DESARROLLO DE APLICACIONES WINDOWS

Alonso Morales Salazar
MCP, MCAD.NET

Microsoft .net

Microsoft .net Framework Windows® Forms

Temas:

- Entender el concepto de clase
- Trabajar con clases
- Uso de miembros compartidos
- Herencia, polimorfismo y espacios de nombres

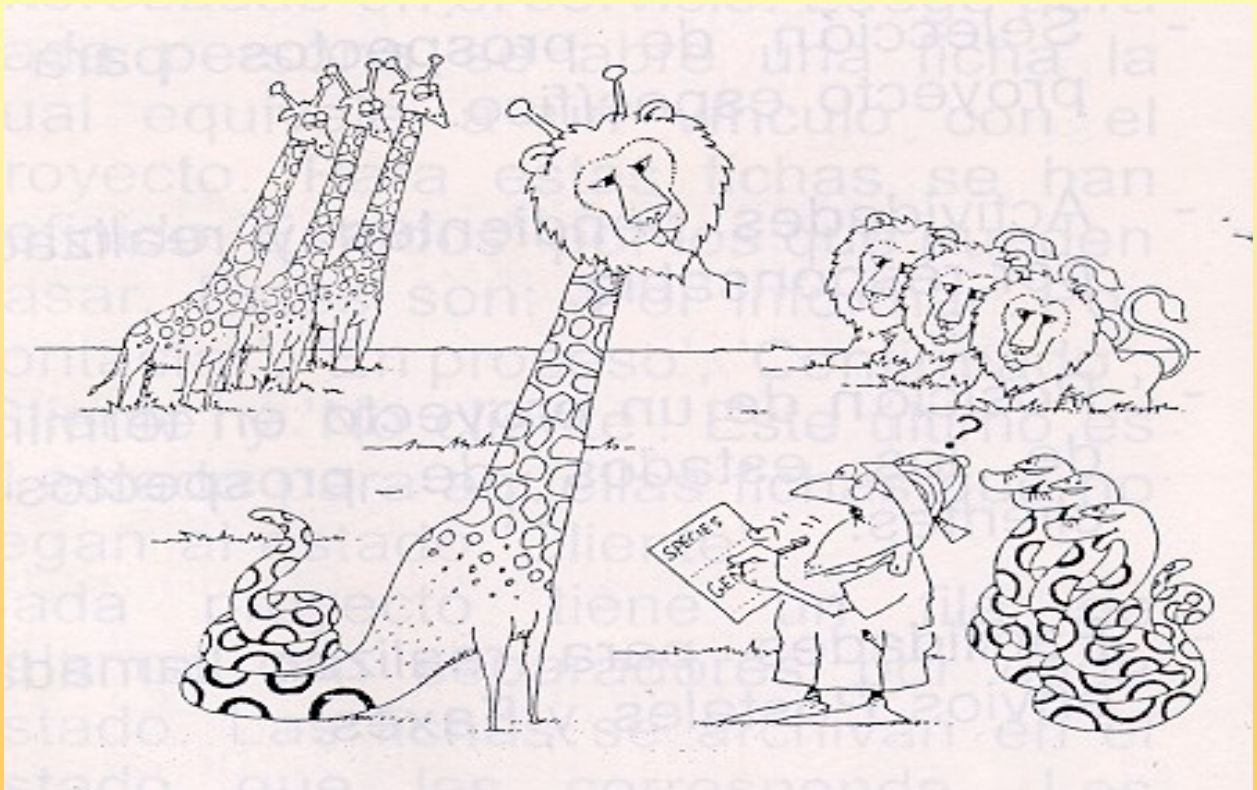
Percibiendo el mundo real

Las personas empleamos tres modalidades de organizar nuestras experiencias:

1. OBJETO – ATRIBUTO
   (Árbol: Dimensión, Colores, Ubicación)
2. OBJETO – COMPONENTES
   (Árbol: Rama, Raíz, Tronco)
3. OBJETO- CLASE
   (Árbol: Pino)

ESTOS CONCEPTOS LOS UTILIZAMOS DESDE LOS PRIMEROS DIAS DE NACIDOS.

LA CLASIFICACIÓN ES EL MEDIO POR EL CUAL ORGANIZAMOS EL CONOCIMIENTO

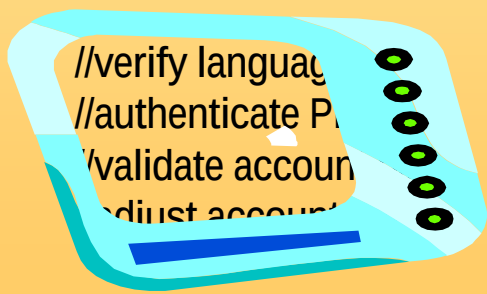


¿Qué es una clase?

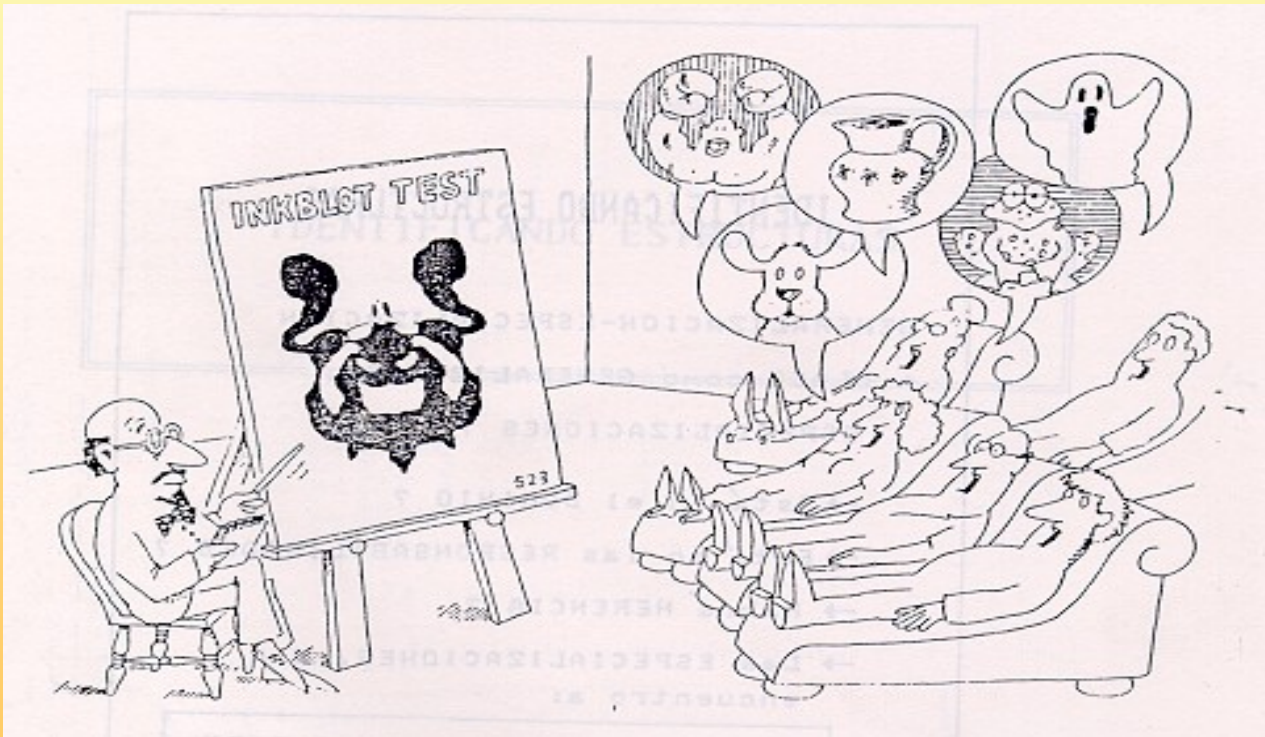
- Una *clase* es una estructura preliminar que describe un objeto y define atributos y operaciones para el objeto
- Las clases utilizan *abstracción* para poner a disposición únicamente los elementos esenciales que definen el objeto
- Las clases utilizan *encapsulación* para garantizar que se cumple una abstracción

Lo que ve el usuario:

Lo que está encapsulado:



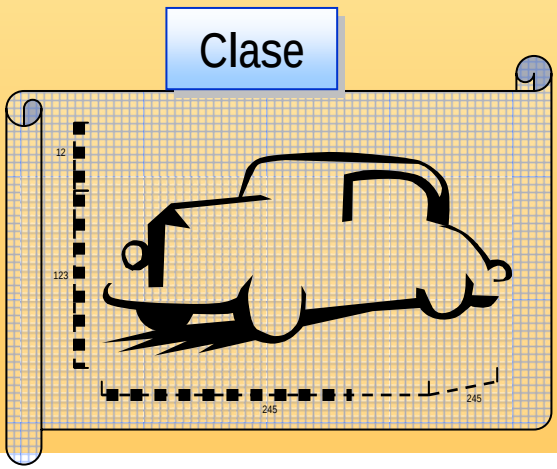
Diferentes observadores pueden clasificar el mismo objeto de diferentes maneras

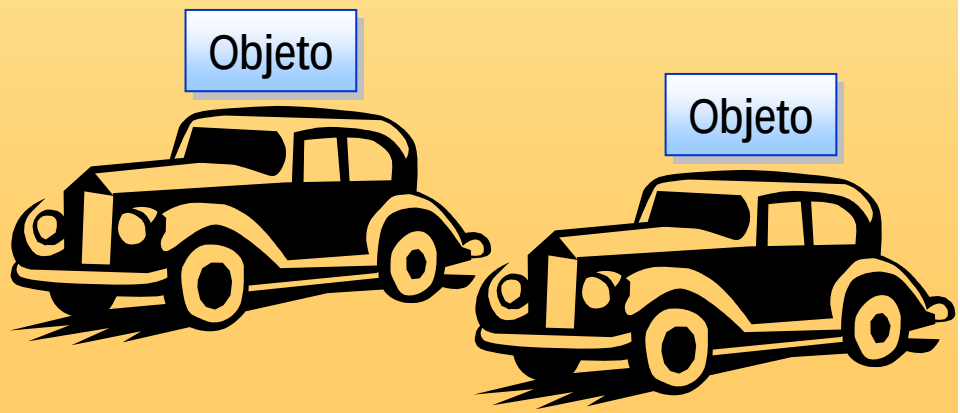


Clases y objetos deberían estar en el correcto nivel de abstracción: ni muy alto ni muy bajo

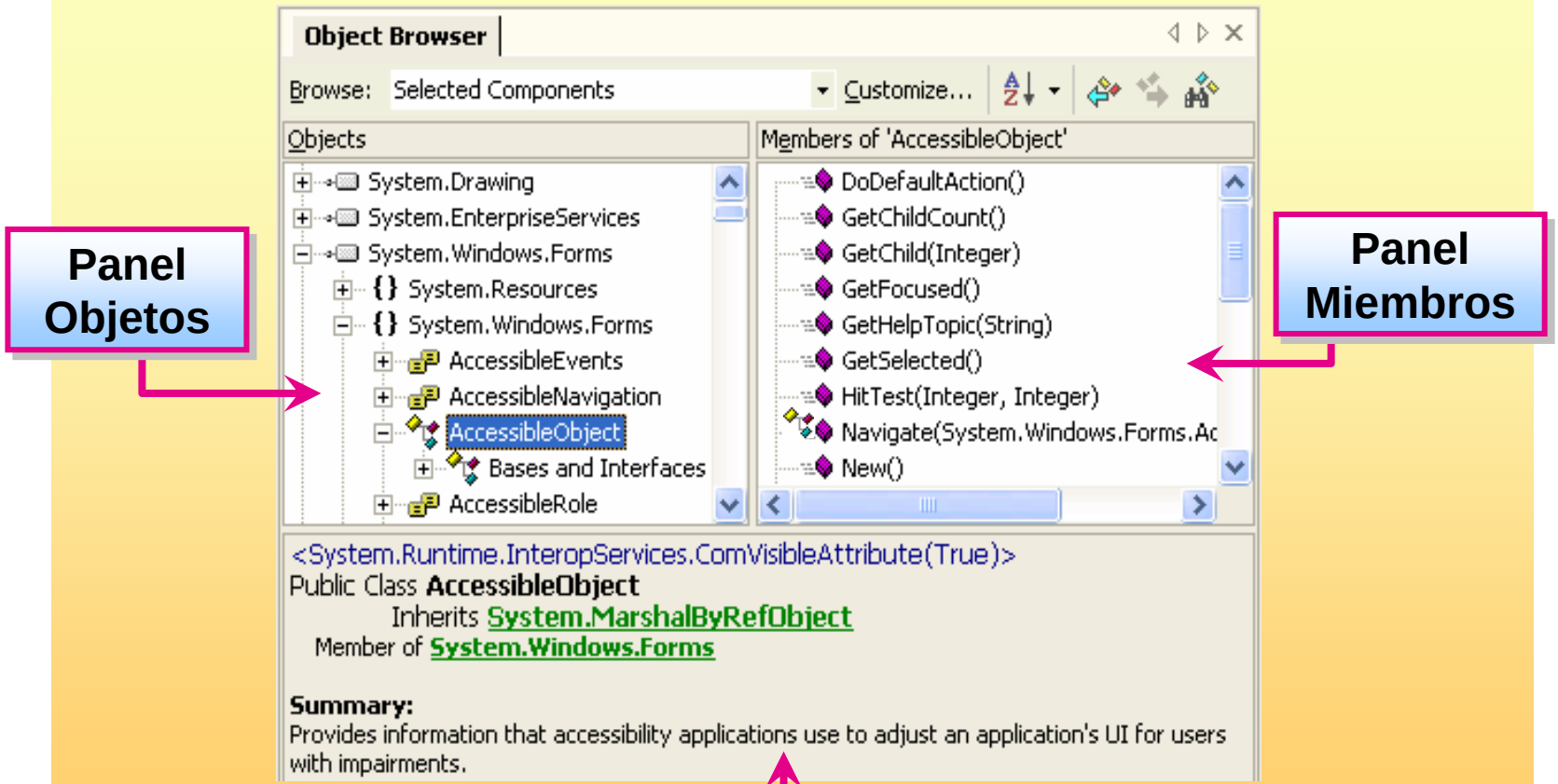
¿Qué es un objeto?

- Un objeto es una instancia de una clase
- Los objetos tienen las siguientes cualidades:
  - Identidad: los objetos se distinguen uno de otro
  - Comportamiento: los objetos pueden realizar tareas
  - Estado: los objetos almacenan información que puede cambiar con el tiempo





Cómo utilizar el Examinador de objetos



Cómo crear una nueva clase

- Crear una nueva clase utilizando el comando Agregar clase del menú Proyecto
- Ejemplo de una nueva clase denominada BankAccount:

```
Public Class BankAccount

End Class
```

Cómo agregar miembros de datos de una instancia

- Agregar un miembro de datos denominado *balance*

```
Public Class BankAccount
  Private balance As Double

End Class
```

Palabra clave	Definición
Public	Accesible en todas partes
Private	Accesible sólo en el propio tipo
Protected	Accesible sólo por clases que heredan de la clase

Cómo agregar métodos

- Agregar un método denominado Deposit

```
Public Class BankAccount

   Private balance As Double

   Public Sub Deposit(ByVal amount As Double)
       balance += amount
   End Sub

End Class
```

- Métodos sobrecargados: dos o más métodos con el mismo nombre pero diferentes firmas

  Ejemplo: MessageBox.Show

Cómo agregar propiedades

- Agregar una propiedad:

```
Public Class BankAccount
    Private customerName As String

    Public Property Name( ) As String
        Get
            Return customerName
        End Get
        Set(ByVal Value As String)
            customerName = Value
        End Set
    End Property

End Class
```

Cómo crear una instancia de una clase

- Uso de la palabra clave New para crear una instancia de la clase BankAccount:

```
Module Bank

   Sub Main
       Dim account As New BankAccount( )
       account.Deposit(500.00)
   End Sub

End Module
```

Creación de una clase BankAccount

demo

Cómo utilizar los constructores

- Ejecutan código cuando el objeto está instanciado

```
Public Sub New( )
    ' Perform simple initialization
    value = 1
End Sub
```

- Pueden sobrecargarse, pero no utilizan la palabra clave Overloads

```
Public Sub New(ByVal i As Integer)
   ' Perform more complex initialization
    value = i
End Sub
```

Cómo utilizar los destructores

- Se utilizan para eliminar recursos
- Se invocan por el entorno de ejecución antes de destruir el objeto
  - Importante: es posible que la destrucción no se produzca de modo inmediato

```
Protected Overrides Sub Finalize( )
    ' Can close connections or other resources
    conn.Close
End Sub
```

Lección: Uso de miembros compartidos

- Cómo utilizar miembros de datos compartidos
- Cómo utilizar métodos compartidos

Cómo utilizar los miembros de datos compartidos

- Los miembros de datos compartidos permiten que múltiples instancias hagan referencia a una única variable de nivel de clase

```
Class SavingsAccount
    Public Shared InterestRate As Double
    Public Name As String, Balance As Double
. . .

End Class
```

```
SavingsAccount.InterestRate = 0.03
```

Cómo utilizar los métodos compartidos

- Pueden utilizarse sin declarar una instancia de una clase
- Únicamente pueden acceder a datos compartidos

```
' TestClass code
Public Shared Function GetComputerName( ) As String
    ...
End Function
```

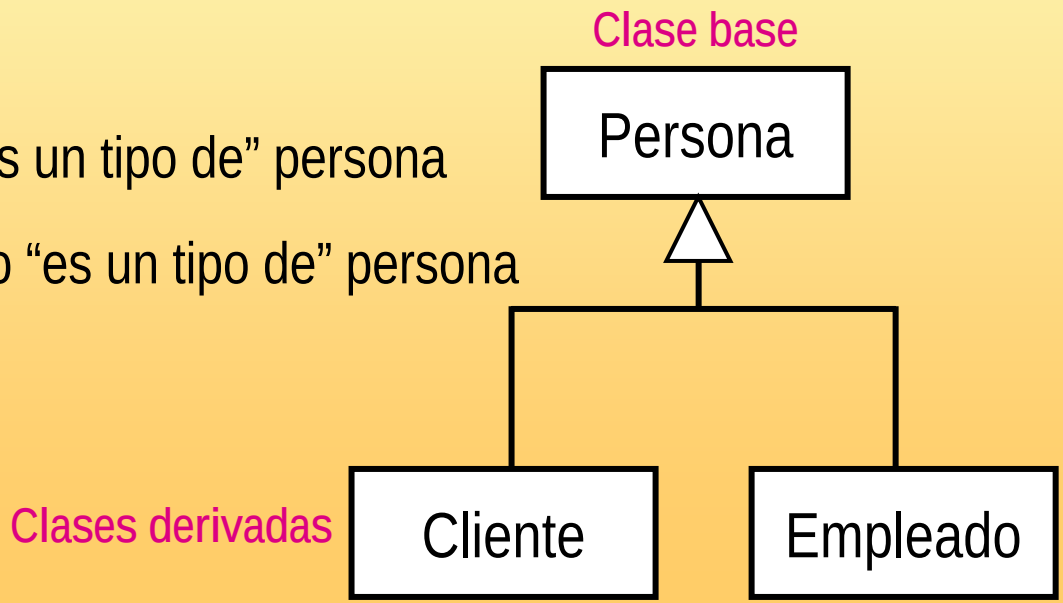
```
' Client code

MessageBox.Show(TestClass.GetComputerName( ))
```

Creación de métodos compartidos

demo

¿Qué es la herencia?

- La herencia especifica una relación “es un tipo de”
- Múltiples clases comparten los mismos atributos y operaciones, permitiendo una eficaz reutilización del código
- Ejemplos:
  - Un cliente “es un tipo de” persona
  - Un empleado “es un tipo de” persona

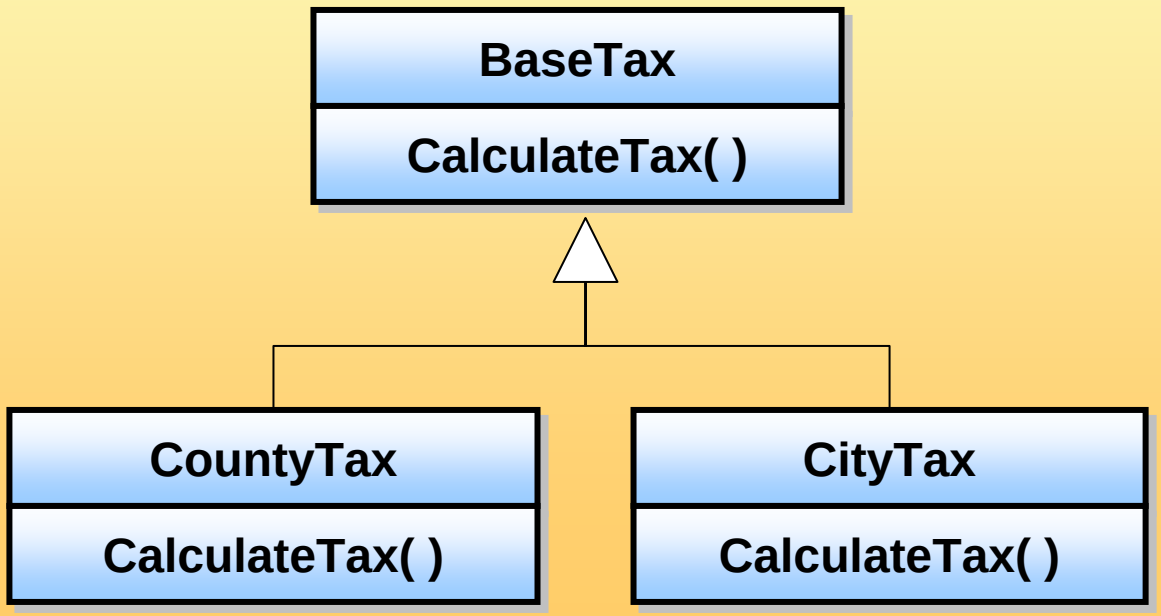


Cómo heredar de una clase

- Una clase derivada hereda de una clase base
- Pueden heredarse propiedades, métodos, miembros de datos, eventos y controladores de eventos (dependiendo del ámbito)
- Palabras clave
  - **Inherits**: hereda de una clase base
  - **NotInheritable**: no es heredable
  - **MustInherit**: no pueden crearse instancias de la clase; debe ser heredada como una clase base

¿Qué es el polimorfismo?

- El nombre del método reside en la clase base
- Las implementaciones del método residen en las clases derivadas

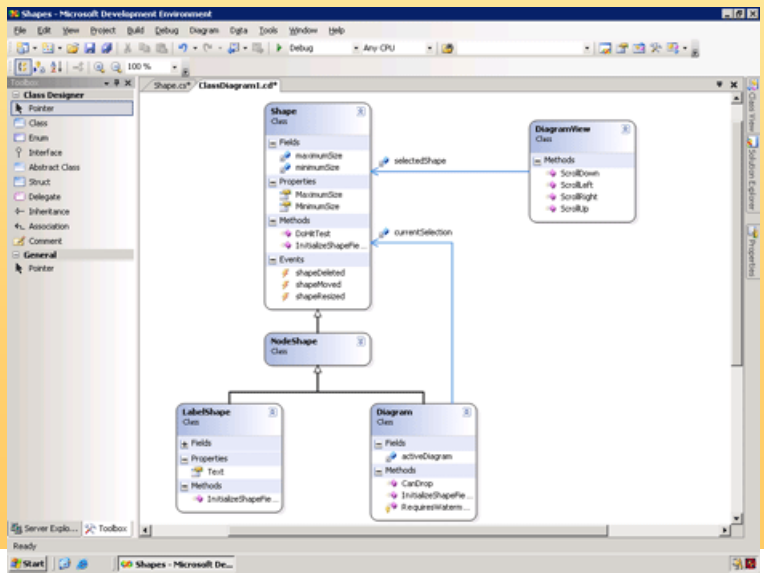


Comparación entre clases y estructuras

Clases	Estructuras
Pueden definir miembros de datos, propiedades y métodos	Pueden definir miembros de datos, propiedades y métodos
Soportan constructores e inicialización de miembros	Sin constructor predeterminado ni inicialización de miembros
Soportan el método **Finalize**	No soportan el método **Finalize**
Extensibles por herencia	No soportan herencia
Tipo Referencia	Tipo Valor

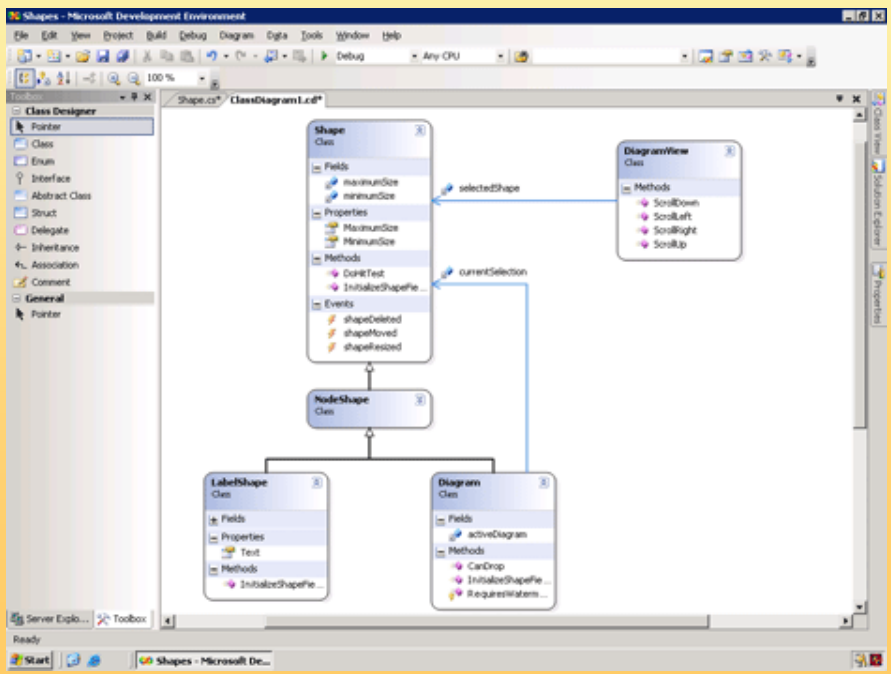
Diseñador de Clases de VS 2005

El diseñador de clases de Visual Studio es una herramienta de diseño de código visual integrada en .NET Framework. La experiencia visual del diseñador de clases está estrechamente relacionada con la de Common Language Runtime. Las formas CLR, como clases, estructuras e interfaces, están representadas por formas visualmente distintas, las cuales indican su identidad.



Razones para usar un diseñador de clases

- El diseño de software es una tarea ardua y compleja.
- Es necesario afrontar numerosos retos durante todo el ciclo de desarrollo, desde las fases tempranas de diseño y las revisiones de código, hasta la documentación del producto final.



Razones para usar un diseñador de clases

Un diseñador de clases visual puede ser una herramienta de gran utilidad durante todo el ciclo de desarrollo, como se muestra en los escenarios siguientes:

- **Comprensión del código existente:** Las bases de código existente pueden resultar complicadas y confusas. El uso de un diseñador de clases visual permite explorar gráficamente las jerarquías de clases existentes y comprender las relaciones establecidas entre las mismas.
- **Diseño de clases:** Un diseñador de clases visual permite crear gráficamente el diseño y la implementación de alto nivel del software.
- **Revisión y refactorización de código:** Un diseñador de clases visual constituye una herramienta eficaz para llevar a cabo revisiones y refactorizaciones. El diseñador permite anotar los diagramas de código existente para su revisión y facilita la refactorización de código, lo que conlleva un ahorro de tiempo considerable.
- **Diagramas de clases para documentación:** Los diagramas de clases se pueden utilizar para documentar las jerarquías de clases existentes. Los árboles de jerarquía se pueden visualizar gráficamente. Los diagramas de clases también resultan útiles para la comunicación de ideas entre colegas, a través de correo electrónico o presentaciones visuales.

Diseñador de clases

demo

Cómo organizar clases en espacios de nombres

- Los espacios de nombres son un sistema organizativo
- Los espacios de nombres proporcionan nombres cualificados para las clases
  - Ejemplo: System.Windows.Forms.Button
- Para importar un espacio de nombres:
  - A nivel de proyecto, agregar una referencia a la DLL que contiene el espacio de nombres
  - Utilizar la palabra clave **Imports**

Cómo organizar clases en espacios de nombres

- Los espacios de nombres son un sistema organizativo
- Los espacios de nombres proporcionan nombres cualificados para las clases
  - Ejemplo: System.Windows.Forms.Button
- Para importar un espacio de nombres:
  - A nivel de proyecto, agregar una referencia a la DLL que contiene el espacio de nombres
  - Utilizar la palabra clave **Imports**

Cómo organizar clases en espacios de nombres

- Los espacios de nombres son un sistema organizativo
- Los espacios de nombres proporcionan nombres cualificados para las clases
  - Ejemplo: System.Windows.Forms.Button
- Para importar un espacio de nombres:
  - A nivel de proyecto, agregar una referencia a la DLL que contiene el espacio de nombres
  - Utilizar la palabra clave **Imports**

demo