

## VB.NET. Resúm del llenguatge







---

## Índex

INTRODUCCIÓ.....	4
CAPÍTOL IV : CLASSES I OBJECTES.....	5
CAPÍTOL VI : GESTIÓ D'EVENTS I DELEGACIÓ.....	13
CAPÍTOL VII : GESTIÓ D'ERRORS EN VB.NET: VIURE AMB LES EXCEPCIONS.....	15
CAPÍTOL VIII : WINDOWS FORMS. DIBUIX I IMPRESSIÓ.....	17
CAPÍTOL IX : ENTRADA / SORTIDA.....	22
CAPÍTOL X : THREADS.....	27
REMOTING.....	34
CAPÍTOL XI : BREU INTRODUCCIÓ AL ACCÉS A BASES DE DADES AMB VB.NET.....	36
CAPÍTOL XII : BREU REVISIÓ D'ASP.NET.....	43
CAPÍTOL XIII : ENSAMBLATS .NET. IMPLEMENTACIÓ I COM INTEROP.....	44
CONCEPTES BÀSICS DE LA PROGRAMACIÓ OO :.....	47
GLOSSARI.....	51
NOTES.....	56



---

## Introducció

### **Introduir comentaris a la 'Lista de tareas' :**

- **TODO** : Feina pendent.
- **HACK** : Revisió.
- **UNDONE** : Feina a desfer o obsoleta.

Es poden configurar paraules reservades per aquest efecte. A proposar : “!(REVISAR)”, “↓(DESFER)” i “(RECORDAR)”

23

---

### **Option Strict ON :**

Hauria d'estar sempre activat. Això ens garanteix que totes les variables i tots els paràmetres de funcions reben dades del tipus correcte.

---

### **Àmbit de variables i mètodes:**

**Public:** Es veu des de tot arreu.

**Private:** Es veu únicament des de la classe, mètode o bloc de codi on s'ha definit.

**Friend:** Es veu únicament des de classes del mateix ensamblat.

**Protected:** Son accessibles només des de la pròpia classe i les seves derivades.

---

### **Dispose :**

Dispose equival a :

A = `Nothing`

Però es programable, (es un mètode). S'ha de fer servir sempre.

**1** Veure : “Dispose i Finalize” del capítol IV.



---

## Capítol IV : Classes i objectes

### **MyClass:**

Referència a la pròpia classe, (a la classe membre i no a la instància concreta). Es fa servir sobretot per evitar duplicació de codi als constructors.

147 – 148

---

### **MyBase i Me :**

**MyBase** fa referència en tot moment a la classe pare de la classe actual.

**Me** fa referència a la classe actual. Encara que el flux estigui a una altra classe, les accions amb **Me** referenciaran a mètodes de la classe actual.

---

### **ReferenceEquals:**

Determina si dos declaracions, (variables), fan referència a la mateixa instància.

203

---

### **TypeName:**

Obté el nom de la classe a la que referència una variable.

```
Console.WriteLine(TypeName(x))
```

On x es una instància de qualsevol classe.

### **Type Of:**

Compara una variable amb una classe. Retorna **True** si la variable referència un objecte d'aquesta classe.

```
Console.WriteLine(typeof x Is Exemple)
```

Retorna **True** si la variable x es una instància de la classe Exemple.

137

---

### **GetType:**

Retorna el tipus d'objecte al que referència una variable.

objecte.**GetType**.toString(): Retorna el nom complert de la classe.

138

---



---

### CollectionBase:

Aquesta classe ens permet muntar llistes d'un únic tipus d'objecte. Es abstracta. A partir de l'espai de noms **{}Collection** podem implementar qualsevol de les llistes estàndards implementades. Piles, cues, arrais, llistes per clau, etc.

**1** Veure també : “Ilist” i “La classe ArrayList”.

199 – 200

---

### IEnumerator:

**IEnumerator** implementa la funció **MoveNext**; que es el mínim imprescindible per implementar una llista :

#### Exemple de recorregut amb For-Next :

```
Dim x As String = "Esto es un texto de prueba"  
Dim y() As String = x.Split(" ")  
Dim z As Integer  
For z = 0 To y.Length - 1  
    Console.WriteLine(y(z))  
Next z
```

#### Exemple de recorregut amb For-Each :

```
Dim subY As String  
For Each subY In y  
    Console.WriteLine(subY)  
Next
```

#### Exemple de recorregut amb IEnumerator :

```
Dim i As IEnumerator = y.GetEnumerator()  
While i.MoveNext  
    Console.WriteLine(i.Current())  
End While
```

---

### Ilist:

Implementa una llista bàsica. Una classe pot implementar la funcionalitat d'una llista amb aquesta interface.

**1** Veure també : “CollectionBase” del capítol VI.

229 – 230

---

### La classe ARRAYLIST:

La classe **ArrayList** substitueix a **Dim Preserve** de VB6.

**ArrayList.Add** : Afegeix un nou element.

**1** Veure: “Ilist”

#### Exemple d'ArrayList :

```
Dim llista As New ArrayList  
Dim t As Integer  
  
llista.Add("Cadena2")  
llista.Add("Cadena1")
```



```
llista.Add("Cadena3")

Console.WriteLine("Valor de la segona posició de l'array : " & CType(llista.Item(2),
String))

Console.WriteLine("Ordenació de l'array.")
llista.Sort()

For t = 0 To llista.Count - 1
    Console.WriteLine(t.ToString() & " --> " & CType(llista.Item(t), String))
Next
```

128 – 129

### HashTable:

Un tipus d'array que permet fer cerques per clau, (no per índex).

#### Exemple de HashTable :

```
Dim eVar As New Hashtable

eVar.Add("st1", "Cadena1")
eVar.Add("st2", "Cadena2")
eVar.Add("st3", "Cadena3")

Console.WriteLine("El HashTable conté " & eVar.Count.ToString() & " variables")

Console.WriteLine("Existeix 'st2'? : " & eVar.ContainsKey("st2").ToString())
Console.WriteLine("Cerquem 'st2' : " & CType(eVar.Item("st2"), String))

Console.WriteLine("Existeix 'st4'? : " & eVar.ContainsKey("st4").ToString())
```

132

### Sobrecàrrega, (Overloads):

No fer servir paràmetres a les funcions per fer coses diferents, (amb [Optional](#) per exemple). És millor fer servir sobrecàrrega, (amb [Overloads](#)).

**1** Veure : “Sobrecàrrega” a Conceptes bàsics de la programació OO

145

### Propietats semipúbliques:

Com fer que [Get](#) sigui públic i [Set](#) sigui privat?

```
Public ReadOnly Property getA...
...
Private WriteOnly Property setA...
...
```

### Default Property:

Marca una propietat com la propietat predeterminada de la classe. Però ha de tenir almenys un paràmetre.

```
Default Public ReadOnly Property propietat(ByVal index As Integer)
```

148 - 149



---

**Classes aniuades:**

```
Public Class a
    ...
    Friend Class b
        ...
    End Class
End Class
```

Una classe definida dins d'una altre i que en depen d'ella.

152

---

**Variables compartides, (estàtiques):**

**Shared** converteix una variable en compartida per a totes les instancies de la classe. Per exemple :

```
Private Shared a
Public ReadOnly pa
    Get
        Return a
    End Get
End Class
```

La propietat *pa* retorna el valor de la variable *a*, que es el mateix per totes les instancies de la classe.

155

1 Veure : “Constructors” a Conceptes bàsics de la programació OO

**Mètodes compartits:**

```
Public Shared Sub x
```

Aquest mètode pot executar-se sense necessitat de que existeixi cap instància viva de la classe.

157

**Constructors compartits:**

- No poden tenir paràmetres.
- Només es criden amb la primera instanciació de la classe. I abans del constructor de la pròpia instància.
- S'ha de fer servir només per inicialitzar variables compartides.

158

---

**Dispose i Finalize:**

No fer servir **Finalize**.

**Dispose** no allibera memòria de l'objecte. Ha de fer-se servir per tancar connexions o altres elements no relacionats amb la pròpia instància.

1 Veure també : “Clone i Dispose”

**El recol·lector d'escombraries :**

El recol·lector es un sistema automàtic que allibera memòria eliminant els objectes que ja no son referenciats per cap variable del nostre programa. Es a dir, que son inabastables per la nostra aplicació.

El recol·lector d'escombraries s'activa automàticament cada cop que el sistema precisa de mes memòria. Però nosaltres el podem invocar explícitament amb l'objecte **gc**. La crida **gc.Collect()** executa el recol·lector.

160



---

### Name Space:

```
Namespace a
  Class b
    ...
  End Class
  Class c
    ...
  End Class
End Namespace
```

Defineix un espai de noms que pot agrupar diverses classes.

168

---

### Estructures :

Una estructura es la evolució de Type ... End Type de VB6. Però ara permet incloure funcions.

```
Structure nom
  Variable As tipus
  Function funció() As tipus
    ...
  End Function
End Structure
```

Les diferències entre estructures i classes son :

- Les estructures no disposen de constructors.
  - Les estructures no implementen el mètode Finalize().
  - Les estructures no implementen herència.
  - Les variables dins una estructura son sempre de tipus valor.
- 

### Overridable:

Indica que un mètode pot ser reemplaçat en una classe filla.

```
Public Overridable Sub metode()
```

- 1 Veure : Conceptes bàsics de la programació OO

### Overrides:

Indica que un mètode d'una classe filla està reemplaçant el mateix mètode existent a la classe pare.

```
Public Overrides Sub metode()
```

Es necessari que el mètode a la classe pare estigui definit com a **Overridable**.

- 1 Veure : Conceptes bàsics de la programació OO
- 

184



---

### **NotOverridable:**

Marca un mètode per que no pugui ser reemplaçat a cap classe filla.

```
Public NotOverridable Overrides Sub metode()
```

Aquest exemple indica que metode() sobrecarrega al mateix mètode de la classe pare i, a la vegada, impedeix que pugui ser sobrecarregat per cap classe filla de la classe actual.

**1** Veure : Conceptes bàsics de la programació OO

### **NotInheritable:**

Marca una classe per que no es pugui heretar d'ella. Per tant, serà una “Classe final”

```
Public NotInheritable Class Exemple
```

**1** Veure : Conceptes bàsics de la programació OO

187

---

### **Classe abstracta:**

```
Public MustInherit Class a ...
```

Indica que la classe no pot fer-se servir per si mateixa. I que ha de ser heretada obligatòriament.

**1** Veure : Conceptes bàsics de la programació OO

### **Mètodes abstractes:**

```
MustOverride
```

Declara un mètode o una propietat com a implementable obligatòriament en una classe filla. El mètode redefinit a la classe filla ha de ser **Overrides**.

196 – 197

---

### **Interface:**

Classe especial en que es defineixen els procediments, funcions i propietats sense implementar-los; i sense precedir-los de l'àmbit, (**Public**, **Private**...)..

Es com definir una classe absolutament abstracta. Però no es una classe.

A la classe que implementa la interface afegim :

```
Implements NomInterface.
```

I a cada mètode o propietat implementada sobre la classe que implementa la interface afegim :

```
Public Function a() Implements NomInterface.FuncióImplementada
```

**1** Veure : “Constructors” a Conceptes bàsics de la programació OO

217



---

### Herencia d'interfases:

Les interfases poden heretar d'altres interfases.

```
Public Interface Interface1
    Inherits Interface2

    Function a
    Sub b
End Interface
```

221

---

### Clone i Dispose:

Els mètodes **Clone** i **Dispose** han de provenir de les interfases:

- **ICloneable**
- **IDisposable**

**CompareTo()** provè de la interface **IComparable**.

Per poder comparar per mes d'una clau fer servir **IComparer**.

Una classe que implementa **IComparer**, conté la funció **Compare**. Aquesta funció retorna un valor sencer que diu resultat de la comparació de dos objectes passats per paràmetre.

233 – 235

---

### La classe Environment :

**Environment** es fa servir principalment per controlar l'execució de la aplicació. Per saber els paràmetres i el path on s'executa. Per finalitzar-la. Informació sobre la màquina, l'usuari i la plataforma on s'executa, etc.

Un exemple :

`Environment.GetCommandLineArgs().GetLength(0)` à Retorna el nombre d'arguments de la línia de comandes. El primer correspon al path.

---

### String :

Mètodes d'string :

**PadLeft** i **PadRight**: Afegeix espais a esquerra o dreta d'una cadena.

```
s.PadLeft(20)
```

**Format**: Transforma una cadena a cadena amb format.

```
Console.WriteLine(String.Format("El valor es: {0:C}. Y el IVA {1}", i, 16))
Console.WriteLine(String.Format("{0}/{1}/{2}", 20, 4, 2006))
```

**Split** i **Join**: **Split** separa una cadena en trobar el caracter especificat. **Join** fa al contrari.

```
Dim x As String = "Este es un texto de prueba"
Dim y() As String = x.Split(" ")
Dim z As Integer
For z = 0 To y.Length - 1
    Console.WriteLine(y(z))
Next z
Dim w As String = String.Join("-", y)
Console.WriteLine(w)
```

**Altres** : Imprimir missatges a l'estil java.

```
Console.WriteLine("Això es {0} exemple d'impressió de missatges a l'estil {1}", 1, "java")
```



**StringBuilder :**

**StringBuilder** es una especialització de **String** amb mes funcionalitats per al tractament de cadenes.

**Exemple de StringBuilder :**

```
Dim sb As New StringBuilder("Esto es una cadena de prueba")  
sb.Insert(7, "[TEXTO INSERTADO]")  
sb.Append(" [TEXTO AÑADIDO]")  
Console.WriteLine(sb)
```



## Capítol VI : Gestió d'events i delegació.

### Paradigma d'events en .NET:

Els events han de seguir el paradigma de .NET.

Han de tenir els paràmetres :

- **sender** à Objecte del tipus de la classe, (**Me**).
- **e** à **EventArgs**. Informació sobre l'event.

241 – 242

### Events personalitzats:

Es pot crear una classe personalitzada de **System.EventArgs** per mostrar mes informació. Per exemple : Un missatge.

### Exemple d'events personalitzats :

#### Una classe personalitzada especialitzada d'EventArgs :

```
Public Class SuperEventPersonalitzar
    Inherits EventArgs

    Public Sub New()
        MyBase.New()
    End Sub

    Public Sub New(ByVal pMissatge As String)
        MyBase.New()
        mMissatge = pMissatge
    End Sub

    Dim mMissatge As String
    Public ReadOnly Property Missatge()
        Get
            Return mMissatge
        End Get
    End Property
End Class
```

### La classe on es fa saltar l'event :

```
Public Class unaClasse
    Public Event elEvent(ByVal sender As Llistes, ByVal e As SuperEventPersonalitzar)

    Public Sub provaEvent()
        RaiseEvent elEvent(Me, New SuperEventPersonalitzar("Hola, soc elEvent"))
    End Sub
End Class
```

### La classe que captura l'event :

```
Private WithEvents x As unaClasse
x = New unaClasse
Private Sub x_elEvent(ByVal sender As Llistes, ByVal e As SuperEventPersonalitzar)
    Handles x.elEvent
        MsgBox("Ha saltat l'event de prova. " & e.Missatge)
    End Sub
```

248



---

### **AddHandler:**

Afegeix un event d'escolta concret. Necessita :

- Un objecte.
- Un mètode existent a escoltar.
- Un mètode propi a executar quan es produeixi un event, (amb [AddressOf](#) a **1** Veure “Introducció”).

250

### **AddHandler i IDisposable:**

Si es fa servir [AddHandler](#), es recomana fer servir [IDisposable](#). Per cridar [RemoveHandler](#) al mètode [Dispose](#).

253

**1** Veure : “Events dinàmics” a Conceptes bàsics de la programació OO

---

### **Delegats:**

Es declara un mètode sense implementació. En temps d'execució es pot definir funcionalitat per aquest mètode, mitjançant un altre mètode existent i amb la mateixa signatura..El mètode [Invoke](#) substitueix el nom del mètode que es tria en temps d'execució.

```
Public Delegate Sub metodeVariable(pParametre As Tipus)
Sub MetodeA(p As Tipus)
    ...
End Sub
Public Sub endavant()
    Dim x As Tipus
    Dim del As metodeVariable
    del = New metodeVariable(AddressOf MetodeA)
    del.Invoke(x)
End Sub
```

259 – 260

---



## Capítol VII : Gestió d'errors en VB.NET: Viure amb les excepcions.

### Try... Catch... Finally:

```
Try
    ...
    Codi normal
    ...
    Exit Try à No recomanable.
    ...
Catch ex As Exception [When expressió]
    ...
Catch ex2 As IO.Exception
    ...
Finally
    ...
End Try
```

Una clàusula **Catch** amb **When**, s'executarà el contingut del bloc catch només si *expressió* es vertadera..

267 – 272

### e.StackTrace:

Imprimeix tota la traça de l'error.

271

### Catch. Altres temes:

Si **Catch** ha de finalitzar el codi, es recomanable cridar **Dispose**, i després cridar **Throws e**. On *e* es l'objecte exception del bloc **Try**.

Això el que fa es pujar la traça de l'error a la funció que va cridar el procediment on es va generar l'error.

274

### Herencia d'exception:

Es pot heretar d'una classe **Exception** per fer-la mes completa. Per exemple :

Heretar d'**IOException** per afegir-li una propietat de només lectura amb la descripció del error. I una propietat personalitzada amb un codi d'error propi.

### Exemple:

```
Public Class ExepcioProva
    Inherits Exception

    Public Sub New(ByVal pCodi As Integer, ByVal pMissatge As String)
        MyBase.New(pMissatge)
        setCodi = pCodi
    End Sub

    Private mCodi As Integer
    Public ReadOnly Property Codi() As Integer
        Get
            Return mCodi
        End Get
    End Property
    Private WriteOnly Property setCodi() As Integer
        Set(ByVal pCodi As Integer)
            mCodi = pCodi
        End Set
    End Property
End Class
```

274



---

***Jerarquia d'exception:***

La jerarquia bàsica del objecte Exception es :

System.Exception

- System.ApplicationException
- System.SystemException



---

## Capítol VIII : Windows Forms. Dibuix i impressió.

### Controls heretats:

Per crear un control especialitzat a partir d'un estàndard :

```
Public Class ExTextBox  
    Inherits System.Windows.Forms.TextBox
```

I només queda afegir els mètodes i propietats especialitzades per *ExTextBox*. Es poden modificar propietats o mètodes existents al control original.

281

---

### Iniciar una aplicació des d'un formulari amb Sub Main:

```
Public Sub Main()  
    Dim a As New Form1  
    Application.Run(a)  
End Sub
```

Aquest tipus de construcció es molt recomanable.

291

---

### MessageBox:

Substitueix *MsgBox()*. Te mes opcions, mes possibilitats. I es una classe.

292

---

### Menús:

#### Com crear un menú amb codi...

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)  
    Handles MyBase.Load  
        Dim mm As New MainMenu  
        Dim mnOpc As New MenuItem("Menú")  
  
        mnOpc.MenuItems.Add("Opció 1", AddressOf opcl_Click)  
        mnOpc.MenuItems.Add("Opció 2")  
  
        mm.MenuItems.Add(mnOpc)  
  
        Me.Menu = mm  
End Sub  
  
Public Sub opcl_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
    MsgBox("Opció 1 click")  
End Sub
```

299

---



---

### Formularis de diàleg:

#### Cridar un formulari de diàleg :

```
Dim dialeg As New ColorDialog  
If dialeg.ShowDialog() = DialogResult.OK Then Me.BackColor = dialeg.Color
```

#### Com crear-los i personalitzar-los amb un formulari normal.

Creem un nou formulari amb les propietats :

```
Me.ControlBox = False  
Me.MaximizeBox = False  
Me.MinimizeBox = False  
Me.TopMost = True
```

Afegim a aquest formulari dos botons. Un per acceptar i un altre per cancel·lar. I al formulari vinclem les propietats AcceptButton i CancelButton, amb aquests botons :

```
Me.AcceptButton = Me.btnAcceptar  
Me.CancelButton = Me.btnCancelelar
```

Als botons omplim la propietat DialogResult amb els valors que després es retornaran com a resultat del diàleg :

```
Me.btnAcceptar.DialogResult = System.Windows.Forms.DialogResult.OK  
Me.btnCancelelar.DialogResult = System.Windows.Forms.DialogResult.Cancel
```

Des d'un altre formulari cridem a aquest nou formulari de diàleg. De la forma :

```
Dim dp As New DialogPerso  
dp.ShowDialog()  
MsgBox(dp.DialogResult)
```

I podem explorar els objectes i propietats del formulari de diàleg per extreure les dades introduïdes per l'usuari.

303

---

### Afegir controls:

Com afegir controls en temps d'execució. I Com gestionar els seus events.  
Aquest exemple crea un botó i activa l'event click.

```
Dim boto As New System.Windows.Forms.Button()  
boto.Visible = True  
boto.Text = "Hola"  
boto.Size = New Size(100, 100)  
  
Me.Controls.Add(boto)  
AddHandler boto.Click, AddressOf Me.boto_Click  
  
Public Sub boto_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  
MsgBox("Click al botó")  
End Sub
```

304



---

### Formularis heretats:

Es poden crear formularis i compilar-los com a DLL, (això no es imprescindible). Seguidament, es poden fer servir com si fossin controls. Mitjançant la modalitat : “Formulari heretat”.

Els formularis s’hereten de la mateixa manera que qualsevol altra classe:

```
Public Class Form1
    Inherits Form2
```

305

---

### Declarar un event predeterminat:

```
<DefaultEvent("NomEvent")>
Public Class xxx
    ...
End Class
```

310

---

### Tractament del teclat a KeyPress:

Per cancel·lar una tecla pitjada al event `KeyPress` :

```
e.Handled = True
```

318

---

### Dibuix de gràfics a OnPaint:

Es on s’han de fer aquesta mena de feines.

Classes de dibuix mes comunes : Pen, Rectangle, etc.

#### Exemple d’OnPaint :

```
Protected Overrides Sub OnPaint(ByVal e As System.Windows.Forms.PaintEventArgs)
    Dim g As Graphics
    g = e.Graphics()

    Dim llapis As New Pen(Color.Azure, 4)
    Dim rectangle As New Rectangle(New Point(10, 10), New Size(100, 100))

    g.DrawRectangle(llapis, rectangle)

    rectangle.X = 20
    rectangle.Y = 20
    g.DrawRectangle(llapis, rectangle)

    g.Dispose()
End Sub
```

320

---



## Com imprimir:

325 – 331

Amb l'objecte **PrintDocument** :

A l'event **PrintPage** d'aquest objecte es on te lloc tota la operativa d'impressió. I on es determina el contingut de cada pàgina i el numero total de pàgines.

### Exemple de l'event PrintPage :

```
Private Sub MyPrintPage(ByVal sender As Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Private MyFont As Font = New Font("Arial", 24) ' la font per imprimir el text
    Private MyBrush As Brush = Brushes.Black ' el pinzell per imprimir el text

    e.Graphics.DrawString("Pàgina " & CurrentPage.ToString(), MyFont, MyBrush, X, Y)
    If CurrentPage < TotalPages Then
        CurrentPage += 1
        e.HasMorePages = True
    Else
        e.HasMorePages = False
    End If
End Sub
```

## Configurar impressores:

Per configurar impressores es poden fer servir els objectes :

**PrintDialog**: Selecció d'impressora i propietats d'impressora.

**PageSetupDialog**: Configuració de pàgina.

L'objecte **PrintDocument** es pot vincular amb aquests objectes a partir de les propietats :

**DefaultPageSettings**: Amb l'objecte **PageSetupDialog**.

**PrinterSettings**: Amb l'objecte **PrintDialog**.

Aquestes dues propietats son editables manualment i dispostes d'una col·lecció de mètodes i propietats per configurar la impressora i pàgina del document a imprimir.

### Exemple de PrintDialog :

```
Dim pt As New PrintDialog
pt.Document = PrintDocument1
pt.AllowSomePages = True

If pt.ShowDialog() = DialogResult.OK Then
    PrintDocument1.PrinterSettings = pt.PrinterSettings
    ...
    Imprimir el document
End If
```

### Exemple de PageSetupDialog :

```
Dim ps As New PageSetupDialog
ps.Document = PrintDocument1
ps.ShowDialog()
PrintDocument1.DefaultPageSettings = ps.PageSettings
```

329



### **Vista prèvia de la impressió:**

El resultat d'una impressió es pot mostrar per pantalla de diverses formes. Una, ([PrintPreviewControl](#)), com a objecte incrustable en qualsevol formulari. L'altre, ([PrintPreviewDialog](#)), com a finestra independent. Per fer-ho, cal vincular un objecte [PrintDocument](#), a un objecte [PrintPreviewDialog](#) o [PrintPreviewControl](#).

### **Exemple d'impressió amb PrintPreviewDialog :**

```
Dim pd As New PrintPreviewDialog  
pd.Document = PrintDocument1  
pd.ShowDialog()
```

### **Exemple d'impressió amb PrintPreviewControl :**

```
Dim FormPreview As Form = New Form  
Dim PreviewControl As PrintPreviewControl = New PrintPreviewControl  
  
PreviewControl.Document = PrintDocument1  
PreviewControl.StartPage = 0  
PreviewControl.Columns = 4  
  
FormPreview.Text = "Print Preview Form"  
FormPreview.WindowState = FormWindowState.Maximized  
FormPreview.Controls.Add(PreviewControl)  
FormPreview.Controls(0).Dock = DockStyle.Fill  
  
FormPreview.ShowDialog()  
FormPreview.Dispose()
```

330



---

## Capítol IX : Entrada / Sortida.

### La classe **Stream**:

La classe **Stream** permet accedir a qualsevol origen de dades. L'especialització per arxius es : **FileStream**.

El mètode **Close** no es pot fer servir si l'origen no està efectivament obert. A diferència de VB6 on això no era necessari.

347

---

### Classes per tractar arxius i directoris:

- **File**
- **FileInfo**
- **Directory**
- **DirectoryInfo**

Excepcions més comunes en el tractament d'aquestes classes:

- **DirectoryNotFoundException**
- **EndOfStreamException**
- **FileLoadException**
- **FileNotFoundException**

334

### Mes sobre **FileInfo** i **DirectoryInfo**:

Es pot donar tota la informació d'un arxiu o un directori i, seguidament, crear-los amb **Create**, fent servir **FileInfo** o **DirectoryInfo**.

El "." en aquestes classes es refereix al directori vigent.

### Exemple de **File**, **FileInfo**, **Directory** i **DirectoryInfo**:

```
' File
Console.WriteLine("Existeix module1.vb?: " & File.Exists("../module1.vb"))

' FileInfo
Dim x As FileInfo
x = New FileInfo("../module1.vb")
Console.WriteLine("Atributs de l'arxiu module1.vb: " & x.Attributes.ToString())
Console.WriteLine("FullName: " & x.FullName)

' Directory.
Console.WriteLine("Existeix el directori arrel :- )?: " & Directory.Exists("C:\"))

' DirectoryInfo
Dim d As New DirectoryInfo("C:\")
Console.WriteLine("Atributs del directori C:\: " & d.Attributes.ToString())
```

340



### La classe Path:

Gestiona camins, (locals i de xarxa), independentment de la plataforma, (caràcters “\”, “/”, etc).  
Aïlla directoris, unitats, noms d’arxiu, extensions, etc.

### Exemple de Path:

```
Console.WriteLine(Path.GetTempPath())
```

335

### BinaryReader i BinaryWriter:

Son classes especialitzades de [Stream](#), per a arxius binaris.

### Exemple d’escriptura i lectura amb BinaryReader i BinaryWriter:

```
Private Shared Sub BinaryStream()  
    Dim bsw As BinaryWriter  
    Dim bsr As BinaryReader  
  
    Try  
        ' Arxiu on s'escriurà/llegirà  
        arxiu = New FileStream("arxiu.txt", FileMode.OpenOrCreate,  
            FileAccess.ReadWrite)  
  
        ' Escriptura.  
        bsw = New BinaryWriter(arxiu)  
        bsw.Write("Hola mon!!!")  
  
        ' Lectura.  
        bsr = New BinaryReader(arxiu)  
        arxiu.Position = 0 ' Coloca al inici de l'arxiu, perquè a la escriptura  
            estava situat al final.  
        Console.WriteLine(bsr.ReadString())  
    Catch ioe As IOException  
        Console.WriteLine(ioe.StackTrace)  
    Catch e As Exception  
        Console.WriteLine(e.StackTrace)  
    Finally  
        bsw.Close()  
        bsr.Close()  
        arxiu.Close()  
    End Try  
End Sub
```

355

### Arxius seqüencials. TextReader i TextWriter:

Especialitzacions de la classe [Stream](#) per arxius seqüencials. Son abstractes. Per poder treballar-hi s’ha de fer servir [StreamReader](#) i [StreamWriter](#).

350 i 358

### Exemple d’arxius seqüencials:

```
Private Shared Sub textStream()  
    Dim tsw As StreamWriter  
    Dim tsr As StreamReader  
  
    Try  
        ' Escriptura.  
        tsw = New StreamWriter("arxiu.txt") ' No necessita FileStream  
        tsw.WriteLine("Hola Mon!!!")  
        tsw.WriteLine("El hombre desactualizado!!!")  
        tsw.Close()  
    End Try
```



```
' Lectura.  
tsr = File.OpenText("arxiu.txt") ' No necessita FileStream  
Dim s As String  
Do  
    s = tsr.ReadLine()  
    If Not s Is Nothing Then  
        Console.WriteLine(s)  
    End If  
Loop Until s Is Nothing  
Catch ex As Exception  
    Console.WriteLine(ex.StackTrace)  
Finally  
    tsw.Close()  
    If Not tsr Is Nothing Then tsr.Close()  
End Try  
End Sub
```

360

### **Console.SetIn i Console.SetOut:**

Redirigeixen l'entrada i la sortida estàndard, a qualsevol classe lectora o escriptora.

359

### **Peek:**

**Peek** determina el final de l'arxiu. Substitueix a **EOF** de VB6.  
El valor **-1** indica el final del arxiu.

360

### **Serialització:**

Representa la capacitat d'emmagatzemar els objectes per que persisteixin en una propera execució del programa. Per fer una classe serialitzable es suficient afegir l'atribut **<Serializable(>** a la capçalera de definició de la classe :  
**<Serializable(>** Public Class classe

Serà necessari que tota classe filla d'una classe serialitzable sigui també al seu torn serialitzable. En cas contrari el procés de serialització fallarà amb la excepció **System.Runtime.Serialization.SerializationException**

361

La serialització pot fer-se en diferents formats, els mes usuals son en binari o en SOAP-XML. Per aquest últim es necessari importar la llibreria **System.Runtime.Serialization.Formatters.Soap**.

### **Exemple de serialització:**

```
Imports System.IO  
Imports System.Runtime.Serialization  
  
Module Module1  
    Sub Main()  
        Dim classe As New Exemple("hola")  
  
        Dim arxiuBinari As FileStream  
        Dim formatBinari As New Formatters.Binary.BinaryFormatter  
  
        Dim arxiuSOAP As FileStream  
        Dim formatSOAP As New Formatters.Soap.SoapFormatter  
  
    Try
```



```
        arxiuBinari = New FileStream("serialBinari.txt", _  
            FileMode.Create, FileAccess.Write)  
        formatBinari.Serialize(arxiuBinari, classe)  
  
        arxiuSOAP = New FileStream("serialSOAP.txt", _  
            FileMode.Create, FileAccess.Write)  
        formatSOAP.Serialize(arxiuSOAP, classe)  
    Catch ex As Exception  
        MsgBox(ex.Message)  
    End Try  
End Sub  
End Module
```

#### Exemple de des-serialització :

```
Private Function desSerialitza() As Exemple  
    Dim arxiuSOAP As FileStream  
    arxiuSOAP = New FileStream("serialSOAP.txt", FileMode.Open, FileAccess.Read)  
  
    Dim formatSOAP As New Formatters.Soap.SoapFormatter  
    Return CType(formatSOAP.Deserialize(arxiuSOAP), Exemple)  
End Function
```

#### Exemple de clonació d'objectes mitjançant serialització:

```
Module Module1  
    Sub Main()  
        ' Clonació.  
        Dim classe3 As Exemple  
        classe3 = classe.Clone()  
        Console.WriteLine(classe3.parametre)  
        classe3.parametre = "Adeu"  
        Console.WriteLine(classe.parametre)  
        Console.WriteLine(classe3.parametre)  
    End Sub  
End Module
```

```
Imports System.IO  
Imports System.Runtime.Serialization  
  
<Serializable(> _  
Public Class Exemple  
    Implements ICloneable  
  
    Private mStream As MemoryStream  
  
    Public Sub New(ByVal p As String)  
        parametre = p  
    End Sub  
  
    Private mParametre As String  
    Public Property parametre() As String  
        Get  
            Return mParametre  
        End Get  
        Set(ByVal Value As String)  
            mParametre = Value  
        End Set  
    End Property  
  
    Public Function Clone() As Object Implements System.ICloneable.Clone  
        Dim formatBinari As New Formatters.Binary.BinaryFormatter  
        Try  
            serialitza()  
            mStream.Position = 0  
            Return formatBinari.Deserialize(mStream)  
        Catch ex As Exception  
            MsgBox(ex.Message())  
        End Try  
    End Function
```



```
        Finally
            mStream.Close()
        End Try
    End Function

    Private Sub serialitza()
        Dim formatBinari As New Formatters.Binary.BinaryFormatter
        Try
            mStream = New MemoryStream
            formatBinari.Serialize(mStream, Me)
        Catch ex As Exception
            MsgBox(ex.Message())
        End Try
    End Sub
End Class
```

361 – 365

**Mes...:**

Exemple complet de serialització i des-serialització d'objectes.

365

**FileSystemWatcher:**

Controla tots els canvis que tenen lloc en un directori i, si es vol, als seus subdirectoris. Amb diferents nivells de configuració.

**Exemple de FileSystemWatcher:**

```
Private WithEvents fsw As New System.IO.FileSystemWatcher

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    fsw.Path = Path.GetFullPath(".")
    fsw.Filter = "*.txt"
    fsw.EnableRaisingEvents = True

    visor.Text = "Inspeccionant: " & fsw.Path
End Sub

Private Sub Canvi(ByVal sender As Object, ByVal e As System.IO.FileSystemEventArgs)
Handles fsw.Changed, fsw.Created, fsw.Deleted
    visor.Text = e.Name & " - " & e.ChangeType().ToString()
End Sub

Private Sub fsw_Renamed(ByVal sender As Object, ByVal e As System.IO.RenamedEventArgs)
Handles fsw.Renamed
    visor.Text = e.Name & " - " & e.ChangeType().ToString()
End Sub
```

375



---

## Capítol X : Threads.

### **Dominis d'aplicació:**

Un domini d'aplicació es l'entorn on s'executa una aplicació i els seus processos. Els processos d'un mateix domini d'aplicació poden compartir dades. Els processos de diferents dominis d'aplicació no poden compartir dades.

### **Exemple d'AppDomain:**

```
Dim apps As AppDomain
Console.WriteLine("ApplicationBase: " &
apps.CurrentDomain.SetupInformation.ApplicationBase())

Console.WriteLine("Assemblies: ")
Dim assemblies() As System.Reflection.Assembly
assemblies = apps.CurrentDomain.GetAssemblies()
Dim ass As System.Reflection.Assembly
For Each ass In assemblies
    Console.WriteLine(ass.FullName)
Next
```

---

### **Funcions amb suport thread:**

Moltes funcions disposen de omònimes compatibles amb thread. Una funció compatible amb thread disposa de 3 signatures :

- Funcio()                   à No compatible amb thread.
- BeginFuncio()           à Inicia la funció en un thread secundari.
- EndFuncio()             à Recull el resultat de la funció executada en un thread secundari.

Aquest funcionament fa que el treball amb Threads sigui transparent per al programador. Per treballar així s'han de seguir un patró proposat per .NET, tal i com es mostra a l'exemple següent.

### **Exemple de Threads amb la implementació de IAsyncResult :**

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Delegate Function delegatLlista() As String
    Private mLlistaDelegada As delegatLlista
    Private delCallBack As New AsyncCallback(AddressOf RespostaCallBack)

    Private Sub btnOmplerLlista_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnOmplerLlista.Click
        Dim laLlista As New Llista
        mLlistaDelegada = New delegatLlista(AddressOf laLlista.omplirLlista)

        mLlistaDelegada.BeginInvoke(delCallBack, Nothing)
    End Sub

    Public Sub RespostaCallBack(ByVal result As IAsyncResult)
        Dim resposta As String

        resposta = mLlistaDelegada.EndInvoke(result)
        llista.Items.Add(resposta)
    End Sub
End Class
```



### Inici i finalització d'un subprocés, (Threads):

Un subprocés pot iniciar-se amb qualsevol mètode d'un objecte prèviament existent. O en mètodes de classe. Termina només quan el mètode que el va iniciar finalitza.

381 – 382

### Restriccions dels mètodes que criden a subprocessos:

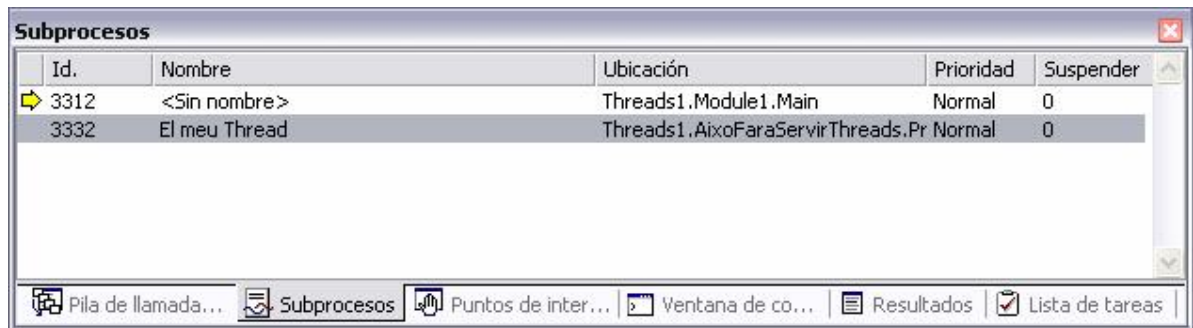
Els mètodes que inicien un subprocés no poden tenir paràmetres ni ser funcions.

382

Es recomanable que un subprocés ha de tenir un nom descriptiu. La propietat Name del subprocés que s'engega permet introduir un nom per al subprocés-

```
elThread.Name = "El meu Thread"
```

Si assignem noms descriptius a subprocessos, després podrem veure amb facilitat aquests subprocessos des de la finestra de "Subprocessos".



389

### Exemple de Threads:

Exemple de com crear un subprocés.

```
Option Strict On
Imports System.Threading

Module Module1

    Sub Main()
        Dim laMevaClasse As New AixoFaraServirThreads

        Dim elMeuThread As New ThreadStart(AddressOf laMevaClasse.Prova)
        Dim elThread As New Thread(elMeuThread)

        elThread.Start()

        Dim count As Integer

        Do While True
            count -= 1
            Console.WriteLine("Estat del comptador Main: " & count)
        Loop
    End Sub

End Module

Public Class AixoFaraServirThreads
    Public Sub Prova()
        Dim count As Integer

        Do While True
            count -= 1
            Console.WriteLine("Estat del comptador Thread: " & count)
        Loop
    End Sub
End Class
```



---

```
End Sub  
End Class
```

383 - 385

### Vincular subprocessos a formularis.

415 - 421

#### Prioritats en subprocessos:

Al assignar una prioritat a un subprocés determinarem la seva velocitat d'execució davant del conjunt dels altres subprocessos. Les prioritats son valors relatius, i sempre sotmesos a les característiques de la màquina, del SO, etc..

```
Valors de ThreadPriority :  
ThreadPriority.Highest  
ThreadPriority.AboveNormal  
ThreadPriority.Normal  
ThreadPriority.BelowNormal  
ThreadPriority.Lowest
```

#### Exemple d'assignació de prioritats a un subprocés:

```
elThread.Priority = ThreadPriority.Higest
```

387

#### IsAlive:

Retorna si un subprocés es viu.

#### Abort:

Abort desencadena una excepció **ThreadAbortException**, a la vez que llama al método **Finalize** del objeto encapsulado por **Thread**. Un cop es crida a Abort generalment el subprocés acaba. Es pot anul·lar Abort cridant **ResetAbort**.

#### Suspend i Resume :

**Suspend** suspen un subprocés fins que es crida a **Resume**. **Resume** desbloqueja un subprocés suspès amb **Suspend**.

#### Join. Ajornar subprocessos:

**Join** ajorna un subprocés, (el congela), fins que finalitzi un altre.

Per exemple: Des del subprocés 1 podem fer **thread2.Join()**. Això farà que el subprocés 2 quedi congelat fins que acabi el subprocés 1.

**Join(milisegons)** fa que el subprocés s'ajorni fins que acabi un altre, o be fins que passi el temps indicat com a paràmetre.

389

#### Sleep. Suspendre subprocessos:

Bloqueja un subprocés durant *X* milisegons. **Sleep(0)** el suspèn indefinidament; i només pot reanimar-se amb **Interrupt**.

391

#### Interrupt. Reanimació de subprocessos:

Trenca la suspensió del subprocés fet amb **Sleep**. Al contrari del que indica el seu nom, **Interrupt** reanima un subprocés prèviament ajornat o congelat, amb **Join** o **Sleep**, per exemple.

392

---

#### Sincronització de subprocessos :

**SyncLock ... End SyncLock:**



No deixa que cap altre subprocés accedeixi al codi que encercla fins que el subprocés en us no acabi.

Exemple de Threads amb SyncLock:

```
Option Strict On
Imports System.Threading

Module MCasa
    Sub Main()
        Dim casa As New Casa(10)
        Console.ReadLine()
    End Sub
End Module

Public Class Casa
    Public Const MAXIMA_TEMPERATURA As Integer = 40
    Private mTemperaturaActual As Integer = 25
    Private mHabitacions() As Habitacio

    Public Sub New(ByVal pNumHabitacions As Integer)
        ReDim mHabitacions(pNumHabitacions - 1)
        Dim i As Integer
        Dim aThreadStart As Threading.ThreadStart
        Dim aThread As Thread

        For i = 0 To pNumHabitacions - 1
            Try
                mHabitacions(i) = New Habitacio(Me, mTemperaturaActual, _
                    "Habitació: " & CStr(i))
                aThreadStart = New ThreadStart(AddressOf _
                    mHabitacions(i).ControlaTemperatura)
                aThread = New Thread(aThreadStart)
                aThread.Start()
            Catch ex As Exception
                Console.WriteLine(ex.Message)
            End Try
        Next
    End Sub

    Public Property Temperatura() As Integer
    Get
        Return mTemperaturaActual
    End Get
    Set(ByVal pTemperatura As Integer)
        mTemperaturaActual = pTemperatura
    End Set
End Property
End Class

Public Class Habitacio
    Private mTemperaturaActual As Integer
    Private mNomHabitacio As String
    Private mCasa As Casa

    Public Sub New(ByVal pCasa As Casa, ByVal pTemperatura As Integer, _
        ByVal pNomHabitacio As String)
        mCasa = pCasa
        mTemperaturaActual = pTemperatura
        mNomHabitacio = pNomHabitacio
    End Sub

    Public Sub ControlaTemperatura()
        CanviaTemperatura()
    End Sub

    Private Sub CanviaTemperatura()
        SyncLock mCasa
            Try
                If mCasa.Temperatura < mCasa.MAXIMA_TEMPERATURA - 5 Then
                    Thread.Sleep(200)
                    mCasa.Temperatura += 5
                End If
            End Try
        End Sub
    End Sub
End Class
```



```
        Console.WriteLine("A l'habitació " & mNomHabitacio & " _  
            la temperatura es de " & mCasa.Temperatura)  
    ElseIf mCasa.Temperatura < mCasa.MAXIMA_TEMPERATURA Then  
        Thread.Sleep(200)  
        mCasa.Temperatura += 1  
        Console.WriteLine("A l'habitació " & mNomHabitacio & " _  
            la temperatura es de " & mCasa.Temperatura)  
    Else  
        Thread.Sleep(200)  
        mCasa.Temperatura -= 2  
        Console.WriteLine("A l'habitació " & mNomHabitacio & " _  
            la temperatura es de " & mCasa.Temperatura)  
    End If  
    Catch tie As ThreadInterruptedException  
        Console.WriteLine(tie.Message)  
    Catch ex As Exception  
        Console.WriteLine(ex.Message)  
    End Try  
End SyncLock  
End Sub  
End Class
```

397 – 403

#### Les classes InterLocked i Monitor :

Per processos en que s'incrementa o disminueix valors de variables, es proporciona la classe [Interlocked](#), que fa això de forma automàtica.

404

#### Sincronization :

La clàusula <Sincronization(> davant d'una classe fa que si un procés accedeix a una variable o mètode d'instància, cap altre procés pugui entrar fins que el primer no acabi.

---

#### **Abraçada mortal:**

Dos processos necessiten x recursos, (els mateixos), per treballar. Un procés te uns i l'altre te la resta. Tots dos resten esperant a que l'altre alliberi els recursos que li falten. Això no passa mai i la aplicació falla.

404 – 410

---

#### **Canviar l'estat d'un control des d'un subprocés:**

##### Formulari :

```
Imports System.Threading  
Public Class Form_Principal  
    Inherits System.Windows.Forms.Form  
  
    ' Variables.  
    Private xThread As Thread  
    Private WithEvents DH As New DirHola(10000000, tMissatge)  
  
    ' Sortir.  
    Private Sub btnSortir_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btnSortir.Click  
        If Not xThread Is Nothing Then  
            If xThread.IsAlive Then xThread.Abort()  
        End If  
        Me.Dispose(True)  
    End Sub  
  
    ' Començar el subprocés.
```



```
Private Sub btnComencar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnComencar.Click
    Dim xThreadStart As New ThreadStart(AddressOf DH.comencar)
    xThread = New Thread(xThreadStart)
    xThread.Priority = ThreadPriority.BelowNormal
    xThread.Start()
End Sub

' Salta l'event del subprocés.
Private Sub DH_procesExecucio(ByVal pMissatge As String) Handles DH.procesExecucio
    tMissatge.Text = pMissatge
End Sub
End Class
```

### Classe que implementa el subprocés:

```
Public Class DirHola
    ' Propietats.
    Private cops As Long
    Private caixaText As Windows.Forms.TextBox

    ' Variables.
    Private missatge As String = ""
    Private mtdd As MethodInvoker = New MethodInvoker(AddressOf updateTextControl)

    ' Events.
    Public Event procesExecucio(ByVal pMissatge As String)

    ' Constructor.
    Public Sub New(ByVal pCops As Long, ByVal pCaixa As Windows.Forms.TextBox)
        cops = pCops
        caixaText = pCaixa
    End Sub

    ' Inici del subprocés.
    Public Sub comencar()
        fesDirHola()
    End Sub

    ' Cos del subprocés.
    Private Sub fesDirHola()
        Dim comptador As Long = 0

        Do While comptador < cops
            comptador += 1
            missatge = "Hola " & CStr(comptador)

            Try
                mtdd.Invoke()
            Catch ex As Threading.ThreadInterruptedException
                MsgBox("Abortat")
            End Try
        Loop
    End Sub

    ' Event.
    Private Sub updateTextControl()
        RaiseEvent procesExecucio(missatge)
    End Sub
End Class
```

420 – 421

### Compartir dades amb Threads :

Si els Threads criden mètodes estàtics, aquests, com en qualsevol altre classe, tindran accés a variables de classe. I les podran manipular i, per tant, ser vistes des d'altres processos i classes.



---

Si afegim `<ThreadStatic(>` davant d'una variable estàtica, farem que aquesta variable tingui un valor independent per a cada procés. Es a dir, que es comporti com a estàtica dins d'un procés, però com a d'instància entre diferents processos.

```
<ThreadStatic(> Public Shared variable As Integer = 5
```

---

### ***Excepcions amb Threads :***

Les excepcions `ThreadAbortException` i `ThreadInterruptedException` s'invoquen quan es criden els mètodes `Abort` i `Interrupt` respectivament.



## Remoting.

### Remoting :

Fabricar una classe que pot operar encapsulantse a través d'un protocol de comunicacions, (HTTP, TCP, etc). I donar serveis a d'altres aplicacions.

Es distingeixen dos formes d'implementar remoting. Amb webservices, o treballant directament amb les classes TCP/IP de .NET.

Una classe que implementa remoting treballa a través d'un **canal**, (TCP, HTTP, etc), i amb un **format**, (XML, SOAP, etc).

Una classe que implementa remoting fa servir la classe `System.Runtime.Remoting`.

Els objectes que implementen Remoting han de ser serialitzables.

### Exemple de Remoting :

#### Una classe que donarà servei a partir de Remoting :

Una classe que ofereix un servei a partir de Remoting, ha d'heretar de `MarshalByRefObject`.

```
imports System
Namespace Lab13
    Public Class aServer
        inherits Marshalbyrefobject

        private aRandom as Random
        Public Sub New()
            aRandom = new Random
            Console.writeline("Server activado")
        End Sub

        public function Authenticate(CustomerId as Integer) as boolean
            dim aRandomNumber as double=aRandom.NextDouble()
            dim passed as boolean
            if (aRandomNumber < 0.5) then
                passed = false
            Else
                passed = true
            End If

            Console.writeline("Authentication server CustomerId: {0}, _
                passed: {1}", customerid,passed.toString())

            return passed
        end function
    End Class
End Namespace
```

#### La classe que implementa Remoting i fa de servidor de la classe que implementa el servei :

```
imports System
imports System.Runtime.Remoting
imports System.Runtime.Remoting.Channels
imports System.Runtime.Remoting.Channels.Http

Namespace Lab13
    public Class Server
        Public Shared Sub Main()
            dim chan1 as new HttpChannel(8086)
            ChannelServices.RegisterChannel(chan1)
            RemotingConfiguration.RegisterWellKnownServiceType(getType(Lab13.aServer), _
```



```
        "DoAuthentication", WellKnownObjectMode.Singleton)

        Console.WriteLine("Press a key to exit")
        Console.ReadLine()
    End Sub
End Class
End Namespace
```

**La classe client que demana el servei :**

```
imports System
imports System.Runtime.Remoting
imports System.Runtime.Remoting.Channels
imports System.Runtime.Remoting.Channels.Http
imports Microsoft.VisualBasic
imports System.IO

Namespace Lab13
    Public class TestClient
        Public Shared Sub Main()
            dim chan1 as new HttpChannel()
            ChannelServices.RegisterChannel(chan1)

            Dim aServ As aServer = CType(Activator.GetObject(GetType(Lab13.aServer), _
                "Http://localhost:8086/DoAuthentication"), aServer)

            try
                Dim passed as boolean = AServ.Authenticate(1234)
                Console.WriteLine("Customer 1234 authentication passed: " & _
                    passed.ToString())
            Catch ex as exception
                Console.WriteLine(ex.message)
            End Try
        End Sub
    End Class
End Namespace
```



---

## Capítol XI : Breu introducció al accés a bases de dades amb VB.NET.

### Nomenclatura SQL :

Nom	Sentències SQL relacionades	Tipus de retorn d'un objecte Command
<b>DML</b>	Insert, Update i Delete	ExecuteNonQuery
<b>DRL</b>	Select	ExecuteScalar o ExecuteReader
<b>DDL</b>	Create, Alter i Drop	ExecuteNonQuery
<b>DCL</b>	Grand, Revoke i Deny	ExecuteNonQuery

---

### Accedir a dades. Descripció:

Tres classes per accedir a dades :

- System.Data.OleDb
- System.Data.SqlClient
- System.Data.Odbc

#### **OleDb :**

Per accedir a orígens de dades diferents a SQL-Server. Per exemple : Access, IBMDB400, FoxPro...

#### **SqlClient :**

Per accedir a SQL-Server.

#### **Odbc :**

Antic protocol de gestió de dades.

425 – 430

---

### Comparativa ADO & ADO.NET :

ADO	ADO.NET	Àmbit
<b>Connection</b>	xxxConnection	SQL i OleDb
	xxxTransaction	SQL i OleDb
<b>Command</b>	xxxCommand	SQL i OleDb
<b>Recordset</b>	DataSet	Desconnectat i sempre amb xxxDataAdapter
	xxxDataAdapter	Desconnectat
	xxxDataReader	Connectat
	DataView	Desconnectat

On **xxxx** es substitueix per "OleDb", "SQL" o "Odbc"



---

### Jerarquia OLEDB i SQLClient:

**xxxxConnection** à Estableix connexió amb una BD.

**xxxxTransaction**  
**xxxxException**  
**xxxxError**

**xxxxCommand** à Executa mandats de la BD.

**xxxxParameter**

Command pot executar-se com :

**ExecuteScalar**: Retorna un únic valor.  
**ExecuteReader**: Retorna un o mes registres de dades.  
**ExecuteNonQuery**: No retorna cap valor.  
**ExecuteXmlReader**: Retorna una estructura de dades XML.

**xxxxDataReader** à Recull registres de la BD.

Substitueix a **RecordSet** de VB6.

On **xxxx** es substitueix per “OleDb”, “SQL” o “ODBC”

**DataSet** à Recull les dades en un entorn desconnectat.

**DataView** à Cursor amb una vista filtrada i/o ordenada d'un conjunt de dades d'un DataSet.  
**DataTable** à Una taula. Un conjunt de dades dels n possibles d'un DataSet  
**DataRelation** à L'especificació de la relació entre dues taules d'un DataSet.  
**DataRow** à Una fila d'una taula.

	425
<b>Exemple d'us d'OleDb:</b>	426
<b>Exemple d'us de SQLClient:</b>	429

---

### DataReader :

**xxxxDataReader** crea un cursor en un ambient connectat. I que **només es de lectura i forwardonly**. No disposa de les propietats BOF i EOF tradicionals.

Per recorre'l es pot fer :

```
Dim obj As DataReader
Do
    Do While (obj.Read() )
        Console.WriteLine(obj.GetString(camp))
    Loop
Loop Until obj.NextResult()
```

**Read()** retornarà false quan ja no hi hagin mes dades en el conjunt de dades actual.

**NextResult()** salta entre conjunt de resultats. Això es útils quan es crida a procediments emmagatzemats que retornen mes d'un conjunt de dades.

Per obtenir dades d'un DataReader es fan servir els mètodes get segons el tipus de dades que vulguem obtenir :

**GetString()** : Cadenes

**GetInteger()** : Valors enters.

**GetValue()** : Qualsevol valor retornat com a Object. Després caldrà emprar **CType** per fer la conversió.

Altres propietats :



---

**RecordsAffected()** : Retorna el nombre de registres afectats per la última sentència SQL del DataReader.

---

### Transaccions:

Les transaccions s'implementen amb la classe xxxTransaction. Les transaccions han de garantir :

- **Atomicitat:** O tots els canvis es reflexen a la base de dades, o cap.
- **Consistència:** Si la BD es consistent abans de la transacció, ho haurà de ser després.
- **Aïllament:** Assegura l'execució concurrent de transaccions. De forma que una transacció no pot afectar a cap altra.
- **Durabilitat:** Un cop la transacció comença, els canvis fets a la base de dades no es perden, encara que hi hagi una falla al sistema.

### Exemple de SQLTransaction :

```
Dim con As New SqlConnection("connexio")
Dim com As SqlCommand()
Dim trans As SqlTransaction

Trans = con.BeginTransaction(nivell d'aïllament)
Com.Transaction = trans ` Vincula l'objecte command amb la transacció.
.
Accions de l'objecte Command
.
Trans.Commit()
Trnas.Rollback()
```

### Propietats de Transaction :

**BeginTrans:** BeginTrans s'aconsegueix de l'objecte [Connection](#). A la vegada es pot donar un nivell d'aïllament.

Nivell d'aïllament	Lectura no confirmada	Lectura no repetible	Fantasma
ReadUncommitted	<del></del>	<del></del>	<del></del>
ReadCommitted		<del></del>	<del></del>
RepeteableRead			<del></del>
Serializable			

**ReadUncommitted:** Protegeix les dades actualitzades, evitant que cap altra transacció les actualitzi, fins que acaba la transacció en curs.

**ReadCommitted:** Protegeix parcialment les lectures, impedit que una altra transacció llegeixi dades que encara no s'han confirmat.

**RepeteableRead:** Impedeix, fins que acaba la transacció en curs, que una altra transacció actualitzi una dada que s'hagi llegit a aquesta.

**Serializable :** Ofereix aïllament total, i evita qualsevol tipus d'interferència incloent-hi els fantasmes.

**Registres fantasma:** La possibilitat de llegir un registre que està afectat per una sentència Delete d'una altra transacció.

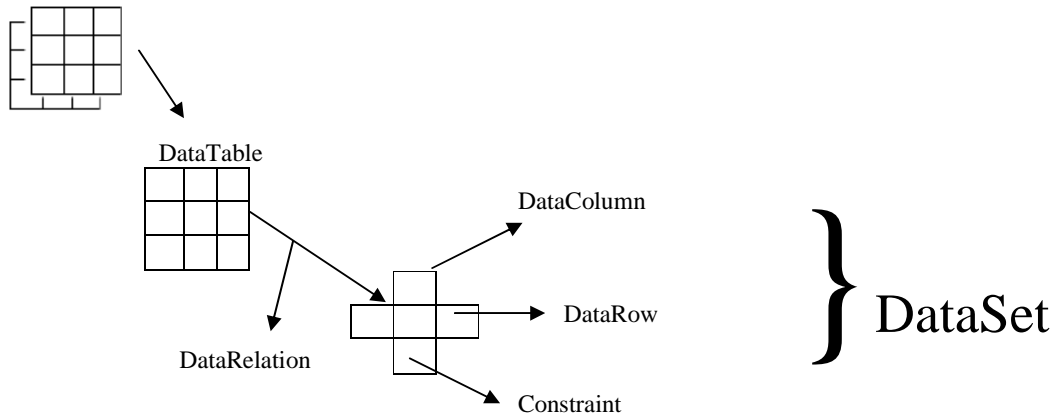
**objCommand.Transaction :** L'objecte o objectes [Command](#) que faran les operacions sobre la base de dades; i que es veuran afectats per la transacció.

**Commit :** S'accepta la transacció.

**Rollback :** Es cancel·la la transacció.



**Entorns desconectats de dades. Conjunt de classes :**



**DataSet:**

DataSet conté tota la informació d'una o diverses taules. Aquestes taules poden correspondre a una base de dades, a una estructura XML, a un arxíu XSD, o poden construir-se dinàmicament.

Un DataSet es pot omplir de dades de forma manual, llegint un arxíu de dades XML o omplint mitjançant un DataAdapter.

**Llegir i guardar XML :**

**DataSet.ReadXml**("Nom d'arxiu", mode): Llegeix el contingut d'un arxíu XML

On mode:

- **ReadSchema**: Si amb l'arxiu de dades no hi ha un esquema, no carregarà les dades.
- **IgnoreSchema**: Ignora l'esquema, si al mateix arxiu de dades n'hi hagués un.
- **InferSchema**: Dedueix l'esquema a partir de les dades.
- **DiffGran**: Obsolet.
- **Fragment**: Carrega un fragment XML. Equival a un conjunt de resultats.

**DataSet.WriteXml**("Nom d'arxiu"): Guarda tota la estructura i dades del DataSet a un arxíu XML.

**DataSet.ReadXMLSchema**("Nom d'arxiu"): Carrega un esquema XSD a partir d'un arxíu d'esquemes XSD.

**Llegir des d'un DataAdapter :**

```
Dim con As New SqlConnection("BBDD") \ Connexió.  
Dim daCustomers As New SqlDataAdapter("select * from customers", con) \ Crea un  
DataAdapter.  
  
Dim dsCustomers As New DataSet  
daCustomers.Fill(dsCustomers, "Customers") \ Omple un DataSet amb la selecció del  
DataAdapter.
```

A l'exemple s'omple el DataSet "dsCustomers" amb el resultat de la execució de la sentència SQL del DataAdapter.



## 1 Veure : “DataAdapter” d’aquest capítol

### Exploració de registres :

A nivell de **DataSet**, **Table** o **DataRow** es pot saber si hi ha hagut canvis, (si s’han modificat, esborrat o afegit files).

Amb **HashChanges(tipus)** à Es pot saber si hi ha hagut canvis d’un tipus determinat, (afegit, esborrat, modificació...), o sobre qualsevol tipus.

Amb **GetChanges(tipus)** à Es poden recollir les files canviades, esborrades, afegides, etc. O totes.

També a nivell **DataSet**, **Table** o **DataRow** :

Amb **AcceptChanges()** à S’accepten els canvis, fent que aquests siguin a partir d’aquell moment els valors originals de la fila. Això no vol dir que els canvis es guardin a la BD.

Amb **RejectChanges()** à Es cancel·len els canvis.

Amb **DataRow** es pot explorar els camps del registre en curs.

**DataRow(camp, RowVersion)** emmagatzema diferents versions de les dades del registre en curs :

**Original**: Les dades del registre al moment de carregar-se.

**Current**: El que hi ha en aquest moment.

**Default**: El registre resultant d’aplicar els valors per defecte a tots els camps del registre.

**Proposed**: El valor abans d’**EndEdit**.

Estats d’un registre. Valors de la propietat **DataRow:RowState** :

**Added**: Fila nova, (afegida)

**Deleted**: Registre esborrat.

**Detached** : Aquesta fila encara no forma part de cap taula.

**Modified**: Registre modificat

**Unchanged**: El registre no ha sofert canvis des de l’última acceptació de canvis.

### Relacions. **DataRelation** :

```
dsProductos.Relations.Add(New DataRelation("Productos_Categorias", _  
    dsProductos.Tables("Categorias").Columns("CategoryID"), _  
    dsProductos.Tables("Productos").Columns("CategoryID"))
```

Especifica una relació entre dues columnes de diferents taules declarades en un **DataSet**. Al exemple es relacionen les taules “Categorias” i “Productos”, mitjançant els seus respectius camps “CategoriaID”.

Després, des d’un **DataRow**; en una fila qualsevol de la taula “pare”, podem veure totes les files relacionades de la taula “Filla”. Mitjançant el mètode **DataRow.GetChildRows()**

El següent exemple recorre totes les “categories” i mostra els “productes” de cada categoria.

```
Dim rowCat As DataRow  
For Each rowCat In ds.Tables("Categorias").Rows  
    Dim nodCat As New TreeNode(rowCat("CategoryName"), 0, 0)  
    Dim rowProd As DataRow  
  
    For Each rowProd In rowCat.GetChildRows("Productos_Categorias")  
        Dim nodProd As New TreeNode(rowProd("ProductName"), 1, 1)  
        nodCat.Nodes.Add(nodProd)  
    Next  
  
    tree.Nodes.Add(nodCat)  
Next
```

### Carregar dades sense aplicar restriccions ni relacions :



Es possible que un **DataSet** tingui mes d'una taula; i que aquestes estiguin relacionades, o tinguin restriccions.

Mentre es carregen les dades a aquestes taules es possible que, temporalment, aquestes relacions i/o restriccions no s'acompleixin. Pot ser necessari deshabilitar les comprovacions mentre es carregen les dades :

**DataSet.EnforceConstraints = False** à Deshabilita totes les restriccions de totes les taules del DataSet. Després de la càrrega, es necessari tornar a habilitar-les.

**DataSet.Table.BeginLoadData** i **DataSet.Table.EndLoadData** à Habilita i deshabilita les comprovacions per una taula concreta del DataSet.

### DataAdapter:

**DataAdapter** serveix de pont entre una connexió a una base de dades i un **DataSet**. Conté les sentències necessàries per fer un SELECT, INSERT, UPDATE i DELETE. Encara que només es obligatòria la sentència SELECT.

#### Exemple de DataAdapter:

```
Dim con As New SqlConnection("data source=asus-08;initial catalog=northwind;integrated security=sspi")
Dim da As New SqlDataAdapter("SELECT * FROM products", con)
Dim ds As New DataSet
Try
    da.Fill(ds, "Products")
    Return ds.Tables("Products")
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

**DataAdapter** no necessita fer un **Open** de la connexió per carregar les dades. L'open al **DataAdapter** es implícit.

#### Mètodes de DataAdapter:

- **Fill**: Omple un DataSet a partir de la selecció declarada al dataAdapter.
- **Update**: Actualitza els canvis que des del DataSet s'hagin fet.

#### Propietats de DataAdapter:

- **SelectCommand**: L'única obligatòria.
- **InsertCommand**: Comanda que s'executarà quan es trobi una dada inserida des del DataSet.
- **UpdateCommand**: Comanda que s'executarà quan es trobi una dada modificada des del DataSet.
- **DeleteCommand**: Comanda que s'executarà quan es trobi una dada esborrada des del DataSet.

1 Veure : "DataSet" d'aquest capítol

### DataView :

**DataView** ordena i filtra una taula prèviament carregada. Només pot fer una vista de les dades d'una sola taula de qualsevol **DataSet**.

#### Exemple de DataView :

```
Dim dvProds As New DataView(TabProd)
dvProds.RowFilter = "categoryId = " & iCatId & " & Filtre.
dvProds.Sort = "ProductName ASC" & " \ Ordenació.
dg.DataSource = dvProds & " Es mostren les dades a un objecte Grid.
```



L'objecte **DataTable** de **DataSet** compta amb una propietat: **DefaultView** que crea un objecte de tipus **DataGridView** amb el bolcat de la totalitat d'una taula del **DataSet**.



---

## Capítol XII : Breu revisió d'ASP.NET

### **Arxiu Web.Config:**

Configuració de la aplicació ASP. De seguretat.. De compilació i depuració, etc.

Amb : `<Identity impersonate="True" />`  
S'habilita l'accés anònim al servei web.

451 – 454

---

### **Serveis web amb ASP:**

Es poden crear serveis web fent servir ASP. Seran mètodes que serien accessibles des de programes VB; amb `WebMethod()`.

455

### **WebMethod :**

Amb :  
`<WebMethod(>Public Function nom() As Tipus`  
Es declara una funció que serà publicada en un servei web, per ser accessible des de qualsevol aplicació.

---

### **WDSL.EXE:**

Amb la utilitat WDSL.EXE es genera un proxy d'un servei web, per poder utilitzar els seus mètodes des de qualsevol aplicació. S'aconsegueix el mateix afegint una referència web del servei web a la nostra aplicació.

460

---

### **WebRequest i WebResponse:**

Obtenir dades d'Internet amb les classes : `WebRequest` i `WebResponse`.

#### **Exemple de WebRequest i WebResponse :**

```
Dim url As New Uri("http://www.google.com")

Dim pregunta As System.Net.WebRequest
pregunta = System.Net.WebRequest.Create(url)

Dim resposta As System.Net.WebResponse
resposta = pregunta.GetResponse()

Dim lector As New System.IO.StreamReader(resposta.GetResponseStream())
Dim dada As String = lector.ReadToEnd()

Console.WriteLine(dada)
```

370

---



---

## Capítol XIII : Ensamblats .NET. Implementació i COM Interop.

### **Paquets d'instal·lació:**

Molts cops no serà necessari crear paquets d'instal·lació per les nostres aplicacions .NET. Serà suficient copiar el directori `\Bin` del nostre projecte.

463

---

### **L'arxiu .Config:**

L'arxiu `.Config` del projecte permet controlar permisos, ubicacions de DLLs dependents, etc.

467

---

### **L'arxiu ASSEMBLY.VB:**

L'arxiu `ASSEMBLY.VB` es l'arxiu de configuració del projecte .NET.

467

#### **Autoincrement de versió:**

Per activar l'autoincrement de versió cal fer servir "\*" a l'arxiu `ASSEMBLY.VB`.

Per publicar una llibreria al GAC es necessari que aquesta tingui "tancada" la versió. Es a dir, un numero de versió complet, en lloc de "\*".

- 1 Veure : "Registrar una biblioteca", d'aquest capítol.

470

---

### **ILDASM.EXE:**



El programa `ILDASM` permet visualitzar tota la informació i el manifest d'un programa compilat amb .NET.

468



### Compilació de components compartits :

#### GAC :

Espai lògic on s'emmagatzemen les llibreries que poden ser compartides entre diversos programes .NET.

471

#### GACUTIL.EXE:

Es pot obtenir una llista dels GACs amb GACUTIL.EXE.

472

#### SN.EXE:

Genera noms segurs, (noms únics), per compartir components. En el procés crea arxius .SNK

474

### Registrar una biblioteca :

- 1) Generar un arxiu de claus amb la comanda: **SN -k arxiu**  
[arxiu] es el nom del nou arxiu de claus.
- 2) A l'arxiu Assembly.vb del projecte del que es vol fer públic el component :
  - Donar un numero de versió tancat a la clàusula AssemblyVersion. Per exemple :  
<Assembly: AssemblyVersion("2.0.0.0")>
  - Declarar l'arxiu de claus anterior per al projecte del component. Amb la línia :  
<Assembly: AssemblyKeyFile("../..\\keyfile.snk")>
- 3) Compilem el projecte. Si consultem el component compilat amb ILDASM veurem la declaració de la clau a l'apartat de manifest.
- 4) Para registrar en la Global Assembly Caché, fem: **GACUTIL -i arxiu**  
On [arxiu] es el nom de l'arxiu de component que es vol publicar, amb extensió.
- 5) Fent **GACUTIL -l component**

On [component] es el nom del component ja publicat a la GAC; veurem la informació del component, així com les seves versions.

```
Microsoft (R) .NET Global Assembly Cache Utility. Version 1.1.4322.520
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.

The Global Assembly Cache contains the following assemblies:
  Name: null
  Version: 2.0.0.0
  Culture: neutral
  PublicKeyToken: e766b2349bb
  Custom: null

The cache of ngen files contains the following entries:
Number of items: 1
Name: null
Version: 2.0.0.0
Culture: neutral
PublicKeyToken: e766b2349bb
Custom: null
```

- 6) Perquè el component aparegui a la llista de components .NET :
  - Obrir **RegEdit**.
  - Sobre la carpeta **HKEY\_LOCAL\_MACHINE/Software/Microsoft/VisualStudio/7.0/AssemblyFolders**.  
Declareu una carpeta que servida de contenidor dels components que fem a partir d'aquell moment. Els nous components registrats hauran de copiar-se en aquesta carpeta.
  - A partir d'aquell moment, els components publicats amb GAC i copiats sobre aquesta carpeta seran visibles a la llista de components .NET utilitzables.



***Fer servir API :***

```
<DllImport("Arxiu.Dll")> Public Function NomFuncioDLL(paràmetres...)
```

---

477



## Conceptes bàsics de la programació OO :

### Abstracció, (classes i mètodes abstractes) :

Una classe abstracta es aquella en la que es defineixen un o mes mètodes sense implementar. Aquesta classe es pot derivar d'ella però no es pot instanciar. Les classes filles d'aquesta hauran d'implementar els mètodes abstractes.

#### Exemple d'abstracció:

```
Public MustInherit Class classe  
    Public MustOverride Sub mètode()  
End Class
```

1 Veure : “Classe abstracta” del capítol VI.

### Constructor :

Amb els constructors es crea una nova instància de la classe. Tota classe te almenys un constructor, encara que no es declari. Es **New()**. El programador pot definir tants constructors com necessiti; sempre amb **New()**. I es diferenciarien uns dels altres per els seus paràmetres, (nombre de paràmetres i/o tipus).

#### Mètodes i variables compartides o estàtiques :

Els mètodes o variables declarats amb **Shared** poden cridar-se des d'altres classes sense necessitat d'instanciar-les. Les accions que facin afecten a totes les instàncies de la classe. I les variables estàtiques son globals per a totes les instàncies. Per exemple :

```
Shared Public Funcion funció()
```

#### Constructor estàtic :

Una classe pot tenir un constructor estàtic sense paràmetres. Amb la signatura :

```
Public Shared Sub New()
```

Aquest constructor es cridarà un sol cop, la primera vegada que es faci servir la classe. I abans del constructor d'instància. Aquest tipus de constructor es pot fer servir per inicialitzar les variables estàtiques de la classe.

### Herència :

Una classe filla adquireix els mètodes i propietats d'una classe pare. Una forma d'*especialitzar* i *estendre* les capacitats de la classe pare. Per exemple :

```
Public Class Filla Inherits Pare
```

Només es pot heretar d'una classe. La herència múltiple es simula emprant interfaces.

### Interface :

Una Interface es una classe buida, en la que només venen definides els mètodes i propietats, sense cap implementació. Posteriorment qualsevol classe pot implementar els mètodes definits a la Interface. Per indicar que una classe implementa una interface es fa servir :

```
Public Class classe Implements interfaced1, interface2...
```

Posteriorment, a cada mètode de la interface que s'implementa a la classe s'indica :

```
Public Sub mètode() Implements Interface.mètode
```

S'utilitza per ordenació del codi i per simular la herència múltiple.



## 1 Veure : “Interface” del capítol VI

### **Polimorfisme :**

Es la possibilitat de definir múltiples classes de funcionalitats diferents, però amb mètodes i/o propietats definits de manera idèntica. Per exemple, una classe filla implementa la mateixa funció que la classe pare. El codi podrà cridar a una o altre segons necessiti el programador. Per indicar que la classe filla reimplementa un mètode de la classe pare, s'indica :

A la classe pare :  
`Overridable Sub funcio()`

A la classe filla :  
`Overrides Sub funcio()`

Un mètode `Overridable` pot ser o no reimplementat a la classe filla. Per indicar la obligatorietat de reimplementar un mètode es fa servir `MustOverride`. Però llavors s'ha de declarar la classe com abstracta.

El polimorfisme també es pot implementar mitjançant **Interfaces**. Els mètodes i funcions definides a una Interface no estan implementats, només declarats. Múltiples classes poden implementar aquests mètodes.

### **Sobrecàrrega :**

Dos definicions :

1. Diversos mètodes d'una mateixa classe tenen el mateix nom però diferent signatura.
2. Una classe filla reimplementa un mètode de la classe pare. Pot ser amb la mateixa signatura o no.

Per indicar la sobrecàrrega es fa servir la paraula `Overloads`.

### **Ombrejat :**

L'ombrejat es un tipus de sobrecàrrega que es fa servir per redefinir un o mes mètodes d'una classe pare en una classe filla i, alhora, impedir que la classe filla vegi la resta de mètodes sobrecarregats de la classe pare. Per aconseguir això es fa servir la paraula `Shadows` :

`Public Shadows Sub procediment(paràmetres...)`

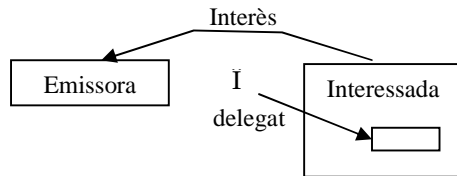
### **Exemple de sobrecàrrega. Exemple d'ombrejat :**

```
Public Class Pare
    Public Sub MetodeA()
    Public Sub MetodeA(int)
    Public Sub MetodeB()
    Public Sub MetodeB(int)
End Class
Public Class Derivada
    Public Overloads Sub MetodeA()
    Public Shadows Sub MetodeB()
End Class
```

Des de Derivada es pot veure `MetodeA()` de Derivada, `MetodeA(int)` i `MetodeB()` de Derivada.



### Declaració d'events :



La classe emissora publica un event amb:  
**Event** nom(paràmetres).

I per “disparar” l’event fa servir:  
**RaiseEvent** nom(paràmetres).

La classe Interessada declara la classe Emissora com:  
**Private WithEvents** obj As Emissora.

I declara un mètode amb qualsevol nom però mateixa signatura que l’event de la classe emissora, de la forma:  
**Private Sub** metode(paràmetrer) **Handler** event1, event2, ..., event\_n

1 Veure : “Gestió d’events i delegació” del capítol VI

### Exemple de declaració d’events :

#### Un formulari per mostrar l’exemple :

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Private WithEvents laClasseEvent As New ClasseEvent("Hola")

    Private Sub laClasseEvent_unEvent(ByVal pParametre As String) Handles
        laClasseEvent.unEvent
        MsgBox(pParametre)
    End Sub

    Private Sub btnDisparaEvent_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btnDisparaEvent.Click
        laClasseEvent.DisparaEvent()
    End Sub
End Class
```

#### Una classe per implementar l’event :

```
Public Class ClasseEvent
    Public Event unEvent(ByVal pParametre As String)

    Public Sub New(ByVal pParametre As String)
        Parametre = pParametre
    End Sub

    Private mParametre As String
    Public Property Parametre() As String
        Get
            Return mParametre
        End Get
        Set(ByVal pParametre As String)
            mParametre = pParametre
        End Set
    End Property

    Public Sub DisparaEvent()
        RaiseEvent unEvent(Parametre)
    End Sub
End Class
```



### Events dinàmics :

En qualsevol moment es pot reenviar un event existent a un mètode. Amb:  
**AddHandler** Objecte.Event, **AddressOf** mètode

I es pot treure amb:  
**RemoveHandler** Objecte.Event, **AddressOf** mètode

### Exemple d'events dinàmics :

El següent exemple implementa el comportament de l'event "click" de Button2, depenent del valor de Button1.

```
Imports System.Windows.Forms

Public Class Form1
    Inherits System.Windows.Forms.Form

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
        AddHandler Button2.Click, AddressOf MensajeCatala
    End Sub

    Private Sub SeleccioIdioma(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        If Button1.Text = "Català" Then
            Button1.Text = "Castellano"
            RemoveHandler Button2.Click, AddressOf MensajeCatala
            AddHandler Button2.Click, AddressOf MensajeCastellano
        Else
            Button1.Text = "Català"
            RemoveHandler Button2.Click, AddressOf MensajeCastellano
            AddHandler Button2.Click, AddressOf MensajeCatala
        End If
    End Sub

    Private Sub MensajeCatala(ByVal sender As Object, ByVal e As EventArgs)
        MsgBox("Missatge en català")
    End Sub

    Private Sub MensajeCastellano(ByVal sender As Object, ByVal e As EventArgs)
        MsgBox("Mensaje en castellanus")
    End Sub
End Class
```

1 Veure : "Dispose" amb AddHandler.



## Glossari

---

<

<Serializable()> · 24, 25  
<Sincronization()> · 31  
<ThreadStatic()> · 33

---

### A

Abort · 31, 33  
Abraçada mortal · 31  
Abstracció, (classes i mètodes abstractes) · 47  
Accedir a dades. Descripció · 36  
AcceptChanges · 40  
Activator · 35  
AddHandler · 14, 50  
AddHandler i IDisposable · 14  
AddressOf · 14, 50  
Afegir controls · 18  
Ajornar subprocessos · 29  
Àmbit de variables i mètodes · 4  
AppDomain · 27  
Append · 12  
ApplicationException · 16  
ArrayList · 6; Sort · 7  
Arxiu Web.Config · 43  
Arxius seqüencials. TextReader i TextWriter · 23  
ASP · 43  
Assembly · 27, 45  
ASSEMBLY.VB · 44  
AssemblyKeyFile · 45  
AssemblyVersion · 45  
AsyncCallback · 27  
Autoincrement de versió · 44

---

### B

BeginInvoke · 27  
BeginLoadData · 41  
BeginTrans · 38  
BinaryFormatter · 25  
BinaryReader · 23  
BinaryReader i BinaryWriter · 23  
BinaryWriter · 23  
Brush · 20

---

### C

canal · 34  
Canviar l'estat d'un control des d'un subprocés · 31  
Carregar dades sense aplicar restriccions ni relacions · 41  
Catch · 15

Catch. Altres temes · 15  
Changed · 26  
Channels · 34  
Classe abstracta · 10  
Classes aniuades · 8  
Classes per tractar arxius i directoris · 22  
Clone i Dispose · 11  
Close · 22  
CollectionBase · 6  
Columns · 40  
Com imprimir · 20  
Command.Transaction · 38  
Commit · 38  
Comparativa ADO & ADO.NET · 36  
CompareTo · 11  
Compartir dades amb Threads · 33  
Compilació de components compartits · 45  
Configuració de pàgina · 20  
Configurar impressores · 20  
Connection · 38  
Console.SetIn i Console.SetOut · 24  
Constructor · 47  
Constructor estàtic · 47  
Constructors compartits · 8  
Controls heretats · 17  
Create · 22  
Created · 26  
CurrentDomain · 27

---

### D

DataAdapter · 41  
DataReader · 37; GetInteger · 37; GetString · 37; GetValue · 37; NextResult · 37; Read · 37; RecordsAffected · 38  
DataRelation · 37, 40  
DataRow: RowState · 40; Added · 40; Deleted · 40; Detached · 40; Modified · 40; Unchanged · 40; RowVersion; Current · 40; Default · 40; Original · 40; Proposed · 40  
DataRow · 37, 40; RowVersion · 40  
DataRow · 40  
DataRow · 40  
DataRow.GetChildRows · 40  
DataSet · 37, 39, 40, 41, 42  
DataSet.ReadXml · 39  
DataSet.ReadXMLSchema · 39  
DataSet.WriteXml · 39  
DataTable · 37  
DataView · 37, 41  
Declaració d'events · 49  
Declarar un event predeterminat · 19  
Default Property · 7  
DefaultEvent · 19  
DefaultPageSettings · 20  
DefaultView · 42  
Delegate · 14  
Delegate Function · 27  
Delegats · 14



DELETE · 41  
DeleteCommand · 41  
Deleted · 26  
Deserialize · 25, 26  
Dibuix de gràfics a OnPaint · 19  
DiffGran · 39  
Dim Preserve · 6  
Directory · 22  
DirectoryInfo · 22  
DirectoryNotFoundException · 22  
Dispose · 4, 8, 14  
Dispose i Finalize · 8  
DLL · 19  
DLLImport · 46  
DLLs · 44  
Document · 21  
Dominis d'aplicació · 27, 34  
DrawRectangle · 19

Exemple de des-serialització · 25  
Exemple de File, FileInfo, Directory i DirectoryInfo · 22  
Exemple de FileSystemWatcher · 26  
Exemple de HashTable · 7  
Exemple de l'event PrintPage · 20  
Exemple de PageSetupDialog · 20  
Exemple de Path · 23  
Exemple de PrintDialog · 20  
Exemple de Remoting · 34  
Exemple de serialització · 24  
Exemple de sobrecàrrega · 48  
Exemple de SQLTransaction · 38  
Exemple de StringBuilder · 12  
Exemple de Threads · 28  
Exemple de Threads amb la implementació de IAsyncResult · 27  
Exemple de WebRequest i WebResponse · 43  
Exploració de registres · 40

---

## *E*

e.StackTrace · 15  
Each · 27  
EnableRaisingEvents · 26  
EndEdit · 40  
EndInvoke · 27  
EndLoadData · 41  
EndOfStreamException · 22  
EnforceConstraints · 41  
Entorns desconectats de dades. Conjunt de classes · 39  
Environment · 11  
EOF · 24  
Estructures · 9  
Event · 13, 49  
Events dinàmics · 50  
Events personalitzats · 13  
Excepcions amb Threads · 33  
Exception · 15, 16  
ExecuteNonQuery · 37  
ExecuteReader · 37  
ExecuteScalar · 37  
ExecuteXMLReader · 37  
Exemple d'abstracció · 47  
Exemple d'AppDomain · 27  
Exemple d'ArrayList · 6  
Exemple d'arxius seqüencials · 23  
Exemple d'assignació de prioritats a un subprocès · 29  
Exemple d'escriptura i lectura amb BinaryReader i BinaryWriter · 23  
Exemple d'events dinàmics · 50  
Exemple d'events personalitzats · 13  
Exemple d'impressió amb PrintPreviewDialog · 21  
Exemple d'impressió amb PrintPreviewControl · 21  
Exemple d'ombrejat · 48  
Exemple d'OnPaint · 19  
Exemple d'us d'OLEDB · 37  
Exemple d'us de SQLClient · 37  
Exemple de clonació d'objectes mitjançant serialització · 25  
Exemple de com crear un subprocés · 28  
Exemple de DataAdapter · 41  
Exemple de DataView · 42  
Exemple de declaració d'events · 49

---

## *F*

Fer servir API · 46  
File · 22  
FileInfo · 22  
FileLoadException · 22  
FileNotFoundException · 22  
FileStream · 24, 25  
FileSystemWatcher · 26  
Filter · 41  
Filter · 26  
Finalize · 8, 29  
Finally · 15  
Finestra de "Subprocessos" · 28  
Font · 20  
format · 34  
Format · 11  
Formatters · 24  
Formularis de diàleg · 18  
Formularis heretats · 19  
Fragment · 39  
Friend · 4, 8  
Funcions amb suport thread · 27

---

## *G*

GAC · 45  
GACUTIL · 45  
GACUTIL.EXE · 45  
GetChanges · 40  
GetChildRows · 40  
GetCommandLineArgs · 11  
GetType · 5  
Global Assembly Caché · 45  
Graphics · 19

---

## *H*

Handled · 19  
Handler · 49



Handles · 49  
HashChanges · 40  
Hashtable · 7  
HashTable · 7; ContainsKey · 7  
HasMorePages · 20  
Herència · 47  
Herència d'exception · 15  
Herència d'interfaces · 11  
HTTP · 34  
HttpChannel · 34, 35

---

## I

IAsyncResult · 27  
ICloneable · 11, 25  
IComparable · 11  
IComparer · 11  
IDisposable · 11, 14  
IgnoreSchema · 39  
ILDASM · 44  
ILDASM.EXE · 44  
Ilist · 6  
Implements · 10, 47  
InferSchema · 39  
Inherits · 11, 13, 17, 47  
Inici i finalització d'un subprocés, (Threads) · 28  
Iniciar una aplicació des d'un formulari amb Sub Main · 17  
Insert · 12  
INSERT · 41  
InsertCommand · 41  
Interface · 10, 11, 47, 48  
Interfaces · 48  
InterLocked · 31  
Interrupt · 29, 33  
Introduir comentaris a la 'Lista de tareas' · 4  
Invoke · 14, 32  
IsAlive · 29, 31

---

## J

Jerarquia d'exception · 16  
Jerarquia OLEDB i SQLClient · 37  
Join · 11, 29

---

## K

KeyPress · 19

---

## L

L'arxiu .Config · 44  
L'arxiu ASSEMBLY.VB · 44  
La classe ARRAYLIST · 6  
La classe Environment · 11  
La classe on es fa saltar l'event · 13  
La classe Path · 23  
La classe que captura l'event · 13

La classe Stream · 22  
Llegir des d'un DataAdapter · 39  
Llegir i guardar XML · 39

---

## M

MarsalByRefObject · 34  
Marshalbyrefobject · 34  
Me · 5  
MemoryStream · 25, 26  
Menús · 17  
Mes sobre FileInfo i DirectoryInfo · 22  
MessageBox · 17  
MethodInvoker · 32  
Mètodes abstractes · 10  
Mètodes compartits · 8  
Mètodes de DataAdapter · 41  
mètodes estàtics · 33  
Mètodes i variables compartides o estàtiques · 47  
MsgBox · 17  
MustInherit · 10, 47  
MustOverride · 10, 47  
MyBase · 5  
MyBase i Me · 5  
MyClass · 5

---

## N

Name Space · 9  
NextResult · 37  
Nodes · 40  
Nomenclatura SQL · 36  
NotInheritable · 10  
NotOverridable · 10

---

## O

ODBC · 36  
OLEDB · 36  
Ombrejat · 48  
OnPaint · 19  
OpenText · 24  
Option Strict ON · 4  
Optional · 7  
Overloads · 7, 48  
Overridable · 9, 48  
Overrides · 9, 10, 48

---

## P

PadLeft · 11  
PadRight · 11  
PageSetupDialog · 20  
Paquets d'instal·lació · 44  
Paradigma d'events en .NET · 13  
Path · 23, 26  
Peek · 24



Pen · 19  
Polimorfisme · 48  
Position · 23, 25  
PrintDialog · 20  
PrintDocument · 20  
PrinterSettings · 20  
PrintPage · 20  
PrintPreviewDialog · 21  
Prioritats en subprocessos · 29  
Priority · 29  
Private · 4  
Property · 7, 13  
Propietats de DataAdapter · 41  
Propietats semipúbliques · 7  
Protected · 4  
Public · 4  
Public Event · 32

---

## R

RaiseEvent · 13, 32, 49  
Read · 37  
ReadCommitted · 38  
ReadLine · 24  
ReadOnly · 7, 13  
ReadSchema · 39  
ReadString · 23  
ReadUncommitted · 38  
Reanimació de subprocessos · 29  
RecordSet · 37  
Rectangle · 19  
ReferenceEquals · 5  
Reflection · 27  
RegEdit · 45  
RegisterChannel · 34, 35  
RegisterWellKnownServiceType · 35  
Registrar una biblioteca · 45  
RejectChanges · 40  
Relacions. DataRelation · 40  
Relations · 40  
Remoting · 34  
RemoveHandler · 14, 50  
Renamed · 26  
RepeatableRead · 38  
ResetAbort · 29  
Restriccions dels mètodes que criden a subprocessos · 28  
Resume · 29  
Rollback · 38  
RowFilter · 42

---

## S

Selecció d'impressora i propietats d'impressora · 20  
SELECT · 41  
SelectCommand · 41  
sender · 13  
Serialització · 24  
Serializable · 38  
SerializationException · 24  
Serialize · 25, 26

Serveis web amb ASP · 43  
Shadows · 48  
Shared · 8, 33, 47  
ShowDialog · 21  
Sincronització de subprocessos · 30  
Sincronization · 31  
Sleep · 29  
SN · 45  
SN.EXE · 45  
Soap · 24  
SOAP · 34  
SoapFormatter · 25  
SOAP-XML · 24  
Sobrecàrrega · 48  
Sobrecàrrega, (Overloads) · 7  
Sort · 42  
Split · 11  
SQLClient · 36  
SqlCommand · 38  
SqlConnection · 38, 39  
SqlDataAdapter · 39, 41  
SqlTransaction · 38  
Start · 28, 32  
Stream · 22, 23  
StreamReader · 23  
StreamWriter · 23  
String · 11  
StringBuilder · 12  
subprocés · 28, 29  
Suspend · 29  
Suspendre subprocessos · 29  
SyncLock ... End SyncLock · 30  
System.Reflection · 27  
System.Reflection.Assembly · 27  
System.Runtime.Remoting · 34, 35  
System.Runtime.Remoting.Channels · 34, 35  
System.Runtime.Remoting.Channels.Http · 34, 35  
System.Runtime.Serialization · 24  
SystemException · 16

---

## T

Tables · 40  
TCP · 34  
TextReader · 23  
TextWriter · 23  
Thread · 28, 29, 31, 32  
ThreadAbortException · 33  
Threading · 31  
ThreadInterruptedException · 33  
ThreadPriority · 29, 32; AboveNormal · 29; BelowNormal · 29;  
Highest · 29; Lowest · 29; Normal · 29  
Threads · 28  
ThreadStart · 28, 32  
Tractament del teclat a KeyPress · 19  
Transaccions · 38  
TreeNode · 40  
ThreadAbortException · 29  
Try · 15  
Try... Catch... Finally · 15  
Type Of · 5  
TypeName · 5



---

## *U*

Una classe personalitzada especialitzada d'EventArgs · 13  
Update · 41  
UPDATE · 41  
UpdateCommand · 41

---

## *V*

Valors de ThreadPriority · 29  
Variables compartides, (estàtiques) · 8  
Vista prèvia de la impressió · 21

---

## *W*

WDSL.EXE · 43  
Web.Config · 43  
WebMethod · 43  
WebRequest · 43  
WebRequest i WebResponse · 43  
WebResponse · 43  
When · 15  
WithEvents · 13, 31, 49  
WriteOnly · 7

---

## *X*

XML · 34



## Notes



