C# and VB.NET Comparison Cheat Sheet

Comments

VB.NET

```
'Single line only
Rem Single line only
```

C#

```
// Single line
/* Multiple
line */
/// XML comments on single line
/** XML comments on multiple lines */
```

Program Structure

VB.NET

```
Imports System
Namespace MyNameSpace
  Class HelloWorld
    'Entry point which delegates to C-style main
Private Function
    Public Overloads Shared Sub Main()
      Main(System.Environment.GetCommandLineArgs())
    End Sub

  Overloads Shared Sub Main(args() As String)
    System.Console.WriteLine("Hello World")
  End Sub 'Main
  End Class 'HelloWorld End Namespace 'MyNameSpace
```

C#

```
using System
Namespace MyNameSpace
{
  class HelloWorld
  {
    static void Main(string[] args)
    {
      System.Console.WriteLine("Hello
World")
    }
  }
}
```

Data Types

VB.NET

```vb
'Value Types
Boolean
Byte
Char (example: "A")
Short, Integer, Long
Single, Double
Decimal
Date


'Reference Types
Object
String



Dim x As Integer
System.Console.WriteLine(x.GetType())
System.Console.WriteLine(TypeName(x))


'Type conversion
Dim d As Single = 3.5
Dim i As Integer = CType (d, Integer)
i = CInt (d)
i = Int(d)
```

C#

```cs
//Value Types
bool
byte, sbyte
char (example: 'A')
short, ushort, int, uint, long, ulong
float, double
decimal
DateTime


//Reference Types
object
string



int x;
Console.WriteLine(x.GetType())
Console.WriteLine(typeof(int))


//Type conversion
float d = 3.5;
int i = (int) d
```

Constants

VB.NET

```vb
Const MAX_AUTHORS As Integer = 25
ReadOnly MIN_RANK As Single = 5.00
```

C#

```cs
const int MAX_AUTHORS = 25;
readonly float MIN_RANKING = 5.00;
```

Enumerations

VB.NET

```vb
Enum Action
  Start
  'Stop is a reserved word
[Stop]
  Rewind
  Forward
End Enum

Enum Status
   Flunk = 50
   Pass = 70
   Excel = 90
End Enum

Dim a As Action = Action.Stop
If a <> Action.Start Then _
'Prints "Stop is 1"
   System.Console.WriteLine(a.ToString & " is " &
a)

'Prints 70
System.Console.WriteLine(Status.Pass)
'Prints Pass
System.Console.WriteLine(Status.Pass.ToString())


Enum Weekdays
   Saturday
   Sunday
   Monday
   Tuesday
   Wednesday
   Thursday
   Friday
End Enum 'Weekdays
```

C#

```cs
enum Action {Start, Stop, Rewind, Forward};
enum Status {Flunk = 50, Pass = 70, Excel =
90};

Action a = Action.Stop;
if (a != Action.Start)
//Prints "Stop is 1"
  System.Console.WriteLine(a + " is " +
(int) a);

// Prints 70
System.Console.WriteLine((int)
Status.Pass);
// Prints Pass
System.Console.WriteLine(Status.Pass);


enum Weekdays
{
  Saturday, Sunday, Monday, Tuesday,
Wednesday, Thursday, Friday
}
```

Operators

VB.NET

```
'Comparison
=  <  >  <=  >=  <>


'Arithmetic
+  -  *  /
Mod
\  (integer division)
^  (raise to a power)


'Assignment
=  +=  -=  *=  /=  \=  ^=  <<=  >>=  &=


'Bitwise
And  AndAlso  Or  OrElse  Not  <<  >>


'Logical
And  AndAlso  Or  OrElse  Not


'String Concatenation
&
```

C#

```
//Comparison
==  <  >  <=  >=  !=


//Arithmetic
+  -  *  /
%  (mod)
/  (integer division if both operands are
ints)
Math.Pow(x, y)


//Assignment
=  +=  -=  *=  /=  %=  &=  |=  ^=  <<=
>>=  ++  --


//Bitwise
&  |  ^  ~  <<  >>


//Logical
&&  ||  !


//String Concatenation
+
```

Choices

VB.NET

```vb
greeting = IIf(age < 20, "What's up?", "Hello")


'One line doesn't require "End If", no "Else"
If language = "VB.NET" Then langType = "verbose"


'Use: to put two commands on same line
If x <> 100 And y < 5 Then x *= 5 : y *= 2


'Preferred
If x <> 100 And y < 5 Then
  x *= 5
  y *= 2
End If



'or to break up any long single command use _
If henYouHaveAReally < longLine And _
itNeedsToBeBrokenInto2   > Lines  Then _
  UseTheUnderscore(charToBreakItUp)


If x > 5 Then
  x *= y
ElseIf x = 5 Then
  x += y
ElseIf x < 10 Then
  x -= y
Else
  x /= y
End If


'Must be a primitive data type
Select Case color
  Case "black", "red"
    r += 1
  Case "blue"
    b += 1
  Case "green"
    g += 1
  Case Else
    other += 1
End Select
```

C#

```cs
greeting = age < 20 ? "What's up?" :
"Hello";









if (x != 100 && y < 5)
{
  // Multiple statements must be enclosed
in {}
  x *= 5;
  y *= 2;
}









if (x > 5)
  x *= y;
else if (x == 5)
  x += y;
else if (x < 10)
  x -= y;
else
  x /= y;



//Must be integer or string
switch (color)
{
  case "black":
  case "red":    r++;
   break;
  case "blue"
   break;
  case "green": g++;
   break;
  default:    other++;
   break;
}
```

Loops

VB.NET

```vb
'Pre-test Loops:
While c < 10
  c += 1
End While Do Until c = 10
  c += 1
Loop


'Post-test Loop:
Do While c < 10
  c += 1
Loop


For c = 2 To 10 Step 2
  System.Console.WriteLine(c)
Next



'Array or collection looping
Dim names As String() = {"Steven", "SuOk", "Sarah"}
For Each s As String In names
  System.Console.WriteLine(s)
Next
```

C#

```cs
//Pre-test Loops: while (i < 10)
  i++;
for (i = 2; i < = 10; i += 2)
  System.Console.WriteLine(i);




//Post-test Loop:
do
  i++;
while (i < 10);



// Array or collection looping
string[] names = {"Steven", "SuOk",
"Sarah"};
foreach (string s in names)
  System.Console.WriteLine(s);
```

Arrays

VB.NET

```vb
Dim nums() As Integer = {1, 2, 3}
For i As Integer = 0 To nums.Length - 1
  Console.WriteLine(nums(i))
Next

'4 is the index of the last element, so it holds 5
elements
Dim names(4) As String
names(0) = "Steven"
'Throws System.IndexOutOfRangeException
names(5) = "Sarah"


'Resize the array, keeping the existing
'values (Preserve is optional)
ReDim Preserve names(6)




Dim twoD(rows-1, cols-1) As Single
twoD(2, 0) = 4.5


Dim jagged()() As Integer = { _
  New Integer(4) {}, New Integer(1) {}, New
Integer(2) {} }
jagged(0)(4) = 5
```

C#

```cs
int[] nums = {1, 2, 3};
for (int i = 0; i < nums.Length; i++)
  Console.WriteLine(nums[i]);


// 5 is the size of the array
string[] names = new string[5];
names[0] = "Steven";
// Throws System.IndexOutOfRangeException
names[5] = "Sarah"


// C# can't dynamically resize an array.
//Just copy into new array.
string[] names2 = new string[7];
// or names.CopyTo(names2, 0);
Array.Copy(names, names2, names.Length);



float[,] twoD = new float[rows, cols];
twoD[2,0] = 4.5;


int[][] jagged = new int[3][] {
  new int[5], new int[2], new int[3] };
jagged[0][4] = 5;
```

Functions

VB.NET

```vb
'Pass by value (in, default), reference
'(in/out), and reference (out)
Sub TestFunc(ByVal x As Integer, ByRef y As
Integer,
ByRef z As Integer)
  x += 1
  y += 1
  z = 5
End Sub


'c set to zero by default

Dim a = 1, b = 1, c As Integer
TestFunc(a, b, c)
System.Console.WriteLine("{0} {1} {2}", a, b, c) '1
2 5


'Accept variable number of arguments
Function Sum(ByVal ParamArray nums As Integer()) As
Integer
  Sum = 0
  For Each i As Integer In nums
    Sum += i
  Next
End Function 'Or use a Return statement like C#

Dim total As Integer = Sum(4, 3, 2, 1) 'returns 10


'Optional parameters must be listed last
'and must have a default value
Sub SayHello(ByVal name As String,
Optional ByVal prefix As String = "")
  System.Console.WriteLine("Greetings, " & prefix
& " " & name)
End Sub


SayHello("Steven", "Dr.")
SayHello("SuOk")
```

C#

```cs
// Pass by value (in, default), reference
//(in/out), and reference (out)
void TestFunc(int x, ref int y, out int z)
{
  x++;
  y++;
  z = 5;
}



int a = 1, b = 1, c; // c doesn't need
initializing
TestFunc(a, ref b, out c);
System.Console.WriteLine("{0} {1} {2}", a,
b, c); // 1 2 5


// Accept variable number of arguments
int Sum(params int[] nums) {
  int sum = 0;
  foreach (int i in nums)
    sum += i;
  return sum;
}


int total = Sum(4, 3, 2, 1); // returns 10


/* C# doesn't support optional
arguments/parameters.
Just create two different versions of the
same function. */
void SayHello(string name, string prefix) {
  System.Console.WriteLine("Greetings, " +
prefix + " " + name);
}

void SayHello(string name) {
  SayHello(name, "");
}
```

Exception Handling

VB.NET

```
Class Withfinally
   Public Shared Sub Main()
      Try
         Dim x As Integer = 5
         Dim y As Integer = 0
         Dim z As Integer = x / y
         Console.WriteLine(z)
      Catch e As DivideByZeroException
         System.Console.WriteLine("Error occurred")
      Finally
         System.Console.WriteLine("Thank you")
      End Try
   End Sub 'Main
End Class 'Withfinally
```

C#

```
class  Withfinally
{
  public static void Main()
  {
    try
    {
      int x = 5;
      int y = 0;
      int z = x/y;
      Console.WriteLine(z);
    }
    catch(DivideByZeroException e)
    {
      System.Console.WriteLine("Error
occurred");
    }
    finally
    {
      System.Console.WriteLine("Thank
you");
    }
  }
}
```

Namespaces

VB.NET

```
Namespace ASPAlliance.DotNet.Community
  ...
End Namespace


'or


Namespace ASPAlliance
  Namespace DotNet
    Namespace Community
      ...
    End Namespace
  End Namespace
End Namespace


Imports ASPAlliance.DotNet.Community
```

C#

```
namespace ASPAlliance.DotNet.Community {
  ...
}


// or


namespace ASPAlliance {
  namespace DotNet {
    namespace Community {
      ...
    }
  }
}


using ASPAlliance.DotNet.Community;
```

Classes / Interfaces

VB.NET

```
'Accessibility keywords
Public
Private
Friend
Protected
Protected Friend
Shared


'Inheritance
Class Articles
  Inherits Authors
  ...
End Class


Imports System


Interface IArticle
   Sub Show()
End Interface 'IArticle
 _


Class IAuthor
   Implements IArticle

   Public Sub Show()
      System.Console.WriteLine("Show() method
Implemented")
   End Sub 'Show

   'Entry point which delegates to C-style main
Private Function
   Public Overloads Shared Sub Main()
      Main(System.Environment.GetCommandLineArgs())
   End Sub


   Overloads Public Shared Sub Main(args() As
String)
      Dim author As New IAuthor()
      author.Show()
   End Sub 'Main
End Class 'IAuthor
```

C#

```
//Accessibility keywords
public
private
internal
protected
protected internal
static


//Inheritance
class Articles: Authors {
  ...
}


using System;


interface IArticle
{
  void Show();
}


class IAuthor:IArticle
{
  public void Show()
  {
    System.Console.WriteLine("Show() method
Implemented");
  }


  public static void Main(string[] args)
  {
    IAuthor author = new IAuthor();
    author.Show();
  }
}
```

Constructors / Destructors

VB.NET

```
Class TopAuthor
  Private _topAuthor As Integer

  Public Sub New()
    _topAuthor = 0
  End Sub

  Public Sub New(ByVal topAuthor As Integer)
    Me._topAuthor = topAuthor
  End Sub

  Protected Overrides Sub Finalize()
   'Desctructor code to free unmanaged resources
    MyBase.Finalize()
  End Sub
End Class
```

C#

```
class TopAuthor {
  private int _topAuthor;

  public TopAuthor() {
     _topAuthor = 0;
  }

  public TopAuthor(int topAuthor) {
    this._topAuthor= topAuthor
  }

  ~TopAuthor() {
    // Destructor code to free unmanaged
resources.
    // Implicitly creates a Finalize method
  }
}
```

Objects

VB.NET

```
Dim author As TopAuthor = New TopAuthor
With author
  .Name = "Steven"
  .AuthorRanking = 3
End With

author.Rank("Scott")
author.Demote() 'Calling Shared method
'or
TopAuthor.Rank()


Dim author2 As TopAuthor = author 'Both refer to
same object
author2.Name = "Joe"
System.Console.WriteLine(author2.Name) 'Prints Joe


author = Nothing 'Free the object


If author Is Nothing Then _
  author = New TopAuthor


Dim obj As Object = New TopAuthor
If TypeOf obj Is TopAuthor Then _
  System.Console.WriteLine("Is a TopAuthor
object.")
```

C#

```
TopAuthor author = new TopAuthor();

//No "With" construct
author.Name = "Steven";
author.AuthorRanking = 3;


author.Rank("Scott");
TopAuthor.Demote() //Calling static method



TopAuthor author2 = author //Both refer to
same object
author2.Name = "Joe";
System.Console.WriteLine(author2.Name)
//Prints Joe


author = null //Free the object


if (author == null)
  author = new TopAuthor();


Object obj = new TopAuthor();
if (obj is TopAuthor)
  SystConsole.WriteLine("Is a TopAuthor
object.");
```

Structs

VB.NET

```vb
Structure AuthorRecord
  Public name As String
  Public rank As Single

  Public Sub New(ByVal name As String, ByVal rank
As Single)
    Me.name = name
    Me.rank = rank
  End Sub
End Structure


Dim author As AuthorRecord = New
AuthorRecord("Steven", 8.8)
Dim author2 As AuthorRecord = author

author2.name = "Scott"
System.Console.WriteLine(author.name) 'Prints
Steven
System.Console.WriteLine(author2.name) 'Prints
Scott
```

C#

```cs
struct AuthorRecord {
  public string name;
  public float rank;

  public AuthorRecord(string name, float
rank) {
    this.name = name;
    this.rank = rank;
  }
}



AuthorRecord author = new
AuthorRecord("Steven", 8.8);
AuthorRecord author2 = author

author.name = "Scott";
SystemConsole.WriteLine(author.name);
//Prints Steven
System.Console.WriteLine(author2.name);
//Prints Scott
```

Properties

VB.NET

```
Private _size As Integer

Public Property Size() As Integer
  Get
    Return _size
  End Get
  Set (ByVal Value As Integer)
    If Value < 0 Then
      _size = 0
    Else
      _size = Value
    End If
  End Set
End Property


foo.Size += 1


Imports System


Class [Date]

   Public Property Day() As Integer
      Get
         Return day
      End Get
      Set
         day = value
      End Set
   End Property
   Private day As Integer


   Public Property Month() As Integer
      Get
         Return month
      End Get
      Set
         month = value
      End Set
   End Property
   Private month As Integer


   Public Property Year() As Integer
      Get
         Return year
      End Get
      Set
         year = value
      End Set
   End Property
   Private year As Integer


   Public Function IsLeapYear(year As Integer) As
Boolean
      Return(If year Mod 4 = 0 Then True Else
```

C#

```
private int _size;

public int Size {
  get {
    return _size;
  }
  set {
    if (value < 0)
      _size = 0;
    else
      _size = value;
  }
}



foo.Size++;


using System;
class Date
{
    public int Day{
        get {
            return day;
        }
        set {
            day = value;
        }
    }
    int day;


    public int Month{
        get {
            return month;
        }
        set {
            month = value;
        }
    }
    int month;


    public int Year{
        get {
            return year;
        }
        set {
            year = value;
        }
    }
    int year;


    public bool IsLeapYear(int year)
    {
        return year%4== 0 ? true: false;
    }
    public void SetDate (int day, int
```

```
False)
   End Function 'IsLeapYear

   Public Sub SetDate(day As Integer, month As
Integer,
year As Integer)
      Me.day = day
      Me.month = month
      Me.year = year
   End Sub 'SetDate
End Class '[Date]
```

```
month, int year)
    {
        this.day   = day;
        this.month = month;
        this.year  = year;
    }
}
```

Delegates / Events

VB.NET

```
Delegate Sub MsgArrivedEventHandler(ByVal message
As String)


Event MsgArrivedEvent As MsgArrivedEventHandler


'or to define an event which declares a
'delegate implicitly
Event MsgArrivedEvent(ByVal message As String)


AddHandler MsgArrivedEvent, AddressOf
My_MsgArrivedCallback
'Won't throw an exception if obj is Nothing
RaiseEvent MsgArrivedEvent("Test message")
RemoveHandler MsgArrivedEvent, AddressOf
My_MsgArrivedCallback



Imports System.Windows.Forms


'WithEvents can't be used on local variable
Dim WithEvents MyButton As Button
MyButton = New Button


Private Sub MyButton_Click(ByVal sender As
System.Object, _
  ByVal e As System.EventArgs) Handles
MyButton.Click
  MessageBox.Show(Me, "Button was clicked", "Info",
_
    MessageBoxButtons.OK,
MessageBoxIcon.Information)
End Sub
```

C#

```
delegate void MsgArrivedEventHandler(string
message);


event MsgArrivedEventHandler
MsgArrivedEvent;


//Delegates must be used with events in C#


MsgArrivedEvent += new
MsgArrivedEventHandler
  (My_MsgArrivedEventCallback);
//Throws exception if obj is null
MsgArrivedEvent("Test message");
MsgArrivedEvent -= new
MsgArrivedEventHandler
  (My_MsgArrivedEventCallback);



using System.Windows.Forms;


Button MyButton = new Button();
MyButton.Click += new
System.EventHandler(MyButton_Click);


private void MyButton_Click(object sender,
System.EventArgs e) {
  MessageBox.Show(this, "Button was
clicked", "Info",
    MessageBoxButtons.OK,
MessageBoxIcon.Information);
}
```

Console I/O

VB.NET

```
'Special character constants
vbCrLf, vbCr, vbLf, vbNewLine
vbNullString
vbTab
vbBack
vbFormFeed
vbVerticalTab
""
Chr(65) 'Returns 'A'


System.Console.Write("What's your name? ")
Dim name As String = System.Console.ReadLine()
System.Console.Write("How old are you? ")
Dim age As Integer = Val(System.Console.ReadLine())
System.Console.WriteLine("{0} is {1} years old.",
name, age)
'or
System.Console.WriteLine(name & " is " & age & "
years old.")

Dim c As Integer
c = System.Console.Read() 'Read single char
System.Console.WriteLine(c) 'Prints 65 if user
enters "A"
```

C#

```
//Escape sequences
\n, \r
\t
\\
\


Convert.ToChar(65) //Returns 'A' -
equivalent to Chr(num) in VB
// or
(char) 65


System.Console.Write("What's your name? ");
string name = SYstem.Console.ReadLine();
System.Console.Write("How old are you? ");
int age =
Convert.ToInt32(System.Console.ReadLine());
System.Console.WriteLine("{0} is {1} years
old.", name, age);
//or
System.Console.WriteLine(name + " is " +
age + " years old.");


int c = System.Console.Read(); //Read
single char
System.Console.WriteLine(c); //Prints 65 if
user enters "A"
```

File I / O

VB.NET

```vb
Imports System.IO


'Write out to text file
Dim writer As StreamWriter = File.CreateText
  ("c:\myfile.txt")
writer.WriteLine("Out to file.")
writer.Close()


'Read all lines from text file
Dim reader As StreamReader = File.OpenText
  ("c:\myfile.txt")
Dim line As String = reader.ReadLine()
While Not line Is Nothing
  Console.WriteLine(line)
  line = reader.ReadLine()
End While
reader.Close()


'Write out to binary file
Dim str As String = "Text data"
Dim num As Integer = 123
Dim binWriter As New BinaryWriter(File.OpenWrite
  ("c:\myfile.dat"))
binWriter.Write(str)
binWriter.Write(num)
binWriter.Close()


'Read from binary file
Dim binReader As New BinaryReader(File.OpenRead
  ("c:\myfile.dat"))
str = binReader.ReadString()
num = binReader.ReadInt32()
binReader.Close()
```

C#

```cs
using System.IO;


//Write out to text file
StreamWriter writer = File.CreateText
  ("c:\\myfile.txt");
writer.WriteLine("Out to file.");
writer.Close();


//Read all lines from text file
StreamReader reader = File.OpenText
  ("c:\\myfile.txt");
string line = reader.ReadLine();
while (line != null) {
  Console.WriteLine(line);
  line = reader.ReadLine();
}
reader.Close();


//Write out to binary file
string str = "Text data";
int num = 123;
BinaryWriter binWriter = new
BinaryWriter(File.OpenWrite
  ("c:\\myfile.dat"));
binWriter.Write(str);
binWriter.Write(num);
binWriter.Close();


//Read from binary file
BinaryReader binReader = new
BinaryReader(File.OpenRead
  ("c:\\myfile.dat"));
str = binReader.ReadString();
num = binReader.ReadInt32();
binReader.Close();
```