AISHWARYA COLLEGE OF EDUCATION

DEPARMENT OF COMPUTER SCIENCE

VISUAL PROGRAMMING

Unit : 1 & 2

CLASS : BCA2

Prepared By - Joseph Dickenson

UNIT-1

Introduction to .Net Framework

What is .NET Framework?

.NET is a framework is not a language. It is a framework (collection of software) to develop various types of software applications. It was designed and developed by Microsoft and the first beta version got released in 2000. It provides a controlled programming environment where software is often developed, installed and executed on Windows-based operating systems. It contains many class libraries known collectively as Framework Class Library (FCL). .NET is a framework that provides various services like memory management, networking, security and type safety. Also, it supports more than 60 programming languages such as C#, F#, VB.NET, J#, VC++, JScript.NET, APL, COBOL, Perl, Oberon, ML, Pascal, Eiffel, Smalltalk, Python, Cobra, Ada, etc.

Features of .NET Framework

1. **Less Coding and Increased Reuse of Code**

   .Net framework supports Object-oriented programming (OOP) languages, which eliminates unnecessary codes and involves less coding. .NET consists of reusable code and many reusable components. This translates into less time and consequently less cost to develop applications.

2. **Language-independent**

   .NET framework allows to develop Desktop, Web, Mobile and Console based applications. Also, it's promoted as a language-independent framework, which means that development can happen in several compliant languages that include C#, VB.NET, etc.

3. **Used for Service-Oriented Architecture**

   .NET is often used for Web Services, WCF and Web API creation; which helps create REST API and SOAP based services that can communicate and transmit data utilizing standard Internet protocols.

4. **Easy Versioning & Deployment Process**

   With features like no-impact applications, private components, controlled code sharing, side-by-side versioning and partially trusted code, .NET framework makes deployment easier post development. The code execution environment supports safe code execution that minimizes software deployment and versioning conflicts. Moreover, it eliminates the performance problems of scripted or interpreted environments as well.

5. Security

.NET offers enhanced application security as web applications developed using ASP.NET have Windows confirmation and configuration. Managed code and CLR offer safeguard features like 'role-based security' and 'code access security'.

6. Multi-tiered Software Architecture

.NET makes use of multi-tiered software architecture. It is known as multi-tiered because it physically separates functions for presentation, business logic and data management. It helps developers to build flexible applications. Moreover, developers also can add or edit a layer without being required to transform on the whole app.

7. Feature-rich

There are a variety of features which will be explored by the developers to make powerful apps. But then .NET Framework has been found to be a great platform for re-designing current applications in order to make it line up with the growing needs of an organization as well. For this, let us consider the case of its 'rich toolbox' and the 'Designer' in the visual studio. They let you access features such as automatic deployment, WYSIWYG editing, and drag-and-drop controls, etc. Any third-party control integration is also easily achievable with .Net Framework.

8. Caching & Exception Handling

The caching & exception handling features in .Net system, make your code more robust and easier-to-use.

9. Types of applications that can be created with .NET

A. Desktop Applications:
- Windows GUI application or Windows Forms (or WinForms) applications
- Windows Presentation Foundation (WPF) applications
- Console based application
- Windows Service applications
- Product/Inventory applications
- Warehousing applications using handheld devices
- Web based applications:
- Websites
- Web Applications – ASP.NET, MVC

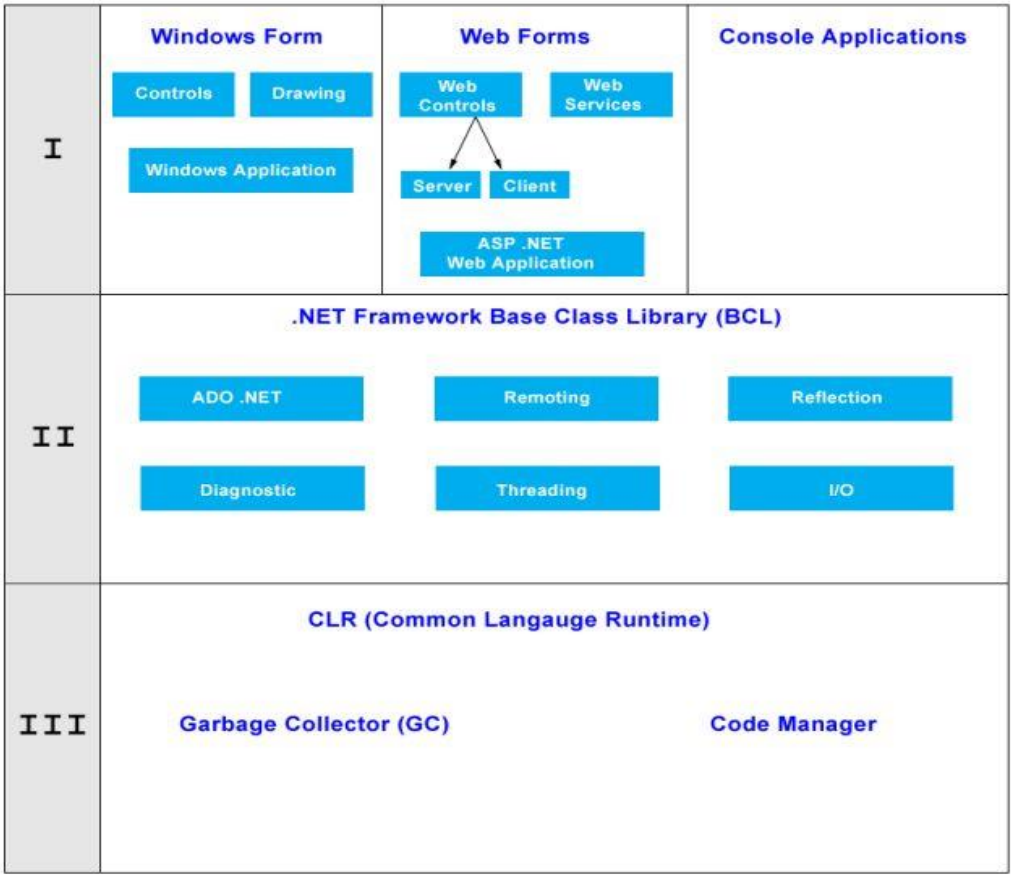
B. Mobile Application
- Operating system independent Mobile app creation using Xamarin

C. Service-oriented Applications:
- SOAP based services creation
- REST API creation

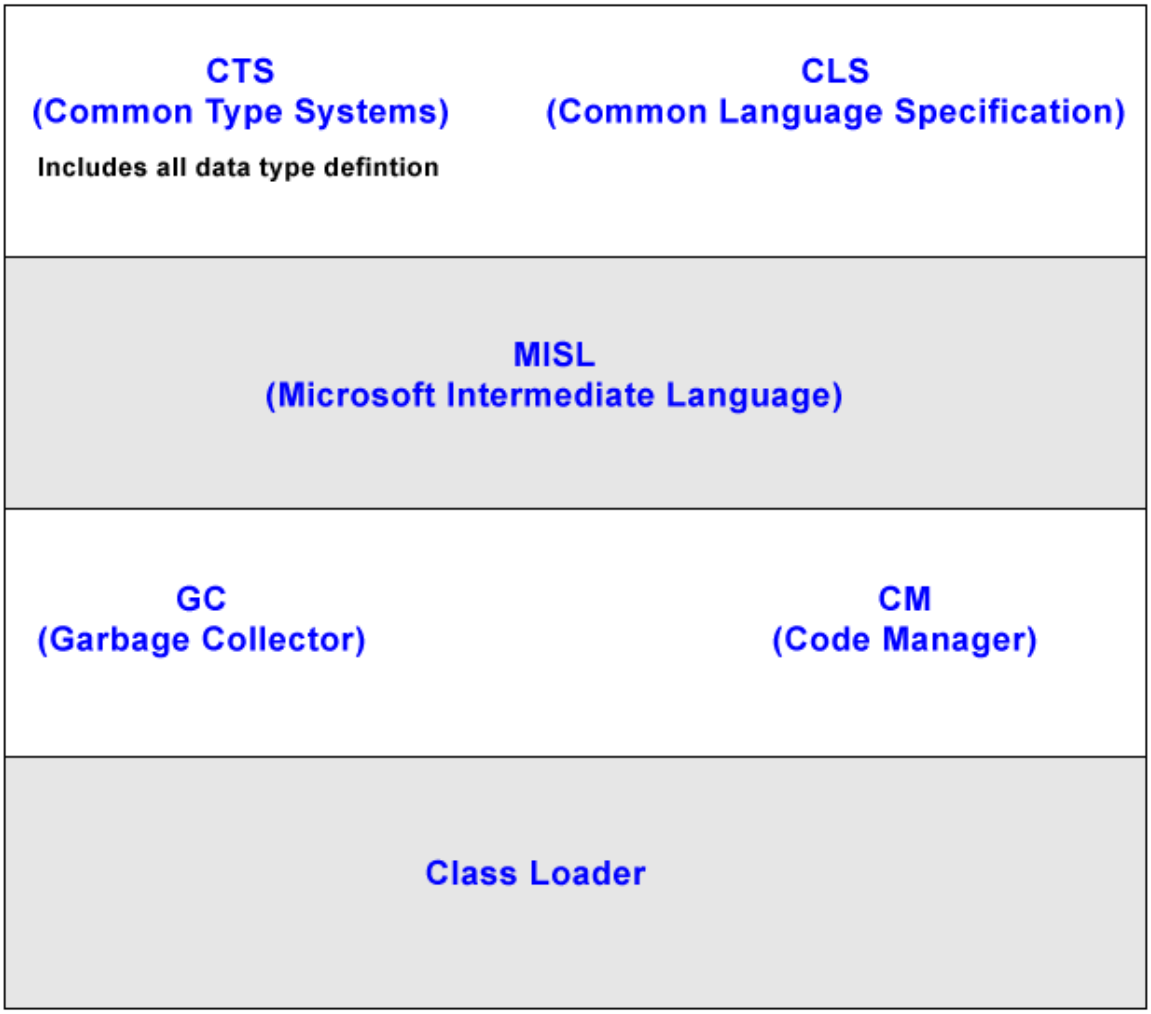
.Net Framework Architecture

.Net Framework Architecture is the programming model for the . **NET** platform. The . **NET Framework** provides a managed execution environment, simplified development and deployment and integration with a wide variety of programming languages.

Architecture of .NET Framework



Architecture of CLR

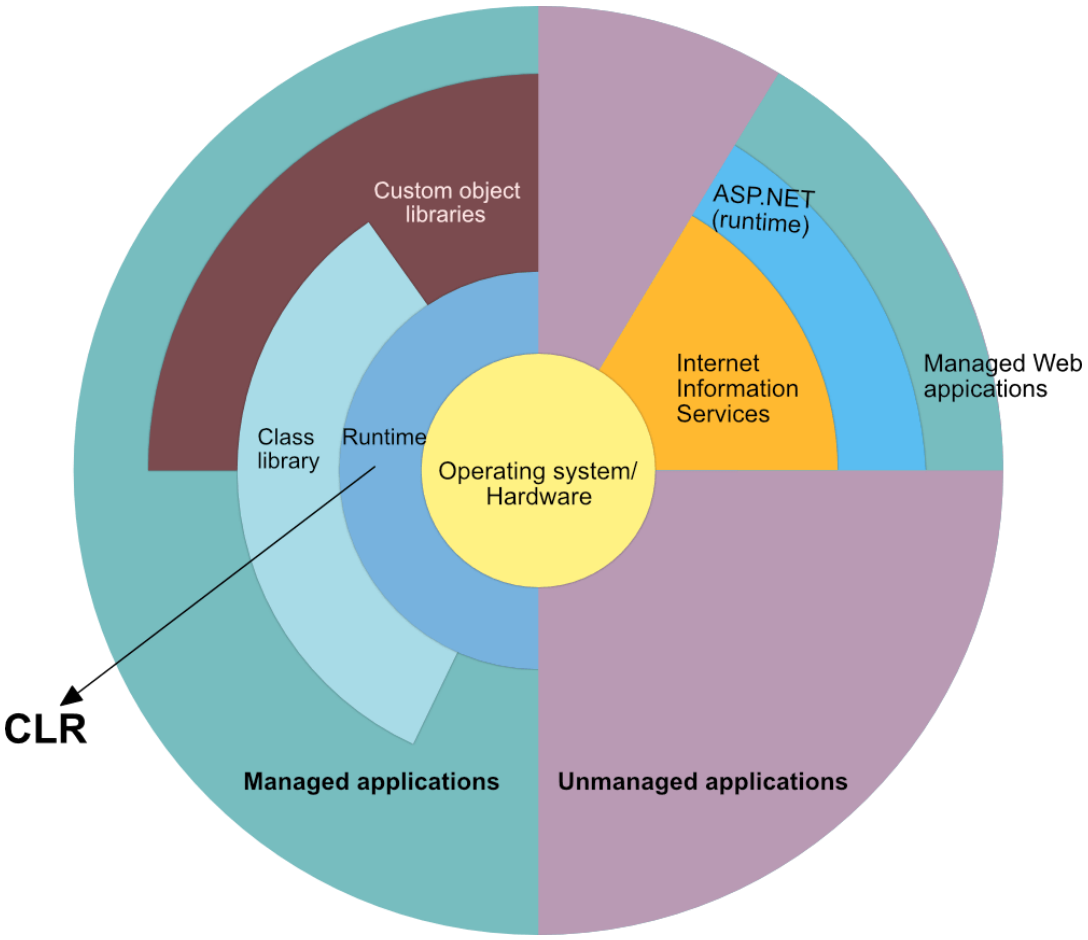


Common Language Runtime (CLR)

CLR is the basic and Virtual Machine component of the .**NET Framework**. It is the **run-time environment in the .NET Framework** that runs the codes and helps in making the development process easier by providing the various services. Basically, it is responsible for managing the execution of *.NET programs* regardless of any *.NET* programming language. Internally, CLR implements the *VES(Virtual Execution System)* which is defined in the Microsoft's implementation of the *CLI(Common Language Infrastructure).* The code that runs under the Common Language Runtime is termed as the Managed Code. In other words, you can say that CLR provides a managed execution environment for the *.NET* programs by improving the security, including the cross language integration and a rich set of class libraries etc. CLR is present in every .NET framework version

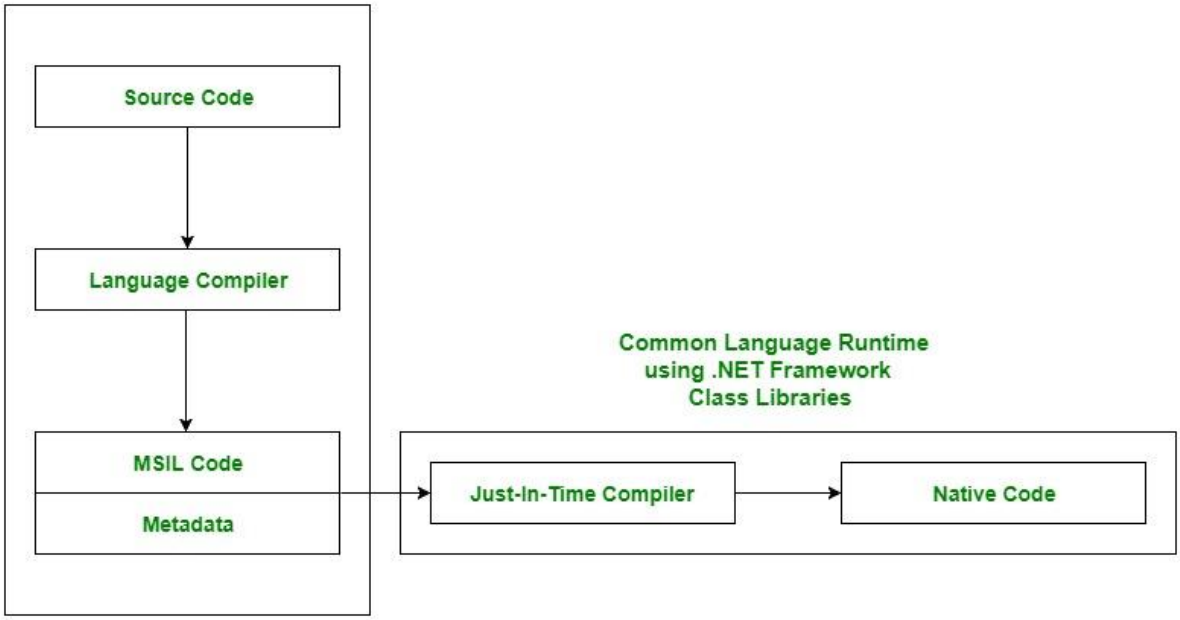
.

Below diagram illustrate how CLR is associated with the operating system/hardware along with the class libraries. Here, the runtime is actually CLR.



Role of CLR in the execution of a VB.NET program

- Suppose you have written a VB.NET program and save it in a file which is known as the Source Code.
- Language specific compiler compiles the source code into the ***MSIL(Microsoft Intermediate Language)*** which is also known as the ***CIL(Common Intermediate Language)*** or ***IL(Intermediate Language)*** along with its metadata. *Metadata* includes the all the types, actual implementation of each function of the program. MSIL is machine independent code.
- Now CLR comes into existence. CLR provides the services and runtime environment to the MSIL code. Internally CLR includes the JIT(Just-In-Time) compiler which converts the MSIL code to machine code which further executed by CPU. CLR also uses the .NET Framework class libraries. Metadata provides information about the programming language, environment, version, and class libraries to the CLR by which CLR handles the MSIL code. As CLR is common so it allows an instance of a class that written in a
- different language to call a method of the class which written in another language.



Main Components of CLR

As the word specify Common which means CLR provides a common runtime or execution environment as there are more than 60 .NET programming languages.

Main componenets of CLR:

- **Common Language Specification (CLS)**
- **Common Type System (CTS)**
- **Garbage Collection (GC)**
- **Just In – Time Compiler (JIT)**

Common Language Specification (CLS):

It is responsible for converting the different .NET programming language syntactical rules and regulations into CLR understandable format. Basically, it provides the Language Interoperability. Language Interoperability means to provide the execution support to other programming languages also in .NET framework.

Language Interoperability can be achieved in two ways :

i. **Managed Code:** The MSIL code which is managed by the CLR is known as the Managed Code. For managed code CLR provides **three** .NET facilities:

- **CAS(Code Access Security)**
- **Exception Handling**
- **Automatic Memory Management.**

ii. **Unmanaged Code:** Before .NET development the programming language like .COM Components & Win32 API do not generate the MSIL code. So these are not managed by CLR rather managed by Operating System which is called unmanaged code.

Common Type System (CTS):

Every programming language has its own data type system, so CTS is responsible for the understanding all the data type system of .NET programming languages and converting them into CLR understandable format which will be a common format.

There are 2 Types of CTS that every .NET programming language have :

a. **Value Types:** Value Types will directly store the value directly into the memory location. These types work with stack mechanism only. CLR allots memory for these at Compile Time.
b. **Reference Types:** Reference Types will contain a memory address of value because the reference types won't store the variable value directly in memory. These types work with Heap mechanism. CLR allots memory for these at Runtime.

Garbage Collector:

It is used to provide the *Automatic Memory Management* feature. Suppose if there is no garbage collector then programmers have to write the memory management codes which will be a kind of overhead on programmers.

JIT(Just In Time Compiler):

It is responsible for converting the CIL(Common Intermediate Language) into machine code or native code using the Common Language Runtime environment.

Benefits of CLR:

- It improves the performance by providing a richly interact between programs at the run time.
- Enhance portability by removing the need of recompiling a program on any operating system that supports it.
- Security also increases as it analyzes the MSIL instructions whether they are safe or unsafe. Also, the use of delegates in place of function pointers enhance the type safety and security.
- Support automatic memory management with the help of Garbage Collector.
- Provides cross-language integration because CTS inside CLR provides a common standard that activates the different languages to extend and share each other's libraries.
- Provides support to use the components that developed in other .NET programming languages.
- Provide language, platform, and architecture independency.
- It allows the creation of the scalable and multithreaded applications in an easier way as a developer has no need to think about the memory management and security issues.

CTS(Common Type System)

Common Type System (CTS) describes a set of types that can be used in different .Net languages in common . That is , the Common Type System (CTS) ensure 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 .

These types can be Value Types or Reference Types . The Value Types are passed by values and stored in the stack. The Reference Types are passed by references and stored in the heap. Common Type System (CTS) provides base set of Data Types which is responsible for cross language integration. The Common Language Runtime (CLR) can load and execute the source code written in any .Net language, only if the type is described in the Common Type System (CTS) .Most of the members defined by types in the .NET Framework Class Library (FCL) are Common Language Specification (CLS) compliant Types.

Microsoft Intermediate Language – MSIL

MSIL stands for Microsoft Intermediate Language. We can call it as Intermediate Language (IL) or Common Intermediate Language (CIL). During the compile time , the compiler convert the source code into Microsoft Intermediate Language (MSIL) .Microsoft Intermediate Language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to the native code. During the runtime the Common Language Runtime (CLR)'s Just In Time (JIT) compiler converts the Microsoft Intermediate Language (MSIL) code into native code to the Operating System.

When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata. The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file . Microsoft Intermediate Language (MSIL) includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations

Just In Time Compiler - JIT

The .Net languages , which is conforms to the Common Language Specification (CLS), uses its corresponding runtime to run the application on different Operating Systems . During the code execution time, the Managed Code compiled only when it is needed, that is it converts the appropriate instructions to the native code for execution just before when each function is called. This process is called Just In Time (JIT) compilation, also known as Dynamic Translation . With the help of Just In Time Compiler (JIT) the Common Language Runtime (CLR) doing these tasks.

The Common Language Runtime (CLR) provides various Just In Time compilers (JIT) and each works on a different architecture depending on Operating System. That is why the same Microsoft Intermediate Language (MSIL) can be executed on different Operating Systems without rewrite the source code. Just In Time (JIT) compilation preserves memory and save time during application initialization. Just In Time (JIT) compilation is used to run at high speed, after an initial phase of slow interpretation. Just In Time Compiler (JIT) code generally offers far better performance than interpreters.

Assemblies

is a logical unit of code, that contains code which the Common Language Runtime (CLR) executes. It is the smallest unit of deployment of a .net application and it can be a **.dll** or an **exe** . Assembly is really a collection of types and resource information that are built to

work together and form a logical unit of functionality. It include both executable application files that you can run directly from Windows without the need for any other programs (.exe files), and libraries (.dll files) for use by other applications.

Assemblies are the building blocks of .NET Framework applications. During the compile time Metadata is created with Microsoft Intermediate Language (MSIL) and stored in a file called Assembly Manifest . Both Metadata and Microsoft Intermediate Language (MSIL) together wrapped in a Portable Executable (PE) file. Assembly Manifest contains information about itself. This information is called Assembly Manifest, it contains information about the members, types, references and all the other data that the runtime needs for execution.

Every Assembly you create contains one or more program files and a Manifest. There are two types program files : Process Assemblies (EXE) and Library Assemblies (DLL). Each Assembly can have only one entry point (that is, DllMain, WinMain, or Main).

NET Framework Class Library

.NET Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.

It contains thousands of classes that supports the following functions.

- Base and user-defined data types
- Support for exceptions handling
- input/output and stream operations
- Communications with the underlying system
- Access to data
- Ability to create Windows-based GUI applications
- Ability to create web-client and server applications
- Support for creating web services

Introduction to Visual Studio

Visual Studio is an **Integrated Development Environment(IDE)** developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

Visual Studio Editions

There are 3 editions of Microsoft Visual Studio as follows:

1. Community:

It is a **free** version which is announced in 2014. *All other editions are paid*. This contains the features similar to Professional edition. Using this edition, any individual developer can develop their own free or paid apps like *.Net applications*, Web applications and many more. In an enterprise organization, this edition has some limitations. For example, if your organization have more than 250 PCs and having annual revenue greater than $1 Million(US Dollars) then you are not permitted to use this edition. In a non-enterprise organization, up to five users can use this edition. Its main purpose is to provide the Ecosystem(Access to thousands of extensions) and Languages(You can code in C#, VB, F#, C++, HTML, JavaScript, Python, etc.) support.

2. Professional:

It is the commercial edition of Visual Studio. It comes in Visual Studio 2010 and later versions. It provides the support for XML and XSLT editing and includes the tool like Server Explorer and integration with Microsoft SQL Server. Microsoft provides a free trial of this edition and after the trial period, the user has to pay to continue using it. Its main purpose is to provide Flexibility(Professional developer tools for building any application type), Productivity(Powerful features such as CodeLens improve your team's productivity), Collaboration(Agile project planning tools, charts, etc.) and Subscriber benefits like Microsoft software, plus Azure, Pluralsight, etc.

3. Enterprise:

It is an integrated, end to end solution for teams of any size with the demanding quality and scale needs. Microsoft provides a 90-days free trial of this edition and after the trial period, the user has to pay to continue using it. The main benefit of this edition is that it is highly scalable and deliver high-quality software.

Types of Project in .Net

The Following are some the different projects that can be created with Visual Studio.NET:

1) Console applications
2) Windows Applications
3) Web applications
4) Web services
5) Class library
6) Windows Control Library
7) Web Control Library

Console applications:

Are light weight programs run inside the command prompt (DOS) window. They are commonly used for test applications.

Windows Applications:

Are form based standard Windows desktop applications for common day to day tasks. (Ex: Microsoft word).

Web applications:

Are programs that used to run inside some web server (Ex:IIS) to fulfill the user requests over the http. (Ex: Hotmail and Google).

Web services:

Are web applications that provide services to other applications over the internet.

Class library:

Contains components and libraries to be used inside other applications. A Class library can not be executed and thus it does not have any entry point.

Windows Control Library:

Contains user defined windows controls to be used by Windows applications.

Web Control Library:

Contains user defined web controls to be used by web applications.

What is Visual Basic ,Net?

Visual Basic .NET (VB.NET) is an object-oriented computer programming language implemented on the .NET Framework. Although it is an evolution of classic Visual Basic language, it is not backwards-compatible with VB6, and any code written in the old version does not compile under VB.NET.

Like all other .NET languages, VB.NET has complete support for object-oriented concepts. Everything in VB.NET is an object, including all of the primitive types (Short, Integer, Long, String, Boolean, etc.) and user-defined types, events, and even assemblies. All objects inherits from the base class Object.

VB.NET is implemented by Microsoft's .NET framework. Therefore, it has full access to all the libraries in the .Net Framework. It's also possible to run VB.NET programs on Mono, the open-source alternative to .NET, not only under Windows, but even Linux or Mac OSX.

The following reasons make VB.Net a widely used professional language −

- Modern, general purpose.
- Object oriented.
- Component oriented.
- Easy to learn.

- Structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- Part of .Net Framework.

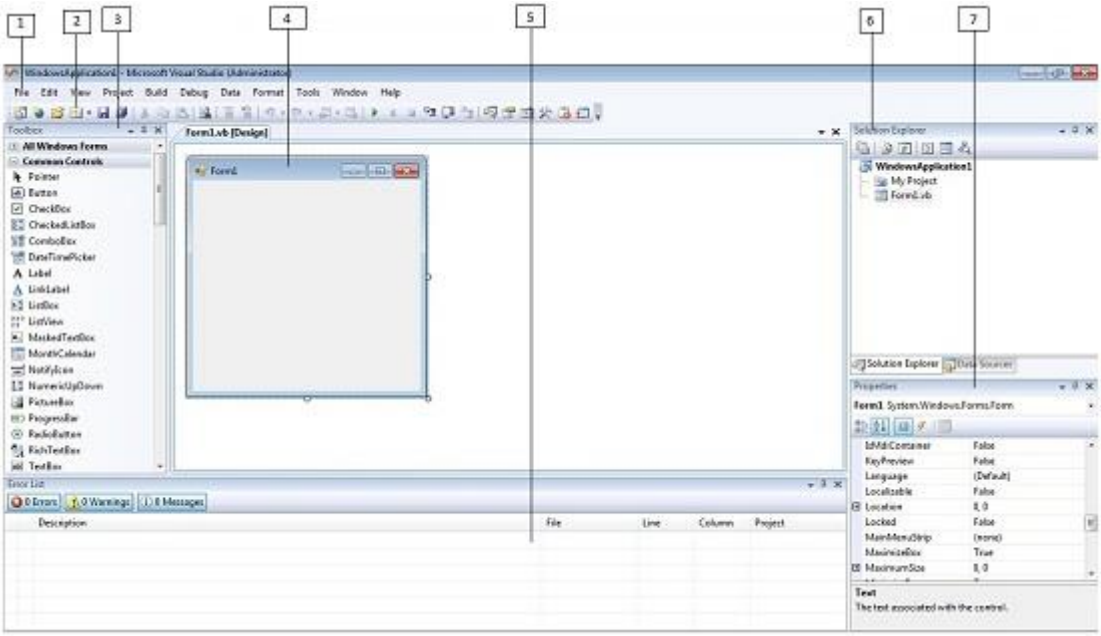
Integrated Development Environment (IDE)

An **integrated development environment** (**IDE**) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools and a debugger

Visual Studio IDE

Visual Studio is a powerful and customizable programming environment that contains all the tools you need to build programs quickly and efficiently. It offers a set of tools that help you write and modify the code for your programs, and also detect and correct errors in your programs.

Before you start learning more about VB.NET programming, it is important to understand the development environment and identify some of the frequently using programming tools in Visual Studio IDE.



The Visual Studio IDE consists of several sections, or tools, that the developer uses while programming.

Menu Bar:

is the rectangular box consisting of button and text. When a user click on these buttons , a program associated with these buttons get executed.

Tool Bar:

is the rectangular box consisting of button and icons. When a user click on these buttons , a program associated with these buttons get executed.

Toolbox

The Toolbox is a palette of developer objects, or controls, that are placed on forms or web pages, and then code is added to allow the user to interact with them. An example would be *TextBox*, *Button* and *ListBox* controls. With these three controls added to a Windows Form object the developer could write code that would take text.

Form

Form is the container for all the controls that make up the user interface. Every window you see in a running visual basic application is a form, thus the terms form and window describe the same entity. Visual Studio creates a default form for you when you create a **Windows Forms Application**.

Every form will have title bar on which the form's caption is displayed and there will be buttons to close, maximize and minimize the form

Error List Window

This window is populated by the compiler with error messages, if the code can't be successfully compiled. You can double-click an error message in this window, and the IDE will take you to the line with the statement in error — which you should fix. Change the MsgBox() function name to MssgBox(). As soon as you leave the line with the error, the name of the function will be underlined with a wiggly red line and the f**ollowing error** description will appear in the Error List window:

Name 'MsssgBox' is not declared

Solution Explorer

This is a section that is used to view and modify the contents of the project. A Visual Studio Windows Application Project will generally have a Form object with a code page, references to System components and possibly other modules with special code that is used by the application.

Server Explorer Window

Is shortcut to accessing server, either installed on the system, or connected to the system. These servers are usually database servers such as SQL server. By accessing the server, you access all the database on the specific server, and then you can build the connection needed inside your program.

Properties Windows

The properties windows shows all the control (like *TextBox*) properties to be changed at design time. Most of these properties can be also changed with code at run time, but basically most properties change the way the control is displayed on your application.

Object Browser

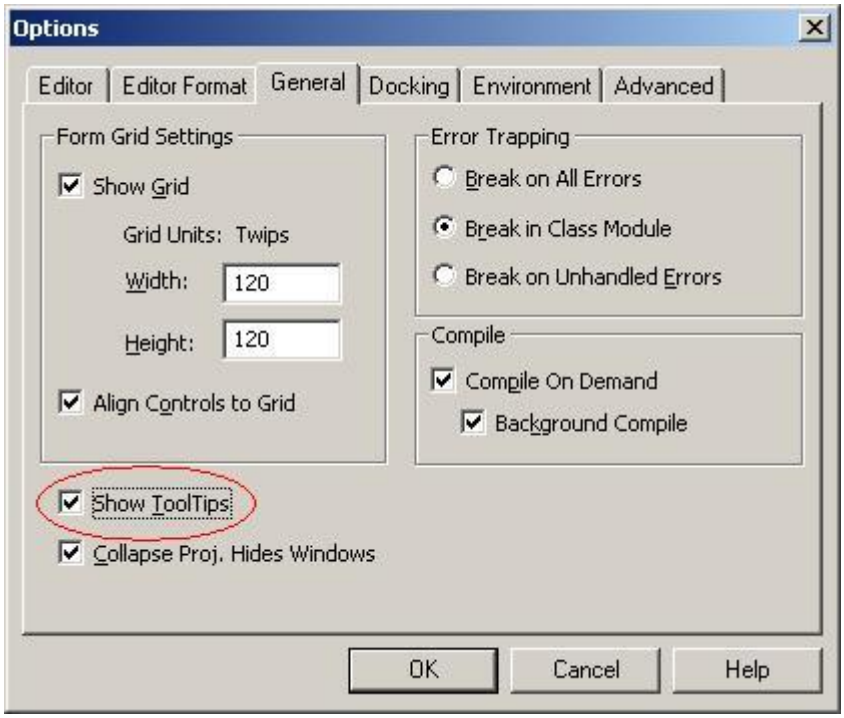
By pressing F2 or selecting it into the *View* menu, it's possible to explore all the available objects of the libraries (types, functions)

UNIT-2

The VB.NET Enviornment

Provides the following tabs to change the default settings of Visual Basic Development Environment.

To change the default setting click on Tool on menu on menu bar and on submenu Option then the following tabs are displayed



Editor Tab

Auto Syntax Checker - Determines whether Visual Basic should automatically verify correct syntax after you enter a line of code. This is useful when you are learning VBA initially although once you are familiar with the syntax and you are editing lots of code you may want to switch this off. When this is unchecked, the line is displayed in a different colour (red by default) to indicate a syntax error. This is ticked by default.

Require Variable Declaration - This should definitely be checked as it prevents incorrect variables from being defined in your code. Selecting this check box will automatically add the statement **Option Explicit** to any **new modules** (not existing ones). When this statement is used you must explicitly declare all your variables before using them. This is not ticked by default.

Auto List Members - This should definitely be checked as it will assist you in finding the correct properties and methods for any objects. You can insert the relevant property or method by selecting then by pressing Tab. This is a feature of Intellisense. This is ticked by

default.

Auto Quick Info - This should definitely be checked as it displays information about functions and their corresponding parameters. This is a feature of Intellisense. This is ticked by default.

Auto Data Tips - This should definitely be checked as it displays the value of a variable or expression when you hover over with the cursor. This is only available in Break Mode when you are Debugging > Step Through Code. This is ticked by default. Auto Indent - Repeats the indent of the preceding line when you press Enter - all subsequent lines will start at the indent. You can press Backspace to remove automatic indents. You should use the Tab key to indent your code (not the spacebar). You can then also use (Shift + Tab) to remove an indent. This is ticked by default.

Tab Width - Defines the number of characters that are passed over by a tab. This can range from 1 to 32 characters (the default is 4). Remember to always use the Tab key and not the spacebar. You can also use (Shift + Tab) to un-indent lines of code. Indenting your code makes it easier to read. Changing this option can cause problems when code is read or edited on different machines. The default is 4.

Drag & Drop Text Editing - Allows you to drag and drop selections of text from the Code Window into the Immediate & Watch Windows. This is ticked by default.

Default in Full Module View - This decides if you only see one procedure at a time or all the procedures, as a single scrollable list in any **new modules** (not existing ones). When this option is switched off you will only see one procedure at a time. This is ticked by default.

Procedure Seperator - Allows you to display or hide seperator bars that appear at the end of each procedure in the Code Window. This is ticked by default.

Editor Format Tab

All these options allow you to control the text colour and background colours for various items in your Code Window.

Most people do not change these options.

Colour Code - It is possible to customise the formatting for all these items: Normal Text, Selection Text, Syntax Error Text, Execution Point Text, BreakPoint Text, Comment Text, Keyword Text, Identifier Text.

Foreground - This is the colour that specifies the Text colour.

Background - This is the colour that specifies the Background colour of the text.

Indicator - This is the colour of the margin indicator that will appear next to that particular item.
Font - The font used to display your code. The default font is Courier New (Western). To make your code easy to read you should always use a fixed-width font, (ie all characters have the same width).

Size - The font size used to display your code. The default font size is 10.

Margin Indicator Bar - This is the margin that appears on the left of your code module. This option lets you toggle whether the margin indicator bar is visible or not.

Docking Tab
Controls precisely which of the windows are dockable and which are floating. When a window is dockable it can be anchored to one of the sides of the Visual Basic Editor. Whenever you move a dockable window a rectangle will appear to show you where it will be docked if released from the current position.

These are global settings used across Excel, Word, PowerPoint, Access and Outlook.

Immediate Window - This is ticked by default.
Locals Window - This is ticked by default.
Watch Window - This is ticked by default.
Project Explorer - This is ticked by default.
Properties Window - This is ticked by default.
Object Browser - This is not ticked by default.

Event Driven Programming

Event-driven programming is a programming paradigm in which the flow of program execution is determined by *events*.

What is event?

Events determine the control's reactions to external conditions. Controls recognize events, but your application handles them. A Command button will recognize that it was clicked. but it won't react to the event unless you provide some code. In other words, you must tell Visual Basic.net what to do when the user clicks the specific ,Command button. Once you specify a subroutine for the control's Click event, this subroutine executes each time the control is clicked. The subroutine that determines how a control reacts to an event .

To write an event handler for a control, follow these steps:

1. Switch to the Code window, or double-click the control for which you want to write the event handler.

2. At the top of the Code window,, you will see two drop-down lists. The first contains the names of all the controls on the Form. Select the control for which you want to write an event handler. The second list contains all the events the selected control can recognize. Select the event for which you want to write an event handler.

The combination of the control's name and the event's name is unique and is the . name of the event handler. Each time an event takes place, Visual Basic looks for the subroutine made up of the name of the control on which the event took place and the name of the event. If such a handler exists, it's executed. If not, your application won't react to the event.

The two most. common groups of events are mouse (events caused with the mouse) and keyboard (events caused with the keyboard).

- Mouse Click event
- KeyPress event

What is Method?

Objects have methods too, which are the actions they can carry out. You can think of methods as the actions of an object. For example, the methods of your VCR are the Play, Fast Forward, Rewind, Pause, and Record buttons. After you press one of these buttons, your VCR can perform without any further assistance from you. A Form also knows how to hide itself, an action that you can invoke from within your code with the Hide method.

Questions

[Marks 2]

1. Define .net framework
2. What is the use of assembly?
3. What is JIT?
4. What is IDE?
5. Define data type ?
6. Differentiate between CUI and GUI.
7. What is the main purpose of Garbage Collector?
8. What is MSIL
9. Explain the role of CTS
10. What is event driven programming?

[Marks 4]

1. Explain is scope and life time of memory variable.
2. Differentiate between Visual Basic and Visual Basic .Net
3. What is Object Oriented Programming approach?
4. Differentiate between even and method

[Marks 8]

1. Explain different type of applications which can developed in .net framework
2. Explain the components of VB.Net IDE.