Proyectos MDI

PROYECTOS SDI (Interface de Documento simple) ***versus MDI-*** *(*interfaz de múltiples documentos*)*

- Hasta ahora hemos estado estudiando algunos de los controles básicos incluidos en el Cuadro de Herramientas de la interface de desarrollo de Visual Studio.
- Para realizar cada una de las prácticas, hemos empleado Proyectos SDI; es decir, '*todo ocurre' en un único* Formulario.
- Pero, si deseamos presentar al usuario final un proyecto a partir del cual éste pueda seleccionar cual/les es/son los formularios –desde un conjunto- que quiere ejecutar … no podremos hacerlo desde un proyecto SDI.
- Bueno, quizás podríamos intentarlo incluyendo un menú de opciones y '*jugar' a hacer visibles / invisibles recursos* de tipo contenedor al pulsar sobre las distintas opciones de este menú.
- *Demasiado complicado ! … a medida que -en modo diseño fuésemos* añadiendo opciones al menú y recursos contenedores, la interface se haría ilegible.
- Para este tipo de proyectos, mejor utilizar la estrategia de diseño y desarrollo MDI - Proyectos multificha.

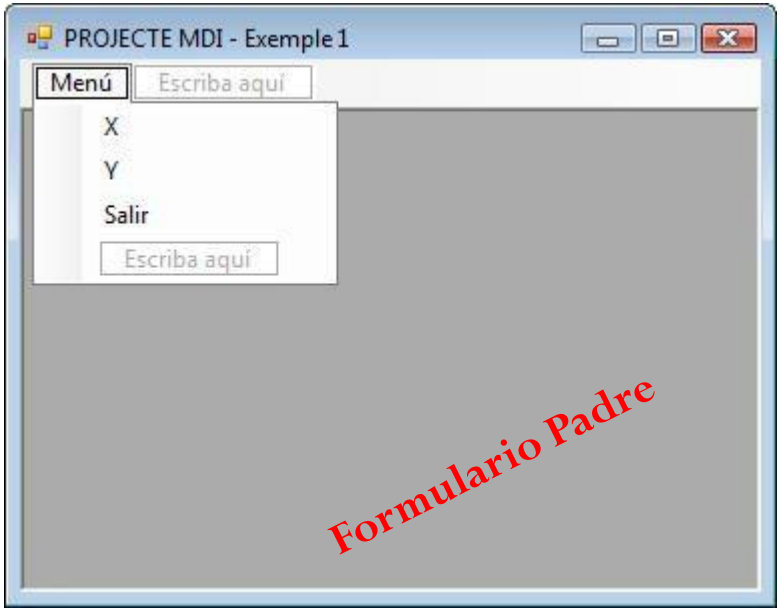
MDI

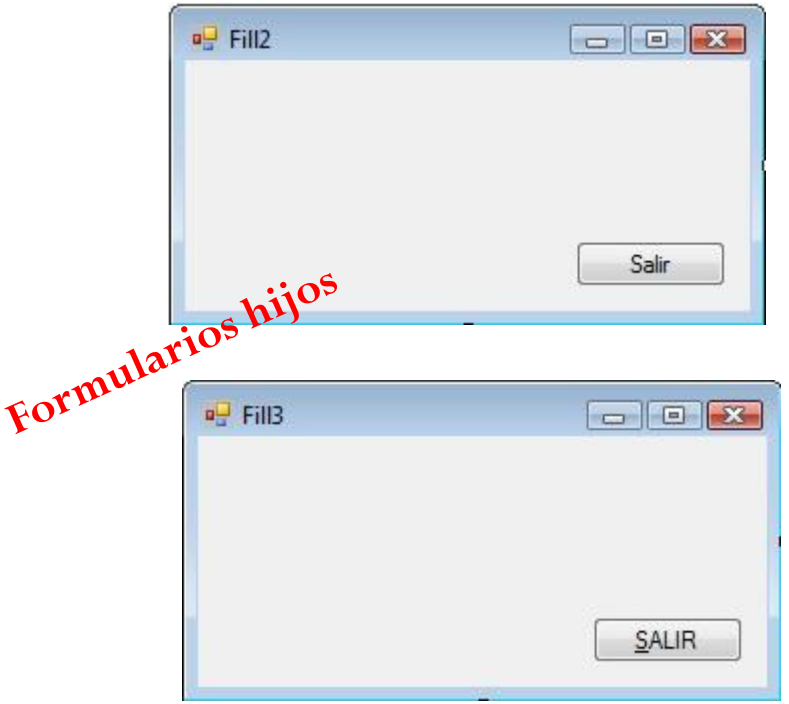
- Un proyecto MDI se basa en el diseño de un Formulario que actúa como 'base' para que, a partir de él, se vayan presentando la resta de formularios.
- Esta descripción de MDI nos llevará a utilizar/seguir la siguiente terminología:
  - Formulario PADRE.
  - Formularios HIJOS.
  - Otros tipos de formularios.

MDI

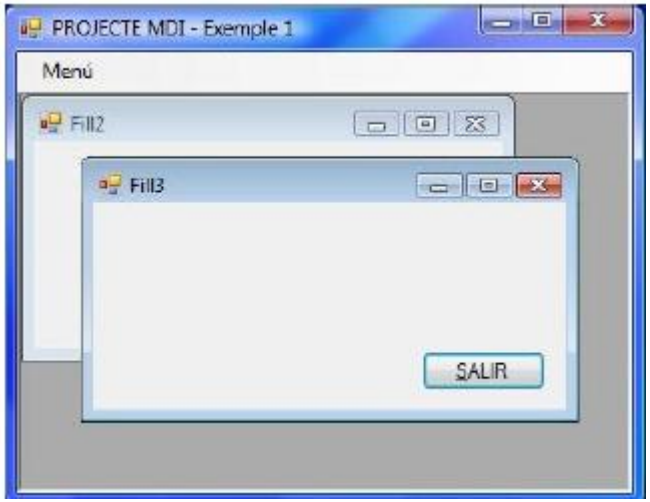
- El diseño y presentación del **formulario Padre,** suele seguir un modelo estándar que contiene algunos controles básicos:
  - Menú.
  - Barra/s de herramienta/s
  - Área cliente.
  - Barra de estado.
- Resulta evidente que el diseño de la interface de comunicación debe adaptarse:
  - Al tipo de dispositivo que realiza el proceso de interactuar visualmente ó no con el usuario de la aplicación, y
  - A la funcionalidad de la aplicación

- Para estudiar la estrategia y gestión de los proyectos MDI, vamos a emplear unos ejercicios prácticos comentados.



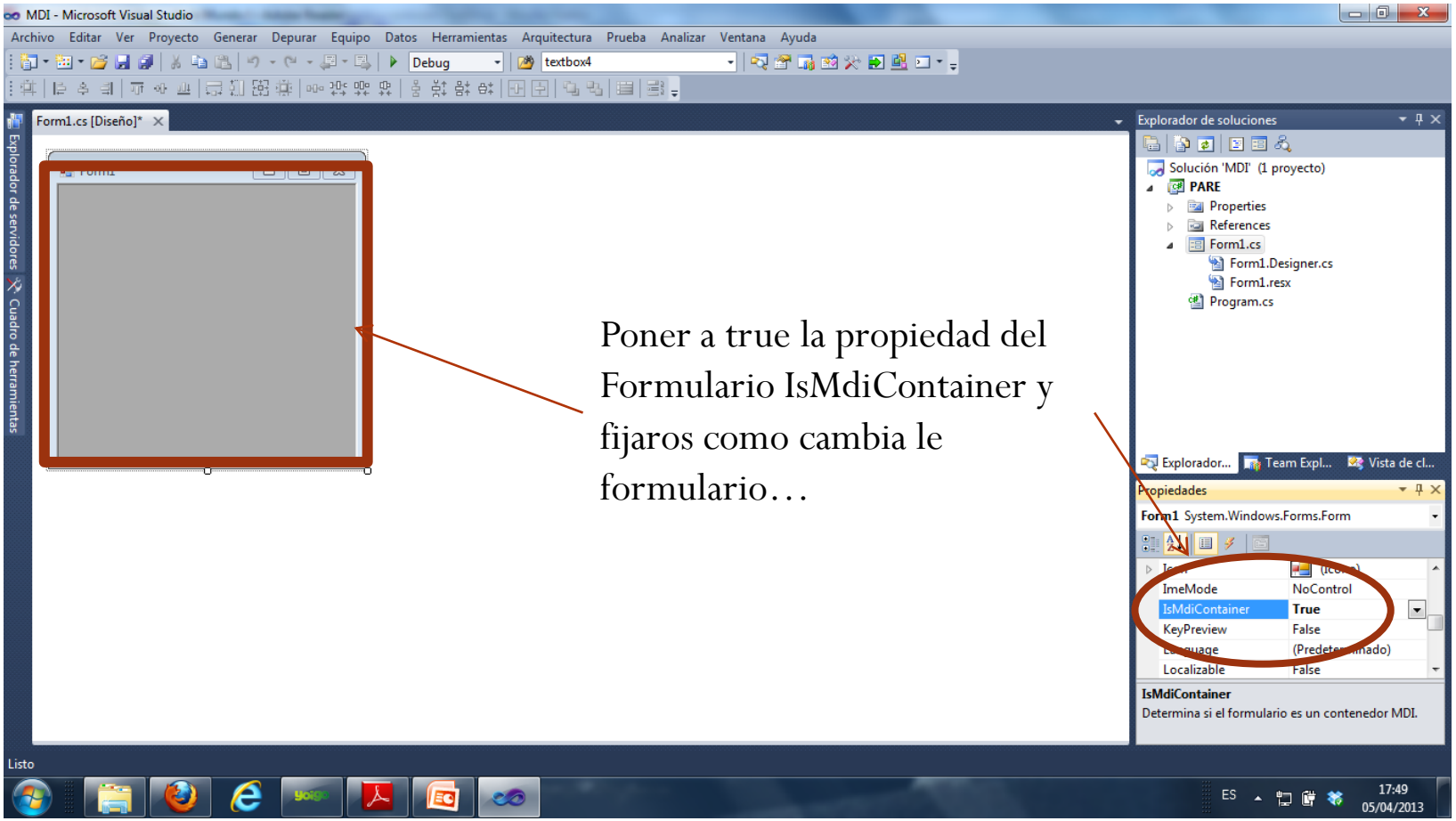


- Veamos el proyecto en ejecución

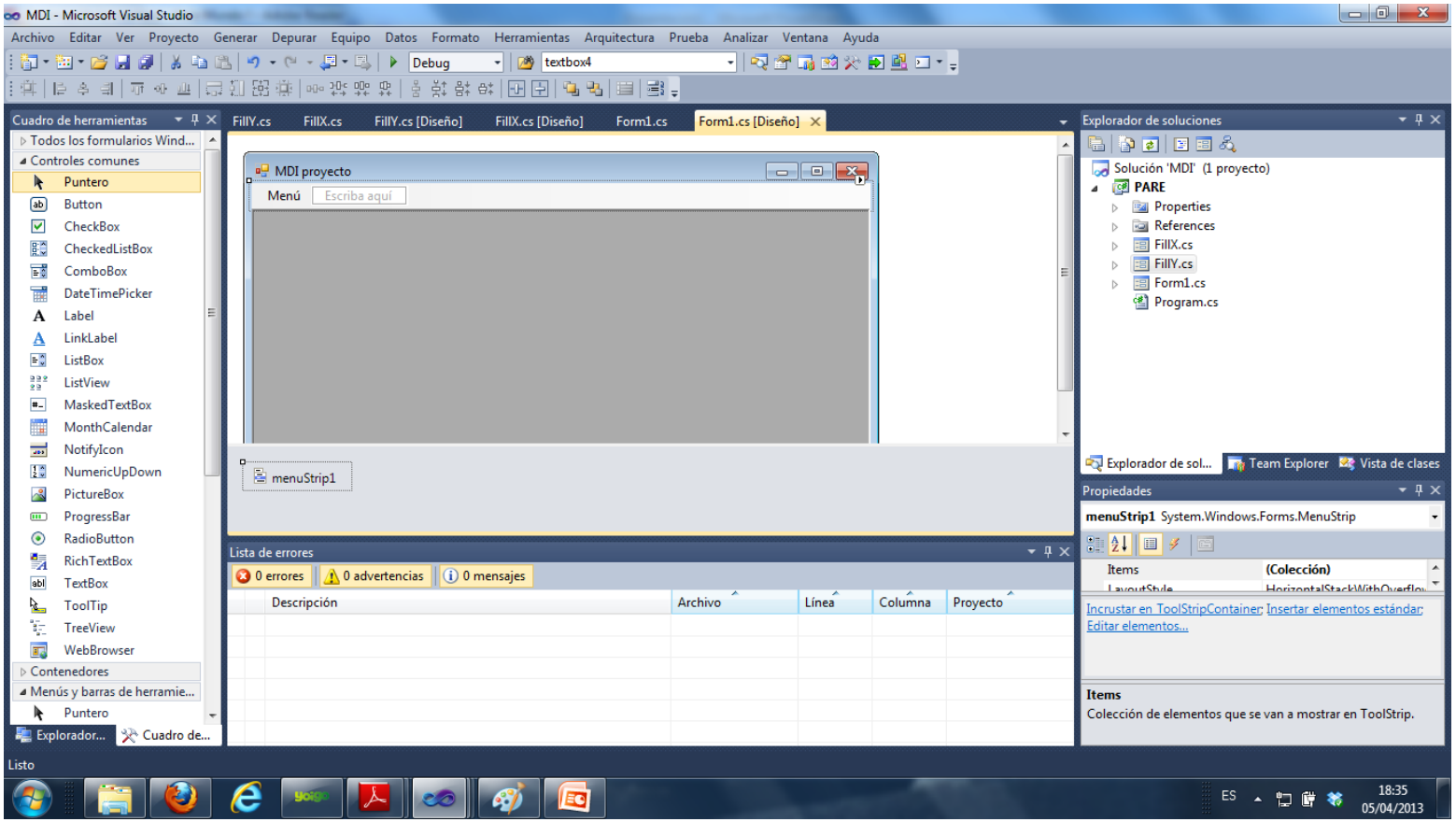


Crear un formulario MDI con combinación de menús y controles ToolStrip

- A continuación vamos a presentar el código fuente de cada uno de los formularios-Proyectos que integran la Solución:



- Añadimos el menú y otros formularios, FillX y FillY



```
private void xToolStripMenuItem_Click(object sender, EventArgs e)
{
    FillX meufill = new FillX();
    meufill.MdiParent = this; // Mi padre es this -> Pare2
    meufill.Show(); // Presentar el formulario
}

private void yToolStripMenuItem_Click(object sender, EventArgs e)
{
    FillY meufill = new FillY();
    meufill.MdiParent = this; // Mi padre es this -> Pare2
    meufill.Show(); // Presentar el formulario
}
```

Si analizamos detenidamente su contenido, interpretamos lo siguiente:

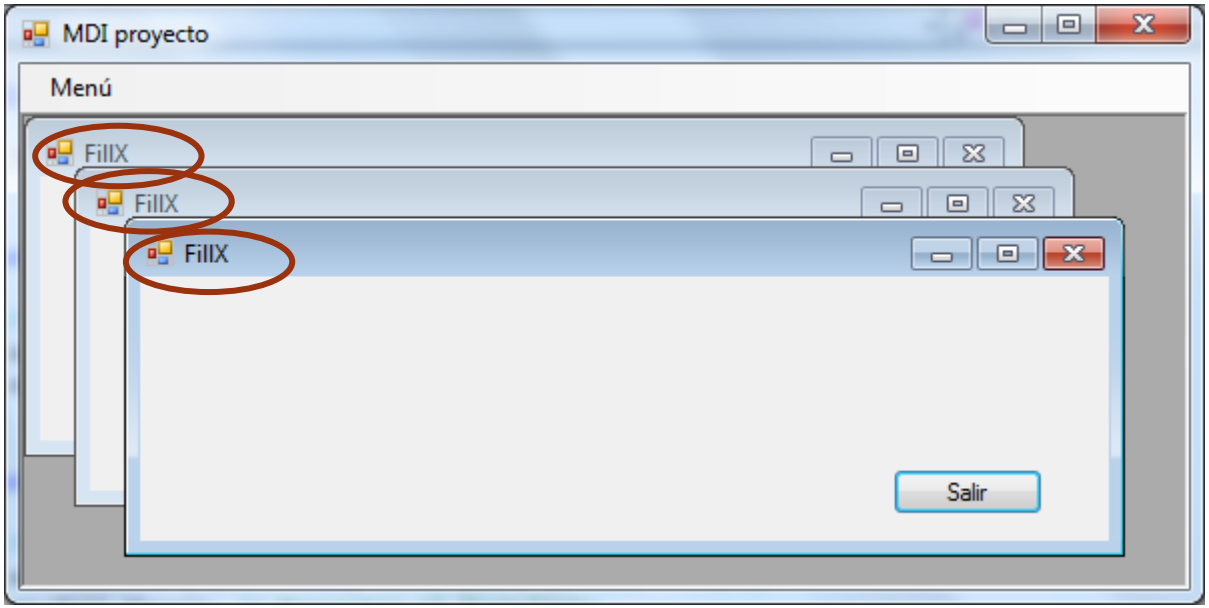
o Se crea un objeto llamado meufill a partir de la clase FillX (*que se corresponde con un formulario hijo).*

o Se emplea la propiedad MdiParent para decirle a este objeto (*que es un formulario) quien es su Formulario Padre.*

o Se ejecuta el método Show() del objeto meufill, cuyo resultado es la presentación sobre el formulario Padre del formulario Hijo (*meufill).*

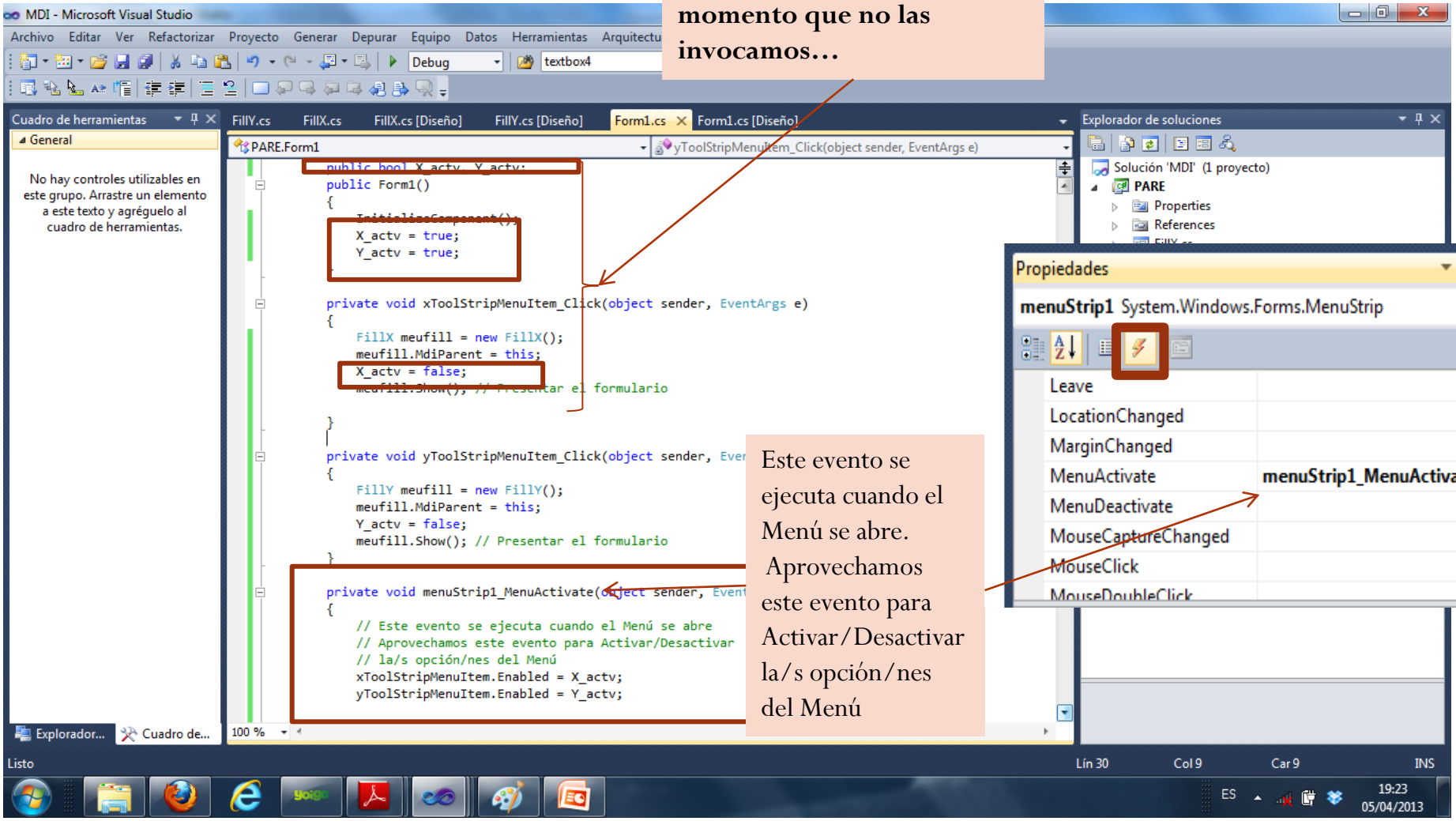
LO MISMO PARA FillY.

- Si os fijáis en la ejecución del ejemplo podéis invocar tantas veces como queráis cualquiera de las opciones, la opción del menú se encuentra operativa, cosa que no es muy recomendable…



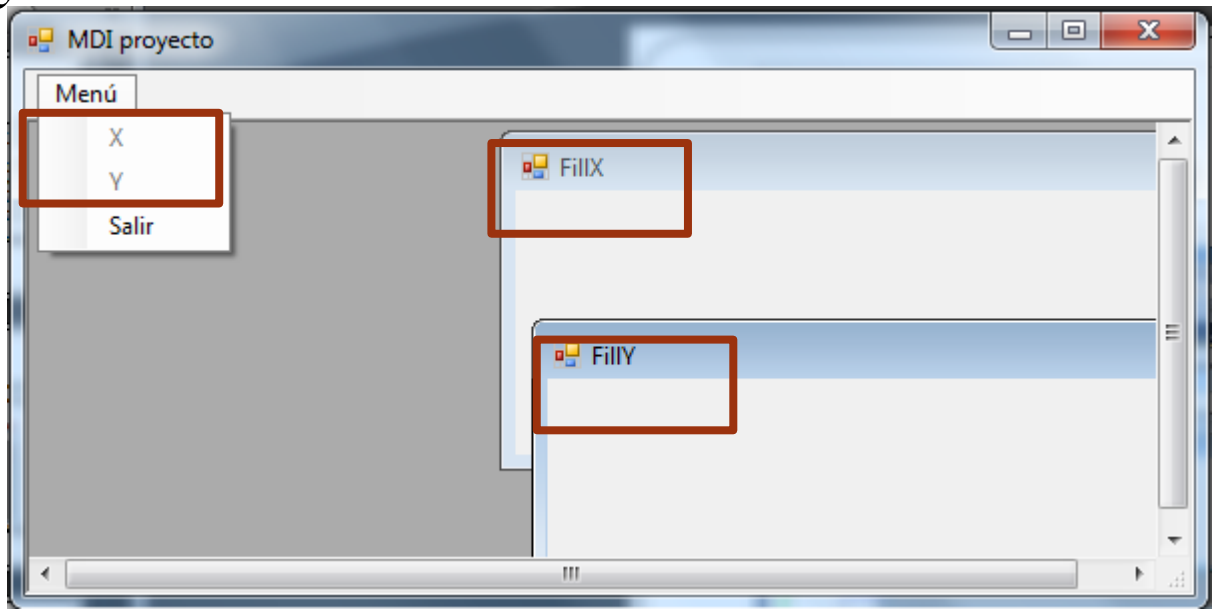
- Veamos como solucionarlo utilizando las técnicas de menú.

Creamos las variables X_actv e Y_actv las inicializamos y dentro de lacad opción de menú las ponemos a false para desactivarlas en el momento que no las invocamos…



Este evento se ejecuta cuando el Menú se abre. Aprovechamos este evento para Activar/Desactivar la/s opción/nes del Menú

Ahora están Desactivados



- … y ahora hay que hacer el efecto contrario cuando cerramos el hijo que se active la opción de menú.
- Tendremos que hacer que interactúen padre e hijo…
- Entonces haremos:
  1. Instanciar al padre desde el hijo
  2. Sobrecargar el constructor del hijo para pasarle el objeto padre…
  3. Solo nos queda cuando cerramos el formulario hijo activar las opciones del menú..
- Mirar las siguientes imágenes:

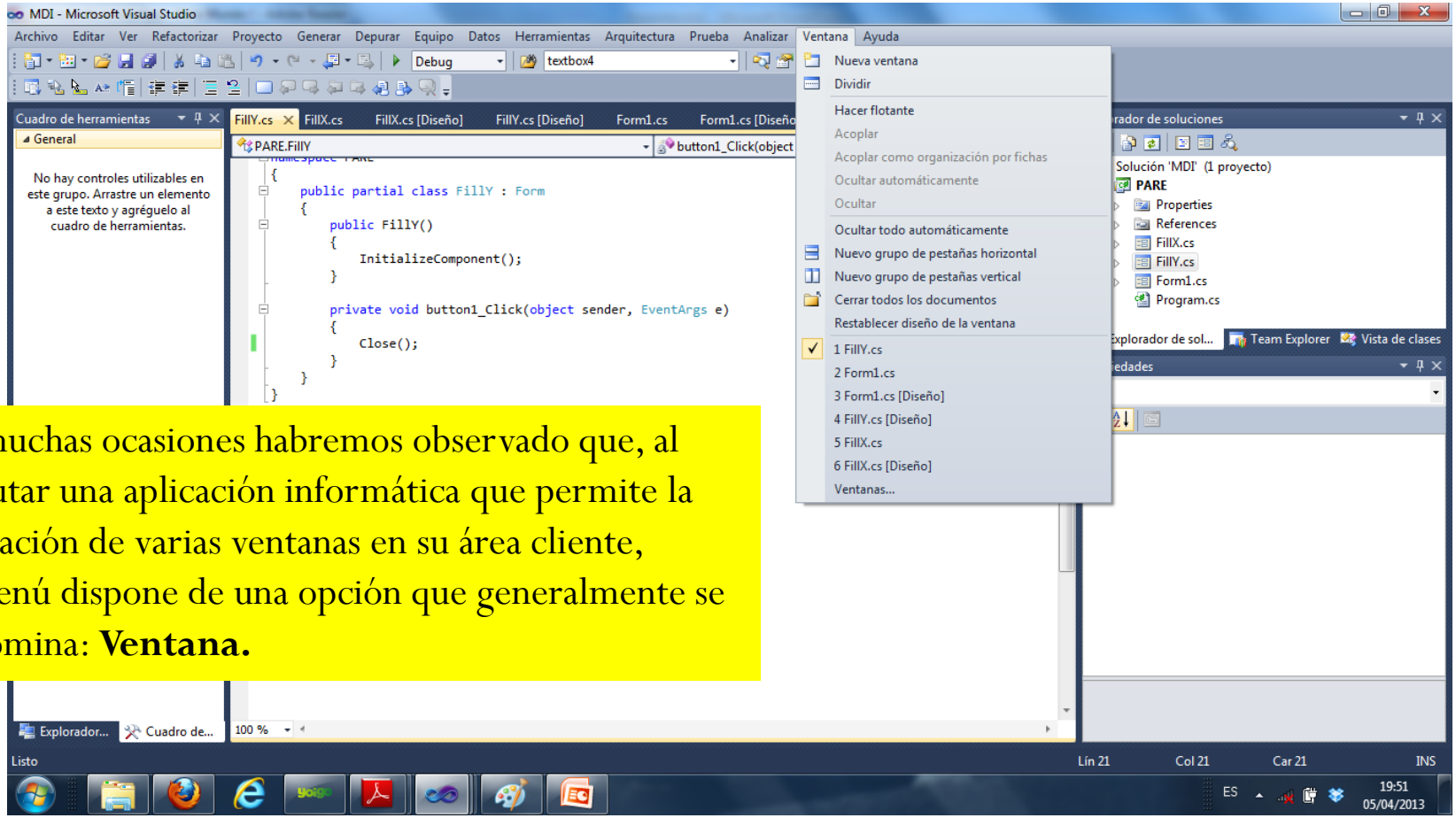
```
// INSTANCIA DE LA CLASE: PARE
Pare meupare;
```

```
// METODO CONSTRUCTOR SOBRECARGADO
public Fill_X(Pare m)
{
    InitializeComponent();

    // Aquí meupare ya contiene el valor del
    // parámetro recibido de tipo Pare
    meupare = m;
}
```

```
// Durante el proceso de cierrre del formulario
private void Fill_X_FormClosing(object sender, FormClosingEventArgs e)
{
    // Al cerrar el Formulario actualizamos el valor de la variable
    // para que al acceder al Menú, la opción esté activa.
    meupare.X_actiu = true;
}
```

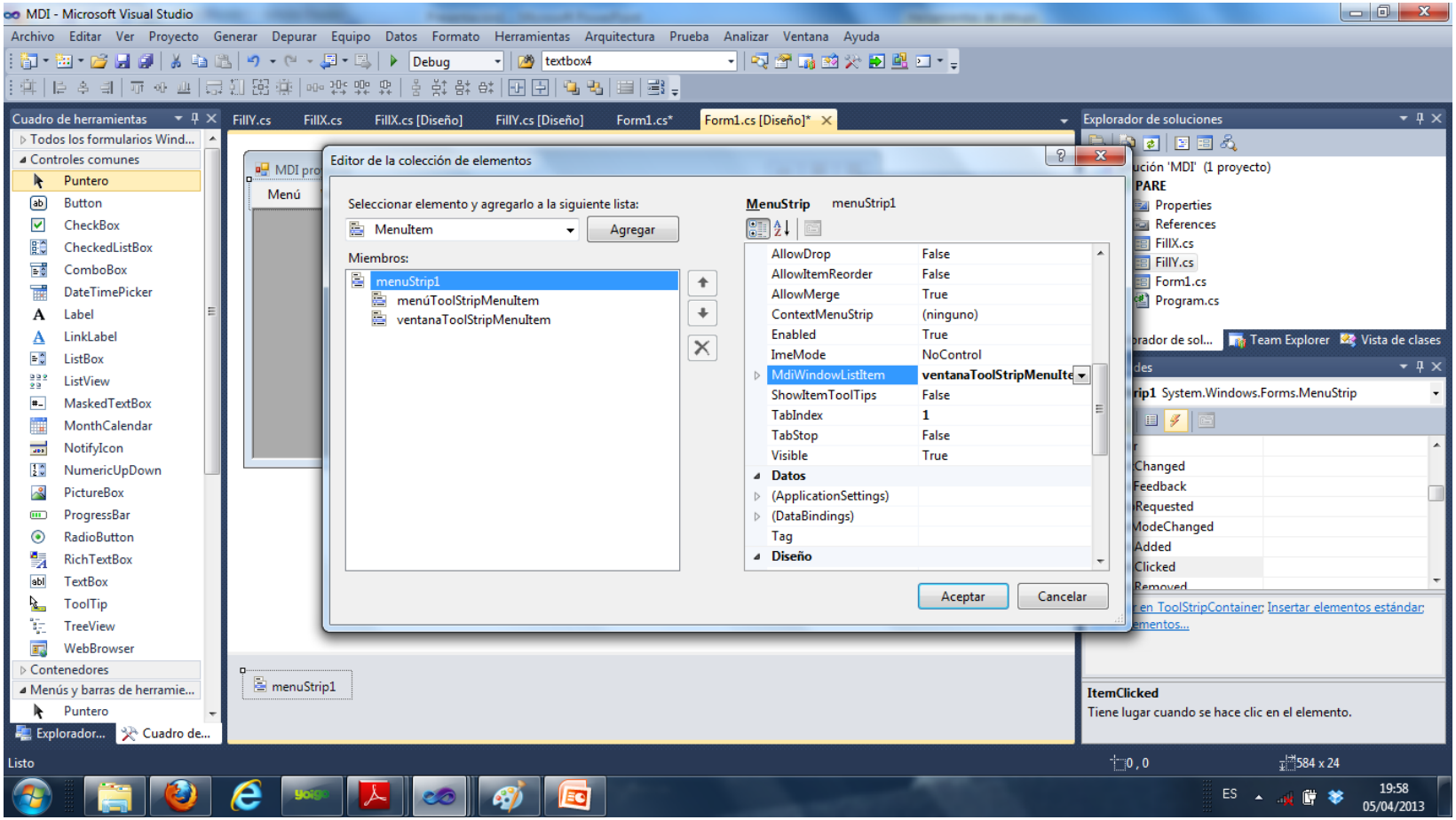
Gestión de Ventanas



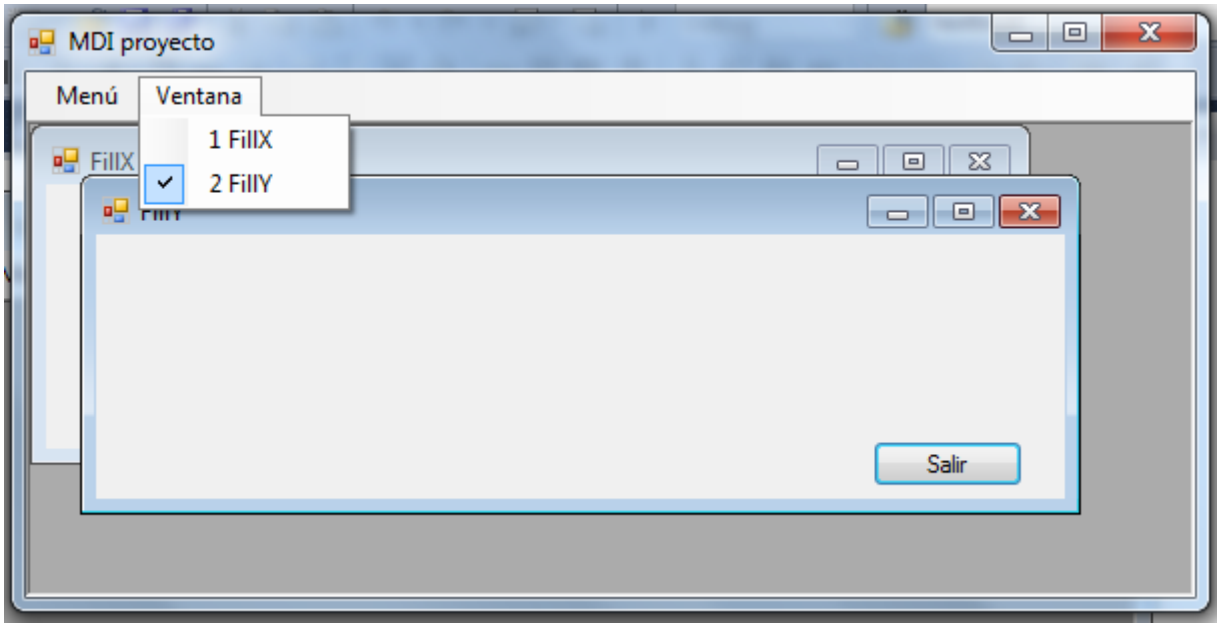
En muchas ocasiones habremos observado que, al ejecutar una aplicación informática que permite la activación de varias ventanas en su área cliente, el menú dispone de una opción que generalmente se denomina: **Ventana.**

Gestión de Ventanas

- Implementar esta funcionalidad desde los controles que gestionan menús es sencillo.
- Todos estos controles disponen de una propiedad en el control de tipo Menú que utilizaremos para indicar cuál es la Opción del Menú que utilizaremos para gestionar la lista de las ventanas Hijas abiertas (*creadas y mostradas)*.
- Para el control MenuStrip, esta propiedad será:
  - **MDIWindowListItem**
- Además, al elemento de Menú generalmente se le suele asignar un valor numérico alto (*Ej. 99) en su propiedad* MergeIndex para que siempre sea mostrado el último a la derecha en la representación visual del Menú.

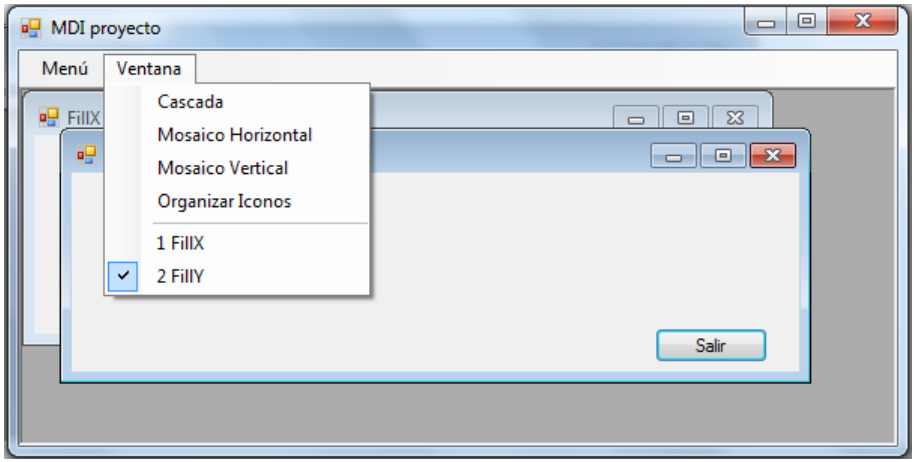

MDI - Microsoft Visual Studio
Archivo Editar Ver Proyecto Generar Depurar Equipo Datos Formato Herramientas Arquitectura Prueba Analizar Ventana Ayuda
Debug
textbox4
Cuadro de herramientas
Todos los formularios Wind...
Controles comunes
Puntero
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Contenedores
Menús y barras de herramie...
Puntero
Explorador...
Cuadro de...
FillY.cs
FillX.cs
FillX.cs [Diseño]
FillY.cs [Diseño]
Form1.cs*
Form1.cs [Diseño]*
MDI pro
Menú
Editor de la colección de elementos
Seleccionar elemento y agregarlo a la siguiente lista:
MenuItem
Agregar
Miembros:
menuStrip1
menúToolStripMenuItem
ventanaToolStripMenuItem
MenuStrip menuStrip1
AllowDrop False
AllowItemReorder False
AllowMerge True
ContextMenuStrip (ninguno)
Enabled True
ImeMode NoControl
MdiWindowListItem ventanaToolStripMenuIte
ShowItemToolTips False
TabIndex 1
TabStop False
Visible True
Datos
(ApplicationSettings)
(DataBindings)
Tag
Diseño
Aceptar
Cancelar
menuStrip1
Explorador de soluciones
ución 'MDI' (1 proyecto)
PARE
Properties
References
FillX.cs
FillY.cs
Form1.cs
Program.cs
Team Explorer
Vista de clases
rip1 System.Windows.Forms.MenuStrip
Changed
Feedback
Requested
ModeChanged
Added
Clicked
Removed
ItemClicked
Tiene lugar cuando se hace clic en el elemento.
Listo
0 , 0
584 x 24
ES
19:58
05/04/2013

- … y sin más al ejecutar tendremos:



DISPOSICIÓN DE LAS VENTANAS HIJAS.

- Otra funcionalidad que generalmente se suele emplear en la gestión de las Ventanas Hijas, es facilitar al usuario final la posibilidad de manejar la disposición de éstas sobre el área cliente del Formulario Padre.
- El proceso de configuración es el siguiente:
  - Una vez que hemos definido la propiedad **MdiWindowListItem** para la gestión de la lista de ventanas…
- En el menú: *Ventana, añadimos los elementos Menu:*
  - ToolStripMenuItem cuyas propiedades Text contengan los siguientes valores:
    - Cascada
    - Mosaico Horizontal
    - Mosaico Vertical
    - Organizar iconos



- A continuación, puede utilizar el método **LayoutMdi al que** pasaremos como parámetro uno de los valores de la ***enumeración MdiLayout.***
  - ***Cascade.***
  - ***TileHorizontal.***
  - ***Tile vertical***
  - ***ArrangeIcons***
- Este método lo emplearemos en el Formulario Padre (*formulario primario MDI) para reorganizar los formularios* secundarios.
- Para finalizar, escribiremos el contenido del evento que responde a la pulsación de cada una de estas opciones; por ejemplo:

```
private void cascadaToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.Cascade);
}

private void mosaicoHorizontalToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileHorizontal);
}

private void mosaicoVerticalToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}

private void organizarIconosToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.ArrangeIcons);
}
```