActivated(WindowEvent e)`: Invocado cuando la ventana es la ventana activa.
- ☑ `public void windowDeactivated(WindowEvent e)`: Invocado cuando la ventana deja de ser la ventana activa.
- ☑ `public void windowClosing(WindowEvent e)`: Cuando la ventana se ha cerrado.
- ☑ `public void windowClosed(WindowEvent e)`: Cuando la ventana se minimiza.
- ☑ `public void windowIconified(WindowEvent e)`: Cuando la ventana se restablece.
- ☑ `public void windowDeiconified(WindowEvent e)`: La primera vez que la ventana se minimiza.
- ☑ `public void windowOpened(WindowEvent e)`: La primera vez que la ventana se hace visible.

Cada interfaz de escucha contiene sus propios métodos para la gestión de eventos. Ejemplos de eventos que son generados por componentes de la swing son:

- ❑ **ActionListener**: Captura cierto tipo de acción realizada sobre ciertos componente. Por ejemplo, pulsar un botón, seleccionar un elemento en una lista desplegable o una opción en un menú.
- ❑ **ChangeListener**: Registra los cambios sobre ciertos componentes.
- ❑ **ItemListener**: Recoge el cambio de estado en un componente tipo listas desplegables.
- ❑ **MouseListener**: Eventos producidos por la manipulación del ratón.
- ❑ **MouseMotionListener**: Movimiento del ratón sobre un componente.
- ❑ **ComponentListener**: Visibilidad de componentes.
- ❑ **FocusListener**: Captura eventos relacionados con la recepción o pérdida del foco.
- ❑ **KeyListener**: Captura pulsaciones del ratón sobre el componente activo.
- ❑ **ListSelectionListener**: Selección de elementos dentro de una lista.

Para responder a un evento dentro de una aplicación, deberás definir una clase que interprete la interfaz de escucha correspondiente al tipo de evento que quieras gestionar. A los objetos de esa clase de escucha se les denomina escuchadores.

Si se quiere responder a un determinado evento en una aplicación, hay que crear un objeto que escuche cuando se produce el evento y responda a él.



### Autoevaluación

#### Un manejador de evento.

- ☐ El el componente que provoca el evento.
- ☐ Es el componente que origina el evento.
- ☐ Recibe un objeto evento, descifrando el evento y procesando lo solicitado por el usuario.

## Anexo I.- Creación de la base de datos.

Para trabajar estos contenidos vamos a usar una base de datos **MySQL** que contiene la información de los contactos de una agenda. Se supone que ya tienes un servidor **MySQL** funcionando y sabes cómo crear y rellenar las tablas de una base de datos. Para tenerlo todo listo para el ejemplo crea una base de datos que se llame **agenda** e importa un archivo en el que hayas escrito las siguientes sentencias:

```
insert into CONTACTOS values(1, 'PABLO', 'ALMERIA');
insert into CONTACTOS values(2, 'MARIO', 'GRANADA');
insert into CONTACTOS values(3, 'LUCIA', 'MADRID');

create table CORREOS (
    CORREO_ID int primary key,
    ID_CONTACTO int NOT NULL,
    CORREO varchar(100) NOT NULL,
    INDEX (ID_CONTACTO),
    FOREIGN KEY (ID_CONTACTO) REFERENCES CONTACTOS(ID)
)ENGINE=INNODB;

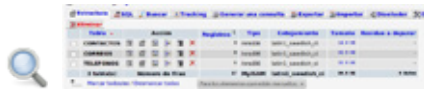
insert into CORREOS values(1, 1, 'pablo@yahoo.com');
insert into CORREOS values(2, 1, 'pablo@gmail.com');
insert into CORREOS values(3, 2, 'mario@yahoo.com');
insert into CORREOS values(4, 2, 'mario@empresa.com');
insert into CORREOS values(5, 2, 'mario@jaspersoft.com');
insert into CORREOS values(6, 3, 'lmc@dominio.es');
insert into CORREOS values(7, 3, 'lucy@algunemail.com');
insert into CORREOS values(8, 3, 'luciamartos@organizacion.org');

create table TELEFONOS (
    TELEFONO_ID int primary key,
    ID_CONTACTO int NOT NULL,
    TELEFONO varchar(10) NOT NULL,
    INDEX (ID_CONTACTO),
    FOREIGN KEY (ID_CONTACTO) REFERENCES CONTACTOS(ID)
)ENGINE=INNODB;

insert into TELEFONOS values(1, 1, '111111111');
insert into TELEFONOS values(2, 1, '222222222');
insert into TELEFONOS values(3, 1, '333333333');
insert into TELEFONOS values(4, 2, '444444444');
insert into TELEFONOS values(5, 3, '555555555');
insert into TELEFONOS values(6, 3, '666666666');
```

Puedes hacerlo usando alguna herramienta como phpmyadmin o directamente en mysql. Si lo haces con phpmyadmin quedaría así:





## Registrar el servidor MySQL.

En el IDE, tenemos que registrar el servidor de bases de datos MySQL (si no lo está). Para esto, abrimos la pestaña Prestaciones y hacemos clic derecho en el nodo Bases de datos. En el menú contextual, hacemos clic en Registrar servidor MySQL. Se mostrará una ventana en la que ingresaremos la configuración del servidor, normalmente con los siguientes datos:

- ☒ Nombre de servidor: localhost.
- ☒ Número de puerto del servidor: 3306.
- ☒ Nombre del usuario administrador: root (o el que tu hayas decidido).
- ☒ Contraseña: la que hayas indicado al instalar MySQL.















Una vez registrado debes iniciar el servidor haciendo clic con el botón derecho y seleccionado iniciar y después conectarte siguiente el mismo procedimiento.

## Crear una conexión en NetBeans.

Una vez que el motor de bases de datos está funcionando desplégalo y elige el nombre de la base de datos contra la que ten quieres conectar, en este caso agenda, aparecerá una conexión debajo que será la que debas elegir al crear la ventana principal de la aplicación.

## Anexo.- Licencias de recursos.

### Licencias de recursos utilizados en la Unidad de Trabajo

Recurso (1)	Datos del recurso (1)	Recurso (2)	
	Autoría: Netbeans. Licencia: CDDL, GNU General Public License 2. Procedencia: Captura de pantalla de Netbeans.		Autoría: Licenci Proced
	Autoría: Netbeans. Licencia: CDDL, GNU General Public License 2. Procedencia: Captura de pantalla de Netbeans.		Autoría: Licenci Proced <a href="http://www.frame-1.com">http://www.frame-1.com</a>
	Autoría: Silveira Neto. Licencia: CC by-sa 2.0. Procedencia: <a href="http://www.flickr.com/photos/silveiraneto/2579658422/">http://www.flickr.com/photos/silveiraneto/2579658422/</a>		Autoría: Licenci 2. Proced
	Autoría: Netbeans. Licencia: CDDL, GNU General Public License 2. Procedencia: Captura de pantalla de Netbeans.		Autoría: Licenci 2. Proced
	Autoría: Netbeans. Licencia: CDDL, GNU General Public License 2. Procedencia: Captura de pantalla de Netbeans.		Autoría: Licenci 2. Proced
	Autoría: Netbeans. Licencia: CDDL, GNU General Public License 2. Procedencia: Captura de pantalla de Netbeans.		Autoría: Licenci 2. Proced
	Autoría: Scott Schram. Licencia: CC by 2.0. Procedencia: <a href="http://www.flickr.com/photos/schram/21742249/">http://www.flickr.com/photos/schram/21742249/</a>		Autoría: Licenci 2. Proced Título: Captur



Autoría: Francisco Javier Cabrerizo.  
Licencia: Uso educativo no comercial.  
Procedencia: Elaboración propia a partir de una  
captura de pantalla de NetBeans.



Autoría  
Licenci  
2.  
Proced