UNIT 1

INTRODUCTION TO MICROSOFT .NET FRAMEWORK & BASICS OF VB .NET

Outline of the Chapter

1. Overview of Microsoft .NET Framework
2. Principal Design Features
3. Features of VB .NET
4. Architecture of .NET Framework
5. The .NET Framework components
6. Common Language Runtime (CLR) Environment
7. Common Type System
8. Common Language Specification
9. The .NET Framework class Library
10. Managed and Unmanaged Code
11. Namespaces
12. Getting Started with Visual Basic .NET
13. Visual Basics Language Concepts
14. GTU Questions

1. OVERVIEW OF MICROSOFT .NET FRAMEWORK:

The Dot Net or .Net is a technology that is an outcome of Microsoft's new strategy to develop window based robust applications and rich web applications and to keep Windows the dominant operating system in the market.

Dot NET technology enables the creations and use of

- COM+ (Component Object Model – Binary Interface Standard) component services,
- XML and objects oriented design,
- Support for new web services protocols such as SOAP (Simple Object Access Protocol),
- WSDL (Web Service Definition Language) and
- UDDI (Universal Description, Discovery and Integration - standard method for publishing and discovering the network-based software components)

and a focus on the internet, all integrated within the distributed internet applications architecture.

Microsoft has devoted enormous resources to the development of .NET and its associated technologies. The results of this commitment to date are impressive.

Using Dot Net architecture, it is easy to reduce time and cost for developing and maintaining enterprise business applications.

2. PRINCIPAL DESIGN FEATURES:

I. Interoperability:

The .Net framework provides a lot of **backward support**. Suppose if you had an application built on an older version of the .Net framework, say 2.0. And if you tried to run the same application on a machine which had the higher version of the .Net framework, say 3.5.

The application would still work. This is because with every release, Microsoft ensures that older framework versions gel well with the latest version.

II. Portability:

Applications built on the .Net framework can be made to **work on any Windows platform**. And now in recent times, Microsoft is also envisioning to make Microsoft products work on other platforms, such as iOS and Linux.

III. Security:

The .NET Framework has a good security mechanism. The inbuilt security mechanism helps in both **validation and verification** of applications.

IV. Language Independence:

.NET framework includes a Common Type System or CTS. CTS define all possible data types and programming constructs support by the CLR.

Due to this feature it supports exchange of types and object instances between libraries and applications written using any .NET language.

V. Common Runtime Engine:

All .NET **programs execute under the supervision of Common Language Runtime** guarantees behavior and certain properties in the area of memory management, security and exception

handling.

The .Net framework has all the capability to see those resources, which are not used by a running program. It would then release those resources accordingly. This is done via a program called the "**Garbage Collector**" which runs as part of the .Net framework.

The garbage collector **runs at regular intervals** and keeps on checking which system resources are not utilized, and frees them accordingly.

VI. Base Class Library:

Base class library (BCL) provides classes which encapsulate numbers of common functions, including the file reading and writing, graphics rendering, database interaction, XML document manipulation and so on.

VII. Debugging Support:

The .NET framework has debugging built into it at a very low level, which runs all the code anyway, manages debugging and sends event to the connected debugger whenever needed.

VIII. Simplified deployment:

The .Net frameworks also have tools, which can be used to package applications built on the .Net framework.

These packages can then be distributed to client machines. The packages would then automatically install the application. It also confirms to security requirements.

3. FEATURES OF VB .NET:

OOP: Vb.net is pure object oriented Programming language. It has many functionalities like inheritance, polymorphism etc. that helps developer to create robust application.

Windows Forms: Using Vb.net developer can easily make windows form application. By inherit existing classes user can create windows form application.

Structured exception handling: Enables you to create more efficient and robust error handlers by using structured exception handling, which allows you to detect and remove errors at runtime.

Multithreading: Enables your applications to handle multiple tasks simultaneously.

Automatic garbage collection: Vb.net does automatic garbage collection. So developer needs to focus on user requirements only.

Events: Vb.net is **event driven programming**. It provides best GUI experience to users.

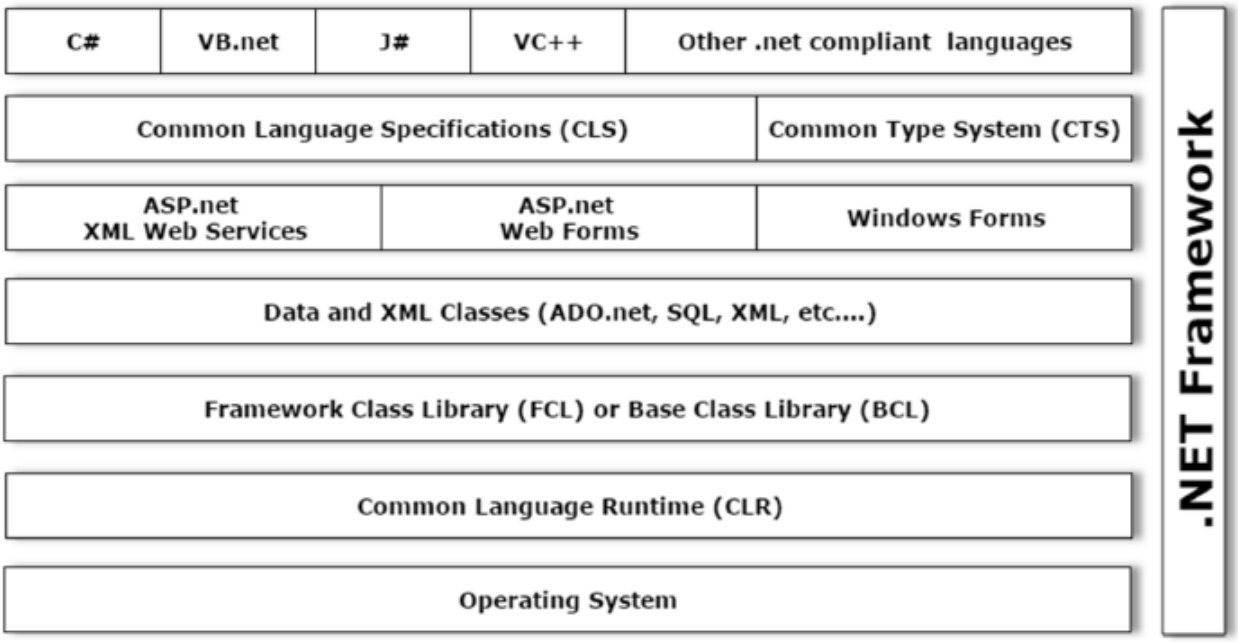
Assemblies: An assembly is a portable executable file that contains a compiled portion of code. All the .NET assemblies contain the definition of types, versioning information for the type, meta-data, and manifest.

4. ARCHITECTURE OF .NET FRAMEWORK:

The .NET Framework sits on top of the operating system, which can be any flavor of Windows and consists of a number of components, currently including.

- Five official languages: C#, VB, Visual C++, Visual J# and Jscript.NET, ASP .NET
- The CLR, an object oriented platform for windows and web development that all that language shares.

- A number of related class libraries, collectively known as the framework class library.



5. .NET FRAMEWORK COMPONENTS:

A. Language Compiler:

- The **top layer** contains the language compiler.
- In .NET Framework developers has freedom to use any language to develop any kind of application.
- .NET Framework is fully based on CLR **(Common Language Runtime)**. It is core component of Microsoft .NET development platforms.
- Programmers uses any languages like VB, C# etc. to write the program and **CLR is responsible for executing that code to Intermediate Language** [MSIL (Microsoft Intermediate Language) / IL (Intermediate Language)].
- **Language compilers** (e.g. C#, VB.NET J#) will convert the Source code / Program to Microsoft Intermediate International Language (MSIL) intern this will be **converted to Native Code by CLR**.

B. Common Language Specification:

- Common Language Specification (CLS) is a **set of basic language features** and **set of rules** that .Net Languages needed to develop Applications and Services, which are compatible with the .Net Framework.
- When there is a **situation to communicate Objects written in different .Net Complaint languages**, those objects must expose the features that are common to all the languages. It helps in cross language inheritance and cross language debugging.
- Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language used to create the application.

C. Web Application and Windows Application:

- The types of applications that can be built in the .Net framework are classified broadly into the following categories.

- **Win Forms** will allow you to developing Forms-based applications (traditional executable applications, which would run on an end user machine. Notepad is an example of a client-based application.
- **ASP.Net** should not be viewed as the next version of (Active Server Pages) ASP, but as dramatically new shift in web application development.
- Using ASP.NET, it is now possible to build robust **web-based** applications, which are made to run on any browser such as Internet Explorer, Chrome or Firefox.
- Web forms and Windows forms allows you to apply rapid Application development techniques to build web and windows applications.

D. Data and XML classes (ADO .NET):

- These classes extend the FCL (Framework Class Library) to support data management and XML manipulation.
- ADO.Net (ActiveX Data Object): This technology is used to develop applications to interact with Databases such as Oracle or Microsoft SQL Server.
- A set of classes called ADO.NET extends the base classes and allows you to manipulate or managing the data to database and maintains the XML data manipulations.
- These all classes support database management with the help of Structured Query Language SQL and it also allows you to change the data.
- The .NET framework also supports the classes which will help you to maintain the XML data and perform searching and translations.

E. Base Class Library:

- The .Net Framework class library (FCL) provides the core functionality of .Net Framework architecture.
- It includes a huge collection of reusable classes, interfaces, and value types that expedite and optimize the development process and provide access to system functionality.
- The .Net Framework class library (FCL) organized in a hierarchical tree structure and it is divided into Namespaces.
- Namespaces is a logical grouping of types for the purpose of identification. Framework class library (FCL) provides the consistent base types that are used across all .NET enabled languages.
- The .Net Framework class library (FCL) classes are object oriented and easy to use in program developments.

F. Common Language Runtime:

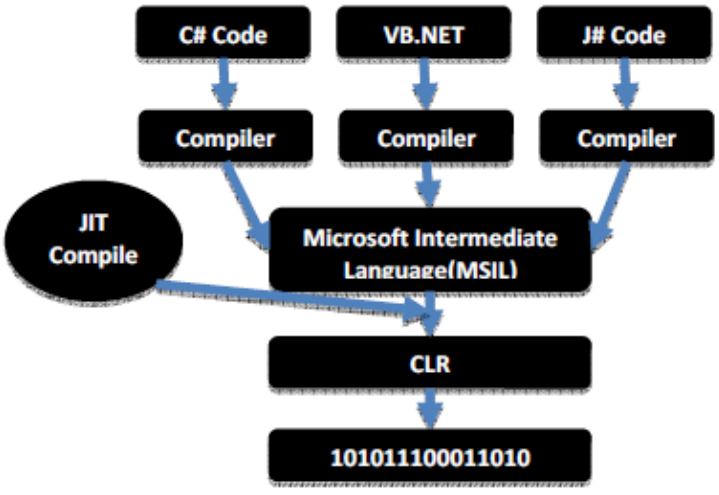
- In .NET Framework developers has freedom to use any language to develop any kind of application.
- .NET Framework is fully based on CLR. It is core component of Microsoft .NET development platforms.
- Programmers uses any languages like VB, C# etc to write the program and CLR is responsible for executing that code to Intermediate Language [MSIL (Microsoft Intermediate Language) / IL (Intermediate Language)].

G. Operating System:

- All most all the Microsoft operating system supports the .NET framework.
- Microsoft .NET supports not only language independence but also language integration. This means that you can inherit from classes, catch exceptions, and take advantages of polymorphism across different languages.

6. COMMON LANGUAGE RUNTIME (CLR):

- Common Language Runtime (CLR) is a **core element of the .NET Framework** that is **responsible** for the **runtime management** of a .NET program. CLR provides the environment where .NET programs are executed.
- The CLR **includes a virtual machine**, which is similar to Java virtual machine in many ways.
- CLR **manages the execution of programs** written in different supported languages. CLR transforms source code into a form of byte code known as Common Intermediate Language (CIL). At run time.
- Developers write code in a supported .NET language, such as C# or VB.Net. The .NET compiler then converts it into CIL code. During run time, the CLR converts the CIL code into something that can be understood by the operating system.
- CLR activates objects, perform security checks on them, lays them out in memory, execute them and garbage collects them.
- CLR can manage the execution of all supported languages by transforming them to byte code and then into the native code for the chosen platform.
- It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation of your .NET program.



- In .NET, programs are not compiled into executable files, they are compiled into Microsoft Intermediate Language (MSIL) files, which the CLR then executes.
- The MSIL files that VB .NET produced are identical to the IL files that other .NET language produces.
- Key fact about CLR is that it is common, the same runtime support development in C# as well as in VB. NET.

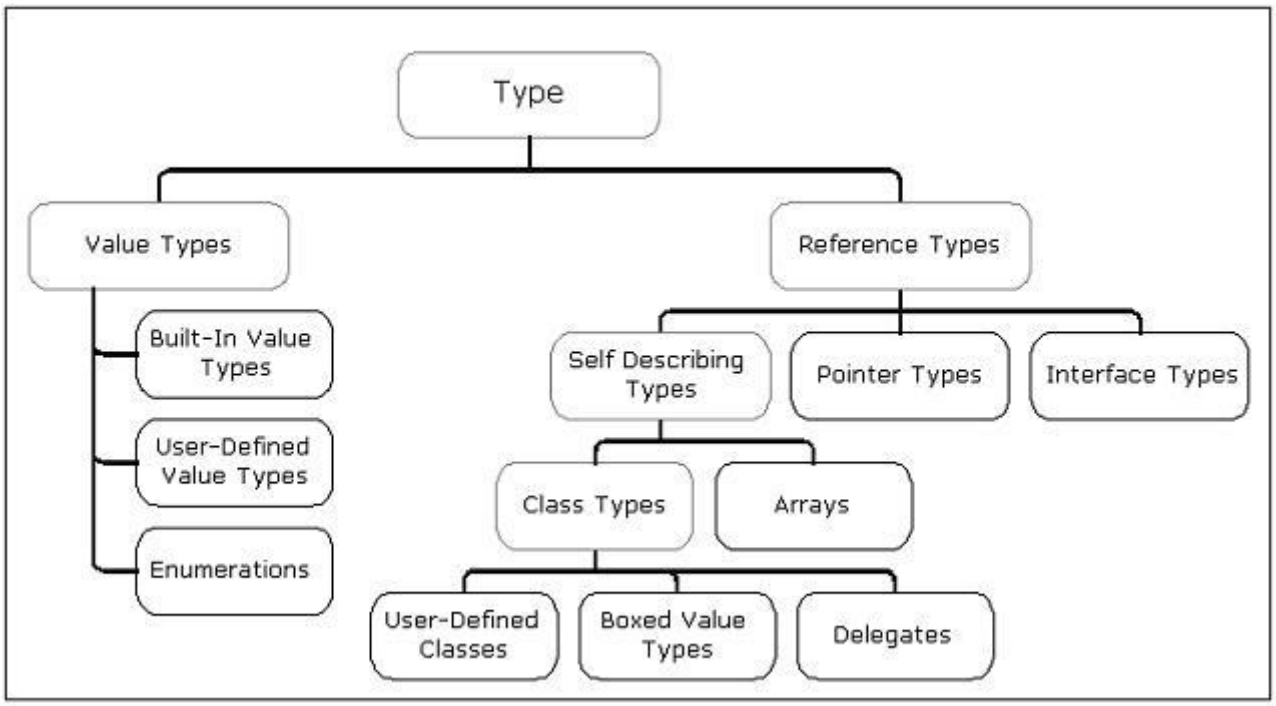
- VB .NET code is compiled into IL when you build your project. The IL is saved in a file on a disk.
- When you run your program, the IL is compiled again, using the just in time compiler. This process is called as JITing. This results in machine code, executed by machine's processor.
- The standard JIT compiler runs on demand. When a method is called, the JIT compiler analyzes the IL and produces highly efficient machine code, which runs very fast.
- The JIT compiler smart enough to recognize that is the code has already been compiled or not, so as application runs, compilation happens only as needed.
- As .NET applications run, they tend to become faster and faster, as the already compiled code is reused.
- The CLS means that all .NET languages produce very similar IL code. As a result, objects created in one language can be accessed and derived from another.

Just-In-Time Compiler:

- In **just-in-time (JIT) compilation**, also known as **dynamic translation** is compilation done during execution of a program – at run time – rather than prior to execution just-in-time (JIT) compilation, also known as dynamic translation.
- When the Microsoft .NET Common Language Runtime is installed on a computer then it can run any language that is Microsoft .NET compatible.
- Before the Microsoft Intermediate Language (MSIL) can be executed, it must be converted by a .NET Framework Just-In-Time (JIT) compiler to native code, which is CPU-specific code that runs on the same computer architecture as the JIT compiler.

7. COMMON TYPE SYSTEM (CTS):

- It describes set of data types that can be used in different .Net languages in common. i.e. CTS ensures that objects written in different .Net languages can interact with each other.
- For Communicating between programs written in any .NET complaint language, the types have to be compatible on the basic level.
- As .NET Framework is language independent and support over 20 different programming languages, many programming language will write data types in their own programming language.
- For Example, an integer variable in C# written as int, where as in visual basic it is written as integer.
- Therefore in .NET Framework you have single class called as System.Int32 to interpret these variables. Similarly for the ArrayList data type .NET Framework has a common type called System.Collections.ArrayList.
- CTS is a standard that specifies how type definitions and specific values of types are represented in computer memory. It is intended to allow programs written in different programming languages to easily share information.
- CTS in .NET Framework define how data types are going to be declared and managed runtime.



- The Common Type System (CTS) supports two general categories of types:
- **Value types:** Value types directly contain their data, and instances of value types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.
- **Reference types:** Reference types store a reference to the value's memory address, and are allocated on the heap structure.

  Reference types can be self-describing types, pointer types, or interface types.

  The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types, and delegates.

Functions of CTS:

- To establish a framework that helps enable cross-language integration, type safety, and high performance code execution.
- To provide an object-oriented model that supports the complete implementation of many programming languages.
- To define rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.
- The CTS also defines the rules that ensures that the data types of objects written in various languages are able to interact with each other.
- The CTS also specifies the rules for type visibility and access to the members of a type, i.e. the CTS establishes the rules by which assemblies form scope for a type, and the Common Language Runtime enforces the visibility rules.
- The CTS defines the rules governing type inheritance, virtual methods and object lifetime.
- Languages supported by .NET can implement all or some common data types.

8. COMMON LANGUAGE SPECIFICATION (CLS):

- CLS defines the base rules necessary for any language targeting common language infrastructure to interoperate with other CLS-compliant languages.
- For example, a method with parameter of "unsigned int" type in an object written in C# is not CLS-compliant, just as some languages, like VB.NET, do not support that type.
- CLS represents the guidelines to the compiler of a language, which targets the .NET Framework. CLS-compliant code is the code exposed and expressed in CLS form.
- Even though various .NET languages differ in their syntactic rules, their compilers generate the Common Intermediate Language instructions, which are executed by CLR.
- Hence, CLS allows flexibility in using non-compliant types in the internal implementation of components with CLS-compliant requirements. Thus, CLS acts as a tool for integrating different languages into one umbrella in a seamless manner.
- Common Language Specification (CLS) is a set of basic language features that .NET languages needed to develop applications and services, which are compatible with the .NET framework.
- Where there is situation to communicate objects written in different .NET compliant languages, those objects must expose the features that are common to all the languages.
- Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language use to create the application.
- Common Language Specification (CLS) defines a subset of Common Type System (CTS). It describes a set of types that can use different .NET languages have in common, which ensure that objects written in different languages can interact with each other.
- Most of the members defined by types in the .NET Framework Class Library (FCL) are common language specification compliant types.

9. THE .NET FRAMEWORK CLASS LIBRARY (FCL):

- The .NET Framework class library is a library consisting of namespaces, classes, interfaces and data types included in the .NET Framework.
- FCL contains huge collection of reusable classes, interfaces and value types that expedite and optimize the development process and provide access to system functionality.
- FCL is organized into hierarchical structure and it is divided into Name Spaces. This library is organized into namespaces that contain functionally related groups of classes.
- These namespaces are divided among different categories, such as data access, common types, debugging, file access, network communication, security, Windows applications, Web applications, Web services, XML data etc. The system library is the root for types in the .NET framework.
- FCL is object oriented and easy to use in program developments. We can also use third party components with .NET Framework.
- The Base Class Library (BCL) is the core of the FCL and provides the most fundamental functionality, which includes classes in namespaces System, System.CodeDom, System.Collections, System.Diagnostics, System.Globalization, System.IO, System.Resources and System.Text.

10. MANAGED CODE AND UNMANAGED CODE:

Managed Code:

- Managed code is the code that is written to target the services of the managed runtime execution environment such as Common Language Runtime in .Net Technology.
- The Managed Code running under a Common Language Runtime cannot be accessed outside the runtime environment as well as cannot call directly from outside the runtime environment.
- Managed code uses CLR which in turns looks after your applications by managing memory, handling security allowing cross language debugging, and so on.
- The MSIL is kept in a file called as assembly, along with metadata that describes classes, methods and attributes of code created by programmer.
- This assembly is the one-stop shopping unit of deployment in the .NET world. You can copy it to the server to deploy the assembly there use it for deployment.
- The runtime CLR offers wide verity of services to your running code, It first loads and verifies the assembly to make sure the IL is ok.
- Then JIT, as methods are called, the runtime arranges for them to be compiled to machine code suitable for the machine, assembly is running on, and caches this machine code to be used the next time the method is called.

Unmanaged Code:

- The code, which is developed outside .NET framework is known as unmanaged code. It do not run under the control of CLR. Unmanaged code compiles straight to machine code and directly executed by the OS.
- The generated code runs natively on the host processor and the processor directly executes the code generated by the compiler.
- It is always compiled to target a specific architecture. Unmanaged executable files are basically a binary image.
- The memory allocation, type safety, security, etc. needs to be taken care of by the programmer.

11. NAMESPACES:

- **FCL is organized into hierarchical structure** and it is **divided into Name Spaces**. This library is organized into namespaces that contain **functionally related groups of classes**.
- These **namespaces are divided among different categories**, such as data access, common types, debugging, file access, network communication, security, Windows applications, Web applications, Web services, XML data etc. The system library is the root for types in the .NET framework.
- **Namespace is the Logical group of types** or we can say **namespace is a container**. It is a way of **grouping classes and struct together** in a way **that limit their scope and avoid name conflicts** with other classes.
- Namespaces are a **mechanism for controlling the visibility of names within a program**. It allows you to create a system to **organize your code**. A good way to organize your namespaces is via a hierarchical system.

- They are intended to help facilitate the combination of program components from various sources by **minimizing the name conflicts**.
- A namespace restricts a **name's scope, making it meaningful only within the defined namespace**. Namespace provides general solution to the problem of global name collision.

.NET Framework Namespaces

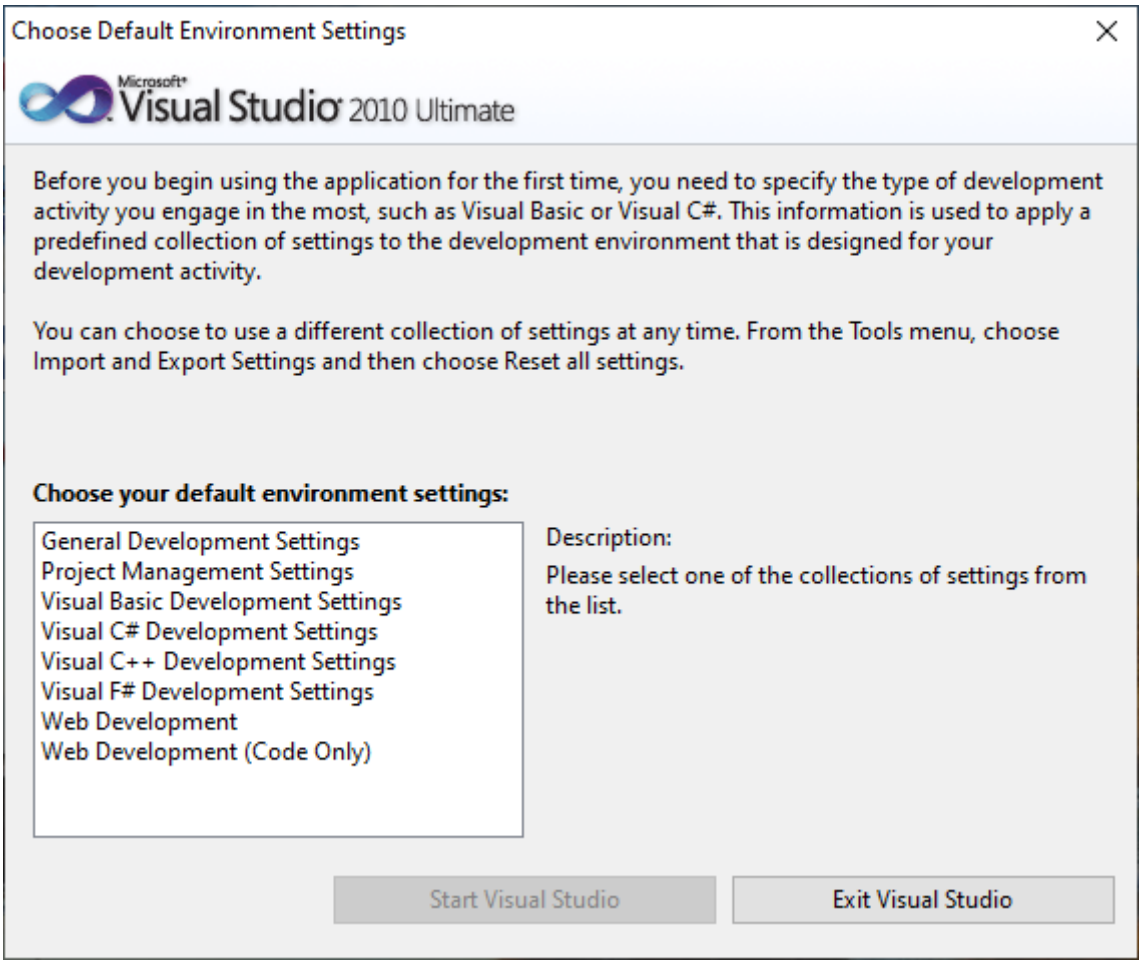
Namespace	Description
System	Contains fundamental classes and base classes that define common value and reference data types, events and event handlers, interfaces, processing exceptions, data conversion, mathematics, as well as garbage collection and application environment management.
System.IO	Provides classes that support asynchronous and synchronous reading from and writing to data streams and files. Contains classeslike FileStream, MemoryStream, Path, and Directory.
System.Collections	Contains interfaces and classes that define various collections of objects, such as lists, arrays, hash tables, stacks, queues, and dictionaries. The classes defined in this namespace.
System.Reflection	Contains classes that provide dynamic binding and a managed view of loaded types, methods, and fields. Contains classes such as Assembly, Module, and MethodInfo.
System.Security	Implements a security system, including base classes for permissions. Includes classes of the likes of SecurityManager, PermissionSet, and CodeAccess-Permission.
System.Net	Provides support for network programming. Includes, for example, the classes HttpWebRequest, IPAddress, Dns, and Connection.
System.Data	Contains classes that implement ADO.NET. Child namespaces include OLEDB, and SqlClient.

12. GETTING STARTED WITH VISUAL BASIC .NET (IDE):

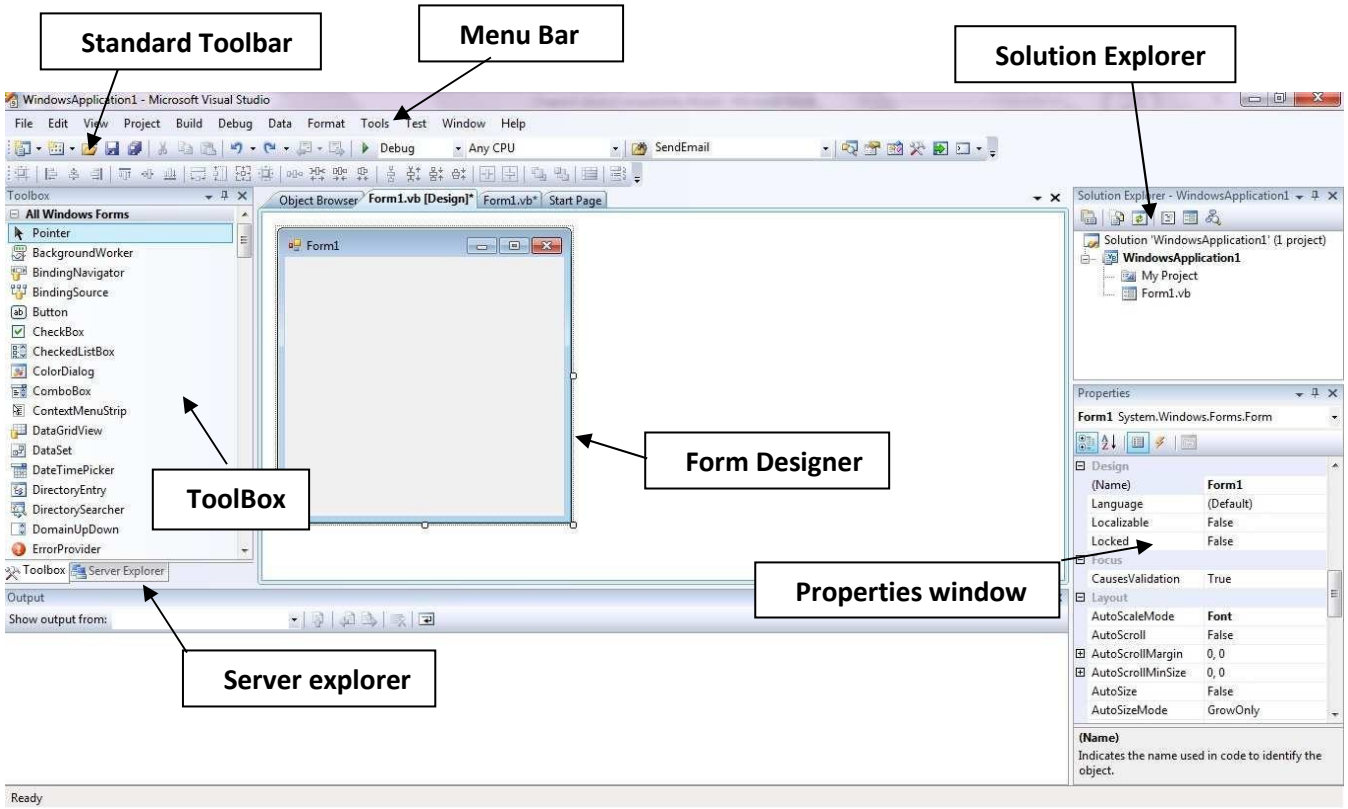
- **Integrated Development Environment (IDE)** consist of inbuilt compiler, debugger, editors, and automation tools for easy development of code. Visual Basic.net IDE can be accessed by opening a new project.
- Visual Basic.net **Integrated Development Environment (IDE)** consist of Solution Explorer, Toolbox, Form, Properties Window, and Menu Bar which is shown in Figure.
- In Visual Studio windows related to a project are combined together and placed at certain locations on the screen. This type of IDE is known as Multiple Document Interface or MDI.

Menu System / Toolbars

- Menu bar consist of the commands that are used for constructing a software code. These commands are listed as menus and submenus
- Menu bar also includes a standard Toolbar that lists the commonly used commands as buttons.

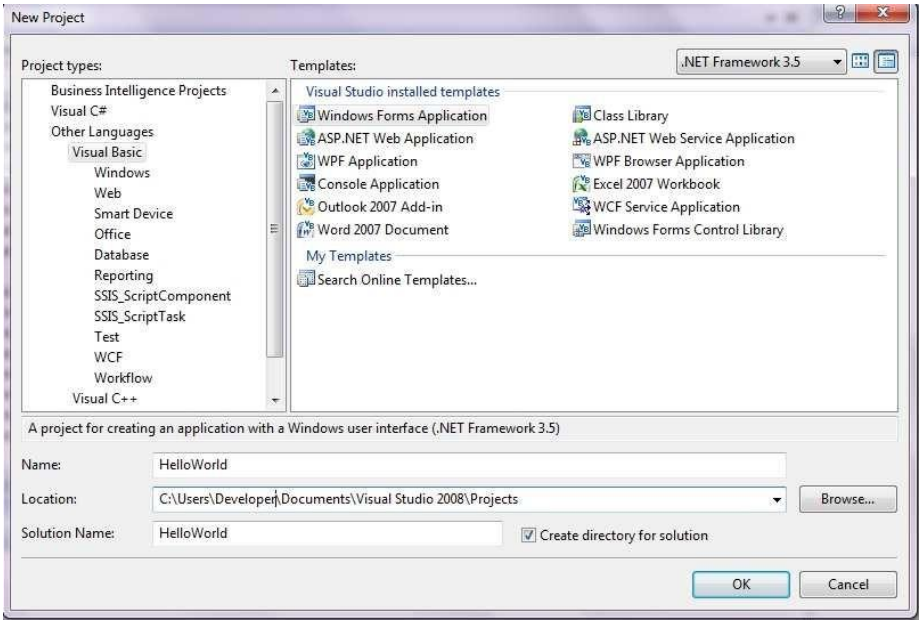


- This Toolbar can be customized, so that the buttons can be added as required.



The new Project dialog box

- Creating a new Visual Basic project is a simple task.
- You can either click **File>New>Project or File or Website command** or **the New Project link** from the Start Page. In both cases, Visual Studio shows the New Project window.
- In Visual Studio, you can choose what version of the .NET Framework your application targets. This can be useful if you plan to develop applications with a high compatibility level and without new language features.
- After that enter name for the new project and click OK to create the project.



Graphical designers

- VB.Net programmers have made **extensive use of forms** to build user interfaces. Each time you create a Windows application, Visual Studio will display a default blank form, onto which you can **drag and drop controls from the Visual Studio Tool box window**.

Code designers

- The **code editor** in Visual Studio is now **based on Windows Presentation Foundation**. This innovation introduces new features to the Visual Basic code editor.
- We can focus on the **zoom functionality in the code editor, the gradient selection, and IntelliSense.**

IntelliSense

- IntelliSense is one of the most important feature in the coding experience with Visual Studio.
- IntelliSense is represented by a **pop-up window that appears** in the code editor each time you begin **typing a keyword** or an **identifier and shows options for auto completing words.**

The Object Explorer/Browser

- The Object Browser is a special **tool window** that enables you to **browse the .NET Framework class library**.
- You can get a **hierarchical view of the Base Class Library** and of all the types defined in your

solution, including types defined in referenced external assemblies.

The Toolbox

- **Toolbox in Visual Basic.net consist of** Controls, Containers, Menu Options, Crystal Report Controls, Data Controls, Dialogs, Components, Printing controls that are used in a form to design the interfaces of an application.

The Solution Explorer

- Solution Explorer is a **special tool window** that enables **managing solutions, projects, and files in the projects or solution**.
- It provides a **complete view of what files compose your projects** and enables adding or removing files and organizing files into subfolders.

Properties Window

- The Properties window allows user to **view and modify the properties of the form and of the controls** that it contains.
- You often need to **set properties for your code**, for .NET objects you use in your code, and for your files.
- To make things easier, **Visual Studio provides a tool window named Properties window**, which is a graphical tool for setting items' properties.

The Dynamic help window

- Visual Studio ships with the **MSDN (Microsoft Developers Network) Library**, which is the place where you can **find documentation for Visual Basic version** and the .NET Framework versions.
- There are basically two ways to access the MSDN Library: offline and online.
- You can also **quickly find information** on particular objects using built-in tools, such as the Object Browser.

The Server explorer

- Server Explorer is the **server management console** for Visual Studio.
- Use this window to **open data connections and to log on to servers and explore their databases and system services**. To access Server Explorer, choose Server Explorer on the View menu.

The Command window

- The .NET Framework allows you to access tools from the command prompt, so if you wish to use these utilities from any window command window, you will need to register these paths with the operating system.

13. VISUAL BASIC LANGUAGE CONCEPTS:

13.1 Variables:

- A variable is nothing but a name given to a storage area that our programs can manipulate.
- Each variable in VB.Net has a specific type, which determines the size and layout of the variable's memory, the range of values that can be stored within that memory; and the set of

operations that can be applied to the variable.

- The basic value types provided in VB.Net can be categorized as:

Type	Example
Integral types	SByte, Byte, Short, UShort, Integer, UInteger, Long, ULong
Floating point types	Single and Double
Decimal types	Decimal
Boolean types	True or False values, as assigned
Date types	Date

- There are some rules to choose variable name by the programmer.
  - Variable name **should not be a keyword**.
  - Variable name **cannot start with a digit**.
  - Upper case and lower case are distinct. Variable **Height** is not the same as **height** or **HEIGHT**.
  - **White space** is not allowed with variable Name.
  - Variable name can be any length

Variable Declaration in VB.NET:

- The **Dim (Declare in memory)** statement is used for variable declaration and storage allocation for one or more variables.

  Syntax: **Dim variable_name As Data_type**

  Example: **Dim PI As Double**

- In above example, we declared a one variable "pi" and data type of that variable is Double.

Variable Initialization in VB.NET:

- The **Dim** statement is used for variable declaration and storage allocation for one or more variables.
- In above example, we declared a one variable "pi" and data type of that variable is Double and assign it a value3.14.

  Syntax: **Dim variable_name As Data_type = Value**

  Example: **Dim pi As Double = 3.14**

Accepting Values from User:

- The Console class in the System namespace provides a function ReadLine for accepting input from the user and store it into a variable. For example,

  Dim var as Integer

  var = Console.ReadLine()

Full Syntax for Declare Variable:

[<attributelist>] [access modifier] [[Shared] [Shadows] | [Static]][Read Only] **Dim**

[WithEvents] variablelist

- **Attribute list** is a list of attributes that apply to the variable. Optional.

- **Access modifier** defines the access levels of the variables, it has values as - Public, Protected, Friend, Protected Friend and Private. Optional.
- **Shared** declares a shared variable, which is not associated with any specific instance of a class or structure, rather available to all the instances of the class or structure. Optional.
- **Shadows** indicate that the variable re-declares and hides an identically named element, or set of overloaded elements, in a base class. Optional.
- **Static** indicates that the variable will retain its value, even when the after termination of the procedure in which it is declared. Optional.
- **ReadOnly** means the variable can be read, but not written. Optional.
- **With Events** specifies that the variable is used to respond to events raised by the instance assigned to the variable. Optional.
- **Variable list** provides the list of variables declared.
- **Bounds list** − optional. It provides list of bounds of each dimension of an array variable.
- **New** − optional. It creates a new instance of the class when the Dim statement runs.
- **datatype** − Required if Option Strict is On. It specifies the data type of the variable.
- **initializer** − Optional if New is not specified. Expression that is evaluated and assigned to the variable when it is created.

Example:

```
Dim StudentID As Integer
Dim StudentName As String
Dim Salary As Double
Dim count1, count2 As Integer
Dim status As Boolean
Dim exitButton As New System.Windows.Forms.Button
Dim lastTime, nextTime As Date
```

13.2 Constant:

- The **constants refer to fixed values** that the program may **not alter during its execution**. These fixed values are **also called literals**.
- **Constants can be of any of the basic data types** like an integer constant, a floating constant, a character constant, or a string literal.
- In VB.Net, constants are **declared** using the **Const statement**. The Const statement is used at module, class, structure, procedure, or block level for use in place of literal values.

  Syntax: Const Dim variable_name As Data_type

  Example: Const Dim pi As Double = 3.1415

Sr. No.	Constant & Description
1	**VbCrLf**

	Carriage return / linefeed character combination.
2	**VbCr** Carriage return character.
3	**VbLf** Linefeed character.
4	**VbNewLine** Newline character.
5	**VbNullChar** Null character.
6	**VbNullString** Not the same as a zero-length string (""); used for calling external procedures.
7	**VbObjectError** Error number. User-defined error numbers should be greater than this value. For example: Err.Raise(Number) = vbObjectError + 1000
8	**VbTab** Tab character.
9	**VbBack** Backspace character.

13.3 Data Types:

- Data types refer to an **extensive system** used for **declaring variables or functions** of different types. The type of a variable determines **how much space it occupies in storage and how the bit pattern stored** is interpreted.

Data Type	Storage Allocation	Value Range
Boolean	Depends on Implementing Platform	True or False (1 or 0)
Byte	1 byte	0 through 255 (unsigned)
Char	2 bytes	0 through 65535 (unsigned)
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point, 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal

Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for **negative values** 4.94065645841246544E-324 through 1.79769313486231570E+308, for **positive values**
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)
Object	4 bytes on 32-bitplatform 8 bytes on 64-bitplatform	Any type can be stored in a variable of type Object
SByte	1 byte	-128 through 127 (signed)
Short	2 bytes	-32,768 through 32,767 (signed)
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
String	Depends on implementing Platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned)
User-Defined	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	2 bytes	0 through 65,535 (unsigned)

13.4 Operators:

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations.

Arithmetic Operators:

Following table shows all the arithmetic operators supported by VB.Net. Assume variable A holds 2 and variable B holds 7, then:

Operator	Description	Example
^	Raises one operand to the power of another.	B^A will give 49
+	Adds two operands.	A + B will give 9
-	Subtracts second operand from the first.	A - B will give -5
*****	Multiplies both operands.	A * B will give 14
/	Divides one operand by another and returns a floating point result.	B / A will give 3.5

****	Divides one operand by another and returns an integer result.	B \ A will give 3
MOD	Modulus Operator and remainder of after an integer division.	B MOD A will give 1

Comparison Operator:

Following table shows all the comparison operators supported by VB.Net. Assume variable A holds 10 and variable B holds 20, then:

Operator	Description	Example
==	Checks if the values of two operands are equal or not; if yes, then condition becomes true.	(A == B) is not true.
<>	Checks if the values of two operands are equal or not; if values are not equal, then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand; if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand; if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then condition becomes true.	(A <= B) is true.

Assignment Operator:

- There are following assignment operators supported by VB.Net:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assigns the result to left operand.	C += A is equivalent to C (C = C + A)
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assigns the result to left operand.	C -= A is equivalent to C (C = C – A)
***=**	Multiply AND assignment operator, It multiplies right operand with the left operand and assigns the result to left operand.	C *= A is equivalent to C (C = C * A)
/=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (floating point division).	C /= A is equivalent to C (C= C / A)
\=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (Integer division).	C \= A is equivalent to C = C \A
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	C^=A is equivalent to C = C ^ A

Bit shift / Bitwise Operator:

- The bit shift operators perform the shift operations on binary values.
- Bitwise operators work on bits and perform bit-by-bit operations. The truth tables for &, |, and ^ are as follows:

A	B	A & B	A | B	A ^ B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Assume if A= 60 and B=13, now in binary format they will be as follows:

A = 0011 1100		A = 0011 1100	A = 0011 1100
B = 0000 1101	A = 0011 1100	B = 0000 1101	B = 0000 1101
---	---	---	---
A&B = 0000 1100	**~A = 1100 0011**	**A|B = 0011 1101**	**A^B = 0011 0001**

Operator	Description	Example
And	Bitwise AND Operator copies a bit to the result if it exists in both operands.	(A AND B) will give 12, which is 0000 1100
Or	Binary OR Operator copies a bit if it exists in either operand.	(A Or B) will give 61, which is 0011 1101
Xor	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A Xor B) will give 49, which is 0011 0001
Not	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(Not A) will give -61, which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240, which is 1111 0000
>>	Binary Right Shift Operator. The Right operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15, which is 0000 1111

Logical Operators:

- Following table shows all the logical operators supported by VB.Net. Assume variable A holds Boolean value True and variable B holds Boolean value False, then:

Operator	Description	Example
And	It is the logical as well as bitwise AND operator. If both the operands are true, then condition becomes true.	(A And B) is False.
Or	It is the logical as well as bitwise OR operator. If any of the two operands is true, then condition becomes true.	(A Or B) is True.
Not	It is the logical as well as bitwise NOT operator. Use to reverses the logical state of its operand.	Not(A And B) is True.
Xor	It is the logical as well as bitwise Logical Exclusive OR operator. It returns True if both expressions are True or both expressions are False; otherwise, it returns False.	A Xor B is True.
AndAlso	It is the logical AND operator. It works only on Boolean data.	(A AndAlso B) is False.
OrElse	It is the logical OR operator. It works only on Boolean data.	(A OrElse B) is True.
IsFalse	It determines whether an expression is False.	
IsTrue	It determines whether an expression is True.	

- AndAlso Operator: If both expressions evaluate to True, result is True. The following table illustrates how result is determined.

If `expression1` is	And `expression2` is	The value of `result` is
`True`	`True`	`True`
`True`	`False`	`False`
`False`	(not evaluated)	`False`

- OrElse Operator: If either or both expressions evaluate to True, result is True. The following table illustrates how result is determined.

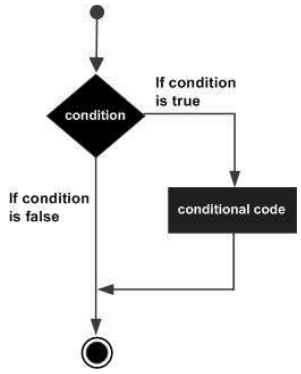
If `expression1` is	And `expression2` is	The value of `result` is
`True`	(not evaluated)	`True`
`False`	`True`	`True`
`False`	`False`	`False`

13.5 Conditional Structure / Decision Making:

- The conditional structure helps to conditionally execute a group of statements or statement, depending on the value of an expression.
- VB.Net provides the following types of decision making statements which given below:
  - If...Then Statement
  - If...Then...Else Statement
  - Nested If Statements
  - Select Case Statement

If...Then Statement:

- It is the simplest form of control statement.
- It frequently used in decision making and changing the control flow of the program execution.
- The Flow chart for If… Then statement is given below:



Syntax:

```
If condition Then
     [Statement(s)]
End If
```

Example

```
Module Module1
Sub Main()
Dim a As Integer = 0
If (a < 1) Then
     Console.WriteLine("a is less than 1")
End If
Console.WriteLine("value of a is : {0}", a)
Console.ReadLine()
End Sub
End Module
```

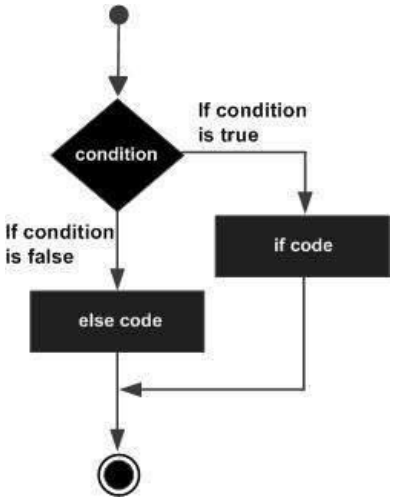
```
file:///C:/Users/Chauhan/AppData/Local/Temporary Projects/UNIT 1 1 IF STATEMENT/bin/Debug/UNIT 1 1 IF STATEMENT.EXE
a is less than 1
value of a is : 0
```

- If the condition **(a < 1) evaluates to true**, then the block of **code inside the If statement** will be executed.
- If condition **(a < 1) evaluates to false**, then the code after the **End If** of the If statement will be

executed.

If...Then...Else Statement:

- It conditionally executes a group of statements, depending on the value of an expression.
- If the Boolean expression evaluates to true, then the block of **If** part will be executed, otherwise the block of **Else** part will be executed.
- The Flow chart for If… Then…Else statement is given below:



Syntax:

```
If(boolean_expression or condition) Then
     'statement(s) will execute if the Boolean expression is
     true
Else
     'statement(s) will execute if the Boolean expression is
     false
End If
```

Example:

```
Module Module1
Sub Main()
Dim a As Integer = 0
'check the boolean condition using if statement
If (a < 1) Then
Console.WriteLine("a is less than 1")
Else
Console.WriteLine("a is not less than 1")
End If
Console.WriteLine("value of a is :"& a)
Console.ReadLine()
End Sub
End Module
```

```
file:///C:/Users/Chau
a is less than 1
value of a is :0
```

If...Else If...Else Statement

- The if statement can be followed by an optional Else if...Else statement, which is used to test various conditions using single If...Else If statement.
- It useful when you want to check more than one condition in a program. The syntax for this statement is:

Syntax:

```
If(boolean_expression 1 or condition 1)Then
   ' Executes when the boolean expression 1 is true
ElseIf(boolean_expression 2 or condition2)Then
   ' Executes when the boolean expression 2 is true
ElseIf(boolean_expression 3 or condition3)Then
   ' Executes when the boolean expression 3 is true
Else
   ' executes when the none of the above condition is true
EndIf
```

Example:

```
Module Module1
Sub Main()
Dim a As Integer = 10
If (a = 1) Then
     Console.WriteLine("Value of a is 1")
ElseIf (a = 2) Then
     Console.WriteLine("Value of a is 2")
ElseIf (a = 3) Then
     Console.WriteLine("Value of a is 3")
Else
     Console.WriteLine("No values is matching")
EndIf
Console.WriteLine("Exact value of a is: {0}", a)
Console.ReadLine()
End Sub
End Module
```

```
file:///C:/Users/Chauhan/AppData/Local/Temporary Projects/UNIT 1 3 IF ELSEIF ELSE STATEMENT/bin/Debug/UNIT 1 3 IF ELSEIF ELSE STATEMENT.EXE
No values is matching
Exact value of a is: 10
```

Nested If Statements

- By using If-Then-Else statements, you can use one If or ElseIf statement inside another If ElseIf statement(s).

Syntax:

```
If(boolean_expression 1)Then
     'Executes when the boolean expression 1 is true
     If(boolean_expression 2)Then
          'Executes when the boolean expression 2 is true
     End If
End If
```

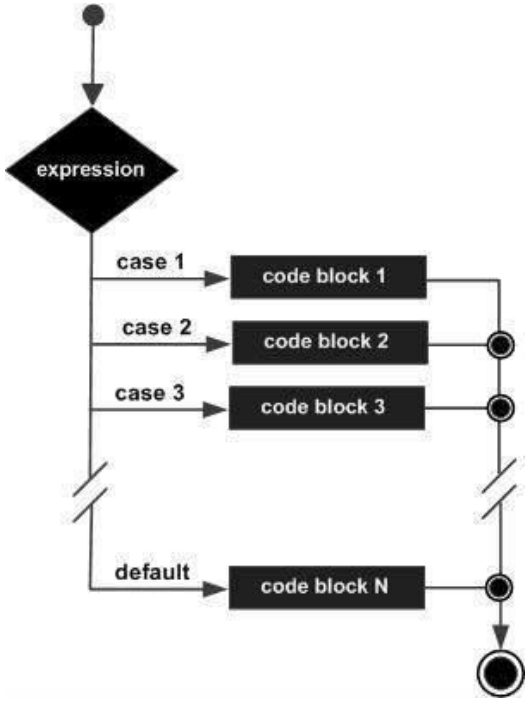
Example:

```
Module Module1
      Sub Main()
      Dim a As Integer
      Dim b As Integer

      Console.writeline(“enter a”)
      A = console.readline()

      If (a = 1) Then
            If (b = 2) Then
                  Console.WriteLine("Value of a is 1 and b is 2")
            Else
                  Console.WriteLine("Value of a is 1 and b is 2")
            End If
      End If
      Console.WriteLine("Exact value of a is : {0}", a)
      Console.WriteLine("Exact value of b is : {0}", b)
      Console.ReadLine()
      End Sub
End Module
```

```
file:///C:/Users/Chauhan/AppData/Local/Temporary Projects/UNIT 1 4 NESTED IF STATEMENT/bin/Debug/UNIT 1 4 NESTED IF STATEMENT.EXE
Value of a is 1 and b is 2
Exact value of a is : 1
Exact value of b is : 2
```

Select Case Statement:

- In **Select Case statement,** it allows a variable to be tested for equality against a given list of values.
- Each value is called a **case**, and the variable being switched on is checked for each select case.



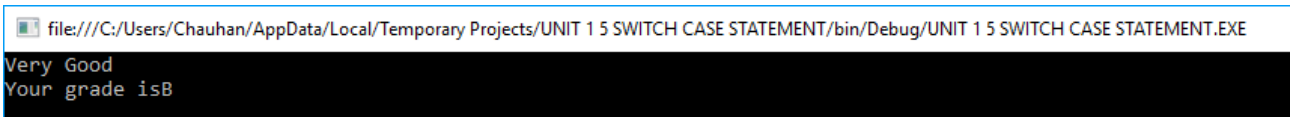
Syntax:

```
Select Case[expression]
```

```
        Case expression_list
            [statements]
        Case Else
            [else statements]
    End Select
```

Example:

```
Module Module1
    Sub Main()
    Dim grade As Char
    grade = "B"
    Select Case grade
        Case "A"
            Console.WriteLine("Excellent!")
        Case "B", "C"
            Console.WriteLine("Very Good")
        Case "D"
            Console.WriteLine("Good")
        Case "F"
            Console.WriteLine("Better try again")
        Case Else
            Console.WriteLine("Invalid grade")
    End Select
    Console.WriteLine("Your grade is{0}", grade)
    Console.ReadLine()
    End Sub
End Module
```



13.6 Loops:

- There may be a situation when you need to execute a block of code several number of times.
- A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages:

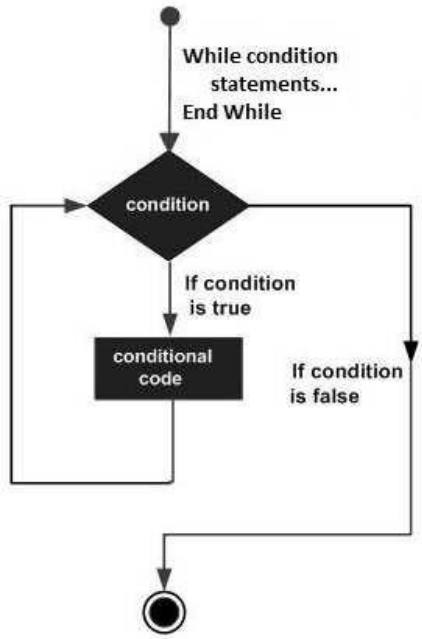
While…End While Loop:

- It executes a series of statements as long as a given condition is true.
- Key point of the While loop is that the loop might not ever run. When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.

Syntax:

```
While condition
    [statements]
    [Continue While]
    [statements]
    [Exit While]
    [statements]
End While
```

- Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is logical true. The loop iterates while the condition is true.
- When the condition becomes false, program control passes to the line immediately following the loop.



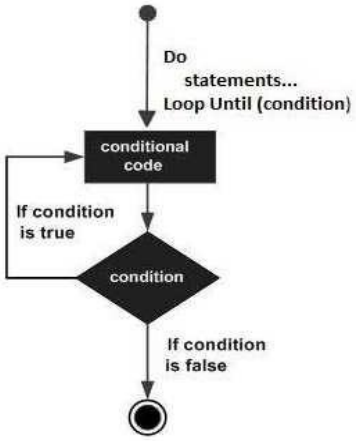
Example:

```
Module Module1
    Sub Main()
        Dim a As Integer = 10
        'while loop execution'
        While a < 20
        Console.WriteLine("value of a: {0}", a)
                    a = a + 1
        End While
        Console.ReadLine()
    End Sub
End Module
```

```
file:///C:/Users/Chauhan/AppData/Local/Temporary Projects/UNIT 1 6 WHILE LOOP/bin/Debug/UNIT 1 6 WHILE LOOP.EXE
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Do Loop:

- It repeats the enclosed block of statements while a Boolean condition is true or until the condition becomes true.
- It could be terminated at any time with the Exit Do statement



Syntax:

```
Do { While | Until } condition
[statements]
[continue Do ]
[statements]
[Exit Do]
[statements]
Loop
                    Or
Do
[statements]
[Continue Do]
[statements]
[Exit Do]
[statements]
Loop {While | Until} Condition
```

Example:

```
Module Module1
    Sub Main()
        Dim iNum As Integer = 3
        Dim iCounter As Integer = 0
        Do Until iNum = 6
            If iNum <= 0 Then
                Exit Do
            End If
            iNum -= 1
            iCounter += 1
            Console.WriteLine(iNum)
        Loop
        Console.WriteLine("The loop run "& iCounter &"times.")
        Console.ReadLine()
    End Sub
End Module
```

```
2
1
0
The loop run 3times.
```

```
Module Module1
    Sub Main()
        'local variable definition
        Dim a As Integer = 10

        Do
            Console.WriteLine ("value of a: {0}", a)
            a = a + 1
        Loop While (a < 20)
        Console.ReadLine ()
    End Sub
End Module
```

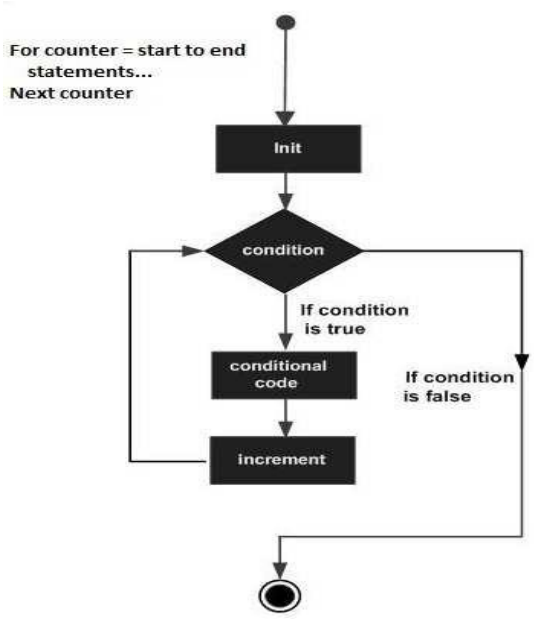
```
file:///C:/Users/Chauhan/AppData/Local/Temporary Projects/UNIT 1 7 DO WHILE LOOP/bin/Debug/UNIT 1 7 DO WHILE LOOP.EXE
value of a:10
value of a:11
value of a:12
value of a:13
value of a:14
value of a:15
value of a:16
value of a:17
value of a:18
value of a:19
```

For…Next Loop:

It repeats a group of statements a specified number of times and a loop index counts the number of loop iterations as the loop executes.

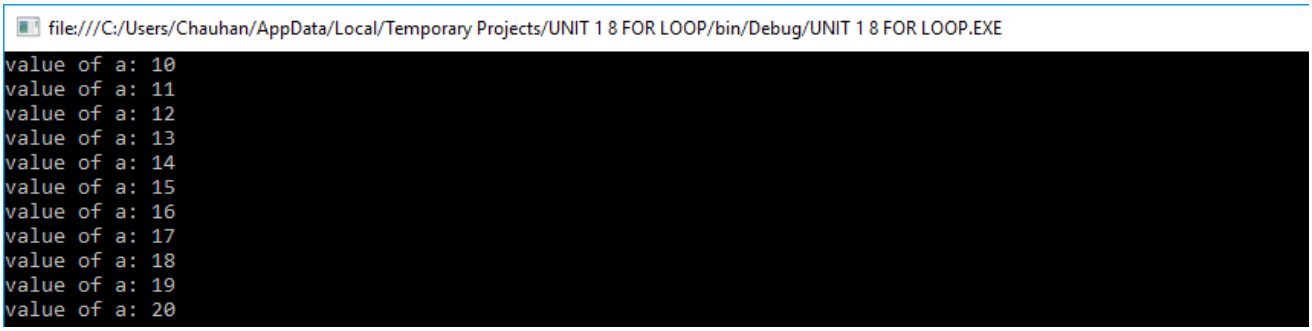
Syntax:

```
For counter [As datatype] = start to end [Step step]
[statements]
[Continue For]
[statements]
[Exit For] [statements]
Next [counter]
```



Example:

```
Module Module1
    Sub Main()
        Dim a As Integer
        For a = 10 To 20
            Console.WriteLine ("value of a: {0}", a)
        Next
        Console.ReadLine ()
    End Sub
End Module
```



For Each… Next Loop:

- It repeats a group of statements for each element in a collection. This loop is used for accessing and manipulating all elements in an array or a VB.Netcollection.

Syntax:

```
For Each element [As datatype] In group
    [statements]
    [Continue For]
    [statements]
    [Exit For]
    [statements]
Next [element]
```

Example:

```
Module Module1
    Sub Main()
        Dim anArray() As Integer = {1, 3, 5, 7, 9}

        Dim arrayItem As Integer

        For Each arrayItem In anArray
            Console.WriteLine (arrayItem)
        Next
        Console.ReadLine ()
    End Sub
End Module
```

With…End With Statement:

- It is not exactly a looping construct. It executes a series of statements that repeatedly refers to a single object or structure.

Syntax:

```
With object
      [statements]
End With
```

Example:

```
Module Module1
      Public Class Book
            Public Name As String
            Public author As String
            Public Subject As String
      End Class
      Sub Main()
            Dim obj As New Book
            With obj
                        .Name = ".Net Programming"
                        .author = "ABCD-XYZ"
                        .Subject = "VB.NET"
            End With
            With obj
            Console.WriteLine(.Name)
            Console.WriteLine(.author)
            Console.WriteLine(.Subject)
            End With
            Console.ReadKey()
      End Sub
End Module
```

```
.Net Programming
ABCD-XYZ
VB.NET
```

Nested…Loop:

Syntax: (For Loop)

```
For counter [As datatype1] = start1 To end1 [Step step1]
      For counter [As datatype2] = start2 To end2 [Step step2]
            ……
      Next [counter2]
Next [counter1]
```

Syntax: (While Loop)

```
While conditional1
      While condition2
            ……
      End While
End While
```

Syntax: (Do While Loop)

```
Do{While |Until} condition1
      Do {While | Until} condition2
            ……
      Loop
```

```
Loop
```

13.7 Array:

- An array stores a fixed-size sequential collection of elements of the same type.
- All arrays consist of contiguous memory locations.
- The lowest address corresponds to the first element and the highest address to the last element.
- In VB.NET array are actually objects and not just region of contiguous memory as in C and C++.

First Element						Last Element
Number[1]	Number[2]	Number[3]	Number[4]	Number[5]	Number[6]	.....

Creating Arrays in VB.NET:

To declare an array in VB.Net, you use the Dim statement.

Example:

```
Dim Data(30) As Integer
'an array of 31 elements


Dim strData(20) As String
'an array of 21 strings

Dim twoDarray(10, 20) As Integer
'a two dimensional array of integers

Dim ranges(10, 100)
'a two dimensional array
```

You can also initialize the array elements while declaring the array.

Example:

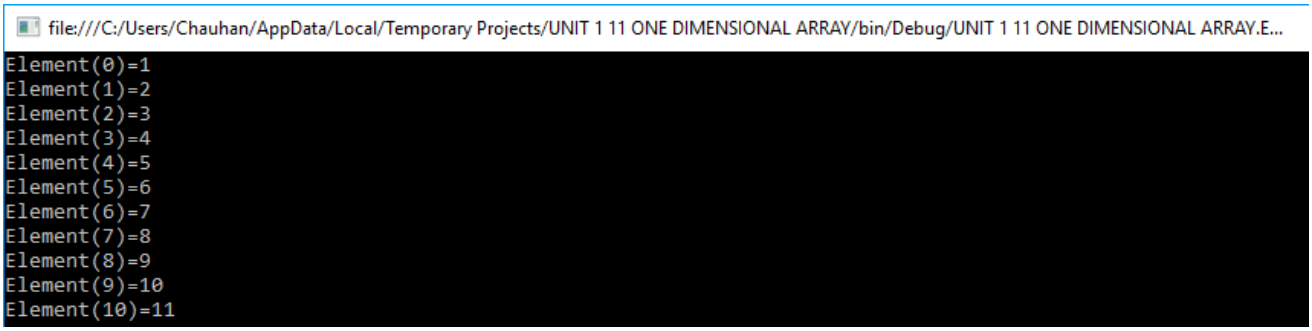
```
Dim intData() As Integer = {12, 16, 20, 24, 28, 32}
Dim branches() As String = {"CE","IT","EC","MECH","CIVIL"}
```

The elements in an array can be stored and accessed by using the index of the array. The following program demonstrates this:

Example:

```
Module Module1
      Sub Main()
            Dim n(10) As Integer
            Dim i, j As Integer
            For i = 0 To 10
                  n(i) = i + 1
            Next i
            For j = 0 To 10
```

```
                Console.WriteLine("Element({0})={1}", j, n(j))
            Next j
            Console.ReadKey()
        End Sub
    End Module
```



Multidimensional Array:

- VB.Net allows multidimensional arrays. Multidimensional arrays are also called rectangular arrays.
- You can declare a 2-dimensional array of strings as:

  Dim twodisplay(10,20) As String
- or, a 3-dimensional array of Integer variables:

  Dim three dispalay(10,20,30) As Integer

Example:

```
Module Module1
Sub Main()
'an array with 5 rows and 2 columns
Dim a( , ) As Integer = {{0,0}, {1,2}, {2,4}, {3,6}, {4,8}}
Dim i, j As Integer

For i = 0 To 4
    For j = 0 To 1
        Console.WriteLine("a[{0},{1}] = {2}",i,j,a(i,j))
    Next j
Next i

Console.ReadKey()
End Sub
End Module
```

Dynamic Array:

- Dynamic arrays are arrays that can be dimensioned and re-dimensioned as par the need of the program. You can declare a dynamic array using the ReDim statement.

Syntax:

Dim Array_Name(Size) As Data Type

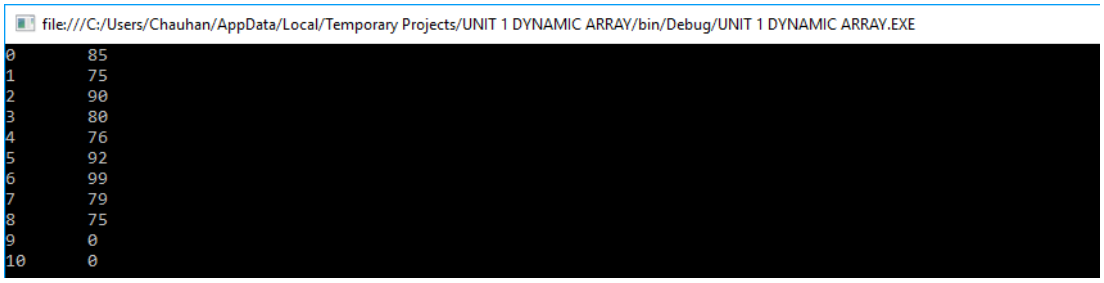
ReDim [Preserve] array_name (subscripts)

Where,

- The **Preserve** keyword helps to preserve the data in an existing array, when you resize it.
- **Array_name** is the name of the array.
- **Subscripts** specifies the new dimension.

Example:

```
Module Module1
    Sub Main()
        Dim marks() As Integer
        Re Dim marks(2)
        marks(0) = 85
        marks(1) = 75
        marks(2) = 90
        ReDim Preserve marks(10)
        marks(3) = 80
        marks(4) = 76
        marks(5) = 92
        marks(6) = 99
        marks(7) = 79
        marks(8) = 75
        For i = 0 To 10
        Console.WriteLine(i & vbTab & marks(i))
        Next i
        Console.ReadKey()
    End Sub
End Module
```



14. GTU QUESTIONS:

Sr. No	Questions	Marks
	18/02/2021	
1	List out the components of .Net Framework	2
2	Give the list of data type in vb.net and explain date data type in brief.	2
3	Explain constant variable declaration in VB.net in brief.	2
4	Explain For…Next Loop.	2
5	Explain Dynamic Array declaration in Vb.Net.	2
6	Explain Common Language Runtime in detail.	3
7	List out the various operators. Explain any two in detail.	3
8	Explain Do ….While loop with example.	4
9	Explain Select case with example.	4
10	What is IDE? Explain server explorer and toolbox in brief.	4
11	Explain multidimensional Array with example.	4
	18/02/20	
1	Give Full form of following. (i) FCL (ii) JIT (iii) MSIL (iv) CTS.	2
2	What is ReDim keyword?	2
3	List out the features of .Net.	2
4	Explain JIT in brief.	2
5	Describe Managed code and Unmanaged code.	3
6	Briefly explain (i) select… case (ii) while…. loop.	3
7	Draw architecture of .Net framework. Explain CLR in detail.	4
8	Explain array in .Net.	4
9	Explain CLR.	4
10	Explain for….each statements with example.	4
	29/10/2020	
1	Give Full form of following. (i) FCL (ii) JIT (iii) MSIL (iv) CTS.	2
2	Define While...End While loop.	2
3	Describe Common language runtime (CLR).	3
4	Describe Managed code and Unmanaged code.	3
5	Define Array and Explain multidimensional array.	4
6	Write a program to show Relational Operators.	3
	16/11/2019	
1	Draw .NET Framework Architecture.	2
2	Give full form for CTS, JIT, MSIL, FCL	2
3	Explain common language runtime (CLR) environment.	4
4	Explain For…each loop with example.	2
5	Give the list of data type in VB and explain Numeric data type in brief	2
6	Explain Variable Declaration in VB.Net in brief	2
7	Define (i) Properties (ii) Methods	2
8	What is the type and function of these two operators: / , \	2
9	Briefly explain: (i) if.. then.. else (ii) while…. loop.	3
10	Describe various parts of VB.Net IDE.	4
11	List out the various operators. Explain arithmetic and logical operator in detail.	4
12	Explain dynamic array with appropriate example.	3
	17/05/2019	

1	What is the difference between VB and VB.Net?	2
2	What is namespace?	2
3	What is JIT?	2
4	Define Managed code and Unmanaged code.	2
5	Draw architecture of .Net framework. Explain CLR in detail	4
6	What is ReDim keyword?	2
7	What is Array? Explain jagged array with example.	3
8	Explain For….Next and For Each….Next statements.	3
9	List different types of operators. Explain logical operators.	3
10	Explain Select…Case statements with example.	4
11	Explain .Net framework IDE in brief.	7
28/11/2018		
1	What is Namespace in .Net?	2
2	Explain DOT NET Framework	4
3	Explain CLR and CTS	4
4	Explain FCL and BCL.	3
5	Explain Property Window.	2
6	Explain Solution Explorer.	2
7	Explain array in .Net	4
8	List out data type in .Net and explain Date Type.	4
9	Explain For…Each loop.	3
04/05/2018		
1	Explain .NET Framework Architecture with diagram.	2
2	Explain Namespace in brief	2
3	Explain the following components (i) .NET Framework Class library	2
4	List types of operator?	2
5	Explain logical operators with example.	3
6	What is Array List its types and explain any one in detail	3
7	Describe various parts of VB.Net IDE.	4
8	Explain while loop with example.	3
9	Explain menus in .NET.	3
10	Explain Data types.	2
03/05/2017		
1	List the data types available.	2
2	Write the syntax to declare an array of 10 integers. Assume the name of the array is ‘marks’	2
3	Differentiate these two logical operators with proper example: ‘And’, ‘AndAlso’.	3
4	What is the type and function of these two operators: / , \	3
5	Explain all the variations of Do .. Loop. Explain all loops with the same program to add numbers between 1 and 10.	7
24/11/2016		
1	Draw .net Framework Architecture	2
2	Explain namespace in brief	2
3	List out the component of .net framework and Explain CLR in detail.	3
4	Explain .net IDE.	4
5	Explain do…while …loop.	2

6	List out the types of operator?	2
7	Explain I) variable II) Constant in brief.	2
8	List out types of array?? Explain Dynamic array with example.	4
20/05/2016		
1	Explain common language runtime (CLR) environment.	4
2	Describe various parts of VB.Net IDE.	4
3	List various data types available. Explain any two with example.	3
4	List various loops available and explain any one with example.	3
5	Explain dynamic array with appropriate example.	3
7/12/2015		
1	Draw .net Framework Architecture	2
2	Explain (i) Variables (ii) Constants. In brief	2
3	Explain namespace in brief.	2
4	Explain CLR and CLS in detail.	3
5	Explain the following components (i) .NET Framework Class library (ii) Data types.	4
6	Explain For…Next loop with example.	2
7	Explain .NET Framework IDE(Integrated Development Enviornment) in brief	3
8	List the various types of operators .Explain assignment operators with example.	3
9	What is Array. List it's types and explain any one in detail.	3
10	Explain menus in .NET.	4
11	Explain (i)do.. Loop (ii)while….End While Loop	4
12	Explain select…case statements with example.	4
12/05/2015		
1	Write a short note on Namespace.	2
2	List and explain the features of VB.Net.	3
3	What is managed code and unmanaged code?	3
4	Explain common type specification in brief	3
5	Draw .Net framework and list its components and Explain CLR and CLS	4
6	Explain .Net class library in detail.	3
7	Explain variable and constants.	2
8	Explain operator precedence.	2
9	List the type of Arrays and Explain Multidimensional Array in brief	3
10	List the control structure of VB.Net and Explain Do…..While loop	3
11	List the type of operators and Explain logical operators in brief	4
04/12/2014		
1	Explain any 7 of the following terms. i)FCL ii) CLS iii)Reference Types iv)Value types v)MSIL vi)Assembly vii) JIT vii)IDE ix)Solution Explorer	7
2	List and explain any two control structures in VB.NET with example.	7