VB.NET - BASIC SYNTAX

VB.Net is an object-oriented programming language. In Object-Oriented Programming methodology, a program consists of various objects that interact with each other by means of actions. The actions that an object may take are called methods. Objects of the same kind are said to have the same type or, more often, are said to be in the same class.

When we consider a VB.Net program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods and instant variables mean.

- **Object** - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating, etc. An object is an instance of a class.
- **Class** - A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- **Methods** - A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instant Variables** - Each object has its unique set of instant variables. An object's state is created by the values assigned to these instant variables.

A Rectangle Class in VB.Net

For example, let us consider a Rectangle object. It has attributes like length and width. Depending upon the design, it may need ways for accepting the values of these attributes, calculating area and displaying details.

Let us look at an implementation of a Rectangle class and discuss VB.Net basic syntax on the basis of our observations in it:

```
Imports System
Public Class Rectangle
    Private length As Double
    Private width As Double

    'Public methods
    Public Sub AcceptDetails()
        length = 4.5
        width = 3.5
    End Sub

    Public Function GetArea() As Double
        GetArea = length * width
    End Function
    Public Sub Display()
        Console.WriteLine("Length: {0}", length)
        Console.WriteLine("Width: {0}", width)
        Console.WriteLine("Area: {0}", GetArea())

    End Sub

    Shared Sub Main()
        Dim r As New Rectangle()
        r.Acceptdetails()
        r.Display()
        Console.ReadLine()
    End Sub
End Class
```

When the above code is compiled and executed, it produces the following result:

```
Length: 4.5
Width: 3.5
Area: 15.75
```

In previous chapter, we created a Visual Basic module that held the code. Sub Main indicates the entry point of VB.Net program. Here, we are using Class that contains both code and data. You use classes to create objects. For example, in the code, r is a Rectangle object.

An object is an instance of a class:

```
Dim r As New Rectangle()
```

A class may have members that can be accessible from outside class, if so specified. Data members are called fields and procedure members are called methods.

Shared methods or **static** methods can be invoked without creating an object of the class. Instance methods are invoked through an object of the class:

```
Shared Sub Main()
    Dim r As New Rectangle()
    r.Acceptdetails()
    r.Display()
    Console.ReadLine()
End Sub
```

Identifiers

An identifier is a name used to identify a class, variable, function, or any other user-defined item. The basic rules for naming classes in VB.Net are as follows:

- A name must begin with a letter that could be followed by a sequence of letters, digits $0-9$ or underscore. The first character in an identifier cannot be a digit.
- It must not contain any embedded space or symbol like ? - +! @ # % ^ & * [ ] { } . ; : " ' / and \. However, an underscore _ can be used.
- It should not be a reserved keyword.

VB.Net Keywords

The following table lists the VB.Net reserved keywords:

AddHandler	AddressOf	Alias	And	AndAlso	As	Boolean
ByRef	Byte	ByVal	Call	Case	Catch	CBool
CByte	CChar	CDate	CDec	CDbl	Char	CInt
Class	CLng	CObj	Const	Continue	CSByte	CShort
CSng	CStr	CType	CUInt	CULng	CUShort	Date
Decimal	Declare	Default	Delegate	Dim	DirectCast	Do
Double	Each	Else	ElseIf	End	End If	Enum
Erase	Error	Event	Exit	False	Finally	For
Friend	Function	Get	GetType	GetXML Namespace	Global	GoTo
Handles	If	Implements	Imports	In	Inherits	Integer

Interface	Is	IsNot	Let	Lib	Like	Long
Loop	Me	Mod	Module	MustInherit	MustOverride	MyBase
MyClass	Namespace	Narrowing	New	Next	Not	Nothing
Not Inheritable	Not Overridable	Object	Of	On	Operator	Option
Optional	Or	OrElse	Overloads	Overridable	Overrides	ParamArray
Partial	Private	Property	Protected	Public	RaiseEvent	ReadOnly
ReDim	REM	Remove Handler	Resume	Return	SByte	Select
Set	Shadows	Shared	Short	Single	Static	Step
Stop	String	Structure	Sub	SyncLock	Then	Throw
To	True	Try	TryCast	TypeOf	UInteger	While
Widening	With	WithEvents	WriteOnly	Xor		