# Conocimiento: Clase Container

La clase **Container** es la raíz de las clases que hacen el papel de contenedores en la interfaz gráfica.

* Constructor: **Container ()**, este crea una nueva ventana que no tiene título ni bordes, por tanto es invisible. La utilidad es crear la base de cualquier contenedor.
* Utilidades: esta clase contiene una serie de métodos que permiten agregar, eliminar, consultar, organizar y controlar los componentes que deban estar dentro del contenedor, entre ellos tenemos:
  + **Component add (Component comp)**: este método permite agregar un componente al contenedor.
  + **void remove (Component comp)**: este método permite eliminar o remover un componente desde el contenedor.
  + **void setFont (Font font)**: este método permite cambiar el tipo de letra que se utilizará en los textos dentro del contenedor.
  + **void print (Graphics g)**: este método dibuja en el contenedor
  + **LayoutManager getLayout ()**: este método permite obtener el gestor de diseño que está utilizando el contenedor.
  + **void setLayout (LayoutManager layout)**: este método permite cambiar el gestor de diseño del contenedor.
  + **void validate ()**: este método valida el contenedor y sus componentes, en especial después de que se ha mutado el estado del contenedor agregando o removiendo componentes.

La jerarquía de clase que heredan de Container, en AWT, es la siguiente:

* Window
  + Frame
  + Dialog
    - FileDialog
* Panel
  + Applet
* ScrollPane

**Clase Window**

Esta clase es una superficie de pantalla de alto nivel (una ventana). Una instancia de la clase Window no puede estar enlazada o embebida en otro Contenedor. Una instancia de esta clase no tiene ni título ni borde.

Generalmente no se crean objetos de la clase Window sino que se utiliza una subclase para instanciarlo.

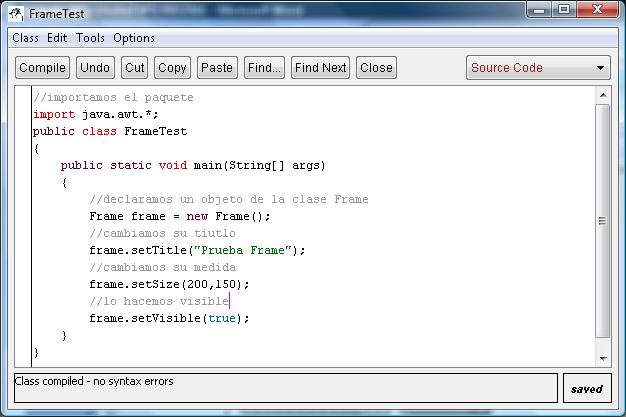
* Constructor: **Window (Frame owner)**, este método crea una ventana cuyo dueño es el Contenedor entregado en el parámetro de entrada, si es **null** se crea una ventana en blanco que no es manipulable.
* Utilidades: la clase Window hereda de la clase Container, por tanto posee la habilidad de utilizar todos los métodos definidos en dicha clase que tenga la accesibilidad suficiente para que esto suceda, además agrega otras funcionalidades, tales como:
  + **void addWindowListener (WindowListener listener)**: este método recibe un evento de ventana que deberá ser activado para provocar algún comportamiento.
  + **List getIconImages ():** este método retorna la imagen que sirve de icono identificatorio de la ventana.
  + **getOwner():** este método retorna el contenedor dueño de esta ventana
  + **boolean isActive ():** este método verifica si la ventana está activa.
  + **void pack ()**: este método permite cambiar el tamaño de la ventana al más apropiado de manera tal que todos los componentes sean contenidos correctamente.
  + **void setIconImage (Image img)**; este método permite cambiar la imagen que sirve de icono identificatorio de la ventana.
  + **void setBounds (int x, int y, int width, int height)**: este método permite cambiar la ubicación y el tamaño de la ventana.

**Clase Frame**

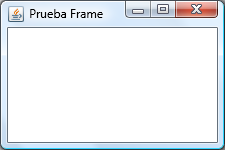
Esta clase hereda de la clase Windows y permite instanciar un contenedor que puede ser visible y que permite ser manipulado en forma y contenido.

* Constructor:
  + **Frame()**, este constructor inicializa una nueva ventana invisible y sin título;
  + **Frame (String titulo)**, este constructor inicializa una nueva ventana invisible y con el título especificado en el parámetro.
  + **Frame (GraphicsConfiguration gc)**, este constructor inicializa una nueva ventana invisible, basado en la configuración gráfica del objeto gc.
* Utilidades: esta clase utiliza todos los métodos heredados de la clase Window, además agrega y mejora otras funcionales, tales como:
  + **Menubar getMenuBar ():** este método permute conocer el MenuBar asociado a este contenedor.
  + **String getTitle ():** este método permite conocer el título del contenedor.
  + **boolean isResizable ():** este método permite verificar si la ventana puede o no cambiar su tamaño original.
  + **void remove (MenuComponent m):** este método permite remover o eliminar el Menubar asignado a la ventana.
  + **void setMenuBar (MenuBar mb):** este método permite asignar un nuevo Menubar a la ventana.
  + **void setResizable (boolean resizable):** este método permite activar o desactivar la opción del cambio de tamaño.
  + **void setTitle (String titulo):** este método permite cambiar el título de la ventana.

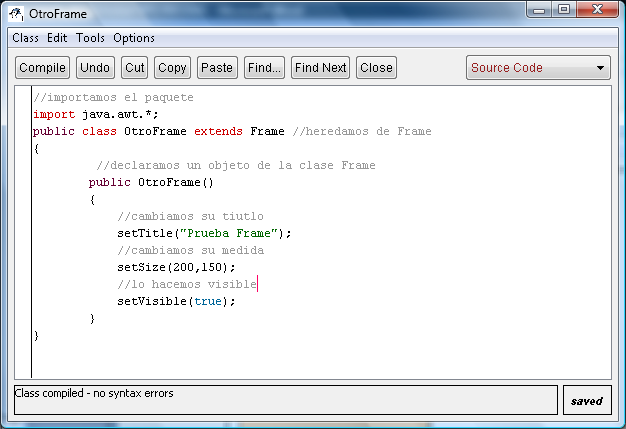
Ejemplo: creación de un Frame:

****

En este ejemplo utilizamos colaboración al declarar un objeto de la clase Frame dentro del método main. El resultado de ejecutar este proceso es:



También se puedeutilizar herencia, ya que la clase Frame no es final, por tanto heredamos de ella y utilizamos los métodos para trabajar en sus características desde la clase hija:

****

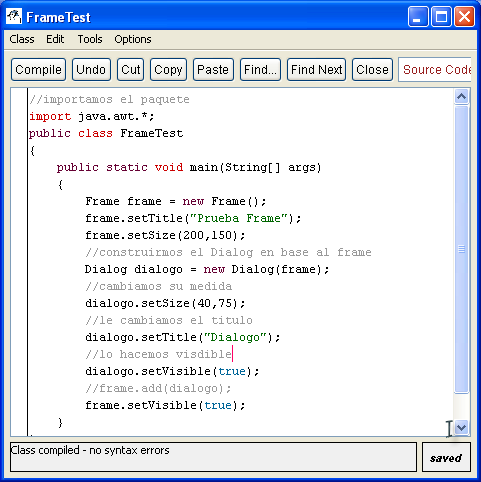
Aquí se puede observar la herencia en la cáscara.

**Clase Dialog**

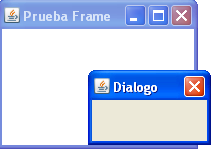
Esta clase hereda de la clase Window y representa a una ventana que va a estar temporalmente visible, simulando un cuadro de diálogo, además debe estar construida en un contenedor padre determinado para que pueda existir.

* Constructor:
  + - * + **Dialog (Frame frame, boolean modal)**, este constructor Crea un cuadro de diálogo cuya ventana padre es frame, donde modal indica si el cuadro de diálogo bloquea o no todos los eventos de teclado y mouse hacia su padre.
        + **Dialog (Frame frame, String titulo, boolean modal),** este constructor Crea un cuadro de diálogo con un título específico, cuya ventana padre es frame, donde modal indica si el cuadro de dialogo bloquea o no todos los eventos de teclado y mouse hacia su padre.
* Utilidades: esta clase utiliza todos los métodos heredados de la clase Window, además agrega y mejora otras funcionalidades, tales como:
  + - * + **String getTitle( )**: este método retorna el título del Dialog
        + **boolean isModal( )**: este método verifica si el cuadro de diálogo bloquea todos los eventos del teclado y mouse para con su padre
        + **boolean isResizable ( )**: este método verifica si se pueden cambiar las dimensiones del cuadro diálogo con el mouse.
        + **void setResizable( boolean valor )**: este método establece si se quiere o no que se puedan cambiar las dimensiones del cuadro usando el mouse
        + **void setTitle (String titulo)**: este método permite cambiar el título de del Diálogo.

Ejemplo: agregar un Dialog al Frame creado anteriormente:



Después de ejecutar el método main en pantalla aparece lo siguiente:

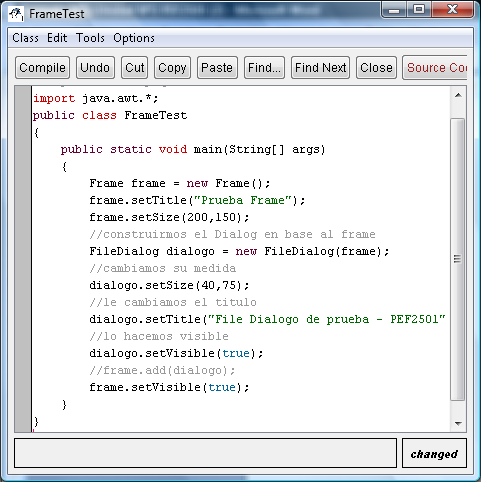


**Clase FileDialog**

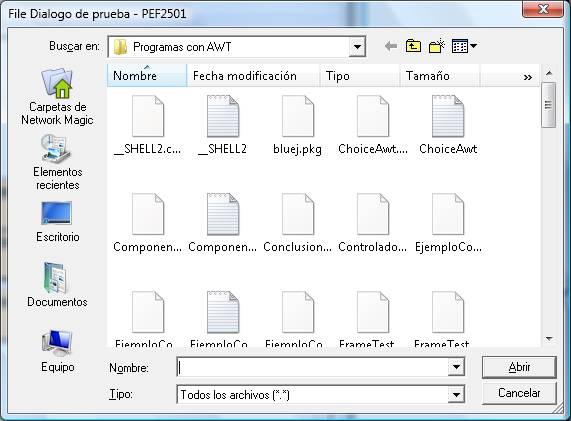
Esta clase hereda de Dialog y permite construir un dialogo para manipular archivos, es decir, abrir o crear un archivo en un directorio especificado.

* **Constructor**
  + - * + **FileDialog (Dialog parent):** este constructor crea un objeto FileDialog para abrir o cargar un archivo.
        + **FileDialog (Dialog parent, String titulo):** este constructor crea un objeto FileDialog para abrir o cargar un archivo, con el título de la ventana especificado por el parámetro.
        + **FileDialog (Dialog parent, String titulo, int mode):** este constructor crea un objeto FileDialog para abrir o cargar un archivo, con el título de la ventana especificado por el parámetro y con el modo de carga o creación según lo especificado en el parámetro mode.
        + **FileDialog(Frame parent):** este constructor crea un objeto FileDialog para abrir o cargar un archivo
        + **FileDialog (Frame parent, String titulo):** este constructor crea un objeto FileDialog para abrir o cargar un archivo, con el título de la ventana especificado por el parámetro.
        + **FileDialog (Frame parent, String titulo, int mode):** este constructor crea un objeto FileDialog para abrir o cargar un archivo, con el título de la ventana especificado por el parámetro.
* Utilidades**:** esta clase utiliza todos los métodos heredados de la clase Window y Dialog, además agrega y mejora otras funcionalidades, tales como:
  + - * **String getDirectory ():** este método retorna un String que contiene el directorio donde se ubica o ha accedido el FileDialog.
      * **String getFile ():** este método retorna un String que contiene el nombre del archivo seleccionado por el FileDialog, si se ha escogido la opción cancelar, retorna un **null.**
      * **int getMode ():** este método retorno un valor int que corresponde al modo de trabajo del FileDialog, si es 0 indica que se desea abrir un archivo, si es 1 indica que se desea grabar un archivo.
      * **void setDirectory (String dir):** este método permite cambiar el directorio donde se abrirá el FileDialog.
      * **void setFile (String file):** este método permite cambiar el archivo que será seleccionado por el FileDialog.
      * **void setMode (int mode):** este método permite cambiar el modo de operar del FileDialog, para abrir o para grabar un archivo.

Ejemplo: Crear un FileDialog que permita abrir un archivo:

****

El resultado de ejecutar el método main es:



**Clase Panel**

La clase Panel es un Contenedor genérico de Componentes. Una instancia de la clase Panel no tiene bordes ni título, simplemente proporciona un Contenedor al que se le debe ir añadiendo Componentes. Un Panel constituye una pantalla concreta que puede anidarse de forma recursiva.

* Constructor:
  + - * **Panel():** construye un objeto de la clase panel invisible
      * **Panel(LayoutManager layout):** crea un objeto de la clase Panel con el gestor de diseño indicado en el parámetro de entrada.
    - Utilidades: esta clase no agrega nuevas utilidades pero es la clase padre de Applet

**Clase Applet**

En Java un Applet (Subprograma), es un programa que puede incrustarse en un documento HTML; es decir en una pagina Web, Cuando un Navegador carga una pagina Web que contiene un Applet, este se descarga en el navegador Web y comienza a ejecutarse esto nos permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la pagina Web en su navegador.

Para describir la disposición de una página Web, se emplea el lenguaje de marcación para hipertextos (HTML), el que permite identificar los elementos de que consta una página. En el caso de querer levantar un Applet es necesario que exista un fichero HTML que utilice la etiqueta <applet>, de la siguiente manera:

<applet code=”NombreApplet” width = 200 height = 60>

</applet>

En el recuadro anterior encontramos la etiqueta que ejecutará el applet llamado NombreApplet, en una ventana de 200 píxeles de ancho y 60 píxeles de alto.

La clase Applet se encuentra en el paquete java.applet.

Para trabajar con esta clase es necesario heredar de ella y sobrescribir los siguientes métodos para lograra el buen funcionamiento: init (), start (), stop ().

* Constructor**:**
  + - * **Applet():** este método construye un objeto de la clase Applet
* Utilidades: esta clase hereda de Panel, por tanto puede usar los métodos heredados, además agrega nuevas utilidades tales como:
  + - * **String getAppletInfo ():** este método retorna un String conteniendo información sobre el autor y la versión del Applet, debe ser sobrescrito ya que en la clase padre retorna null.
      * **void init ():** este método permite inicializar el applet en el browser o el AppletViewer. Es invocado la primera vez que se ejecuta el applet.
      * **void start ():** este método es invocado cada vez que el applet es vuelto a visitar y después de que se ha ejecutado el método init(). Debe ser sobrescrito para especificar las acciones que se van a realizar en el Applet, ya que el método en la clase padre tiene una implementación vacía.
      * **void stop ():** este método es invocado cada vez que el Applet queda invisible, pero sigue estando vivo. Debe ser sobrescrito ya que en la clase padre tiene una implementación vacía y debe contener las acciones que se desea se ejecuten cuando el applet pierde el foco.

Ciclo de vida de un Applet

El siguiente esquema muestra el orden de ejecución de los métodos definidos en la clase Applet y que es necesario sobrescribir en la clase hija para implementar correctamente dicho Applet.

init

start

stop

destroy

Llama a stop()

cada vez que se

deshabilita

Llama a start()

cada vez que se

habilita

Se ejecuta init()

al principio cuando

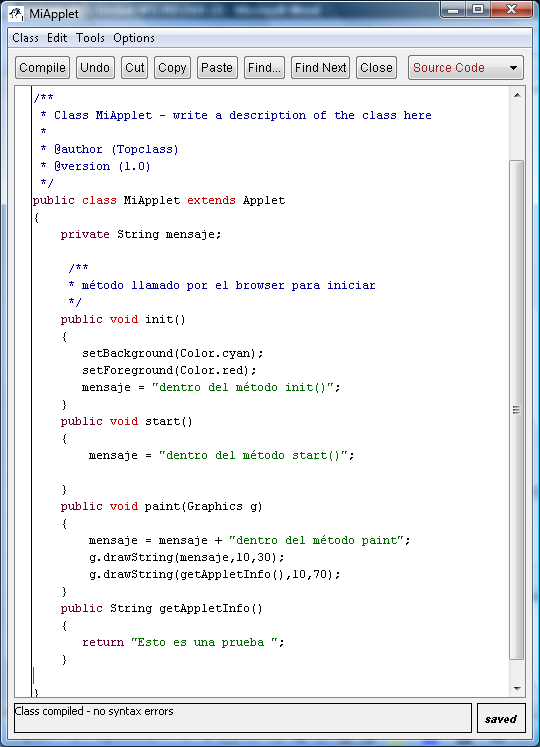
se levanta el Applet

Se ejecuta destroy()

cuando el applet se

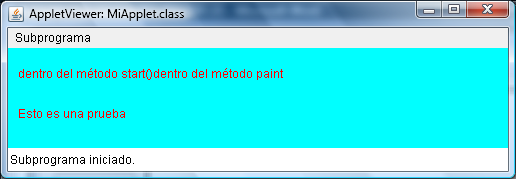
cierra definitivamente

Ejemplo de un applet simple:

****

En este ejemplo la clase MiApplet hereda de Applet, sobrescribe init, start, paint y getAppletinfo para que funcione según los requerimientos específicos de este Applet

El resultado de levantar el applet con el AppletViewer es:



Al ejecutarlo observamos que el mensaje dentro del método init() no aparece, sino que los valores que le dimos en start() y paint().