Unit 1 Introduction to VB.NET

Structure:

1.0 Objectives

After studying this unit, you will be able to:

- explain the advantages of event driven programing
- list and explain the components of .NET framework
- define the .NET architecture
- explore the visual studio .NET environment

1.1 Introduction

The .NET is the product of Microsoft that allows the users to have access to their information, files, or programs on any platform or device. When the Microsoft launched their first windows operating system, it gave a new scenario towards the application and the system design by supporting the multitasking environment. The updated versions of windows driven towards the distributed environment in that way .NET is the next exploration in this path. This course helps you to explore the details of how to develop Visual programming in .NET platform. Before we could start up with the Visual Basic .NET, it is mandatory to become familiar with the .NET Framework, Visual Studio .NET, and the Microsoft Development Environment. First we are going to discuss the concept of event driven programming and its benefits. Also we are going to learn the architecture of .NET platform. This unit is also discussing the role of JIT compiler. We are concluding this unit with the discussion of .NET framework class library is considered as one of the important component in .NET framework.

1.2 Event Driven Programming

In event driven or event based programming the flow of program is determined by events. It relies on events, triggers and handlers, program executed based on the event will be triggered by the GUI elements.

Event driven programming can also be defined as an application architecture technique here

the application has a main loop and further it has been divided in to two sections.

- selection/detection of event
- handling event.

Event: Events are the actions that are performed by the user during the applications usage. It is a signal that informs an application that something has occurred. For example if a user clicks a mouse button on any object then the Click event occurs.

Event handler: Event handlers are nothing but the procedures that are called when a corresponding event occurs. If the user clicks on any object available here the object click procedure will be called with without arguments.

Advantages of event driven programming

Following are the major advantages and uses of event driven programming

- Flexibility
- Suitability for graphical interfaces
- Programming simplicity
- Ease of development

Flexibility:

Event driven programming is considered as one of most flexible programming languages. It provides the flexible way to the programs to respond for many different inputs or events. It helps the programmer to design visually with the existing objects. It provides the technique of separation of the event detection and the event handling leads to design the program in simple and flexible way.

Suitability for graphical interfaces:

In event driven programming programmers creates the program in a graphical way. Developers get the options of selecting the different controls available and place them on the container according to their needs. Executions of these controls are decided by the programmers through the events with the very minimal code. Event driven programming leave the control of program execution to the programmers also the user does not follow any

routine procedure to use the program.

Programming Simplicity:

A visual effect of event driven programming simplifies the complexity of the programmer's task. Here the programmer's will not involve in the task of designing and alignment of control since it is readily available. It is enough to drag and place the controls wherever necessary. After placing if you want to change the look and feel of the control, it is easy to update these features in the respective property column. Such readymade facility makes the programmer easy and comfortable to design an application in even driven programming environment.

Ease of development:

As we discussed earlier developing a program in an event driven language is easy and also here the programmer has to deal with single event at a time. Here various controls in the containers are designed or programmed separately for the different purposes and different point of time, if it necessary it works some time together also.

1.3 .NET Framework

.NET framework is the product of Microsoft has huge number of libraries and provides language interoperability runs primarily on windows platform. The .NET Framework is defining the background to execute Visual Basic

.NET applications and supporting various services to run the application. This .NET framework supports the traditional way of running applications in the windows platform also supports the application to run on the internet environment. The applications written in .NET framework will work on the software platform called CLR (Common Language Runtime). Library class in

.NET framework provides various services like data base connectivity, web application support, cryptography and various communications in network.

The common language runtime and class library together structure the .NET framework. Microsoft Visual studio is the integrated development environment to develop application for the .NET software. This platform encourages the windows application developers to use the .NET framework large in size.

1. Following are the main objectives of the .NET framework

2. Consistent support for object-oriented programming environment immaterial of whether the object code is located and executed locally, Internet-distributed, or executed remotely.

3. Designed to support minimum software deployment and conflict version in Code-execution environment

4. Providing safety in execution of code even it is created by a third party or unknown person

5. Designed to handle the performance problems that may arise through the interpreted environments or scripted codes.

6. Maintain the experience of developer's experience in a consistent way across the applications, based on Windows and Web.

7. Build all communication on industry standards to guarantee that code based on the .NET Framework can integrate with any other code.

Design features of .NET Framework

Common language runtime: CLR act as an execution engine for .NET environment it takes care of execution of code, memory, compilation process, safety services, execution of threads etc.

Base class library: It is a collection of reusable interfaces, classes and value types. Base Class Library is the small subset of class library supports the basic API of the Common Language Runtime.

Language independence: .Net framework supports Common Type System (CTS), it has the details about the allowable programming syntaxes and the data types and how it interacts with each other in the CLR environment. This in turn supports in exchange of instances between the programs that are written in the .NET languages.

Portability: Microsoft .NET framework accelerates third parties to construct or develop compatible implementations of this framework and its languages on platforms apart from the Microsoft platform

1.4 .NET Architecture and Visual Studio .NET

To develop a Visual Basic .NET application, we can use a product called Visual Studio .NET it is a collection of products that includes three programming languages depicted in figure 1.1. Now we are going to discuss the role of Visual Basic .NET, and its role in rapid application development. Visual Studio supports different programming languages using language services; this allows the code editor and debugger to support nearly any programming language, provided a language-specific service exists.

Visual Studio comprises of several other components that make it an outstanding development product. One of the important among this is the

Microsoft Development Environment. Another is the Microsoft SQL Server 2000 Desktop Engine (or MSDE). MSDE is a database engine that runs on PC helps to develop database applications those are compatible with Microsoft SQL Server. SQL Server is a database management system supports in providing data to large networks of users also for an Internet applications. The two other languages that come with Visual Studio .NET are C# and C++. C# .NET (pronounced "C sharp dot net") is a new language that has been developed by Microsoft especially for the .NET

Framework. Visual C++ .NET is Microsoft's version of the C++ language that is used on many platforms besides Windows PCs.

Visual Studio .NET can be used on any PC that runs Windows 2000 or later versions. You can also see that the applications that are developed with Visual Studio .NET can be run on any PC that runs Windows 98 or later, depending on which .NET components are used by the application. From a practical point of view, though, you can assume that the applications that you develop with Visual Basic .NET will be run on PCs that are using Windows 2000 or later. Visual Basic .NET comes in an inexpensive

Standard Edition that includes only the Visual Basic language, not

C# or C++.

Although the three languages shown in figure 1.1 are the only three programming languages you can use within Visual Studio .NET, other vendors are free to develop languages for the .NET Framework. For example, Fujitsu has already developed a version of COBOL for the .NET Framework.

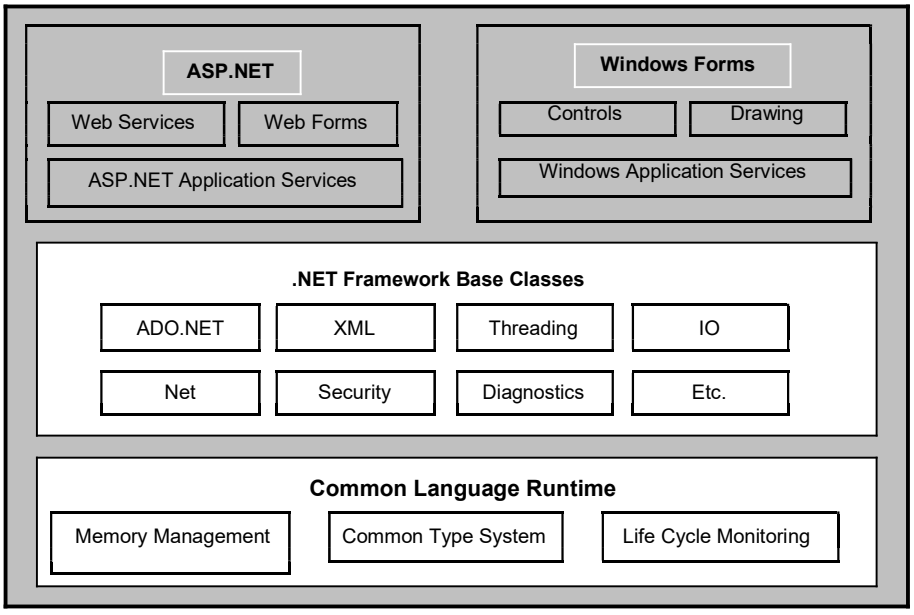

Fig. 1.1: Visual Studio .NET and the .NET Framework

Programming languages supported by Visual Studio .NET

Language	Description
Visual Basic .NET	Designed for rapid application development
Visual C# .NET	A new language that combines the features of Java and C++ and is suitable for rapid application development
Visual C++ .NET	Used to develop performance applications which are high in nature.

Components of Visual Studio .NET

Component	Description
Microsoft SQL Server 2000 Desktop Engine	It is a database engine which can run on PC through which we can develop applications based on visual studio those are having compatibility with the Microsoft SQL Server.

Platforms that can run Visual Studio .NET

- Windows 2000 and later releases of Windows

Platforms that can run Visual Studio .NET applications

- Windows 98 and later releases of Windows, depending on which .NET components the application uses.

Visual Basic .NET Standard Edition

1. An inexpensive alternative to the complete Visual Studio .NET package that supports a limited version of Visual Basic .NET as its only programming language.

   - The .NET Framework defines the environment that you use for executing Visual Basic .NET applications.

   - Visual Studio .NET is a suite of products that includes all three of the programming languages listed above. These languages run within the .NET Framework.
   - You can develop business applications using either Visual Basic

     .NET or Visual C# .NET. Both are integrated with the design environment, so the development techniques are similar although the language details vary.

   - Besides the programming languages listed above, third-party vendors can develop languages for the .NET Framework. However, programs written in these languages can't be developed from within

     Visual Studio .NET.

1.5 Just-In-Time Compiler

The program which is written in high level language requires appropriate runtime while compiling. The architecture on which the language runs has the details to execute its code. All the programming languages require its respective runtime to run the application. For example, to run an application developed using Visual Basic requires the system has the Visual Basic runtime is installed. Because the VB runtime can run the applications that are developed using visual basic code not the other codes like java or c#.

The main benefit of .NET Framework is the interoperability between various languages. In the .NET Framework, all the Microsoft .NET languages use a CLR that resolves the problem

of installing separate runtime for each of the programming languages. Microsoft .NET Common Language Runtime installed on a computer can run any language that is Microsoft .NET compatible.

As all the Microsoft .NET languages share the common runtime language, they all work well together. For example, you can use an object written in C# from Visual Basic.NET. The same applies for all the other Microsoft .NET languages. .NET compatible provide the Microsoft .NET CLR to install on a computer. Before the MSIL (Microsoft Intermediate Language) 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. MSIL is a group of instructions that can be rapidly translated into native code. A Microsoft.NET application can be run only after the MSIL code is translated into native machine code. In .NET Framework, the intermediate language is compiled "just in time" (JIT) into native code when the application or component is run instead of compiling the application at development time.

Types of JIT compilers

Pre-JIT Compiler: It compiles the entire code in to native code in a single cycle during the deployment of application.

Econo-JIT compiler: Here the compilation happens only at the time of requirement as well it will be freed as soon as the work gets over.

Normal-JIT compiler: Here the compilation begins when the particular method is called and the same will be stored in the cache memory. When
the same method is called in future the stored content from cache will be retrieved for execution.

1.6 .NET Framework Class Library

Framework provides a common set of services that application programs written in a .NET language such as Visual Basic .NET can use to run on various operating systems and hardware platforms. As we discussed you can understand that the .NET Framework is divided into two main components: the .NET Framework Class Library and the Common Language Runtime as shown in the figure 1.2.

The .NET Framework Class Library defined as “consists of segments of pre-written code

called classes that provide many of the functions that you need for developing .NET applications". For instance, the ASP.NET classes are used to develop Web Forms applications. The Windows Forms classes are used for developing Windows Forms applications and other classes allow you to work with databases, access files, manage security, and to complete many other functions.

These are not stated obviously in figure 1.2; the hierarchical structure is implemented to organize the classes inside the .NET Framework Class Library. These structures consist of related classes, organized into groups called namespaces. Each function needs to support by the namespace that contains the respective classes. For example, the System. Windows. Forms namespace classes used to create forms and the System. Data namespace classes you use to access data.

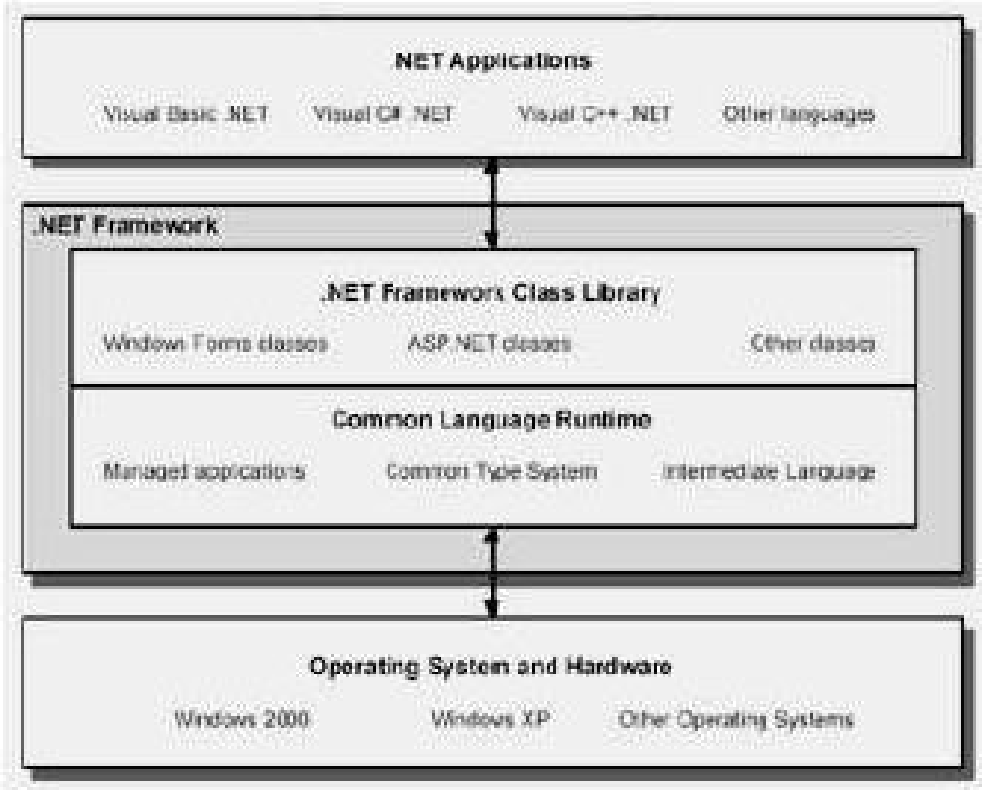

Figure 1.2: The components of the .NET Framework

The .NET Framework Class Library provides pre-written code in the form of classes that are available to all of the .NET programming languages. This class library consists of hundreds of classes, but you can create simple

.NET applications once you learn how to use just a few of them. Few of the most used namespaces are listed below in the table 1.1.

Table 1.1: Name spaces

Name Space	Description
System	Contains fundamental classes which define data types, events and event handlers, interfaces, attributes, and processing exceptions etc.
System.Activities	Pertaining to classes require to create and work with Activities
System.AddIn	contain types used to identify, register, activate, and control add-ins and allow to communicate with host Application
System.Collections	contain types that define various standard, specialized, and generic collection objects
System.Configuration	Contains data pertaining to configuration data, like data in machine or application configuration files
System.Data	Contains classes for managing and accessing data from various sources.
System.Deployment	Contains data supports in deployment of Click One Applications
System.Dynamic	Provides classes and interfaces that support Dynamic Language Runtime.
System.IO	Contains types that support input and output, including the ability to read and write data to streams.
System.Linq	Contain types that support input and output, including the ability to read and write data to streams.

.Net Framework Class Library (FCL)

The .Net Framework class library (FCL) provides the core functionality of .Net Framework architecture . The .Net Framework Class Library (FCL) 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 Classes are accessed by namespaces, which reside within Assemblies. The System Namespace is the root for types in the .NET Framework. The .Net Framework class library

(FCL) classes are managed classes that provide access to System Services . The .Net Framework class library (FCL) classes are object oriented and easy to use in program developments. Moreover, third-party components can integrate with the classes in the .NET Framework.

Common Language Specification - CLS

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. 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 . Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language used to create the application.

Common Language Specification (CLS) defines a subset of Common Type System (CTS) . Common Type System (CTS) 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 (CLS) compliant Types. Moreover Common Language Specification (CLS) standardized by ECMA .

Common Type System - CTS

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 .NETFramework 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

Portable Executable (PE) File Format

The Portable Executable (PE) format is a file format for executables, object code, and DLLs, used in 32-bit and 64-bit versions of Windows operating systems.

The PE file format was defined to provide the best way for the Windows Operating System to execute code and also to store the essential data which is needed to run a program. Portable Executable File Format is derived from the Microsoft Common Object File Format (COFF).

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.

Managed Code

Managed Code in Microsoft .Net Framework, is the code that has executed by the Common Language Runtime (CLR) environment. On the other hand Unmanaged Code is directly executed by the computer's CPU. Data types, error-handling mechanisms, creation and destruction rules, and design guidelines vary between managed and unmanaged object models.

The benefits of Managed Code include programmers convenience and enhanced security . Managed code is designed to be more reliable and robust than unmanaged code , examples are Garbage Collection , Type Safety etc. The Managed Code running in a Common Language Runtime (CLR) cannot be accessed outside the runtime environment as well as cannot call directly from outside the runtime environment. This makes the programs more isolated and at the same time computers are more secure . Unmanaged Code can bypass the .NET Framework and make direct calls to the Operating System. Calling unmanaged code presents a major security risk.

Microsoft .Net Metadata

Metadata in .Net is binary information which describes the characteristics of a resource . This information include Description of the Assembly , Data Types and members with their declarations and implementations, references to other types and members , Security permissions etc. A module's metadata contains everything that needed to interact with another module.

During the compile time Metadata created with Microsoft Intermediate Language (MSIL) and stored in a file called a Manifest . Both Metadata and Microsoft Intermediate Language (MSIL) together wrapped in a Portable Executable (PE) file. During the runtime of a program Just In Time (JIT) compiler of the Common Language Runtime (CLR) uses the Metadata and converts Microsoft Intermediate Language (MSIL) into native code. When code is executed, the runtime loads metadata into memory and references it to discover information about your code's classes, members, inheritance, and so on. Moreover Metadata eliminating the need for Interface Definition Language (IDL) files, header files, or any external method of component reference.

Microsoft .Net Assembly

Microsoft **.Net Assembly** is a logical unit of code, that contains code which theCommon 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).

We can create two types of Assembly:

1. Private Assembly

2. Shared Assembly

A private Assembly is used only by a single application, and usually it is stored in that application's install directory. A shared Assembly is one that can be referenced by more than one application. If multiple applications need to access an Assembly, we should add the Assembly to the Global Assembly Cache (GAC). There is also a third and least known type of an assembly: Satellite Assembly . A Satellite Assembly contains only static objects like images and other non-executable files required by the application.

Garbage Collection

Memory management is the main concern for any application whether application is window based or web based. In .Net, CLR has garbage collector that executes as a part of our program and responsible for reclaiming the memory of no longer used objects. Garbage collector free the memory for objects that are no longer referenced and keeps the memory for future allocations.

Advantage of Garbage Collector

1. Allow us to develop an application without having worry to free memory.
2. Allocates memory for objects efficiently on the managed heap.
3. Reclaims the memory for no longer used objects and keeps the free memory for future allocations.
4. Provides memory safety by making sure that an object cannot use the content of another object.

Conditions for a garbage collection

Garbage collection occurs when one of the following conditions is true:

- The system has low physical memory.
- The memory that is used by allocated objects on the managed heap surpasses an acceptable threshold. This threshold is continuously adjusted as the process runs.
- The System.GC.Collect method is called. In almost all cases, you do not have to call this method, because the garbage collector runs continuously. This method is primarily used for unique situations and testing.

The managed heap

After the garbage collector is initialized by the CLR, it allocates a segment of memory to store and manage objects. This memory is called the managed heap, as opposed to a native heap in the operating system.

There is a managed heap for each managed process. All threads in the process allocate memory for objects on the same heap.

To reserve memory, the garbage collector calls the Win32 VirtualAlloc function, and reserves one segment of memory at a time for managed applications. The garbage collector also reserves segments as needed, and releases segments back to the operating system (after clearing them of any objects) by calling the Win32 VirtualFree function.

The fewer objects allocated on the heap, the less work the garbage collector has to do. When you allocate objects, do not use rounded-up values that exceed your needs, such as allocating an array of 32 bytes when you need only 15 bytes.

When a garbage collection is triggered, the garbage collector reclaims the memory that is occupied by dead objects. The reclaiming process compacts live objects so that they are moved together, and the dead space is removed, thereby making the heap smaller. This ensures that objects that are allocated together stay together on the managed heap, to preserve their locality.+

The intrusiveness (frequency and duration) of garbage collections is the result of the volume of allocations and the amount of survived memory on the managed heap.

The heap can be considered as the accumulation of two heaps: the large object heap and the small object heap.

The large object heap contains very large objects that are 85,000 bytes and larger. The objects on the large object heap are usually arrays. It is rare for an instance object to be extremely large.

UNIFIED CLASSES(Base Class Library)

The term, .NET Framework refers to the group of technologies that form the development foundation for the Microsoft .Net platform the key technologies in this group are the run time and the class libraries.

The unified classes(Base class Library) is a Set of classes. Framework Classes that provide useful functionality to CLR programmers the ,NET Framework class library exposes features of the runtime and simplifies the development of the .Net based Applications in additions, developers can extend classes by creating their own libraries of classes. All applications(web, windows, XML web services) access the same .Net Framework class libraries which are held in namespaces. All .NET Framework class libraries which are held in namespaces all .Net based languages also access the same libraries.

The run time is responsible for managing your code and providing services to it while it executes, playing a role similar to that of the V. B. 6.0 run time

The .NET programming language including visual basic .NET, Microsoft Visual C# and C++ managed extension and many other programming languages form various vendors utilize .Net services and features through a common set of unified classes.

The .NET unified classes provided foundations of which you build your applications, regardless of the language you use whether you simply concatenating a string or building a windows services or a multiple – tier web based applications you will be using these unified classes. The unified classes provide a consistent method of accessing. The plate forms functionality once you learn to use the class Library, you will find that all tasks follow the same uniform Architectures you no longer need to learn and master different API Architecture to write you. applications.

By building your applications on a unified integrated framework, you maximize your return on the time you spend learning this framework and you end up with more robust applications that are easy to deploy and maintain.

1.7 Summary

- The .NET is the product of Microsoft that allows the users to have access to their information on any platform or device.
- The flow of program is controlled or decided by the event in event driven programming.
- Event and event handler are the two main components under the event driven programming concept.
- Flexibility, suitability for graphical interface, programming simplicity and the ease of program development are the advantages of event driven programming.
- .NET framework is the product of Microsoft contains libraries and provides language interoperability.
- To develop a Visual Basic .NET application, Visual Studio .NET is used it has a collection of products and includes three programming languages.

- Visual Basic .NET, Visual C#.NET and Visual C++ .NET are the three programming languages supported by Visual Studio .NET.

- Visual Studio .NET can run on Windows 2000 and later releases.

- The role of Just-in-time compiler is to convert a CIL (Common Intermediate Language) code into native code.

- .NET Framework Class Library consists of classes that support in developing .NET applications.

1.8 Questions and Exercises

1. What is event driven programming?
2. List and explain the advantages of event driven programming.
3. Discuss on .NET framework.
4. Explain the .NET architecture and Visual Studio .NET environment
5. Brief the role of Just-In-Time Compiler.
6. Discuss on .NET Framework class library.

.

1.9 Suggested Readings:

- http://computersight.com/computers/uses-and-advantages-of-event-driven-programming

- http://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.71).aspx

- http://www.c-sharpcorner.com/UploadFile/nipuntomar/jit-just-in-time-compiler

- http://msdn.microsoft.com/en-us/library/gg145045.aspx