# VB.NET y C#

# **VB.NET** y C# - Case Sensitivity

C# es case-sensitivity

```
system.console.writeline("HOLA"); INCORRECTO
System.Console.WriteLine("HOLA"); CORRECTO
```

# VB.NET y C# - Terminación de línea

C# la línea finaliza con un ;

### **VB.NET** y C# - Comentarios

C# soporta dos tipos de comentarios

```
// Comentario de una sola linea
string sName = "Juan";
/* Comentario con mas
de un renglon */
```

### VB.NET y C# - Declaración de Variables

C#: el tipo de variable precede al identificador

```
int x;
decimal y;
rectangle z;
Cliente cli;
```

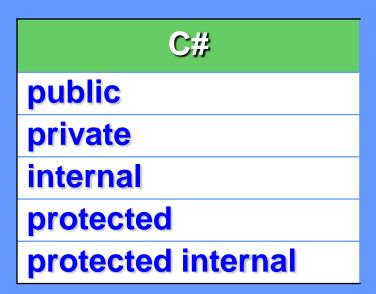
### VB.NET y C# - Inicialización de Variables

 C#: toda variable debe ser inicializada EXPLICITAMENTE antes de ser usada

```
int tempBalance; //variable local
//ERROR: tempBalance NO ha sido inicializada
System.Console.WriteLine(tempBalance);
```

# **VB.NET** y C# - Alcance de miembros

- Miembro: se refiere a los campos, propiedades, métodos, eventos, clases anidadas, etc.
- C#: todo miembro es declarado como PRIVATE por default



### VB.NET y C# - Sentencias condicionales

C#: sentencia if con varios formatos

### **VB.NET** y C# - Operadores Logicos

C#	VB.NET	Operador
&&	And	Operador logico Y
II.	Or	Operador logico O
!	Not	Negacion logica
==	=	Igual
!=	<b>&lt;&gt;</b>	Distinto

## **DEMO**

```
if (porciento >= 0 && porciento <= 100)
if (porciento > 50)
Console.WriteLine("Pasa");
else
Console.WriteLine("Error: fuera del intervalo");
```

# La instrucción switch

```
enum MesNombre { Enero, Febrero, ..., Diciembre }
MesNombre actual:
int mesDias;
switch (actual) {
case MesNombre.Febrero:
mesDias = 28;
break;
case MesNombre.Abril:
case MesNombre.Junio:
case MesNombre.Septiembre:
case MesNombre.Noviembre:
mesDias = 30;
break;
default:
mesDias = 31;
break;
Gabriela Marina Giles MCP – MCTS. Sitio: www.desarrolladoras.org.ar - E-mail: gilesgabriela@yahoo.com.ar
```



#### **VB.NET** y C# - Arreglos

C# utiliza corchetes [] para definición de arrays

```
string[] telefonos; //Definicion de un Arreglo de strings
telefonos = new string[3]; //De <u>3 elementos</u>
telefonos[0] = "1245"; //Seteo del 1er elemento del arreglo
//Definicion y asignacion de una vez
telefonos = new string[] {"1","2","3"};
```

### **VB.NET** y C# - Sentencia for

C#: la sentencia for consta de tres partes

```
//Partes: declaración, prueba, acción
for (int i=1; i < 10; i++)
{
}</pre>
```

### VB.NET y C# - Sentencia for/each

- For/Each permite recorrer arreglos y colecciones
- C#: usa la palabra foreach

```
string[] nombres = new string[5];
foreach(string auxNombre in nombres)
{
    //auxNombre es de SOLO LECTURA
}
```

### VB.NET y C# - Admin. De Excepciones

- Excepción: objeto que se genera cuando en tiempo de ejecución ocurre un error y contiene info sobre el mismo
- C#: usa las palabras try/catch/finally

```
try
   int resultado = x/y;
catch (DivideByZeroException e)
   //Error division por cero
catch
   //Otro error
finally
   //Siempre pasa por aca
}
```

### VB.NET y C# - Comp./Conv. de Tipos

- C# no permite conversiones implícitas de tipos
  - Si falla el casting se devuelve null o InvalidCastException

```
Cuenta cta = new CtaCte();
CtaCte cc = cta; //Error: puede que cta no sea una CtaCte
CtaCte cc = (CtaCte)cta; //Conversion explicita "CASTING"
CtaCte cc = cta as CtaCte; //Usando el operador "as"

if (cta is CtaCte) ... //Comp. con el operador "is"
```

### **VB.NET** y C# - Métodos

- Métodos: acciones que un objeto puede llevar a cabo.
- En C# todo método es una función

```
public void HacerDeposito(int importe) //No devuelve valor
{
}
public int ObtenerInventario(int codArticulo) //Devuelve un entero
{
}
```

### VB.NET y C# - Sobrecarga de Métodos

- Sobrecarga: varios métodos con el mismo nombre pero diferente "firma".
- C#

```
public void HacerDeposito(int importe)
{
}
public void HacerDeposito(int importe, bool acreditar)
{
}
```

### VB.NET y C# - Métodos "estáticos"

Miembros que no requieren de una instancia para ser invocados

• C#

```
public static void HacerDeposito(int importe)
{
}
```