

C#.NET 4.0

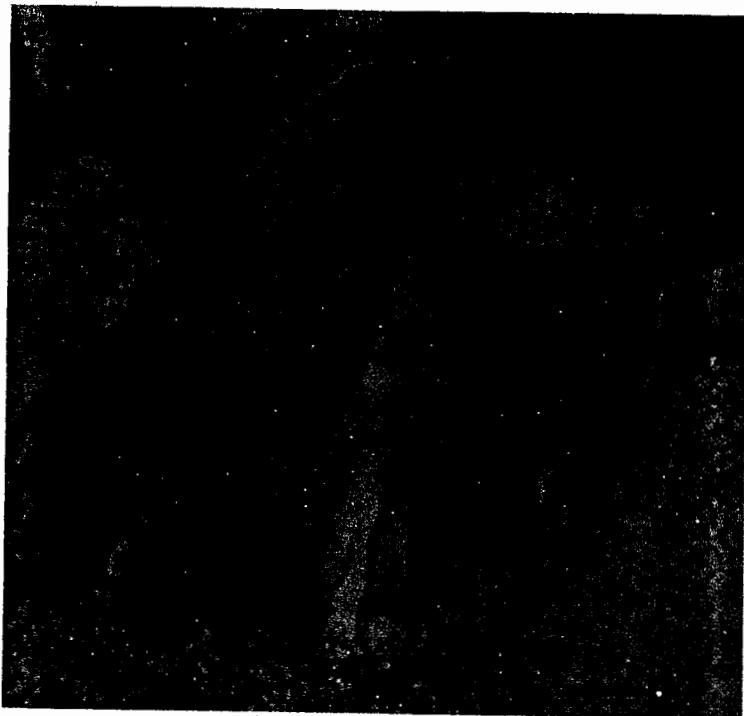
COURSE MATERIAL



2nd Floor, Sri Sai Arcade, Beside: Aditya Trade Centre,
Ameerpet, Hyderabad - 500038

Ph: 040 65538958 / 65538968 / 65538978

B. MADHU



OGIES

Dedicated to the Divine and Lotus feet of my beloved bhagwan Sri Sathya Sai Baba garu

Mahesh Sathya



Prepared By
MaheshBabu ArramaRaju
M.Phil (CS), MTech (CS)
Microsoft Certified Application Developer

INDEX

01	Why .Net and What is .Net	1
02	.Net Framework	16
03	C#.Net	32
04	Object Oriented programming Concepts	68
05	Constructors	77
06	Implementing Polymorphism and Inheritance in C#.NET	88
07	Properties	113
08	Sealed Classes and Partial Classes	122
09	POINTERS	127
10	GENERICs	129
11	DELEGATES	132
12	Exception Handling Mechanism	137
13	Designing .NET components or Assemblies or User Defined Class Libraries	146
14	Windows Forms Applications	153
15	MDI Applications	197
16	ADO.NET	201
17	Working with Connection Object	210
18	Command Object	217
19	Working with Stored Procedures	229
20	Working with DataReader	240
21	Working with Dataset	246
22	Building N - Tier Architecture Applications	265
23	Typed Dataset and Crystal Reports	278
24	Working with XML	289
25	Serialization and Deserialization	300
26	Setup and Deployment	307

Chapter 1

Why .Net and What is .Net

1.1 Introduction

In general .Net is used for network and internet computing so first we look into the different types of Networks and Network computations available in the IT field.

1.2 Types of Networks

1. LAN (Local Area Network)
2. MAN (Metropolitan Area Network)
3. WAN (Wide Area Network)
4. EAN (Enterprise Area Network)

1.2.1 LAN

Network Limited to single organization located at single place is known as LAN

Ex: Network present in a Bank, Network present in an Organization, Network present in an Office etc.

1.2.2 MAN

Network Limited to single city and its sub-urban places is known as MAN

Ex: Citi cable Network

1.2.3 WAN

Network which has no Limit in the universe is known as WAN

Ex: Internet.

1.2.4 EAN:

There are 2 categories available in this Intranet, Extranet

Intranet: Network Limited to group of single organizations located at places is internet.

Ex: Banking network

Extranet: Network limited to group of organizations located at various place is known as extranet.

Ex: ATM Network

1.3 Types of Network Computing

1. Centralized computing.
2. Client – Server computing.
3. Distributed computing.

1.3.1 Centralized Computing

- In this method there is a centrally located server.
- All other computers are connected to this server are client.
- These clients connected to the server are known as dumb terminals because there are no resources present at client side like
 1. Processor
 2. RAM
 3. Hard Disk
 4. Operating System etc.

As there are no resources at client side processing is getting done at server side only.

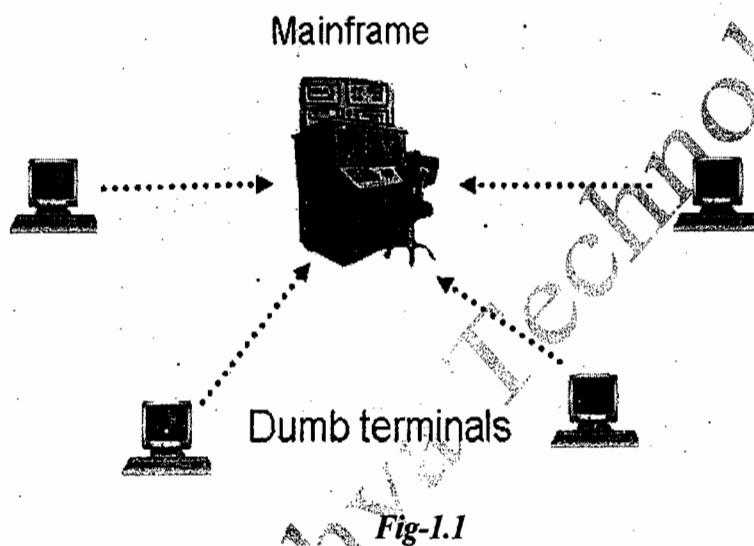


Fig-1.1

Disadvantages of Centralized Computing

When clients increase to large number then

- Burden on the server increases and performance will decrease because every client processing should be done at server only.
 - Data transfer in the network will increases so network will become slow.
- To overcome these disadvantages client server computing is used.

1.3.2 Client Server Computing

- In this method client are connected to centrally located server similar to centralized computing but client are known as smart or intelligent terminals because every client contains all the resources like.
 1. Processor
 2. RAM
 3. Hard Disk
 4. Operating System etc.
- As processor is present at client side processing is getting done at client side only.
- Clients need not depend on server for processing.

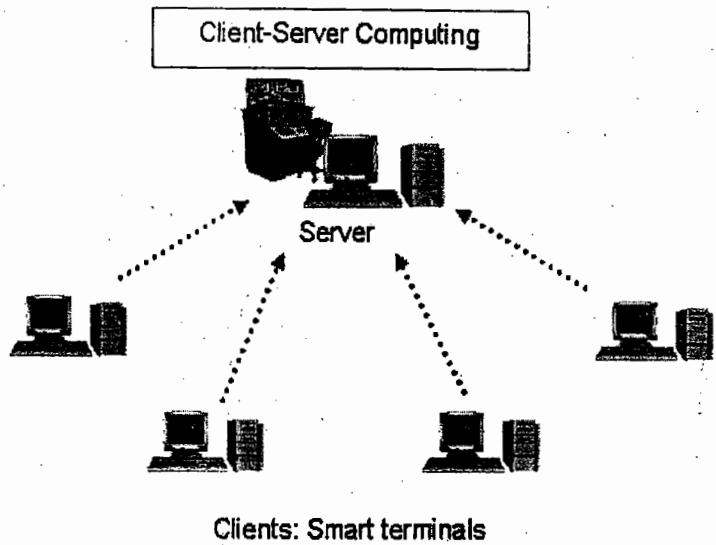


Fig 1.2

Advantages of Client Server Computing

When clients increase to large number

- There doesn't be any burden on the web server.
- There doesn't be any burden on the network.

1.3.3 Distributed Computing:

- In this method, the logic code of an application that is to be executed is divided among many machines in the network, to complete the execution at faster rate and to provide the flexibility.
- After completion of execution result will be integrated.

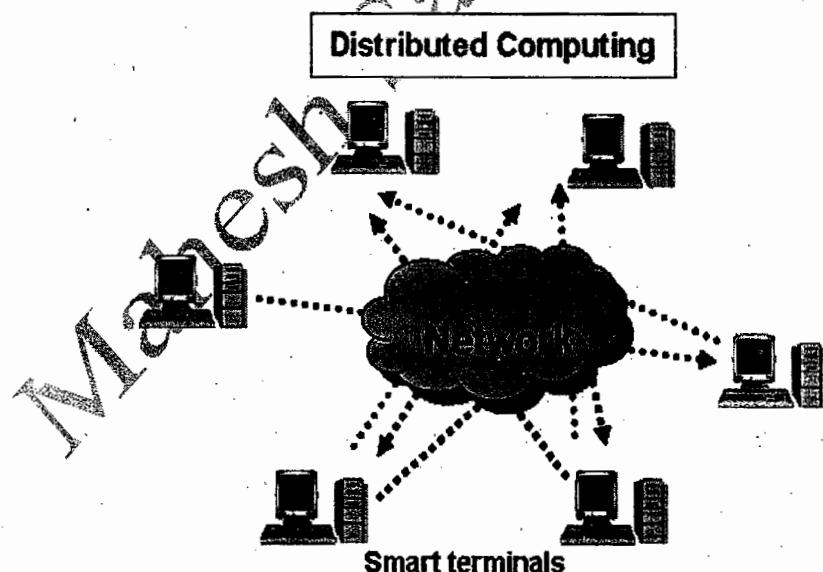


Fig 1.3

1.3.4 Current Scenario Of Distributed Computing

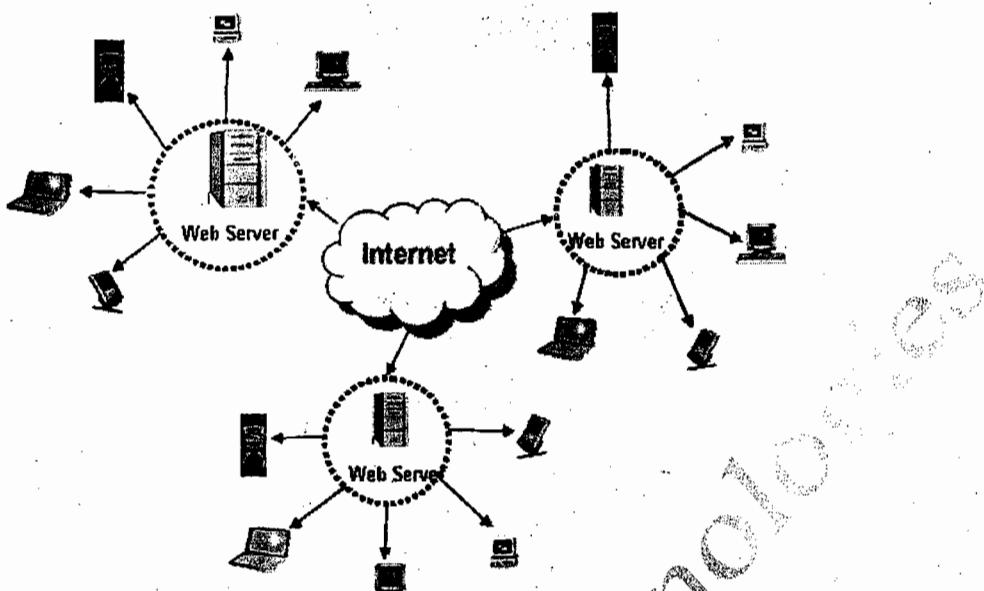


Fig 1.4

- In this method servers are interconnected with in the Internet / Network.
- Current scenario of distributed computing speaks that once data is available with in the server(s), it should be able to be accessed and processed from any kind of client device like computer, Laptop, Mobile Phone, ATM, PDA etc.

1.4 Why .NET

- .NET is very much powerful tool used to build Distributed Computing Applications.

1.5 What Is .Net:

We have various kinds of softwares in IT field like

- 1). Operating system → windows, UNIX, Linux etc
- 2). Applications / packages → MS Office, Tally, Wings, Star Office etc
- 3). Data bases → MS Access, Oracle, Sql Server, DB2, MYSQL etc
- 4). Server Technologies → SMS Server, Share Point Portal Server, Mail Server, BizTalk Server, Microsoft Content Management Server etc.
- 5). ERP Application → SAP, CRM, Seibel, Oracle financials, MSDynamics etc.
- 6). Testing Tools → Win Runner, Win Loader, QTP, etc.
- 7). Programming Languages → FORTAN, COBOL, PASCAL, C, C++, JAVA etc.

Now speaking about .NET

- .NET is not an Operating System
 - .NET is not an Application / Package.
 - .NET is not a Database.
 - .NET is not an ERP Application.
 - .NET is not a Testing Tool
 - .NET is not a Programming Language
-
- .NET is a **Framework tool** which supports many programming languages.
 - .NET supports 61 programming languages
 - In 61 programming languages 9 are designed by Microsoft and 52 are designed by Non Microsoft.

1.5.1 List of All Programming Languages

I have collected this list from the below site

<http://www.dotnetpowered.com/languages.aspx>

1. Ada
 - a. A# (Dr. Martin C. Carlisle)
Port of Ada to .NET
2. APL
 - a. Dyalog APL (Dyalog Ltd)
3. AsmL
 - a. Abstract State Machine Language (MS Research)
4. Assembly
 - a. ASM to IL (bjarke)
5. Basic
 - a. Visual Basic.NET (Microsoft)
 - b. mbas (Novell (Mono Project))
 - c. bmcs (Jambunathan)
6. Basic Variants
 - a. KPL (Morrison Schwartz)
Kid's Programming Language
7. BETA
 - a. BETA.Net (University of Aarhus, Denmark)
8. BF
 - a. BF.NET (Simon Howard)
9. Boo (Python inspired syntax)
 - a. boo (Rodrigo B. de Oliveira & Georges Benatti)

10. C

- a. lcc (Princeton)
ANSI C
- b. csc (Portable.NET)
ANSI C

11. C#

- a. C# (Microsoft)
- b. mcs (Mono/Novell)
- c. csc (DotGNU Portable.NET)

12. C# Variants

- a. CSI (Steve Donovan)
A Simple C# Interpreter
- b. CI% (MS Research)
Comega
- c. eXtensible C# (ResolveCorp)
Language Extension
- d. Spec# (MS Research)
- e. Polymorphic C# (MS Research)
Language has merged with Comega
- f. MC# (Yury P. Serdyuk, Vadim B. Guzey)
Multiprocessor C#
- g. Metaphor (Gregory Neverov @ Queensland University of Technology)
- h. paxScript.NET (VIRT Laboratory)
- i. Parallel C# (Parallel Language Research Project)
- j. Knowledge.NET (Knowledge .NET Project Team) ■■■

13. C++

- a. Managed Extensions for C++ (Microsoft)

14. Caml

- a. F# (MS Research)
(ML and Caml), Abstract IL, ILX
- b. OCAMIL (Emmanuel Chailloux & Raphael Montelatici)

15. CAT

- a. CAT (Christopher Diggins)

16. CFML

- a. BlueDragon (New Atlanta Communications, LLC.)

17. Clarion#

- a. Clarion# (SoftVelocity) ■■■

18. Cobol

- a. NetCOBOL (Fujitsu)
COBOL for .NET
- b. NeoKicks (Fujitsu)
- c. Net Express (Micro Focus)
- d. KICKS for .NET (Intensity Software)

19. Cobra

- a. The Cobra Programming Language (Cobra Language LLC)

20. CULE

- a. CULE.Net (Software Perspectives)

21. E#

- a. E# (Justin Chase)

22. Eiffel

- a. Eiffel ENViSiON! (Eiffel Software)

23. Flash

- a. csswf (Robin Debrouil)

24. Forth

- a. Delta Forth .NET (Valer BOCAN)
- b. MrLoose.Forth (MrLoose) new

25. Fortran

- a. Lahey/Fujitsu Fortran for .NET (Lahey Computer Systems, Inc.)
- b. FTN95 (Salford Software Ltd.)
Fortran for Microsoft .NET

26. G#

- a. G# (Ernest Booth)

27. Haskell

- a. Hugs98 for .NET (Unspecified)
- b. Haskell for .NET (Nigel Perry)
using Mondrian for .NET
- c. Haskell.net Project (Unspecified)

28. IL/MSIL

- a. MSIL (Microsoft)
- b. ilasm (Microsoft)
IL Assembler
- c. Mono Assembler (Novell (Mono Project))
- d. Portable.NET Assembler (dotGNU)

29. Java

- a. Visual J# .NET (Microsoft)

b. IKVM.NET (Jeroen Frijters)

Java VM for .NET

30. JavaScript

a. JScript .NET (GotDotNet)

b. JANET (Unspecified)

JavaScript-compatible language

c. DotGnu JScript (dotGNU)

31. Lexico

a. Lexico (Unspecified)

Spanish

32. LISP

a. clisp (Microsoft)

b. DotLisp (Rich Hickey)

c. L# (L Sharp .NET) (Rob Blackwell)

LISP-based script language

d. FOIL (Rich Hickey and Eric Thorsen)

e. RDNZL (Edi Weitz)

.NET Layer for Common Lisp

33. LISP-like

a. MBase (Meta Alternative)

b. IronLisp (CodePlex Community)

34. LOGO

a. MonoLOGO (Richard Hestilow)

b. TurtleTracks.net Logo (University of Patras)

35. Lua

a. Lua.NET (PUC-RIO)

Integrating Lua with Rotor

36. Mercury

a. Mercury on .NET (Unspecified)

37. Mixal Assembly Language

a. MixNet (SourceForge)

38. Modrian

a. Mondrian for .NET (Nigel Perry)

39. Modula-2

a. GPM/CLR (Queensland University of Technology)

40. Nemerle

a. Nemerle (The University of Wroclaw)

41. Oberon

- a. Active Oberon for .NET (ETH Zuerich)

42. Pan

- a. Pan# (Computer Languages for Secondary Education)

43. Pascal

- a. TMT .NET (TMT)

Pascal Compiler

44. Pascal Variants

- a. Chrome (RemObjects)
- b. Delphi (CodeGear/Borland)
- c. Delphi.NET (Marcus Schmidt)
Interoperability tools
- d. Component Pascal (QUT)

45. Perl

- a. Perl for .NET, PerlNET (ActiveState SRL.)
- b. PerlSharp (Joshua Tauberer)

46. PHP

- a. PHP4Mono (Raphael Romeikat)
- b. PHP4Apps (Daaron)
- c. PHP Sharp (Unspecified)
- d. PHP Mono Extensions (Sterling Hughes)
- e. Phalanger (CodePlex Community)
- f. IronPHP (Ross Girshick)

47. Processing

- a. Processing.NET (Jonatan Rubio, etc. @ SourceForge)

48. Prolog

- a. P# (Jon Cook at Univ. of Edinburgh)
- b. Prolog.NET (Oregon Institute of Technology)

49. Python

- a. IronPython (Microsoft)
- b. KOBRA (Chetan Gadgil)
- c. Python for .NET compiler (Mark Hammond)
- d. Python for .NET (Brian Lloyd)

.NET Integration with Python

50. RPG

- a. Visual RPG for .NET (ASNA)

51. Ruby

- a. IronRuby (Microsoft)

- b. Mono Ruby.NET (Jaen,Mono Developers)
- c. NetRuby (arton)
- d. Rook (Castle Project)
- e. Ruby/.NET Bridge (Ben Schroeder, John Pierce)
- f. RubyCLR (John Lam)
- g. Ruby.NET (Queensland University of Technology)

52. Scala

- a. Scala on Microsoft.NET (Martin Odersky, LAMP at EPFL)

53. Scheme

- a. Common Larceny (Northeastern University)
- b. Dot-Scheme (Pedro Pinto)
PLT Scheme Bridge
- c. Bigloo (Inria Sophia-Antipolis)
- d. Hotdog (Northwestern University)
- e. Tachy (Ken Rawlings)
- f. Scheme.NET (Indiana University)
- g. IronScheme (CodePlex Community)

54. Smalltalk

- a. S# (SmallScript LLC)
- b. #Smalltalk (John Brant & Don Roberts)
- c. VMX Smalltalk (Robowiz Corporation)
- d. LSWVST.NET (Lesser Software)

55. SML

- a. SML.NET (MS Research, University of Cambridge)

56. Spry

- a. Spry (Mark Hahn)

57. Synergy

- a. Synergy (Synergex)

58. Tcl/Tk

- a. TickleSharp (jscottb, Novell Forge)
- b. Jacl.Net (Mailframe)

59. Visual Objects

- a. Vulcan.NET (GrafX Software Development)

60. Windows PowerShell (MS)

61. Zon (Pascal, Modula-2 and Oberon family)

- a. Zonnon (Eugene Zueff)

List Of Microsoft Designed Programming Languages:

1. C# .NET
2. VB. NET
3. J# .NET
4. JSCRIPT .NET
5. C++ .NET
6. F# .NET
7. Windows power shell
8. Iron python
9. Iron Ruby

Very soon Microsoft is going to release C Omega for embeded System Programming

Note:

- ASP. NET is not a programming language.
- ASP. NET is a web technology or server side technology.
- ASP .NET code can be written using VB.NET or C#.NET, or J#.NET.

1.6 Code Execution Process:

1.6.1 In Normal Programming Languages like C, C++, COBOL etc.

- In these programming languages, language compiler will convert the source code and generates the native code.

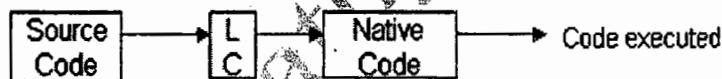


Fig 1.5

- In general at the time of compilation language compiler will consider two factors
 1. CPU / Processor Architecture.
 2. Operating system Architecture.
- Native code will be generated in such a way that it should be understood by the current processor and the Operating System in which the code is being compiled and run.

1.6.2 What is Platform?

- Platform is the combination of CPU Architecture and Operating System Architecture.

1.6.3 What is Platform Dependency?

- Code that has been generated by the language compiler compilation doesn't run / execute on a different Processor and in different Operating System than which it has been compiled, this nature is known as platform dependency.

1.6.4 Platform Independence

- If the code generated by language compiler (LC) compilation runs on any Processor and in any Operating System than which it has been compiled then it is known as platform independent.

1.6.5 Code Execution in Java

- In Java source code is compiled by java language compiler (JLC) and an intermediately code is generated known as byte code.
- This byte code is platform independent, which can be executed on any machine and in any operating system where JVM (Java virtual machine) is available.
- JVM Converts the byte code in to Native code.

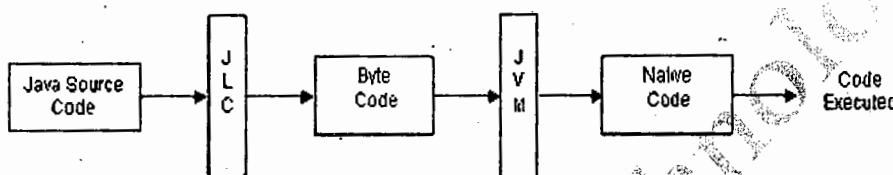


Fig 1.6

JLC :- Java Language Compiler. JVM :- Java Virtual Machine.

- JVM is not a hard ware component rather it is a software component on tool.
- Sun Company designed, separate JVM fro each Operating System, So that Byte code is able to run in any Operating System. (See Fig 1.7)
- If there is no JVM compatible for ay Operating System Byte Code can not be run in that Operating System.

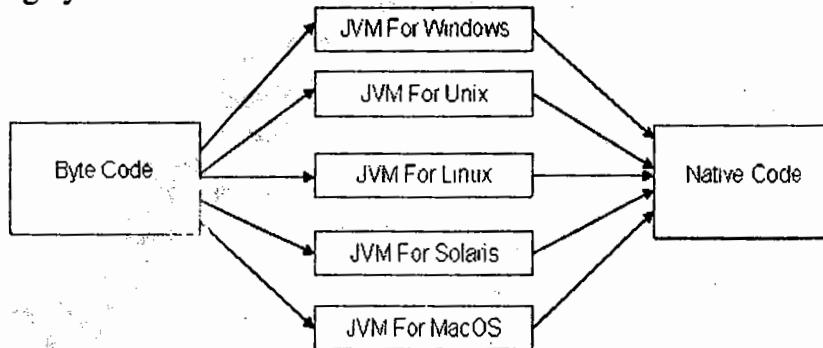


Fig 1.7

1.6.7 Code Execution In .Net

- In .NET code is compiled twice.
- In first compilation source code is compiled by respective language compiler and an intermediately code is generated known as MSIL.
- In second compilation MSIL is converted in to Native code using CLR.

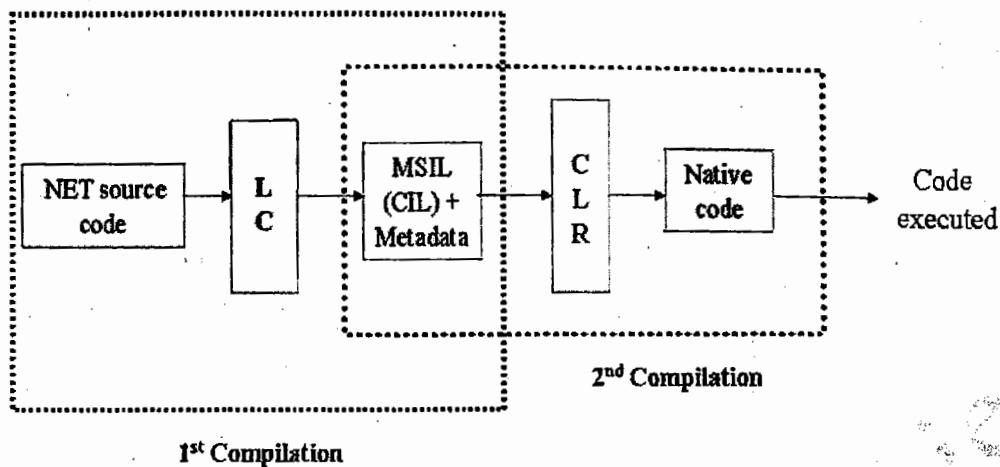


Fig 1.8

LC :	Language Compiler
MSIL :	Microsoft Intermediate language
CIL :	Common Intermediate language
CLR :	Common language Runtime

- Always first compilation is slow and second compilation fast.
- MSIL is only CPU Independent & will run only on windows OS only using .NET framework, because, .NET frame work is designed for windows Operating System only.
- There is another company known as Novel designed separate framework known by MONO framework using this framework we can run MSIL on different Operating Systems like Linux, Solaris, MacOS, BSD, OSX etc. (See Fig 1.9)

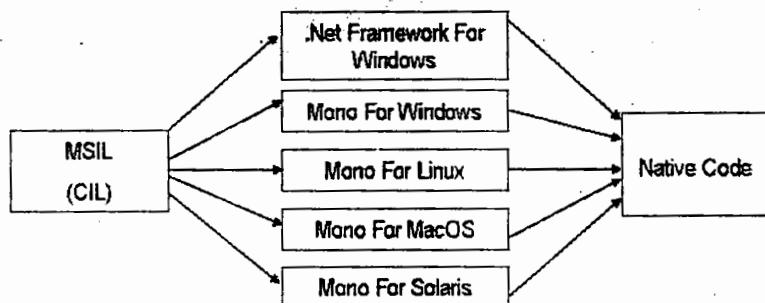


Fig 1.9

- .NET is platform dependent using .NET Frame work but independent using Mono Frame Work.
- Mono Frame Work is designed by the Novell Company. (For More Info on Mono Frame Work please visit (<http://www.mono-project.com>)

- Here in .NET source code means code that has been written in any programming language supported by .NET.
- Every programming language has its own compiler.
- Code might have been written in any programming language and is compiled by any language compiler code will come into a common form i.e. MSIL Like.

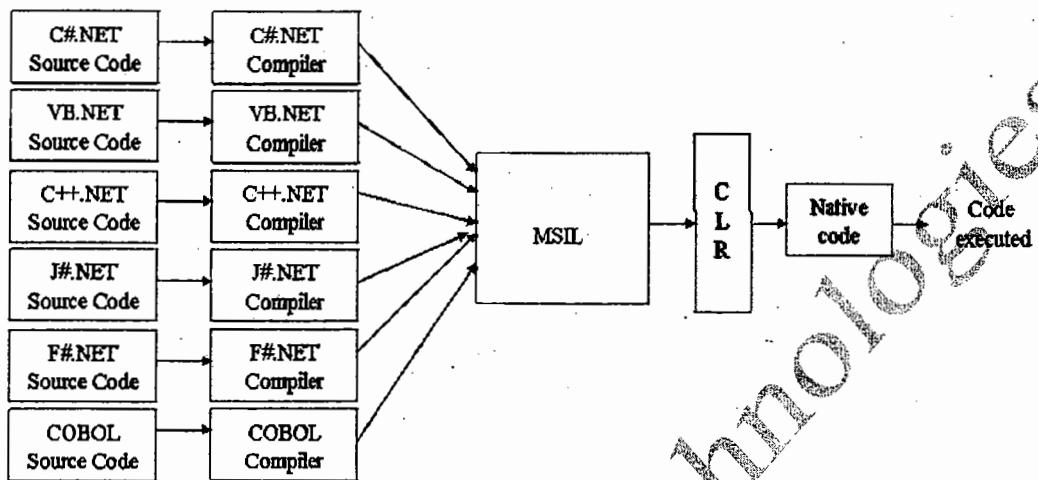


Fig 1.10

Chapter2

.Net Framework

2.1 What is .Net Frame work?

- .Net framework is execution or runtime environment of .net programs or application
- .Net framework is core for complete .NET, with out frame work there is no .Net.
- .Net framework is completely free software and can be downloaded from Microsoft site install and use.

2.2 What is VS.Net?

- VS.Net is just an editor tool used to write .Net code or develop applications using .Net framework.
- VS.Net is most powerful and flexible editor to the programmer.
- VS.Net provides RAD (Rapid Application Development) facilities.
- VS.Net is not free software and is licensed software to use it, it is required to purchase the license from Microsoft.
- Though code is written using VS.Net it will be executed using .Net framework only.

2.3 Difference Between .Net Frame Work And Visual .Net:

Srno	.NET Framework	VS.NET
01	Execution / Runtime Environment of .NET Programs / Applications	Designing / Developing Environment of .NET Programs / Applications
02	Free software	Not Free Software
03	Occupies a memory of ~150MB to ~500MB	Occupies a memory of ~1.5GB to 3.5GB without MSDN
04	Versions:- .NET Framework 1.0 (Released on Jan 15 th 2002, On Jun 22 nd 2000 .NET announced by Bill Gates) .NET Framework 1.1 (Released on April 3 rd 2003) .NET Framework 2.0 (Released on Nov 7 th 2005) .NET Framework 3.0 (Released on Nov 6 th 2006) .NET Framework 3.5 (Released on Nov 19 th 2007) .NET Framework 4.0 Beta Version (Released on May 20 th 2009)	Versions:- VS.NET 2000 (Released on Jan 15 th 2002) VS.NET 2003 (Released on April 3 rd 2003) VS.NET 2005 (Released on Nov 7 th 2005) -----NA----- VS.NET 2008 (Released on Nov 19 th 2007) VS.NET 2010 Beta Version (Released on May 20 th 2009)
05	Compatibility of OS: Win98 (SE) Win XP Professional with SP2 Win 2000 Professional / Server Win 2003 Server Win Vista with SP1 Windows 7	Compatibility of OS: Win XP Professional with SP2 Win 2000 Server Win 2003 Server Win Vista with SP1 Windows 7

2.4 Components Of .Net Frame Work

- .Net framework contains the following components
 1. CLR (Common Language Runtime)
 - i. CLS (Common Language Specification)
 - ii. CTS (Common Type System)
 - iii. GC (Garbage Collector)
 - iv. JIT (Just in Time) Compiler
 2. BCL (Base class libraries) Or FCL (Frame Work Class Libraries)

2.4.1 CLS:

What is the Role of CLS?

- CLS is responsible to provide language interoperability
 - This is achieved in 2 ways 1) Managed Code 2) Unmanaged Code

Language Interoperability:

- Providing code execution support that has been written in other programming language is known as language interoperability.
 - Language Interoperability is achieved using Managed and UN Manager.

Managed Code:

- Code for which MSIL is generated after Language Compiler Compilation is directly executed by the CLR, known as managed code.
 - The cod execution process is known as managed code execution.
 - CLR will provide all the facilities and features of .Net to the Managed Code Execution like Language Interoperability, Automatic Memory Management, Common Data Type System, Exception Handling Mechanism, Code Access Security etc.

Unmanaged Code:

- Code that has written before development of .Net for which MSIL is not available is not executed by the CLR directly, rather CLR redirects the code to OS for Execution, which is known as unmanaged code.
 - The cod execution process is known as Unmanaged code execution.
 - CLR does not provide any facilities and feautes of .Net to the Unmanaged Code Execution like Language Interoperability, Automatic Memory Management, Common Data Type System, Exception Handling Mechanism, Code Access Security etc.
 - Examples for Un Managed Code are COM Componets, Win32APIs etc

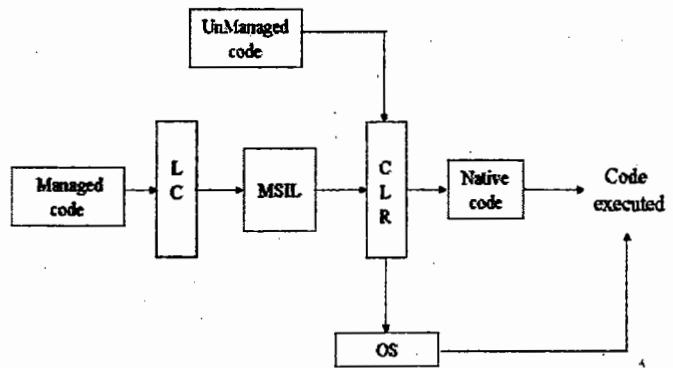


Fig 2.1

- Always managed code execution is faster and unmanaged code execution is slow.

What is CLS?

- **Language Specification:** Every programming Language will have some Syntactical rules used to write the code which is known as **Language Specification**.
- As we know that in Managed code execution process .Net supports many programming languages.
- Every programming Language has its own **Language Specification**.
- One programming Language can not understand the other Programming Languages **Language Specification**.
- But CLR is able to execute all programming Languages code, this because
 - CLR can not understand any Programming Language Language Specification rather CLR has its own Language Specification (Syntactical Rules) for its MSIL.
 - Any Language Compiler should follow this Language Specification of CLR at the time of compilation and should generate MSIL, CLR's JIT Compiler will generate native code from MSIL and CLR will execute it
- This Language Specification of CLR is common to all Programming Languages of Managed code execution of .NET and is known as **CLS (Common Language Specification)**.

2.4.2 CTS:

- As we know that in Managed code execution process .Net supports many programming languages.
- Every programming language has its own data types.
- One programming language cannot understand other programming languages data types.
- But CLR will execute all programming language's data types, this is possible because CLR will contain its own data types, which is common to all the programming languages.
- At the time of compilation all language specific data types are converted in to CLR's Data Type.
- This Data Type System of CLR which is common to all programming languages of .net is known as **Common Type System**.
- This common type system is divided in two categories;
 1. Value types
 2. Reference types

Value Types

- The data types which store the data directly into their memory locations is known as value types

Reference Types

- The data types which do not store the data directly into their memory locations rather refers to other memory locations is known as reference types

Difference Between Value Types And Reference Types

Srno	VALUE TYPES	REFERENCE TYPES
01	Store the value directly into their Memory Locations	Do not Store the value directly into their Memory Locations, rather refers to other Memory Location where value is stored
02	Memory is allotted at Compile Time	Memory is allotted at Run Time
03	Memory allocation is Made with in the Stack i.e. in contiguous Memory Locations	Memory allocation is Made with in the Heap i.e. in Random Memory Locations
04	CLR does not provide Automatic Memory Management	CLR provides Automatic Memory Management

Boxing

- Boxing is the process of converting a variable from value type to reference type.

Unboxing

- UnBoxing is the process of converting a variable from reference type to value type

2.4.3 Garbage Collector

- Garbage collector is responsible to provide automatic memory management.

Why Automatic Memory Management

By using automatic memory management:

- There do not be any problem of insufficient memory
- Burden on the programmer is reduced i.e. programmer need not write code to perform memory manage tasks.

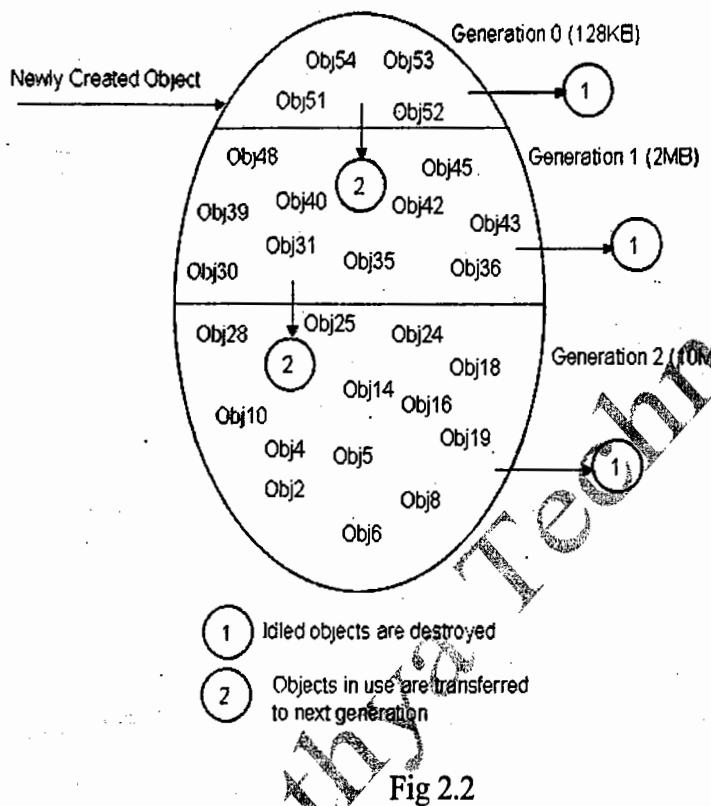
What is Automatic Memory Management

- Garbage collector has its own engine known as optimizing engine, which runs when required and divides objects into two categories.
 1. Objects in use
 2. Idled objects (known as Garbage)
- Objects in use are kept in the memory and idled objects are destroyed from the memory.
- This process is known as Automatic Memory Management.

Working Nature of Garbage Collector

- Garbage collector will divide complete managed heap into three generations internally named as
 1. Generation 0
 2. Generation 1
 3. Generation 2
- **Generation:** - A generation is a portion of memory from the managed heap.
- Maximum number of generations allowed are only three
- In general always generation 2 memory size is bigger than the generation 1 memory size and generation 1 memory size is bigger than the generation 0 memory size.
- This can be represented by mathematically like...
Gen2 > Gen1 > Gen0
(or)
Gen0 < Gen1 < Gen2
- Generally memory to the generations is allotted by the CLR based on application size.
- Garbage collector has a thumb rule that it should place the newly created object only in generation 0.
- GC has no right to place the newly created object into Gen1 or Gen2.
- When application execution starts & application is trying to create a new object then.
 - GC will place this object into Gen0.
 - Like this newly created object will be placed in Gen0 only until the completion of Gen0 memory.
- When Gen0 is completely filled and application is trying to create a new object then.
 - GC will perform an operation known as collection means.
 - Examines all the object present in Gen0, identifies idled objects and objects in use.
 - Idled objects are destroyed or placed in finalization queue.
 - Objects in use are transferred to next generation so that Gen0 will become empty.
 - Newly created object is placed in Gen0.
 - Like this, after performing so many collections of Gen0, Gen1 will be completely filled.
- When Gen 0 and 1 are completely filled and application is trying to create a new object then
 - GC will perform an operation known as collection means.
 - Examines all the object present in Gen0 and 1, identifies idled objects and objects in use.
 - Idled objects are destroyed or placed in finalization queue.
 - Objects in use are transferred to next generation, so that Gen0 and 1 will become empty.
 - Newly created object will be placed in Gen0.
 - Like this, after performing so many collections of Gen0 and Gen1, Gen2 will be completely filled.
- When Gen 0, 1, 2 are completely filled and application is trying to create a new object then

- GC will perform collection of 3 Gens 0, 1, 2 i.e
- Examines all the objects in 3 Gens.
- Divides the objects into idled objects and objects in use
- Idled objects are destroyed or placed in Finalization queue
- Objects in use are adjusted in Gen2 & 3 also in 1 if necessary
- Newly created object is placed in Gen0.



- To perform the collection of Generation 0 GC will take $1/10^{\text{th}}$ of a nano second time which is less than a page cycle.
- Microsoft discovered that searching for an idled object within a small portion of memory is much faster as compared with searching for an object. Within the complete managed Heap, because of this only managed Heap is divided into 3 generations.
- Whereas in JAVA all objects are placed in single pool of memory and there are no generations.
- Complete GC is programmable.

Automatic memory management in .NET is efficient because of maintaining three generations, searching for an idle object within the small portion of memory will be much faster than searching within the complete managed heap and is the most efficient approach.

2.4.5 JIT (Just In Time) Compiler

- This is responsible to compile MSIL code and to generate native code whereas CLR will execute the native code by providing all the facilities like Language Interoperability, Automatic Memory Management, Common Data Type System, Exception Handling Mechanism, Code Access Security etc

.Net Supports Three Types Of Jitters

1. Normal Jitter
2. Echo Jitter
3. Pre Jitter

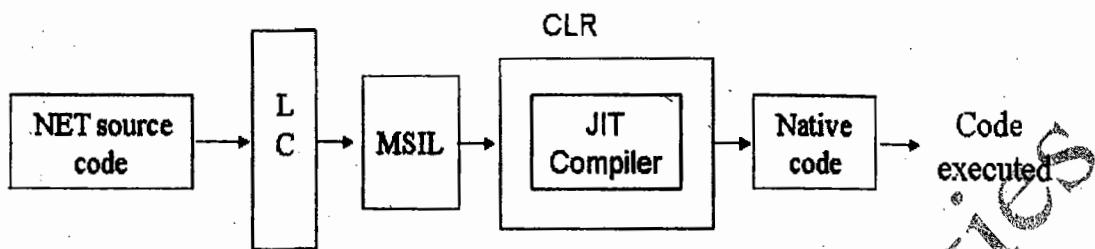
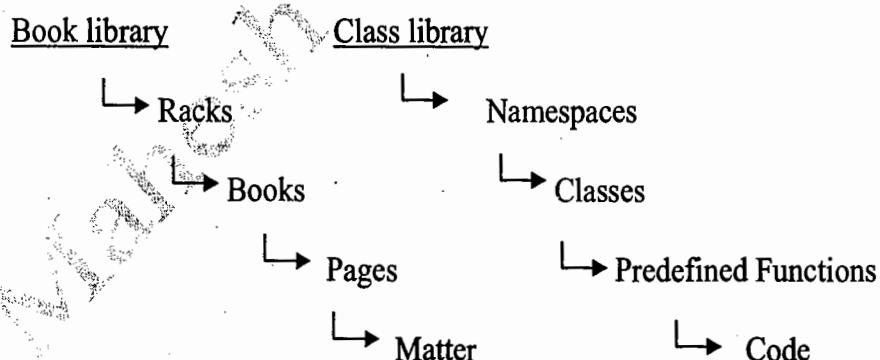


Fig 2.3

2.4.6 Base Class libraries:

- To analyze the Base Class Libraries first we look into the Class Library first.
- In general life we listen like,
- **Book Library:** Means Collection of Books
- **CD Library:** Means Collection of CDs, similarly
- **Class Library:** Means Collection of Classes and Every class contains some predefined functions which are used to write the code in .net
- Class Libraries in .NET are similar to Header Files in C / C++ and Packages in JAVA
- Class Libraries in .Net are also known as Assemblies.
- Class Libraries in .Net are self describing in nature.
- A Class Library in .net is either in the form of DLL (Dynamic Link Library) or in the form of exe (Executable File).

- If we look into Hierarchy of a Class Library:



- This hierarchy is maintained to avoid the confusion and duplicate names of classes and functions while accessing them.
- Like to access any page in any book First we go to particular Book Library in the city, then to a particular Rack in the Library, then to a particular Book in the Rack then to the Required Page in the Book i.e.
 - Book Library → Rack → Book → Page

- Similarly to access any pre defined function First we go to particular Class Library, then to a particular Namespace, then to a particular class and then to a particular function like
 - Class LibraryName .NamespaceName.Class Name.FunctionName
 - Here . is known as Member Access Operator
- In .Net there can be 2 or more classes with same name but they will be stored or present in different namespaces.
- Also there can be 2 or more functions with same name but they will be stored or present in different classes.
- **A Class library can be of two categories**
 1. User defined Class Libraries (UCL)
 2. Base Class Libraries (BCL)

User Defined Class Libraries

- These are created by the programmer for reusability purpose with in the programs or Applications created in .net.

Base Class Libraries

- Base Class Libraries are designed by Microsoft.
- With out these we cannot write any code with in the .net, so Base Class Libraries are known as building blocks of applications or programs of .net.
- These are installed in to the machine when we install .Net framework in to the machine.
- Physical location of Base Class Libraries is
 - Root drive: \ OS folder \ assembly Folder. i.e
 - C: \ windows \ assembly
- There are almost 500 Base Class libraries, 12,909 classes and 4, 01,759 predefined functions are present in .net frame work 2.0 version.
- We can expect more than 5, 00,000 pre defined functions in .net frame work 4.0 version.
- As we are looking that any assembly / Class Library in .Net will exist either in the form of Dll or Exe.

DLL

- A Dll is a Dynamic Link Library can not run itself, used as a supportive file to other applications.
- The Library Functions are Linked to the Application at Runtime (Dynamically) so the name is Dll.
- A Dll does not contain an entry point (main function) so can not run individually.

Exe

- An Exe is executable file and is not a supportive file rather itself an application.
- An Exe will contain an entry point (main function) so runs individually.

Differences between DLL and Exe

Srno	DLL	Exe
01	Can not run Individually	Runs Individually
02	Used as a supportive file for other Applications	Itself an application
03	Does not contain an Entry Point (No Main Function) So can not run individually	Contains an Entry Point (Main Function) So can run individually
04	A Program / Application With out Main Creates a DLL after compilation	A Program / Application With Main Creates an Exe after compilation
05	OS does not create a separate process for any DLL rather DLL will run in the same process created for an exe	OS creates a separate process for each Exe it executes

Note: Creating a process is an overhead to the Operating system so DLL does not contain main function to reduce the burden on the OS

Contents of assembly or class library:

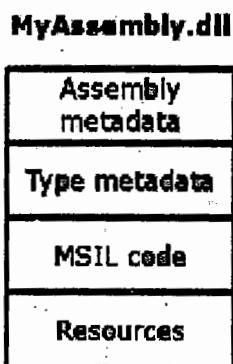


Fig 2.3

Meta Data

- Metadata is binary information describing your program that is stored either in a CLR portable executable (PE) file or in memory.
- When a compiler produces MSIL, it also produces metadata.

Metadata describes:

- The types in your code, including the definition of each type, the signatures of each type's members, the members that your code references, and other data that the runtime uses at execution time.

Metadata stores the following information:

- Description of the assembly.
 - Identity (name, version, culture, public key).
 - The types that are exported.
 - Other assemblies that this assembly depends on.
 - Security permissions needed to run.

- Description of types.
 - Name, visibility, base class, and interfaces implemented.
 - Members (methods, fields, properties, events, nested types).
- Attributes.
 - Additional descriptive elements that modify types and members.

Benefits of Metadata

Metadata is the key to a simpler programming model, eliminating the need for Interface Definition Language (IDL) files, header files, or any external method of component reference. Metadata allows .NET languages to describe themselves automatically in a language-neutral manner, **unseen by both the developer and the user**. Additionally, metadata is extensible through the use of attributes.

Metadata provides the following major benefits:

- **Self-describing files.**
 - Common language runtime modules and assemblies are self-describing.
 - A module's metadata contains everything needed to interact with another module.
 - Metadata automatically provides the functionality of IDL in COM, allowing you to use one file for both definition and implementation.
 - Runtime modules and assemblies do not even require registration with the operating system. As a result, the descriptions used by the runtime always reflect the actual code in your compiled file, which increases application reliability.

Assembly Manifest Contents

- Assembly name
- Version number
- Culture
- Strong name information
- List of all files in the assembly
- Type reference information

Microsoft intermediate language (MSIL)

A language used as the output of a number of compilers and as the input to a just-in-time (JIT) compiler. The common language runtime includes a JIT compiler for converting MSIL to native code

MSIL includes

- Instructions for loading, storing, initializing, and calling methods on objects.
- Instructions for arithmetic and logical operations.
- Instructions for Control flow.
- Instructions for Direct Memory Access.
- Instructions for Exception handling.
- Instructions for and other operations.

2.4.7 Types of Assemblies

1. Private Assembly
2. Public Assembly
3. Static Assembly
4. Dynamic Assembly
5. Satellite Assembly

Note: According to Microsoft there are only 2 types of assemblies Private and Public

Private Assembly

- If an assembly is copied in to individual application in which we want to use then it is known as private (or) local Assembly.
- The copy of the assembly is local to that application and other application cannot use this copy.

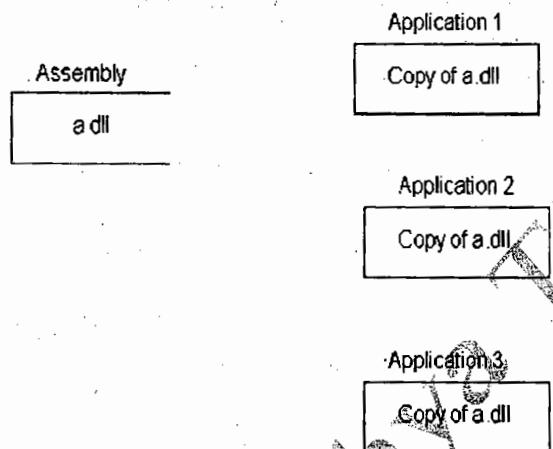


Fig 2.4

Public Assembly

- If an assembly is copied in to a global place and the reference of the assembly is used from all other application then the assembly is known as public or global or shared assembly.

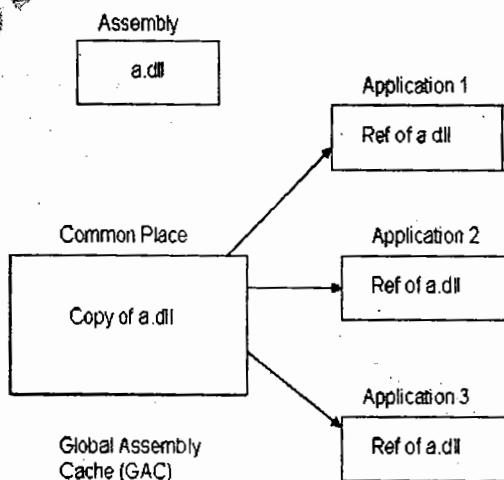


Fig 2.5

- The Global place is known as GAC (Global Assembly Cache).
 - Physical location of GAC is
 - Root drive: \ OS folder \ assembly Folder. i.e
 - C: \ windows \ assembly
 - To make any assembly as public use the following steps
 1. Build the Strong Name for the assembly
 2. Signing of the assembly
 3. Copy the assembly into GAC.
- **Building the Strong Name**
1. A strong Name is the Unique Name used to identify the assembly in GAC.
 2. To build strong name we use a tool known as Sn. Exe.
 3. When we build strong name it will generate public and private key pairs
 4. Public key is similar to ISDN code in telephone directory and which is used to refer the assembly of GAC from other applications.
 5. Following syntax is used to build strong name
Sn.exe -k key file name.snk.
 6. Key File will have a default extension of .snk (means Strong Name Key)

➤ **Signing of Assembly**

1. Copying the Key file path into the assembly is known as signing of the assembly
2. Signing can be done at Design time and also at run time.
3. If signing is made at run time it is known as delay signing.

➤ **Copying the Assembly in to GAC**

1. It is not possible to copy the assembly directly into GAC.
2. We use the following tools to copy the assembly into GAC
 - i. MS Windows Installer 3.1
 - ii. GACUtil.exe.
3. **MS windows installer 3.1:**
 - i. This tool is operating system tool and will work with GUI mode.
4. **GACUtil.exe:**
 - This is .NET framework tool and will work with CUI mode.
 - Following syntax is used to copy assembly into GAC.
 - Gacutil.exe -i assembly name.
 - Ex: Gacutil.exe -i a.dll
 - a.dll is copied into GAC.
 - To delete assembly from the GAC we use
 - Gacutil.exe -u assembly name
 - Ex: Gacutil.exe -u a.dll
 - a.dll is deleted from GAC.

To analyze Static and Dynamic Assemblies first we need to have the idea of PE file.

PE FILE:-

- PE File contains the MSIL and metadata that is based on and extends the published Microsoft PE and common object file format (COFF) used historically for executable content.
- PE stands for Portable Executable file.
- Format of P.E is COFF (Common Object File Format).
- COFF is proprietary of Microsoft ie only Microsoft applications can understand COFF format.
- Tool used to generate PE file is ILASM.exe (Intermediate Language Assembler).
- PE file can be created at Compile Time / at Run Time.

Advantage of common object file format (COFF):

- This file format, which accommodates MSIL or native code as well as metadata, enables the operating system to recognize CLR images. The presence of metadata in the file along with the MSIL enables your code to describe itself (Self Describing Nature), which means that there is no need for type libraries or Interface Definition Language (IDL). The runtime locates and extracts the metadata from the file as needed during execution.
- The MSIL Assembler (Ilasm.exe) generates a portable executable (PE) file from MSIL assembly language

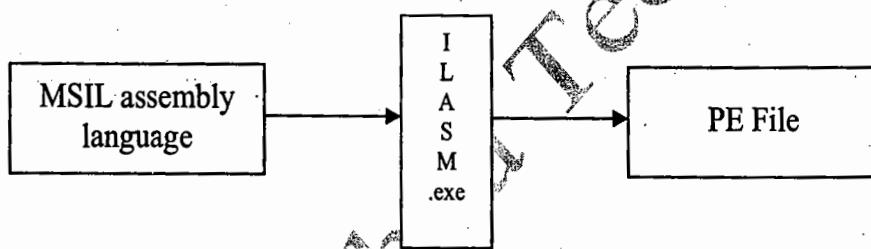


Fig 2.6

- Tool used to convert PE file to text file is ILDASM.exe (Intermediate Language Dis Assembler).
- MSIL Disassembler (Ildasm.exe) is a tool that is included with the .NET Framework SDK.
- The Ildasm.exe parses any .NET Framework .exe or .dll assembly, and shows the information in human-readable format.
- Ildasm.exe shows more than just the Microsoft intermediate language (MSIL) code — it also displays namespaces and types, including their interfaces.

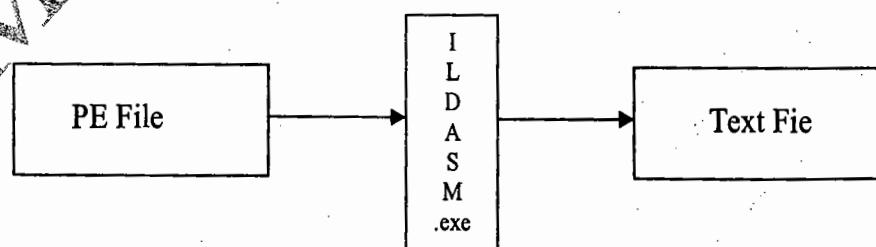


Fig 2.7

Static Assembly

- Static assemblies are stored on disk in portable executable (PE) files before execution.
- Static assemblies can include .NET Framework types (interfaces and classes), as well as resources for the assembly (bitmaps, JPEG files, resource files, and so on).

Dynamic Assembly

- Run directly from memory and are not saved to disk before execution.
- You can also use CLR APIs, such as Reflection, Emit to create dynamic assemblies.

Satellite Assembly

- Satellite assemblies are used to build multilingual applications.
- Provides the Culture information to build an application supportive for multiple languages.
- **Multilingual Application:** Application which has built in supportive of more than one human readable language like English, Hindi, Telugu, Japanese, Chinese etc is known as multilingual application.
- To build multilingual application we need to gather the information known as culture information.
- **Culture information:** - Information that belongs to particular region or local or culture is known as Culture information.
- **Culture information includes like:-**
 - Regional language.
 - Regional Time.
 - Regional Metric System.
 - Regional currency etc.

Check Your Progress or FAQS on .Net Framework

1. What is .NET?
2. Why .NET is used?
3. Explain code Execution process in .NET?
4. How many languages .NET supports?
5. Is .NET platform independent?
6. What is platform?
7. What is platform Dependency?
8. What is platform Independence?
9. How many times code is compiled in .NET?
10. Which code compilation is faster and why?
11. What are the components of .NET framework?
12. What is CLR and explain its role?
13. What is CLS and explain its role?
14. What is Language Interoperability?
15. What is Managed Code and Managed Code execution?
16. What is Un Managed Code and Un Managed Code execution?
17. Which code execution is faster Managed / Unmanaged?

- as
- ne
as
ire
is
18. What is CTS and explain its role?
19. What are the categories of data types in CTS?
20. What are Value Types explain?
21. What are Reference Types explain?
22. Differentiate between Value Types and Reference Types?
23. What is Boxing explain?
24. What is UnBoxing explain?
25. Differentiate between Boxing and Unboxing?
26. What is the role of GC explain?
27. What is Automatic Memory Management?
28. Explain the working nature of Garbage Collector?
29. What is a Generation?
30. How many Generations are maintained?
31. How Generations sizes are allotted?
32. Explain why automatic memory management in .NET is efficient?
33. How much time does GC will take for the collection of Generation 0?
34. What is the role of JIT compiler, explain?
35. List out the types of Jitters available?
36. What is a Class Library / Assembly explain?
37. In which form a Class Library / Assembly in .NET exists?
38. What is the difference between a DLL and an Exe?
39. Why Dll can not execute itself and Exe will execute itself?
40. Why Dll does not contain Main function?
41. Why assemblies in .NET are called self describing in nature?
42. What are the contents of an Assembly?
43. What is Manifest explain?
44. What does Manifest contain?
45. What is MSIL explain?
46. What does MSIL contain?
47. What is Type Metadata explain?
48. What are user defined class Libraries?
49. What are Base Class Libraries / Frame Work Class Libraries?
50. What is the physical location of BCL?
51. List out Types of Assemblies?
52. What is a Private Assembly explain?
53. What is a Public Assembly explain?
54. What is GAC explain?
55. What is the physical location of GAC?
56. What is strong name explain?
57. What is the purpose of SN.exe tool / what tool is used to build strong name?
58. What is a public key?
59. What is the purpose of Gacutil.exe / how do you copy an assembly into GAC?
60. List out tools used to copy assembly into GAC?
61. What is PE file?
62. What is the format of PE file?
63. What is COFF?
64. What is static assembly explain?
65. What is dynamic assembly explain?
66. What tool is used to generate PE file / explain the purpose of ILASM.exe tool?

67. What tool is used to convert PE file to text file / explain the purpose of ILDASM.exe tool?
68. What is Satellite Assembly explain?
69. What is culture information?
70. What does culture information contain?
71. What is DLL hell explain?
72. What is CCW explain?
73. What is RCW explain?

M Venkatesh Sathyamurthy Technologies

Chapter 3

C# .NET

3.1 Introduction

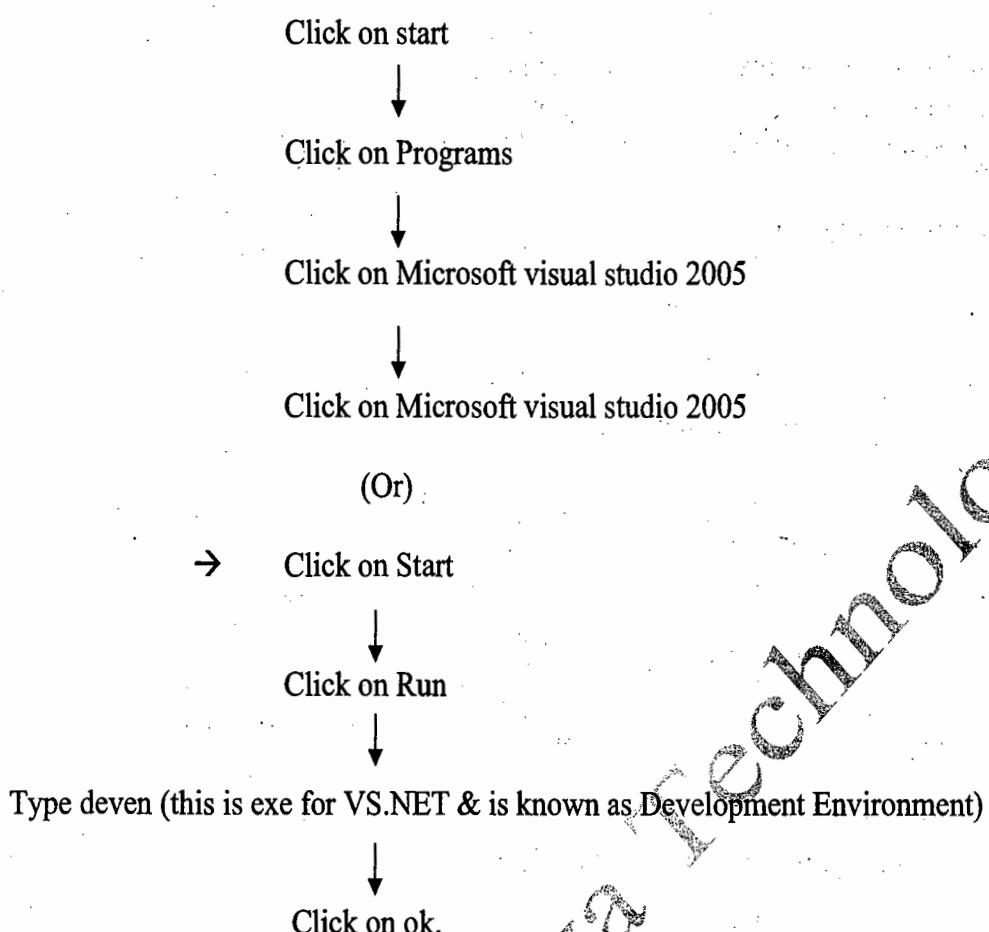
- C#.Net is the most powerful programming language among all programming language in .NET, because C#.NET will contain all the features of C++, VB 6.0 and Java and additional features.
- In C#.NET, the symbol # must and should be pronounced by “sharp” only because Microsoft has taken the symbol from musical note, whose name is “sharp”.
- Among all musical notes, “sharp note” is most powerful note.
- So Microsoft expected that C#.NET will become most power full Programming language among all programming languages.
- We can say that C#.Net = C++ + VB6.0 + Java + Additional Features

3.2 Differences among C++, VB6.0, Java and C#.Net

Sr No	C++	VB 6.0	JAVA	C#.NET
01	Object Oriented but not pure	Object Based Programming Language	Pure Object Oriented Programming Language	Pure Object Oriented Programming Language
02	Supports Operator Over Loading	Does not Support Operator Over Loading	Does not Support Operator Over Loading	Supports Operator Over Loading
03	Supports Multiple Inheritance	Does not Support Inheritance	Supports Multiple Inheritance using Interfaces	Supports Multiple Inheritance using Interfaces
04	Not Platform Independent	Not Platform Independent	Platform Independent	Platform Independent using Mono
05	Does not contain good editor to provide GUI facilities	Contains very good editor to provide GUI facilities, Visual Studio	Contains good editors to provide GUI facilities like SWING, Eclipse and NetBeans	contain very good editor to provide GUI facilities, Visual Studio.Net
06	Code Reusability is achieved using Header Files	Code Reusability is achieved using COM and DCOM	Code Reusability is achieved using Packages	Code Reusability is achieved using Assemblies
07	Does not provide Automatic Memory Management	Does not provide Automatic Memory Management	Provides Automatic Memory Management	Provides Efficient Automatic Memory Management
08	Does not provide Language Interoperability	Provides Limited Language Interoperability	Does not provide Language Interoperability	Provides Enhanced Language Interoperability
09	Supports Pointers	Does not support pointers	Does not support pointers	Support pointers using Unsafe Code
10	Strictly Typed / Type safe programming language	Strictly Typed / Type safe programming language	Strictly Typed / Type safe programming language	Strictly Typed / Type safe programming language

11	Provides structured Exception Handling	Provides structured Exception Handling	Provides structured Exception Handling	Provides structured Exception Handling
12	Supports General Data types using Templates	Does not support General Data types	Supports General Data types using Generics (Included in JSE / JEE 5.0 version)	Supports General Data types using Generics (Included in Framework 2.0 version)
13	Case Sensitive PL	Not Case Sensitive PL	Case Sensitive PL	Case Sensitive PL
14	Supports Enumerations	NA	Supports Enumerations	Supports Enumerations
15	Supports Structures	Supports Structures	Does not supports Structures	Supports Structures
16	Supports Function Pointers	Does not supports Function Pointers / Delegates	Supports Delegates	Supports Delegates
17	Does not Support XML	Does not Support XML	Supports XML	Supports XML
18	Can not be Used to Build Web Applications	Can not be Used to Build Web Applications	Can be Used to Build Web Applications	Can be Used to Build Web Applications
19	Can not be Used to Build Multi Lingual Applications	Can not be Used to Build Multi Lingual Applications	Can be Used to Build Multi Lingual Applications	Can be Used to Build Multi Lingual Applications
20	Can be Used to Build Device Applications	Can not be Used to Build Device Applications	Can be Used to Build Device Applications	Can be Used to Build Device Applications
21	Can be Used for Mobile Programming	Can not be Used for Mobile Programming	Can be Used for Mobile Programming (JME)	Can be Used for Mobile Programming (.NET Framework CE)
22	Supports 3 Access Modifiers private, protected and Public	Supports 2 Access Modifiers private and Public	Supports 4 Access Modifiers private, protected, Public and default	Supports 5 Access Modifiers private, protected, internal, protectedinternal and Public
23	NA	Supports Distributed Programming BUT not efficient	Supports Distributed Programming using RMI	Supports Distributed Programming using Remoting
24	NA	NA	Does not support Explicit Interface Implementation	Supports Explicit Interface Implementation
25	NA	NA	Does not Support LINQ	Supports LINQ
26	NA	NA	Supports security using trusted and un trusted applets	Supports security using Code Access Security

3.3 Entering In To Visual Studio .Net



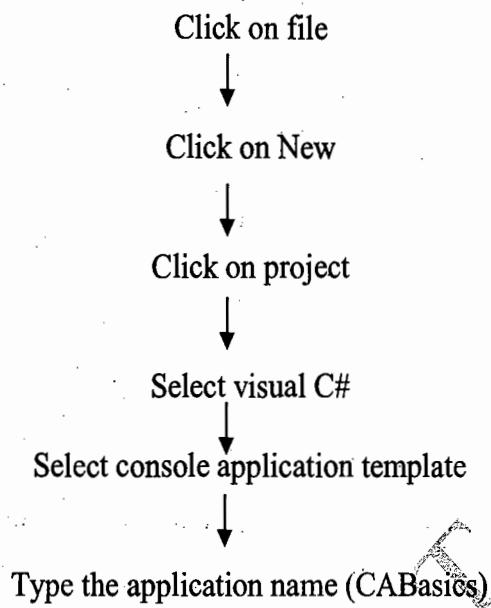
3.4 Types of application that can be created using Visual Studio .Net

1. Console Applications.
2. Class Libraries.
3. Windows Forms Applications.
4. Windows Forms Control Libraries.
5. Windows Services.
6. Crystal Report Applications.
7. Setup and Deployment Applications.
8. WPF Applications.
9. WPF User Control Libraries.
10. WPF Custom Control Libraries.
11. WCF Service Applications.
12. Web Applications.
13. Web Services Applications.
14. WPF Browser Applications
15. AJAX Enabled Web Applications.
16. Mobile Applications.
17. Device Applications.
18. SSIS (SQL Server Integrated services) Applications.
19. Share Point Portal Server Applications.
20. BizTalk Server Applications.

3.5 Console Application

- Console Applications do not provide any GUI facilities.
- Console applications are similar to C, C++ programs.
- In console applications we completely work in CUI based environment.

Steps to create a console application:-



Click on ok.

Note: Any project / application that we create on visual studio .NET is treated as a solution. And solution file will have a default extension of .sln.

3.6 Structure of C# .NET program

List of Class Libraries / namespaces

namespace Namespace name

{
 class class name
 {

 Static void Main (string [] args)

{ Statements

}

}

}

{ → Opening delimiter } → closing delimiter

using

- This keyword is used to include any namespace on the top of the program. (in VB .Net Imports key word is used).

Solution explorer

- Solution explorer is similar to windows explorer in windows. Using solution explorer we can.

- Create files / folders.
- Delete files / folders.
- Rename files / folders.
- Copy file / folders.
- Move files / folders.

With in the same solution.

Renaming the program name or class name:-

Go to Solution explorer



Select the program, cs file name



Click with Right mouse button



Click on rename



Type the new name (Clsexample1.cs)

Example 3.1

Program to Print Welcome on the Screen

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABasics
{
    class ClsExample1
    {
        static void Main(string[] args)
        {
            Console.Write("Welcome");
            Console.Read();
        }
    }
}
```

For compiling the code using C# .Net compiler use the following steps.

Click on Build.



Click on Build CABasics.

(Or) shortcut key is Ctrl + Shift + B

This will generate exe file which contain MSIL code.

For second compilation and to run our program use the following steps

Click on Debug



Click on start Debugging

(Or) shortcut key is – F5.

Output:-



Note:-

C#.Net support all the escape sequences like ‘in’ ‘it’ etc. those are supported by C and C++.

Escape sequence	Character name
\'	Single quote
\"	Double quote
\	Backslash
\0	Null
\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

3.7 Working with Console Class

- Console is a class used to work with Input and Output streams.
- Console class is present in system Namespace.

Methods/Functions with Console class:

- a) Write("Message")
- b) WriteLine("Message")
- c) Read()
- d) ReadLine()
- e) Clear()

Write("Message")

- This method is used to display any Message to the user in the output stream.
- After displaying the message blinking cursor remains in the same line.
- EX: Console. Write("Welcome");
 - o/p: Welcome

WriteLine("Message")

- This method is used to display required message to the user on the output stream.
- After displaying the message blinking cursor moves to a new line.
- Ex: Console .WriteLine("welcome");
 - o/p : welcome
- Note: Console.WriteLine("Message") = Console.Write ("Message\n");

Read()

- This method is used to read a single character from the input stream.

ReadLine()

- This method is used to read a group of character from the input stream.

Clear()

- This method is used to delete the contents of screen and is same as clrscr() in c/c++

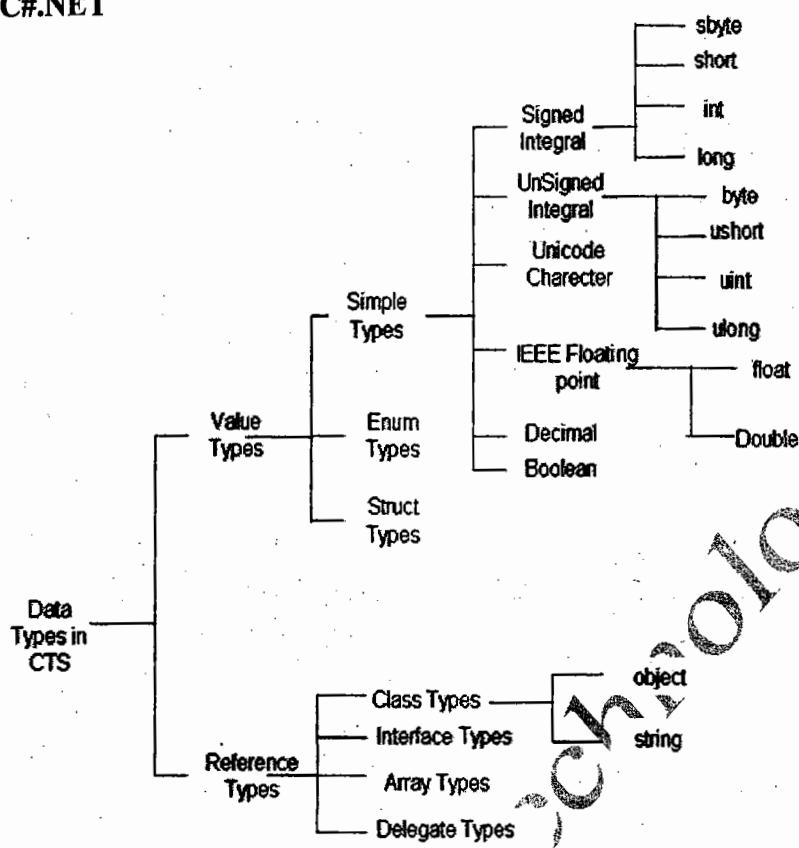
3.8 Working With Variables in C#.NET

How to declare variable:-

Syntax:- Data type variable name[=initializing value];

Ex: int a;
 String s;
 double x;
 int a,b,c;

Data Types in C#.NET



Equivalent Data types chart of CTS, C#.net and VB.net

Category	Class name (CTS Data Type)	Visual Basic data type	C# data type
Integer	Byte	Byte	byte
	SByte	SByte	sbyte
	Int16	Short	short
	Int32	Integer	int
	Int64	Long	long
	UInt16	UShort	ushort
	UInt32	UInteger	uint
	UInt64	ULong	ulong
Floating point	Single	Single	float
	Double	Double	double
Logical	Boolean	Boolean	bool
Other	Char	Char	char
	Decimal	Decimal	decimal
	IntPtr	IntPtr No built-in type.	IntPtr No built-in type.
	UIntPtr	UIntPtr No built-in type	UIntPtr No built-in type
Class objects	Object	Object	object
	String	String	string

Table for Data types their Memory Sizes and Ranges

Type	Range	Size
sbyte	-128 to 127	Signed 8-bit integer
byte	0 to 255	Unsigned 8-bit integer
char	U+0000 to U+FFFF	Unicode 16-bit character
short	-32,768 to 32,767	Signed 16-bit integer
ushort	0 to 65,535	Unsigned 16-bit integer
int	-2,147,483,648 to +2,147,483,647	Signed 32-bit integer
uint	0 to 4,294,967,295	Unsigned 32-bit integer
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer
Floating point	1.5×10^{-45} to 3.4×10^{38} , 7-digit precision	Signed 32-bit integer
double	5.0×10^{-324} to 1.7×10^{308} , 15-digit precision	Signed 64-bit integer
decimal	1.0×10^{-28} to $7.9 \times 10{28}$, 28-digit precision	Signed 128-bit integer

Steps to add a new class file:-

Go to solution Explorer

Select the solution → click with right mouse button

Click on Add

Click on New Item

Select class template

Type the class name (ClsExampe2.cs)

Click on Add

3.9 Key words in C#.NET

- Keywords are predefined reserved identifiers that have special meanings to the compiler.
- Keywords cannot be used as variables in your program unless they include @ as a prefix.
- C#.Net supports 2 types of keywords 1) Reserved keywords 2) Contextual keywords.
- There are 77 Reserved keywords and 6 Contextual keywords present in C#.Net

Reserved Keywords

<u>abstract</u>	<u>event</u>	<u>new</u>	<u>struct</u>
<u>as</u>	<u>explicit</u>	<u>null</u>	<u>switch</u>
<u>base</u>	<u>extern</u>	<u>object</u>	<u>this</u>
<u>bool</u>	<u>false</u>	<u>operator</u>	<u>throw</u>
<u>break</u>	<u>finally</u>	<u>out</u>	<u>true</u>
<u>byte</u>	<u>fixed</u>	<u>override</u>	<u>try</u>
<u>case</u>	<u>float</u>	<u>params</u>	<u>typeof</u>
<u>catch</u>	<u>for</u>	<u>private</u>	<u>uint</u>
<u>char</u>	<u>foreach</u>	<u>protected</u>	<u>ulong</u>
<u>checked</u>	<u>goto</u>	<u>public</u>	<u>unchecked</u>
<u>class</u>	<u>if</u>	<u>readonly</u>	<u>unsafe</u>
<u>const</u>	<u>implicit</u>	<u>ref</u>	<u>ushort</u>
<u>continue</u>	<u>in</u>	<u>return</u>	<u>using</u>
<u>decimal</u>	<u>int</u>	<u>sbyte</u>	<u>virtual</u>
<u>default</u>	<u>interface</u>	<u>sealed</u>	<u>volatile</u>
<u>delegate</u>	<u>internal</u>	<u>short</u>	<u>void</u>
<u>do</u>	<u>is</u>	<u>sizeof</u>	<u>while</u>
<u>double</u>	<u>lock</u>	<u>stackalloc</u>	
<u>else</u>	<u>long</u>	<u>static</u>	
<u>enum</u>	<u>namespace</u>	<u>string</u>	

Contextual Keywords

<u>get</u>	<u>partial</u>	<u>set</u>
<u>value</u>	<u>where</u>	<u>yield</u>

- A contextual keyword is used to provide a specific meaning in the code, but it is not a reserved word in C#.
- **get** Defines an accessor method for a property or an indexer.
- **partial** Defines partial classes, structs, and interfaces throughout the same compilation unit.
- **set** Defines an accessor method for a property or an indexer.
- **where** Adds constraints to a generic declaration.
- **yield** Used in an iterator block to return a value to the enumerator object or to signal the end of iteration.
- **value** Used to set accessors and to add or remove event handlers.

Example 3.2

Program to Print Variable Data on the Screen

- To print variable data on the screen there are 2 methods available.
- If we have `int a=10;`
- We use `Console.WriteLine("Value of a is:- " + a);`
- In the above statement `+` is known as concatenation operator.
- We can also use `Console.WriteLine("Value of a is:- {0}", a);`
- In the above statement `{0}` is known as Output stream argument.

Observe the following Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

namespace CABasics

```
{
    class ClsExample2
    {
        static void Main()
        {
            int a = 10;
            Console.WriteLine("Value of a is:- {0}", a);
            Console.WriteLine("Value of a is:- " + a);
            Console.Read();
        }
    }
}
```

Output:

```
Windows PowerShell - Run as administrator
Value of a is:- 10
Value of a is:- 10
```

Note: When you try to run the above program it raises Compilation Error because an Application can not contain 2 Main() methods.

To avoid this you need to change the start up object

Steps to change startup object:-

Go to solution explore

Double click on properties

Go to start up object

Select the class file name to execute.

Save and close the properties window.

3.10 Reading the Data from the User

To read the data from the user there are 2 methods available.

- 1) ReadLine()
- 2) Read()

Differences between Read and ReadLine:-

Srno	Read	ReadLine
01	Reads Single / Next Character from the Input Stream	Reads Group of Characters from the Input Stream
02	Reads Maximum of only 1 character	Reads Maximum of 255 characters (Limit of OS Command prompt)
03	Reads ASCII value of the Character	Reads string value of the Character(s)
04	Return Type is int / Integer	Return Type is string

➤ Syntax of ReadLine () / Read()

Variable name = Console. ReadLine ();

Variable name = Console. Read ();

➤ Reading the Data from the user using ReadLine () method:-

Example 3.3

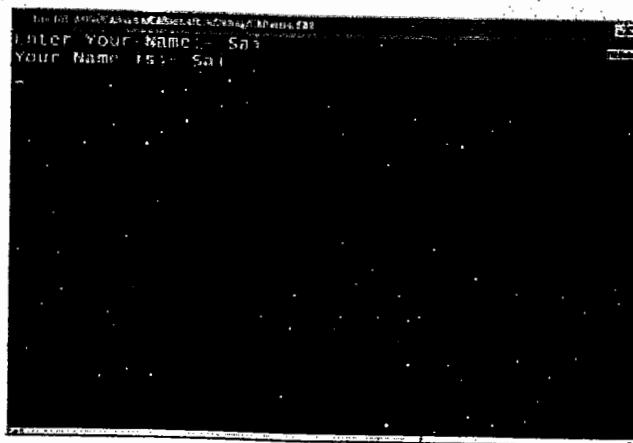
Program to Reading string value:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

namespace CABasics
{
    class ClsExample3
    {
        static void Main()
        {
            Console.Write("Enter Your Name:- ");
            string S = Console.ReadLine();
            Console.WriteLine("Your Name is:- " + S);
            Console.Read();
        }
    }
}

```

Output:-**Example 3.4****Program to Read Integer values:-**

```

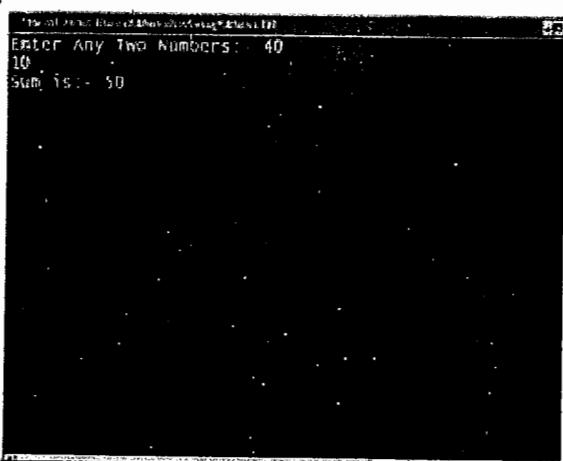
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABasics
{
    class ClsExample4
    {
        static void Main()
        {
            int a, b, c;
            Console.Write("Enter Any Two Numbers:- ");
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            c = a + b;
            Console.WriteLine("Sum is:- " + c);
        }
    }
}

```

```
        Console.Read();
    }
}
}
```

Output:-



Data type conversion

There are two methods available for type Conversions:-

1) Target data type. Parse ("value")

int. parse ("10") → 10

double. Parse ("10.5") → 10.5

2) Convert. to target data type class ("value")

Convert.ToInt32 ("10") → 10

Convert.ToDouble ("10.5") → 10.5

Writing the code In note pad and executing using .net framework

Open the Notepad Application

↓
And type the following code

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace CABasics

{

 class ClsExample5

{

 static void Main()

{

 int a, b, c;

 Console.Write("Enter Any Two Numbers:- ");

 a = Convert.ToInt32(Console.ReadLine());

 b = Convert.ToInt32(Console.ReadLine());

```
c = a + b;  
Console.WriteLine("Sum is:- " + c);  
Console.Read();  
}  
}  
}
```

Save the file with same name as class name and extension with .CS (ClsExample5.cs) in any location. (Say in D:\)

Go to VS.NET command prompt

Click on start

Click on programs

Click on Microsoft Visual Studio 2008

Click on Visual Studio Tools

Click on Visual Studio 2008 Command Prompt

Change the location to the same where we saved our file (D:\)).

Type CSC ClsExample5.cs

This will create exe. ie ; first compilation is made and MSIL code is written in to this exe file .

To Execute the file: Type `ClsExample5.exe`

Programming constructs

Methods used to write the programs are called programming construct.

There are these types' constructs

- 1) Sequential
 - 2) Selection
 - 3) Iteration

1) Sequential:

Code will be executed line by line without missing any statement with in the program.

2) Selection:

- i. Simple If
 - ii. If Else
 - iii. Multiple Ifs
 - iv. Nested Ifs
 - v. Switch – Case.

1) Syntax for Simple If:-

if (condition)

{
 Statements
};

}

2) Syntax for If Else:-

```
if (condition)  
{  
    Statement  
};  
else  
{  
    Statement  
};  
}
```

3) Syntax for Multiple – Ifs

```
if (condition)  
{  
    Statement  
};  
if (condition)  
{  
    Statements  
};  
};  
;
```

4) Syntax for Nested Ifs

```
if (condition)  
{  
    Statements  
};  
}  
else if (condition)  
{  
    Statements  
};  
}  
;  
else  
{  
    Statement  
};  
}
```

5) Syntax of Switch – Case:-

switch ("expression")

```

{
    case expr 1 :
    {
        Statements
        :
        break ;
    }
    case expr 2 :
    {
        Statements
        :
        break ;
    }
    default :
    {
        Statements
        :
    }
}

```

Types of Iterative statements:

- 1). For Loop
- 2). While Loop
- 3). Do While Loop
- 4). For Each Loop

1. Syntax of for Loop:-

```

for (Variable = StartValue; condition; increment / decrement)
{
    Statements
    :
}

```

2. Syntax of While Loop:-

```

While (condition)
{
    Statement s
    :
    [Increment / or decrement]
}

```

For numeric only we take increment / decrement for non numeric there will be no increment / decrement.

3 Syntax of Do while loop:-

```

do
{
    Statements
    :
}

```

[Increment / decrement]

} while (condition);

4 Syntax of for each Loop:-

foreach (datatype variable in array /list collection name)

{

Statements

:

}

Note:- Data type of loop variable must and should be same as the data type of the array / list / collection.

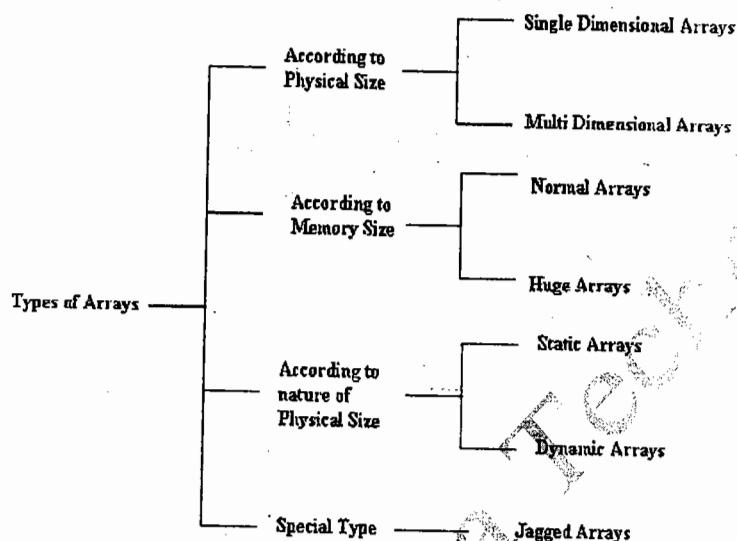
List of operators in C# .NET

Category	Expression	Description
Primary	<code>x.w</code>	Member access
	<code>x(...)</code>	Method and delegate invocation
	<code>x[...]</code>	Array and indexer access
	<code>x++</code>	Post-increment
	<code>x--</code>	Post-decrement
	<code>new T(...)</code>	Object and delegate creation
	<code>new T[...]</code>	Array creation
	<code>Typeof(T)</code>	Obtain <code>System.Type</code> object for T
	<code>checked(x)</code>	Evaluate expression in checked context
	<code>unchecked(x)</code>	Evaluate expression in unchecked context
Unary	<code>+x</code>	Identity
	<code>-x</code>	Negation
	<code>!x</code>	Logical negation
	<code>~x</code>	Bitwise negation
	<code>++x</code>	Pre-increment
	<code>--x</code>	Pre-decrement
	<code>(T)x</code>	Explicitly convert x to type T
Multiplicative	<code>x * y</code>	Multiplication
	<code>/ y</code>	Division
	<code>x % y</code>	Remainder
Additive	<code>x + y</code>	Addition, string concatenation, delegate combination
	<code>x - y</code>	Subtraction, delegate removal
Shift	<code>x << y</code>	Shift left
	<code>x >> y</code>	Shift right
Relational and type testing	<code>x < y</code>	Less than
	<code>x > y</code>	Greater than
	<code>x <= y</code>	Less than or equal
	<code>x >= y</code>	Greater than or equal
	<code>x is T</code>	Return true if x is a T, false otherwise
	<code>x as T</code>	Return x typed as T, or null if x is not a T
Equality	<code>x == y</code>	Equal
	<code>x != y</code>	Not equal
Logical AND	<code>x & y</code>	Integer bitwise AND, boolean logical AND
Logical XOR	<code>x ^ y</code>	Integer bitwise XOR, boolean logical XOR
Logical OR	<code>x y</code>	Integer bitwise OR, boolean logical OR
Conditional AND	<code>x && y</code>	Evaluates y only if x is true
Conditional OR	<code>x y</code>	Evaluates y only if x is false
Conditional	<code>x ? y : z</code>	Evaluates y if x is true, z if x is false
Assignment	<code>x = y</code>	Assignment
	<code>x op= y</code>	Compound assignment, supported operators are “= / = % = += -= <<= >>= &= ^ = =”

3.11 Arrays in C#Net

- An Array is an user defined datatype used to store more than one values with the same name.
- Each value / location is identified using the index.
- Always 1st index of Array starts with zero (0) and is known as Lower Bound of Array.
- Always Last index of Array ends with size-1 of Array and is known as Upper Bound of Array.
- In C#.NET Arrays are Reference Types.

3.11.1 Types of arrays:-



Single dimensional array

- An array which contains one row or one column is known as dimensional array.

Ex: - A

50	70	40	10	30
0	1	2	3	4

A	0	50
	1	70
	2	40
	3	10
	4	30

Multi dimensional array

- An array which contains more than one rows and more than one columns is multi dimensional array.

Normal Array

- An array which occupies a memory size of less than 64 KB is known as normal array.

Huge Array

- An array which occupies a memory size of ≥ 64 KB is known as Huge array.

Static Array

- An array whose physical size is fixed and cannot be changed at runtime is known as static array.

Dynamic Array

- An array whose physical size is not fixed through out the program execution and can be changed at runtime is known as dynamic array.

Note:-

- Array size can be changed during the program execution using Resize () function of array class In both C#.NET and VB.NET
- VB.NET also supports another keyword known as "REDIM" to change size of the Array during Runtime.

Jagged Array

- An array which contains another array within it is known as jagged array,
- A Jagged Array is known as Array of Arrays

Ex:

50	20	40	10	60	30
----	----	----	----	----	----

A

		0	1	2	3
		10	20	30	40
		0	1	2	
0		50	60	70	
1		80	90	10	20
2					

3.11.2 Working with single Dimensional Arrays in C#.NET

A	20	50	10	60	30	40
	0	1	2	3	4	5

Syntax:-

Data Type[] Array Name = new Data Type[size]

Ex: - int[] A = new int[6] ;

A [0] = 20; A [1] = 50; A [2] = 10; A [3] = 60; A [4] = 30; A [5] = 40 ;

(OR)

Syntax:-

Data Type[] Array Name = new data type [size] {initializing element};

Ex: - int[] A = new int [6] {20,50,10,60,40,30} ;

(OR)

Syntax:-

DataType[] Array Name = new Data Type[] {initializing element},

Ex: - int[] A = new int[] { 20,50,10,60,40, 30 } ;

Example 3.5

Program to print Array Elements on the Screen:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray1
    {
        static void Main()
        {
            int[] A = new int[]{50,10,40,60,30,20};
            //A[0] = 50; A[1] = 10; A[2] = 40;
            //A[3] = 60; A[4] = 30; A[5] = 20;
            Console.WriteLine("Elements of Array are:- ");
            for (int i = 0; i < 6; i++)
            {
                Console.Write(A[i] + " ");
            }
            Console.Read();
        }
    }
}
```

Output:-

A screenshot of a Windows Command Prompt window titled "file:///E:/2016/BACKUP/1/4/CABASIC/BASICS/5inBackup/CL0011". The window displays the text "Elements of Array are:-" followed by the elements of an integer array: 50, 10, 40, 60, 30, 20.

Comments:-

C# .Net // Single line comment
 /* Multi Line
 Comment */

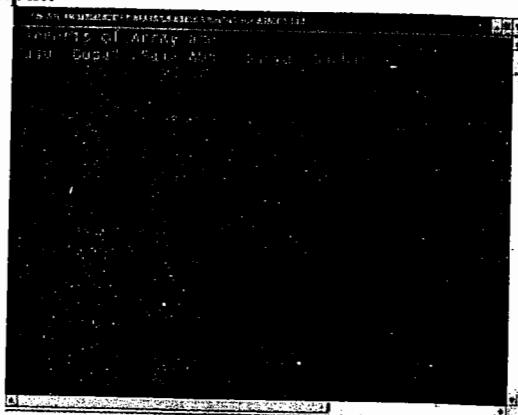
VB .Net ' Single line comment.
 Does not supports multi line Comment.

Example 3.6

Program to print string Array Elements on the Screen:-

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
  
namespace CABASICS  
{  
    class ClsArray2  
    {  
        static void Main()  
        {  
            string[] A = new string[6] { "Raju", "Gopal", "Sai", "Abhi", "Surya", "Sachin" };  
            Console.WriteLine("Elements of Array are:- ");  
            for(int i=0;i<6;i++)  
            {  
                Console.Write(A[i] + " ");  
            }  
            Console.Read();  
        }  
    }  
}
```

Output:-



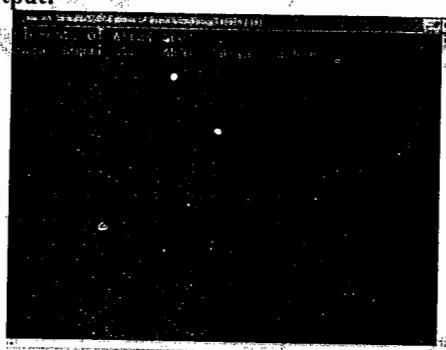
Example 3.7

Program to print Array Elements using For Each Loop:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray3
    {
        static void Main()
        {
            string[] A = new string[6] { "Raju", "Gopal", "Sai", "Abhi", "Surya", "Sachin" };
            Console.WriteLine("Elements of Array are:- ");
            foreach (string S in A)
            {
                Console.Write(S + " ");
            }
            Console.Read();
        }
    }
}
```

Output:-



Working nature of For Each loop

- In for each loop Data Type of loop variable must and should be the same as data type of the Array / Collection that we refer.
- At the time of execution 1st for each loop goes to the array counts number of elements in the Array and repeats the loop that many times automatically.
- At each iteration array location value will be copied into loop variable S.

3.11.3 Array object in C# .NET:-

When we create any array with any name then array is treated as an object.

Methods with Array object:-

1) CopyTo(Destination Array, int Index)

This method is used to copy element of one array to another array.

Ex:-	A	<table border="1"><tr><td>20</td><td>50</td><td>10</td><td>40</td><td>30</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table>	20	50	10	40	30	0	1	2	3	4
20	50	10	40	30								
0	1	2	3	4								

B	<table border="1"><tr><td>25</td><td>15</td><td>35</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr></table>	25	15	35	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11
25	15	35	0	0	0	0	0	0	0	0	0														
0	1	2	3	4	5	6	7	8	9	10	11														

A.CopyTo(B,0) →	<table border="1"><tr><td>20</td><td>50</td><td>10</td><td>40</td><td>30</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr></table>	20	50	10	40	30	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11
20	50	10	40	30	0	0	0	0	0	0	0														
0	1	2	3	4	5	6	7	8	9	10	11														

A.CopyTo(B,3) →	<table border="1"><tr><td>25</td><td>15</td><td>35</td><td>20</td><td>50</td><td>10</td><td>40</td><td>30</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td></tr></table>	25	15	35	20	50	10	40	30	0	0	0	0	0	1	2	3	4	5	6	7	8	9	10	11
25	15	35	20	50	10	40	30	0	0	0	0														
0	1	2	3	4	5	6	7	8	9	10	11														

A.CopyTo(B,8) → Raises Run Time Error

Properties with Array object:-

Length:

This properties returns size of the array

Ex:- A.Length → 5

Rank:-

This properties returns the number of dimensions present with in the array.

Ex:- A.Rank → 1

Example 3.8

Program with Array Object:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray5
    {
        static void Main()
        {
            int[] A = new int[6] { 50, 20, 60, 40, 10, 30 };
            int[] B = new int[12] { 25, 15, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            Console.WriteLine("Size of Array A is:- " + A.Length);
            Console.WriteLine("Size of Array B is:- " + B.Length);
            Console.WriteLine("Rank of Array A is:- " + A.Rank);
            Console.WriteLine("Rank of Array B is:- " + B.Rank);
            Console.WriteLine("Elements of Array A are:- ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            Console.WriteLine("\nElements of Array B are:-");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            A.CopyTo(B, 0);
            Console.WriteLine("\nElements of Array B after Copying are:-");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            Console.Read();
        }
    }
}
```

Output:-

```
Microsoft Windows [Version 10.0.19041]
[Administrator: Command Prompt]
C:\Users\DELL\OneDrive\Desktop\Chalkley> using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray5
    {
        static void Main()
        {
            int[] A = new int[6] { 50, 20, 60, 40, 10, 30 };
            int[] B = new int[12] { 25, 15, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            Console.WriteLine("Size of Array A is:- " + A.Length);
            Console.WriteLine("Size of Array B is:- " + B.Length);
            Console.WriteLine("Rank of Array A is:- " + A.Rank);
            Console.WriteLine("Rank of Array B is:- " + B.Rank);
            Console.WriteLine("Elements of Array A are:- ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            Console.WriteLine("\nElements of Array B are:-");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            A.CopyTo(B, 0);
            Console.WriteLine("\nElements of Array B after Copying are:-");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            Console.Read();
        }
    }
}

C:\Users\DELL\OneDrive\Desktop\Chalkley>
```

3.11.4 Array Class:-

In .Net we have a separate class for array known as Array. This class is present in System namespace and this class will provide methods to perform required operations with the arrays.

Methods with System.Array class:-

1. Copy(Source array, Destination Array, int Length)
(or)
Copy(Source array, int source index, destination array, int destination index, int length)
2. Reverse (Array name)
(or)
Reverse (Anlame int index, int length)
3. Sort (Array name)
(or)
Sort (Array name, int index, int length)
4. BinarySearch (Array name, object value)
(or)
Binary search (Array name, int index, int length, object value)
5. Clear (Array Name, int index, int length)

Copy Method:-

Used To copy the elements from One array to another array

Consider 2 Arrays Like:

A	0	1	2	3	4	5	
	20	40	30	60	10	50	

B	0	1	2	3	4	5	6	7	8	9	10	11
	25	15	35	0	0	0	0	0	0	0	0	0

Array .Copy (A, B, 6) →

20	40	30	60	10	50	0	0	0	0	0	0
----	----	----	----	----	----	---	---	---	---	---	---

It overwrites the existing element in B.

Array .Copy (A, 2, B, 3, 3) →

B	25	15	35	30	60	10	0	0	0	0	0
---	----	----	----	----	----	----	---	---	---	---	---

Reverse Method:- Used to reverse the element of array

Array. Reverse (A) →

50	10	60	30	40	20
----	----	----	----	----	----

To reverse few elements use **Array. Reverse(A, 2, 3)** →

A

20	40	10	60	30	50
----	----	----	----	----	----

Sort Method:- Used to sort the elements of Array in ascending order.

Array.Sort(A) → A

10	20	30	40	50	60
----	----	----	----	----	----

To sort only few elements use **Array.Sort (A, 2, 3)** →

A

20	40	10	30	60	50
----	----	----	----	----	----

To Sort in Descending order:-

For this first sort the array and then reverse the array.

Binary Search ():-

For performing this our array should be in sorted order.

0 1 2 3 4 5
A

10	20	30	40	60	50
----	----	----	----	----	----

And this will return zero or positive integer i.e. index value of the element if elements is found. Otherwise returns negative value, If element is not found.

To find element 40 with in the Array use

Array.BinarySearch(A, 40) → 3

To search only few element of the array first sort the array.

0 1 2 3 4 5
A

10	20	30	40	60	50
----	----	----	----	----	----

Use **Array.BinarySearch (A, 2, 4, 40) → 3**

If we use **Array.BinarySearch (A,35) → -3** (Because 35 lies between 2 & 3 > 30 < 40)

Clear Method:-

This method is used to store default value in to the array locations

	0	1	2	3	4	5
A	20	40	30	60	10	50

Array.Clear(A, 0, 6) →

	0	1	2	3	4	5
A	0	0	0	0	0	0

Example 3.9

Program with Array Class Methods:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray6
    {
        static void Main()
        {
            int[] A = new int[6]{50,10,40,60,20,30};
            int[] B = new int[12] { 15, 25, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            Console.WriteLine("Elements of Array A are:- ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            Console.WriteLine("\nElements of Array B are:- ");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            Array.Copy(A, 0, B, 3, 6);
            Console.WriteLine("\nElements of Array B after Copying:- ");
            foreach (int i in B)
            {
                Console.Write(i + " ");
            }
            Array.Reverse(A);
        }
    }
}
```

```

Console.WriteLine("\nElements of Array A after Reversing:- ");
foreach (int i in A)
{
    Console.Write(i + " ");
}
Array.Sort(A);
Console.WriteLine("\nElements of Array A after Sorting:- ");
foreach (int i in A)
{
    Console.Write(i + " ");
}
Console.Write("\nEnter Search Element:- ");
int S = Convert.ToInt32(Console.ReadLine());
int F = Array.BinarySearch(A, S);
if (F >= 0)
{
    Console.WriteLine("Search Element Found in {0} Location", (F + 1));
}
else
{
    Console.WriteLine("Search Element Not Found");
}
Array.Clear(A, 2, 3);
Console.WriteLine("Elements of Array A after Clearing:- ");
foreach (int i in A)
{
    Console.Write(i + " ");
}
Console.ReadLine();
}
}

```

Output:-

```

C:\Users\DELL\OneDrive\Desktop\Assignment 1\Assignment 1\bin\Debug\Assignment 1.exe
Elements of Array A after Reversing
50 40 30 20 10
Elements of Array B after sorting
10 20 30 40 50
Elements of Array A after Copying
10 20 30 40 50
Elements of Array A after Reversing
50 40 30 20 10
Elements of Array A after sorting
10 20 30 40 50
Enter Search Element:- 40
Search element Found in 4 location
Elements of Array A after Clearing
10 20 30

```

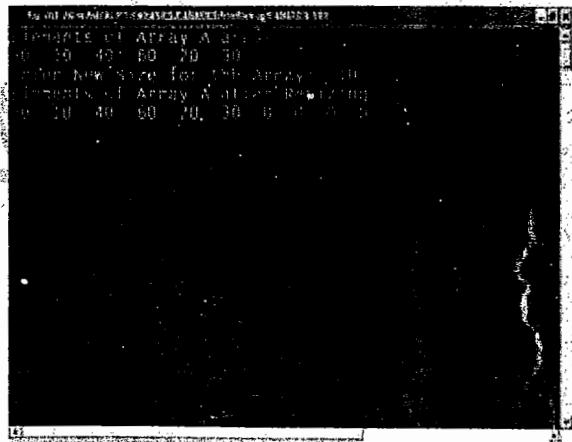
Example 3.10

Program to Change the Size of Array at Run Time:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsArray7
    {
        static void Main()
        {
            int[] A = new int[6] { 50, 10, 40, 60, 20, 30 };
            Console.WriteLine("Elements of Array A are:- ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            Console.Write("\nEnter New Size for the Array:- ");
            int S = Convert.ToInt32(Console.ReadLine());
            Array.Resize(ref A, S);
            Console.WriteLine("Elements of Array A after Resizing:- ");
            foreach (int i in A)
            {
                Console.Write(i + " ");
            }
            Console.Read();
        }
    }
}
```

Output:-



3.11.5 Working with two dimensional array: -

A	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90
3	20	40	60

Syntax: Data Type [,] Array name = new Data Type [Rowsize, colsize];

Ex: int[,] A = new int[4,3]

A[0,0]=10; A[0,1]=20; A[0,2]=30; A[1,0]=40; A[1,1]=50; A[1,2]=60;
A[2,0]=70; A[2,1]=80; A[2,2]=90; A[3,0]=20; A[3,1]=40; A[3,2]=60;

(Or)

Syntax: Data Type[,] Array name = new Data Type [Rowsize, colsize]{{Row1 elements},{Row2 elements},...};

Ex: int[,] A = new int[4, 3] {{10,20,30}, {40,50,60}, {70,80,90} {20,40,60}};

(Or)

Int[,] A = new int [] {{10,20,30}, {40,50,60}, {70,80,90} {20,40,60}};

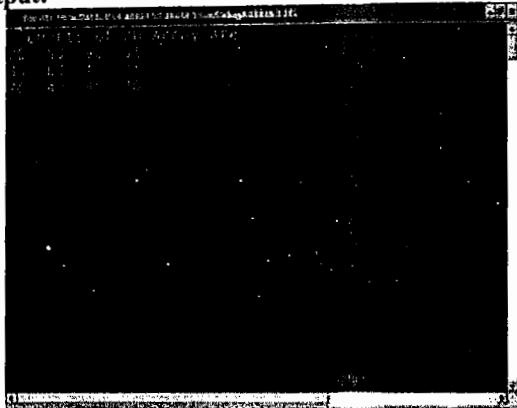
Example 3.11

Program to work with 2D Array:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class Cls2DArray
    {
        static void Main()
        {
            int[,] A = new int[3, 4]{{20,50,90,55},{35,60,10,75},{70,40,80,30}};
            Console.WriteLine("Elements of 2D Array are:- ");
            for (int R = 0; R < 3; R++)
            {
                for (int C = 0; C < 4; C++)
                {
                    Console.Write(A[R, C] + " ");
                }
                Console.WriteLine();
            }
            Console.Read();
        }
    }
}
```

Output:-



3.11.6 Working With Jagged Arrays.

A		0	1	2	
0	40	50	60		
1	0	1	2	3	
2	70	80	90	70	
3	10	20	30	40	50

Syntax: Data Type[][] Array Name = new Data Type [Rowsize][];

Ex: int[][] A = new int[3][];

A [0]= new int[3] {40,50,60};

A [1] = new int[4] {70,80,90,70};

A[2] = new int[5] {10,20,30,40,50}

Identification of elements:

A[0][0] → 40; A[0][1] → 50; A [0][2] → 60;

A[1][0] → 70; A[1][1] → 80; A[1][2] → 90; A[1][3] → 70

A[2][0] → 10; A[2][1] → 20; A[2][2] → 30; A[2][3] → 40; A[2][4] → 50

Example 3.12

Program to work with Jagged Array:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace CABASICS
```

```
{
```

```

class ClsJArray1
{
    static void Main()
    {
        int[][] A = new int[3][];
        A[0] = new int[5] { 20, 15, 35, 40, 25 };
        A[1] = new int[4] { 90, 75, 30, 45 };
        A[2] = new int[6] { 70, 60, 10, 80, 50, 35 };
        Console.WriteLine("Elements of Jagged Array are:- ");
        for (int R = 0; R < A.Length; R++)
        {
            for (int C = 0; C < A[R].Length; C++)
            {
                Console.Write(A[R][C] + " ");
            }
            Console.WriteLine();
        }
        Console.Read();
    }
}

```

Output:-

```

dotnet run
Elements of Jagged Array are:- 
20 15 35 40 25
90 75 30 45
70 60 10 80 50 35

```

Example 3.13

Program to work with Jagged Array using For Each Loop:-

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

namespace CABASICS

```

{
    class ClsJArray2
    {
        static void Main()
        {
            int[][] A = new int[3][];

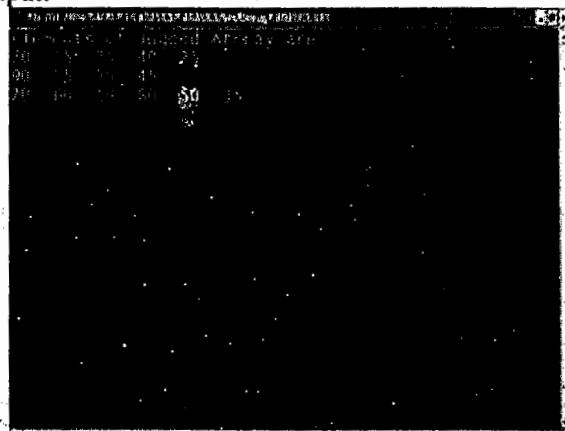
```

```

A[0] = new int[5] { 20, 15, 35, 40, 25 };
A[1] = new int[4] { 90, 75, 30, 45 };
A[2] = new int[6] { 70, 60, 10, 80, 50, 35 };
Console.WriteLine("Elements of Jagged Array are:- ");
foreach (int[] i in A)
{
    foreach (int j in i)
    {
        Console.Write(j + " ");
    }
    Console.WriteLine();
}
Console.Read();
}

```

Output:-



3.12 Boxing and Unboxing:-

- Boxing is the process of converting a variable from value type to reference type
 - UN Boxing is the process of converting a variable from reference type to value type
 - **Difference between boxing and unboxing:-**

Srno	Boxing	UnBoxing
01	Converting a variable from Value Type to Reference Type	Converting a variable from Reference Type to Value Type
02	Supports 2 types <ul style="list-style-type: none"> Implicit Boxing Explicit Boxing 	Supports only 1 Type <ul style="list-style-type: none"> Explicit UnBoxing
03	Boxing is 20 times Costlier than normal Initialization	UnBoxing is 4 times Costlier than normal Initialization

Example 3.14

Program for Boxing and Unboxig:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CABASICS
{
    class ClsBoxing
    {
        static void Main()
        {
            int i = 10;
            object O = i;//Implicit Boxing
            object X =(object) i;//Explicit Boxing
            int j = (int)O;//Explicit UnBoxing
            Console.WriteLine("Value of i is:- " + i);
            Console.WriteLine("Value of O after Implicit Boxing is:- " + O);
            Console.WriteLine("Value of X after Explicit Boxing is:- " + X);
            Console.WriteLine("Value of j after Explicit UnBoxing is:- " + j);
            Console.Read();
        }
    }
}
```

Output:-



- Boxing is 20 times costlier than normal initialization, because whenever a boxing is done following tasks will be performed internally.
 1. Runtime will search for the respective data within the stack.
 2. A copy of this value is made into Heap.
 3. Reference to this copy is maintained from the object variable.
- Unboxing is 4 times costlier than normal initialized because, when unboxing is made following tasks are performed internally.
 1. Object referenced value is searched within the Heap.
 2. A copy of this is made into stack.

When to use Boxing and Un-Boxing: As boxing and Un-boxing is costlier process, than normal initialization so Boxing and Unboxing should be avoided in maximum situations use only if necessary, but we should use boxing and unboxing in case other operations are costlier (takes more time) than boxing and unboxing.

FAQS:

1. What is Boxing?
2. What is UnBoxing?
3. What happens When Boxing is performed?
4. What happens When UnBoxing is performed?
5. List out Types of Boxing?
6. List out Types of UnBoxing?
7. What is Implicit Boxing Explain?
8. What is Explicit Boxing Explain?
9. What is Explicit UnBoxing Explain?
10. Why and how many times Boxing is Costlier than Normal Initialization?
11. Why and how many times UnBoxing is Costlier than Normal Initialization?
12. Differentiate between Boxing and UnBoxing?
13. Why UnBoxing does not support Implicit Explain?
14. Where to use and not to use Boxing?
15. Where to use and not to use UnBoxing?

Chapter 4

Object Oriented programming Concepts

4.1 Class

A class is a collection of things which posses common similarities.

Ex: Birds is a class contain common similarities / Features like.,

- Every bird has two legs.
- Every bird has two wings.
- Every bird has some color.
- Every bird can fly etc.

4.2 Object:

- An object is used to represent the class,
- An object is defined as Instance of a class.

Ex: Parrot is an object for Birds class.

4.3 Features Of Object Oriented Programming:

Any programming language to become as object oriented should follow (satisfy) the following features apart from the Class and Object.

- Abstraction
- Encapsulation
- Polymorphism
- Inheritance

4.3.1 Abstraction:

- It's a process of Hiding the implementation but providing the service.
- There are two types of abstraction available:
 - Data Abstraction
 - Functional Abstraction

4.3.2 Encapsulation:

- It's a process of Binding the member variable of a class along with member functions.
- Encapsulation can be implemented with the help of object and also Access modifiers like private, public and protected etc.

4.3.3 Polymorphism:

- It's derived from a Greek word, where poly means many morph means Faces / Behaviors.
- Same function / operator will show different behaviors when passed different type of values or different number of values.
- **Types of Polymorphism:** There are two types of polymorphism
 - Static Polymorphism /Compile Time polymorphism / Early Binding
 - Dynamic Polymorphism / Run Time polymorphism / Late Binding

- Static polymorphism is achieved using, i) Function Over Loading (ii) Operator Over Loading.
- Dynamic polymorphism is achieved by using (i) Function Overriding.

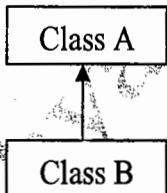
4.3.4 Inheritance:

- This is a process of creating a new class from already existing class.
- In inheritance process existing class is known as Parent / Base class, newly created class is known as Derived / Child class.
- In inheritance process, child class will get all the features of parent / base class.
- Main purpose of inheritance is code Re-usability and providing additional functionality/enhancement.

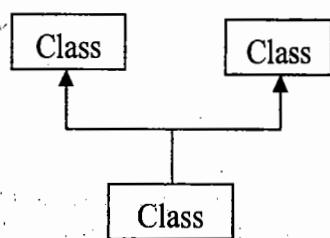
➤ Types of Inheritance:

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hybrid Inheritance
- Hierarchical Inheritance

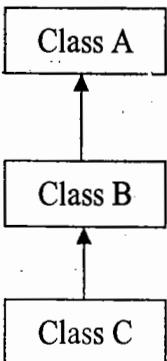
- **Single Inheritance:** Creating a single new class from single base class is known as Single Inheritance.



- **Multiple Inheritance:** Creating a new class from two or more base classes is known as Multiple Inheritance.

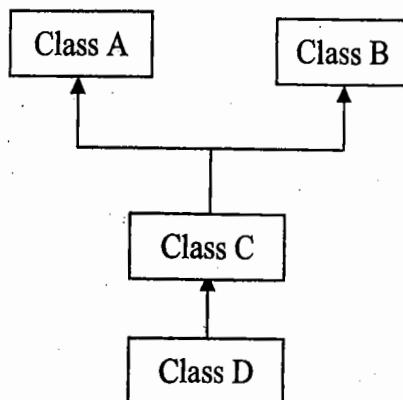


- **Multi Level Inheritance:** Creating a new class from already derived class is known as Multi Level Inheritance.



ver
lass
onal
as
as
as
as

➤ **Hybrid Inheritance:** This is combination of both Multiple and Multi Level Inheritance.



4.4 Class in C#.Net

In C#.NET a class is a user defined Datatype and is known as Reference Type.

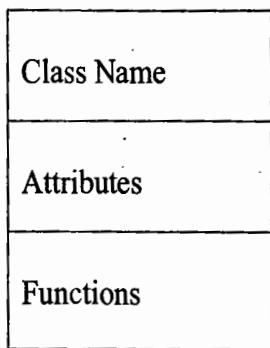
- A class can contain following members in C#.NET

1. Data Fields
2. Functions
3. Constructors
4. Destructors
5. Properties
6. Indexers
7. Events

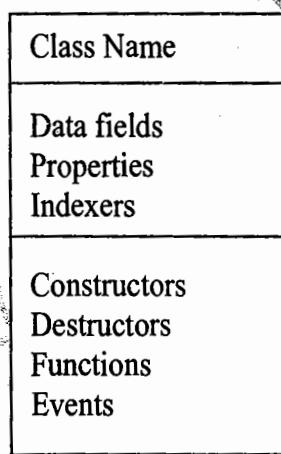
4.4.1 Diagrammatic representation of a class:-

A class is represented using rectangle symbol divided in to three parts

General notation in UML



In C#. Net



4.4.2 Data Fields

- It's used to store the data related to the class.
- Except Data Fields no other member of a class can store the data.
- To declare any Data Field, we use the following
 - Syntax: Access modifier Datatype Data Field Name [=Initialising value];
 - Ex: Public int Empid; String EName; Double salary;
- Access modifier can be any one of
 - i) private iv) protectedinternal
 - ii) protected v) public
 - iii) internal
- Default accessibility of a class is internal

4.4.3 Functions

- Any programming language will have methods.
- A method is a re-usable piece of code which can be called again and again and used to perform required tasks.
- In any programming language a method can be of two types:

(i) Procedures ii) Functions

- A procedure is a method which doesn't return any value to the calling place.
- A function is a method which always will return single value to the calling place.
- As a function will return some value to the calling place, mentioning return type is compulsory if function does not return any value then mention return type as void.

Imp Note: C#.Net will support only functions and doesn't support procedures, whereas VB.Net will support both function and procedures.

Syntax to create a class: Access Modifier class Class Name

Syntax to create an object: Class Name Object Name = new Class Name ([args list])

↓
Constructor name

Example 4.1

Program to Create a Class and to Consume:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

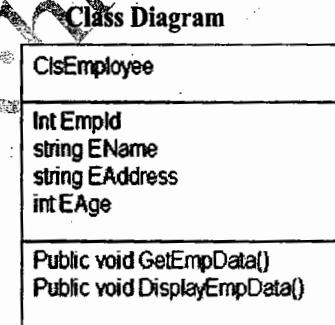
```
namespace CAClassExamples
{
    class ClsEmployee
```

```
    {
        public int EmpId, EAge; public string EName, EAddress;
        public void GetEmpData()
        {
            Console.WriteLine("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }
    }
```

```
    public void DisplayEmpData()
    {
        Console.WriteLine("Employee Id is:- " + this.EmpId);
        Console.WriteLine("Employee Name is:- " + this.EName);
        Console.WriteLine("Employee Address is:- " + this.EAddress);
        Console.WriteLine("Employee Age is:- " + this.EAge);
    }
}
```

```
class ClsExample1
```

```
{
    static void Main(string[] args)
    {
        ClsEmployee Obj1 = new ClsEmployee();
        ClsEmployee Obj2 = new ClsEmployee();
        Obj1.GetEmpData();
        Obj2.GetEmpData();
        Obj1.DisplayEmpData();
```



```
        Obj2.DisplayEmpData();
        Console.Read();
    }
}
```

Output:-

Employee Details Id: 101 Name: SAI
Address: HYDERABAD Age: 28
Employee Details Id: 102 Name: GOPAL
Address: BANGALORE Age: 25
Employee Id: 103 Name: SAI
Employee Name: SAI Address: HYDERABAD
Employee Age: 28 Employee Id: 107 Name: Gopal
Employee Name: Gopal Address: BANGALORE
Employee Age: 25

- To access any members of a class, with the help of object we use . (dot) operator known as **Member Access Operator** like Objectname.Member Name
 - Using member access operator, we can access any member based on its accessibility.
 - When an object to a class is created, runtime will allot memory for the members of a class and reference will be maintained from the object, this will be done by **new Keyword / Operator**.
 - In the above example, memory will be allotted for the members and reference will be maintained from the objects like, (Fig Required)
 - If new Keyword is not used, then memory will not be allotted to the Data Fields and no reference will be maintained from the object and raises an error as Null Reference Error.
 - **this Keyword:** this is a keyword used to access the current class in the same class.
 - **this** is an object for the current class in the same class.
 - **How Encapsulation is Implemented:**

➤ How Encapsulation is Implemented:

- In the above example when Obj1.GetEmpData() function is called Obj1 reference will be copied in to **this** keyword / Object and Obj1 referenced Data Fields will be bound to GetEmpData() function.
 - When Obj2.GetEmpData() function is called Obj2 reference will be copied in to **this** keyword / Object and Obj2 referenced Data Fields will be bound to GetEmpData() function.
 - Similarly for DisplayEmpData() function also, i.e. when ever any function is called with any Object of the class that Object referenced Data Fields(Member Variables) will be bound to the respective Member Function, this feature we call as **Encapsulation**.

Example 4.2

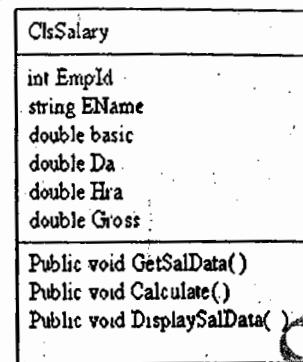
Program to Create a Class and to Consume:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAClassExamples
{
    class ClsSalary
    {
        int EmpId; string EName; double Basic, DA, HRA, Gross;
        public void GetSalData()
        {
            Console.WriteLine("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.Basic = Convert.ToDouble(Console.ReadLine());
        }
        public void Calculate()
        {
            this.DA = 0.4 * this.Basic;
            this.HRA = 0.3 * this.Basic;
            this.Gross = this.Basic + this.DA + this.HRA;
        }
        public void DisplaySalData()
        {
            Console.WriteLine("Employee Id is:- " + this.EmpId);
            Console.WriteLine("Employee Name is:- " + this.EName);
            Console.WriteLine("Employee Basic is:- " + this.Basic);
            Console.WriteLine("Employee DA is:- " + this.DA);
            Console.WriteLine("Employee HRA is:- " + this.HRA);
            Console.WriteLine("Employee Gross is:- " + this.Gross);
        }
    }
    class ClsExample2
    {
        static void Main()
        {
            ClsSalary Obj1 = new ClsSalary();
            Obj1.GetSalData();
            Obj1.Calculate();
            Obj1.DisplaySalData();
            Console.ReadLine();
        }
    }
}
```

Class Diagram



Output:-

```
Enter Employee Details - 101
SAI
20000
Employee Id is:- 101
Employee Name is:- SAI
Employee Basic is:- 20000
Employee DA is:- 6000
Employee Gross is:- 34000
```

Exercise:

- 1) Create a class to accept and print books details.

ClsBook
string BkName
string AuthName
string PubName
double Price
Public void GetBData()
Public void DisplayBData()

- 2) Design a class to accept student details along with marks and calculate result and the3 result on the screen.

ClsStudent
int Rno
string \$ name
int M1
int M2
int M3
int M4
int total
String Result
Public void GetSData()
Public void Calculate()
Public void DisplaySData()

3) Design a class to accept customer details and calculate and print electricity bill

ClsEBill
int CNumber
int PrvRead
int PresRead
int Units
double Price
Public void GetCdata ()
Public void Calculate ()
Public void DisplayCData ()

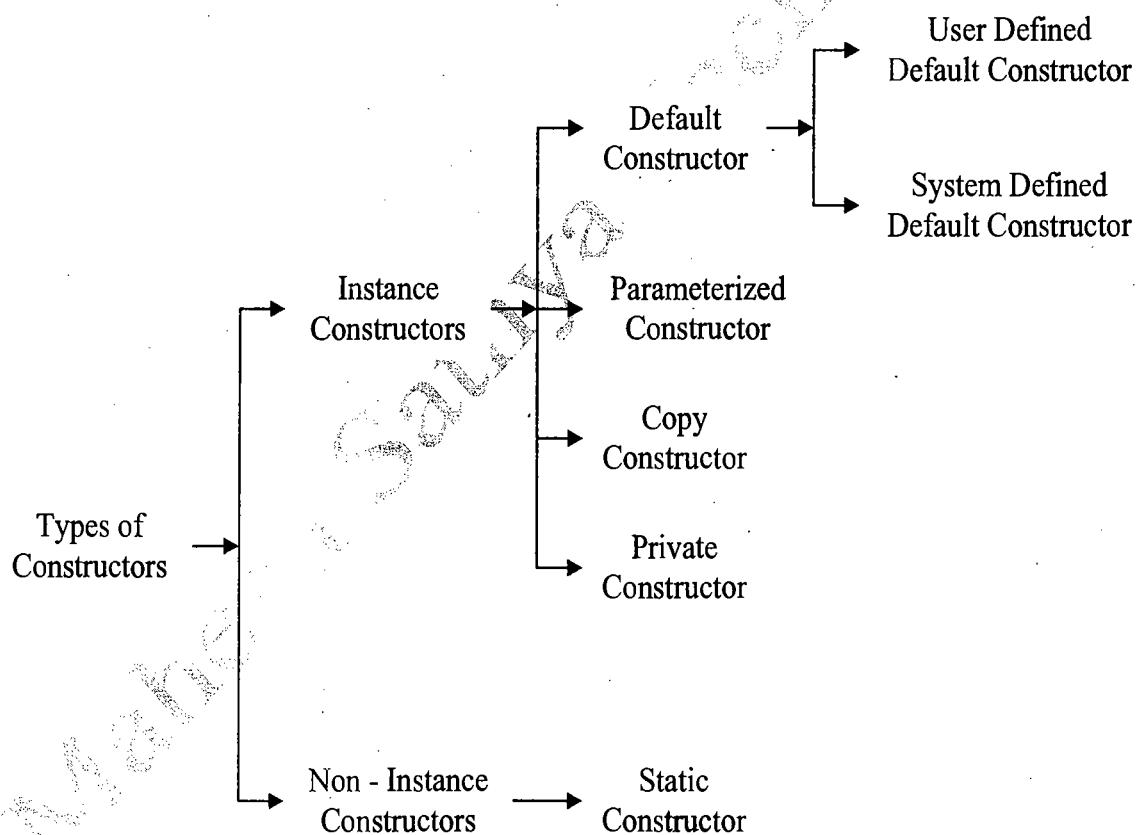
Mangesh Sonwane Technologies

Chapter 5 Constructors

5.1 Introduction

- A constructor is a member method of a class which is invoked automatically when an object to the class is created.
- A Constructor name should be same name as class name.
- A constructor does not have any return type even void also.
- A constructor is used to keep something ready for the object when it is created.
- A Constructor is used to initialize required values in to the data fields of the class or to pass the required value in to the data fields or to make required value in establishing connection to data base, keeping the file open to perform some operations etc, when an object is created.

5.2 Types of Constructor:-



5.2.1 User Defined Default Constructor:-

- This Constructor is created by the programmer.
- This constructor doesn't accept any arguments or parameters.
- In General this constructor is used to initialize required values in to the data fields.
- But there is no restriction to write particular code in to the constructor.

Example 5.1

Program to work with User Defined Default Constructor:-

Code:

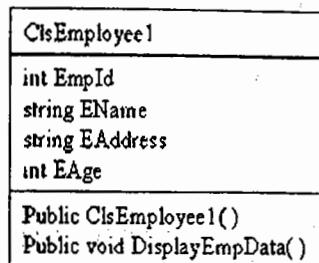
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsEmployee1
    {
        int EmpId, EAge; string EName, EAddress;
        public ClsEmployee1()
        {
            this.EmpId = 101;
            this.EName = "Sai";
            this.EAddress = "Hyderabad";
            this.EAge = 25;
        }
        public void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is:- " + EmpId);
            Console.WriteLine("Employee Name is:- " + EName);
            Console.WriteLine("Employee Address is:- " + EAddress);
            Console.WriteLine("Employee Age is:- " + EAge);
        }
    }
    class ClsUDConstructor
    {
        static void Main(string[] args)
        {
            ClsEmployee1 Obj1 = new ClsEmployee1();
            ClsEmployee1 Obj2 = new ClsEmployee1();
            Obj1.DisplayEmpData();
            Obj2.DisplayEmpData();
            Console.Read();
        }
    }
}
```

Output:-



Class Diagram



5.2.2 System Defined Default Constructor:

- If there is no constructor defined within a class then, system will create its own constructor and will assign default values into the Data Fields.
- This constructor created by the system is known as System Defined Default Constructor.
- When an object to the class is created first system will search for a constructor with in the class, If there is no constructor available with in the class system will create its own constructor and will assign default values in to the Data Fields.

Example 5.2

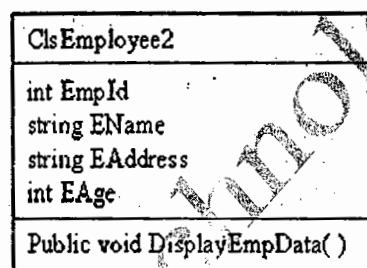
Program with System Defined Default Constructor:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsEmployee2
    {
        int EmpId, EAge; string EName, EAddress;
        public void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is: " + EmpId);
            Console.WriteLine("Employee Name is:- " + EName);
            Console.WriteLine("Employee Address is:- " + EAddress);
            Console.WriteLine("Employee Age is:- " + EAge);
        }
    }
    class ClsSDConstructor
    {
        static void Main()
        {
            ClsEmployee2 Obj1 = new ClsEmployee2();
            ClsEmployee2 Obj2 = new ClsEmployee2();
            Obj1.DisplayEmpData();
            Obj2.DisplayEmpData();
            Console.Read();
        }
    }
}
```

Class Diagram



ctor

the
wn

Output:-

```
Employee Id is: 0
Employee Name is:-
Employee Address is:-
Employee Age is: 0
Employee Id is: 0
Employee Name is:-
Employee Address is:-
Employee Age is: 0
Employee Id is: 0
Employee Name is:-
Employee Address is:-
Employee Age is: 0
Employee Id is: 0
Employee Name is:-
Employee Address is:-
Employee Age is: 0
```

In the above example though we did not write any constructor in the class, default values are stored in to the Data Fields. This is because as there is no constructor with in the class System Created its own constructor and initialized default values into the Data Fields.

Disadvantage of Default Constructors While Initializing The Data:

- In the above examples, any no. of objects we may create for a class, all the object will store same data in their referenced Data Fields.

To overcome this drawback, we use parameterized constructor

5.2.3 Parameterized Constructor:-

- A Parameterized Constructor accepts arguments to store the values in to the Data Fields.
- Using Parameterized Constructor we can store different set of values into different objects created to the class.

Example 5.3

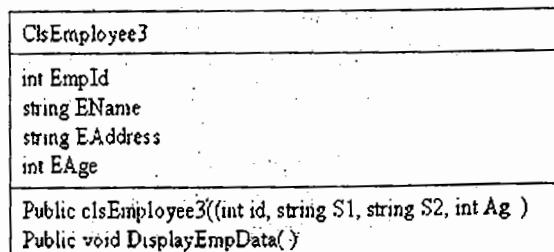
Program with Parameterized Constructor:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsEmployee3
    {
        int EmpId, EAge; string EName, EAddress;
        public ClsEmployee3(int id, string S1, string S2, int Ag)
        {
            this.EmpId = Id;
            this.EName = S1;
            this.EAddress = S2;
        }
    }
}
```

Class Diagram



```

        this.EAge = Ag;
    }
}

class ClsPConstructor
{
    static void Main()
    {
        ClsEmployee3 Obj1 = new ClsEmployee3(101, "Sai", "Hyderabad", 25);
        ClsEmployee3 Obj2 = new ClsEmployee3(102, "Raju", "Banglore", 28);
        Obj1.DisplayEmpData();
        Obj2.DisplayEmpData();
        Console.Read();
    }
}

```

Output:-

```

Employee id is:- 101
Employee Name is:- Sai
Employee Address is:- Hyderabad
Employee Age is:- 25
Employee id is:- 102
Employee Name is:- Raju
Employee Address is:- Banglore
Employee Age is:- 28

```

- In the above example, if we create an object like,
 - ClsEmployee3 obj3=newEmployee3();
 - It raises a compilation error like “CAConstrucotr. Cls employee3 doesn't contain a constructor that takes 0 arguments”.
 - To avoid this, we should write a constructor with zero arg. In our class like,

```

public ClsEmployee3()
{
    this.EmpId=110;
    this.EName="Gopal";
    this.EAddress = "Hyderabad";
    this.EAge = 22;
}

```
- We can't also create an object like by passing two arguments like,
 - ClsEmployee3 obj4 = new ClsEmployee3(101, “Surya”);

- It raises a compilation error like 'CAConstrucotr. Cls employee3 doesn't contain a constructor that takes 2 arguments'.
- To avoid this, we should write a constructor with two arguments in our class like,

```
public ClsEmployee3(int Id, string S1)
{
    this.EmpId = Id;
    this.EName = S1;
}
```

After writing these two constructors in the class, the class code will be like

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsEmployee3
    {
        int EmpId, EAge; string EName, EAddress;
        public ClsEmployee3()
        {
            this.EmpId=110;
            this.EName="Gopal";
            this.EAddress = "Hyderabad";
            this.EAge = 22;
        }
        public ClsEmployee3(int Id, string S1)
        {
            this.EmpId = Id;
            this.EName = S1;
        }
        public ClsEmployee3(int Id, string S1, string S2, int Ag)
        {
            this.EmpId=Id;
            this.EName=S1;
            this.EAddress = S2;
            this.EAge = Ag;
        }
        public void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is:- " + EmpId);
            Console.WriteLine("Employee Name is:- " + EName);
            Console.WriteLine("Employee Address is:- " + EAddress);
            Console.WriteLine("Employee Age is:- " + EAge);
        }
    }
    class ClsPConstructor
    {
        static void Main()
        {
            ClsEmployee3 Obj1 = new ClsEmployee3(101, "Sai", "Hyderabad", 25);
            ClsEmployee3 Obj2 = new ClsEmployee3(102, "Raju", "Banglore", 28);
            ClsEmployee3 Obj3 = new ClsEmployee3();
        }
    }
}
```

```

        ClsEmployee3 Obj4 = new ClsEmployee3(103, "Surya");
        Obj1.DisplayEmpData();
        Obj2.DisplayEmpData();
        Obj3.DisplayEmpData();
        Obj4.DisplayEmpData();
        Console.Read();
    }
}
}

```

Output:-

```

C:\Users\DELL\JAMAL\CAConstructors\CAConstructors\bin\Debug\CAConstructors.exe
Employee Id is:- 101
Employee Name is:- Sai
Employee Address is:- Hyderabad
Employee Age is:- 25
Employee Id is:- 102
Employee Name is:- Raju
Employee Address is:- Bangalore
Employee Age is:- 28
Employee Id is:- 100
Employee Name is:- Gopal
Employee Address is:- Hyderabad
Employee Age is:- 22
Employee Id is:- 103
Employee Name is:- Surya
Employee Address is:-
Employee Age is:- 0

```

From the above points we can conclude that

- A class can contain more than one constructor.
- A constructor can be overloaded for different no. of arguments.

5.2.4 Copy Constructor:

- This Constructor is used to copy the data of an existing object into a newly created object.
- To copy constructor we need to pass argument that belongs to our class data type.

Example 5.4

Program with Copy Constructor:-

Code:

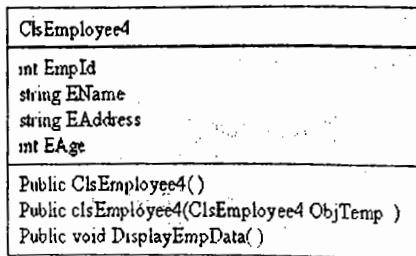
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsEmployee4
    {

```

Class Diagram



```

int EmpId, EAge; string EName, EAddress;
public ClsEmployee4()
{
    Console.WriteLine("Enter Employee Details:- ");
    this.EmpId = Convert.ToInt32(Console.ReadLine());
    this.EName = Console.ReadLine();
    this.EAddress = Console.ReadLine();
    this.EAge = Convert.ToInt32(Console.ReadLine());
}
public ClsEmployee4(ClsEmployee4 ObjTemp)
{
    this.EmpId = ObjTemp.EmpId;
    this.EName = ObjTemp.EName;
    this.EAddress = ObjTemp.EAddress;
    this.EAge = ObjTemp.EAge;
}
public void DisplayEmpData()
{
    Console.WriteLine("Employee Id is:- " + EmpId);
    Console.WriteLine("Employee Name is:- " + EName);
    Console.WriteLine("Employee Address is:- " + EAddress);
    Console.WriteLine("Employee Age is:- " + EAge);
}
class ClsCConstructor
{
    static void Main()
    {
        ClsEmployee4 Obj1 = new ClsEmployee4();
        ClsEmployee4 Obj2 = new ClsEmployee4(Obj1);
        Obj1.DisplayEmpData();
        Obj2.DisplayEmpData();
        Console.Read();
    }
}

```

Output:-

```

File:///F:/J7AM/CAConstructors/CAConstructors/bin/Debug/CAConstructors.EXE
Enter Employee Details:-
101
SAI
HYDERABAD
25
Employee Id is:- 101
Employee Name is:- SAI
Employee Address is:- HYDERABAD
Employee Age is:- 25
Employee Id is:- 101
Employee Name is:- SAI
Employee Address is:- HYDERABAD
Employee Age is:- 25
=
```

5.2.5 Static Constructor

- Static Constructor is used to initialize static Data fields.
- Static Data Fields will have maintained only one instance for any no. of objects created to the class.
- To make any member of a class static, use static keyword.
- Static members are not accessible with object rather; we need to access with class name only.
- There can be only one static constructor in the class.
- The static constructor should be without parameters.
- It can only access the static members of the class.
- There should be no access modifier in static constructor definition.
- If a class is static, then we can't create object for static class.
- Static constructor will be involved only once ie, 1st object created for the class from second object onwards static constructor will not be called.

Example 5.5

Program with Static Constructor:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAConstructors
{
    class ClsSample
    {
        int i;
        static int j;
        public ClsSample()
        {
            i = 100;
        }
        static ClsSample()
        {
            j = 100;
        }
        public void Display()
        {
            Console.WriteLine("Value of i is:- " + i);
            i++;
            Console.WriteLine("Value of j is:- " + j);
            j++;
        }
    }
    class ClsConstructor
    {
        static void Main()
        {
            ClsSample Obj1 = new ClsSample();
            Obj1.Display();
        }
    }
}
```

Class Diagram

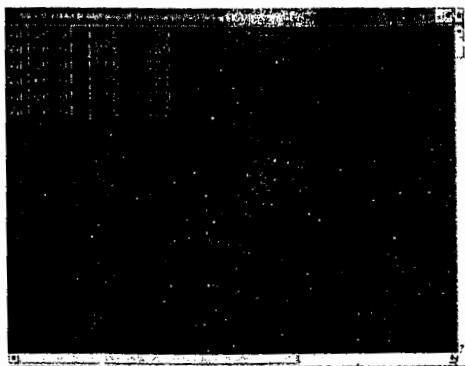
ClsSample
int i;
static int j;
public ClsSample()
static ClsSample()
public void Display()

```

        ClsSample Obj2 = new ClsSample();
        Obj2.Display();
        ClsSample Obj3 = new ClsSample();
        Obj3.Display();
        Console.Read();
    }
}
}
}

```

Output:-



- It's not possible to initialize non-static Data Fields, with in the static constructors, it raises compilation error.

```

static ClsSample()
{
    i = 100; // Not allowed.
    j = 100;
}

```

- We can initialize static Data fields with a non-static constructor but they loose their static nature.

```

public cls sample()
{
    i = 100;
    j = 100; // Allowed But j loses its static nature
}

```

- We can initialize static Data fields in both static and non static constructors but static Data Fields will loose their static nature.

```

public ClsSample()
{
    i = 100;
    j = 100; // Allowed But j loses its static nature
}
static ClsSample()
{
    J = 100; // Allowed
}

```

5.2.6 Private Constructor

- A Constructor whose accessibility is private is known as private constructor.
- If a class contains private constructor then we cannot create an object for the class out side the class.
- Private Constructor are used to create an object for the class with in the same class and to use.
- Generally private constructors are used with in the Remoting concept.

5.2.7 Instance Constructors:-

In stnce constructor will maintain separate instances for each member object of the class, for each object of the class.

5.2.8 Non Instance Constructors:-

Non Instance Constructor will not maintain separate instances for each data field of the class for each object of the class.

FAQs on Constructors

1. What is a Constructor?
2. What should be the name of a constructor?
3. What is the return type of a constructor?
4. What is the purpose of a constructor?
5. List out the types of constructors?
6. Explain about Instance constructors?
7. Explain about Non-Instance constructors?
8. What is User Defined Default constructor Explain?
9. What is System Defined Default constructor Explain?
10. What is the drawback of Default constructor?
11. What is Parameterized constructor Explain?
12. What is Copy constructor Explain?
13. What is private constructor Explain?
14. Can a constructor be overloaded?
15. How many constructors can a class contain?
16. Can we create an object to a class that contains private constructor?
17. Where private constructors are used?
18. What is static constructor explain?
19. Can we initialize static data fields with in the non static constructors?
20. Can we initialize non static data fields with in the static constructors?
21. What happens if a static data field is initialized with in the non static constructor?
22. What happens if a non static data field is initialized with in the static constructor?
23. What keyword is used to make any member as static?
24. How do you access static members of a class?
25. Explain the nature of static members of the class?
26. Can a Static class be instantiated?
27. Where Static Constructors are used?

Chapter 6

Implementing Polymorphism and Inheritance in C#.NET:-

6.1 Function Signature

- The *signature* of a method must be unique in the class in which the method is declared.
- The signature of a method consists of the name of the method, modifiers, types of its parameters the number of parameters.
- The signature of a method does not include the return type.

6.2 Function overloading

- A Function overloading can be compared with person overloading.
- If a person has already some work to do and we are assigning additional work to the person then person will be overloaded.
- In the same way, a function will have already some work to do and if we assign a different work to the same function, then we say function is overloaded.
- A function will be overloading in two Situations.
 1. When data types of the argument are changed.
 2. When number of argument are changed.

6.2.1 When data types of the argument are changed

Consider a function like:

```
Add(int a, int b)
{
}
```

- We can call the above function by using two arguments like Add (10, 20)
- But if we want to call the above function by passing different data type value like Add (10.5, 20), we cannot use the above function and we overload the function like

```
Add(float a, int b)
{
}
```

6.2.2 When numbers of arguments are changed

Consider a function like:

```
Add(int a, int b)
{
}
```

- We can call the above function by passing two arguments like Add (10, 20).
- If we want to call the above function with different number of argument like Add (10, 20, 30) We cannot use the above function and we overload our function and we overload the function like

```

Add(int a, int b, int c)
{
}

```

6.2.3 Definition

- Providing new implementation to a function with different signature is known as function overloading.
- In general function overloading is implemented with in the same class.

6.3 Operator overloading

- Every operator has some pre-defined work given by the designers if we assign additional work apart from the existing work to any operator then we say operator is overloaded.
- Syntax to Overload an operator is:- *type operator operator-symbol (parameter-list)*

Example for Operator overloading

- In general '+' is used to perform addition of 2 numbers but using '+' operator we can't perform addition of 2 objects of a class. So, we overload our '+' operator to perform addition of 2 objects of a class.
- Consider addition of 2 Complex Numbers like (see the table right)
- We can not create any variables and store Real and Complex parts, So we create a class named 'ClsComplex' with 2 Data Fields int Real, Imaginary.
- Now if create any object for this class it contains a reference of Real and Imaginary Parts.
- In two objects of the class we store two complex numbers and perform addition of these two objects and store the result in third object.

	Real	Imaginary
Num1	4	5i
Num2	6	3i
Num3	10	8i

Example 6.1

Program to overload addition Operator:-

Code:

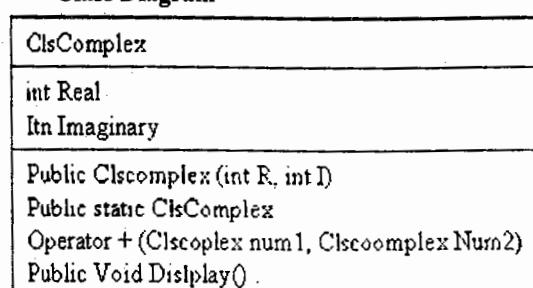
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAPOverLoad
{
    class ClsComplex
    {
        int Real, Imaginary;
        public ClsComplex(int R, int I)
        {
            this.Real = R;
            this.Imaginary = I;
        }
    }
}

```

Class Diagram

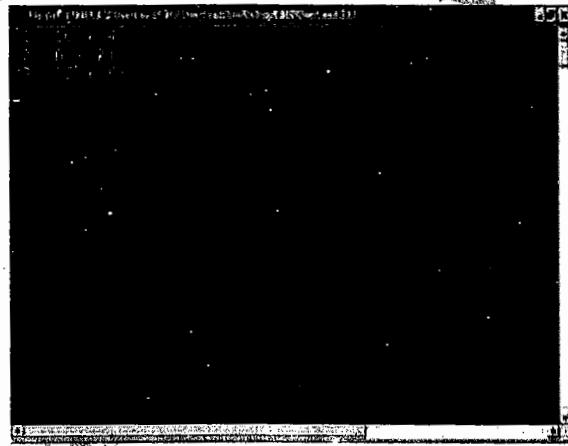


```

public static ClsComplex operator +(ClsComplex Num1, ClsComplex Num2)
{
    ClsComplex Num3 = new ClsComplex(Num1.Real + Num2.Real, Num1.Imaginary +
    Num2.Imaginary);
    return Num3;
}
public void Display()
{
    Console.WriteLine("{0} + {1}i", Real,Imaginary);
}
}
class ClsOpOverLoad
{
    static void Main(string[] args)
    {
        ClsComplex C1 = new ClsComplex(10, 4);
        ClsComplex C2 = new ClsComplex(6, 2);
        ClsComplex C3 = C1 + C2;
        Console.Write("C1= ");
        C1.Display();
        Console.Write("C2= ");
        C2.Display();
        Console.Write("C3= ");
        C3.Display();
        Console.Read();
    }
}

```

Output:-



Exercise: Overload Addition operator to add any two Fractions like $(2/3 + 4/5)$

6.3.1 List of Operators that can be overloaded

- You can redefine the function of most built-in operators globally or on a class-by-class basis.
Overloaded operators are implemented as functions.
- The name of an overloaded operator is operatorx, where x is the operator as it appears in the following table. For example, to overload the addition operator, you define a function called operator+. Similarly, to overload the addition/assignment operator, +=, define a function called operator+=%.

Operator	Name	Type
,	Comma	Binary
!	Logical NOT	Unary
!=	Inequality	Binary
%	Modulus	Binary
%=	Modulus assignment	Binary
&	Bitwise AND	Binary
&	Address-of	Unary
&&	Logical AND	Binary
&=	Bitwise AND assignment	Binary
()	Function call	—
()	Cast Operator	Unary
*	Multiplication	Binary
*	Pointer dereference	Unary
*=	Multiplication assignment	Binary
+	Addition	Binary
+	Unary Plus	Unary
++	Increment	Unary
+=	Addition assignment	Binary
-	Subtraction	Binary
-	Unary negation	Unary
--	Decrement	Unary
--	Subtraction assignment	Binary
→	Member selection	Binary
→*	Pointer-to-member selection	Binary
/	Division	Binary
/=	Division assignment	Binary
<	Less than	Binary
<<	Left shift	Binary
<<=	Left shift assignment	Binary
<=	Less than or equal to	Binary
=	Assignment	Binary
==	Equality	Binary
>	Greater than	Binary
>=	Greater than or equal to	Binary
>>	Right shift	Binary
>>=	Right shift assignment	Binary
[]	Array subscript	—
^	Exclusive OR	Binary
^=	Exclusive OR assignment	Binary
	Bitwise inclusive OR	Binary
=	Bitwise inclusive OR assignment	Binary
	Logical OR	Binary
~	One's complement	Unary
delete	Delete	—
new	New	—
conversion operators	conversion operators	Unary

6.3.2 List of Operators that can not be overloaded

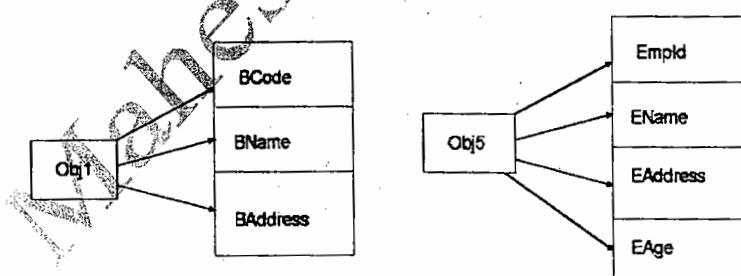
- The operators shown in the following table cannot be overloaded. The table includes the preprocessor symbols # and ##.

Operator	Name
.	Member selection
*	Pointer-to-member selection
::	Scope resolution
? :	Conditional
#	Preprocessor convert to string
##	Preprocessor concatenate

6.4 Implementing Inheritance in C#.NET

6.4.1 Situation where to implement inheritance:-

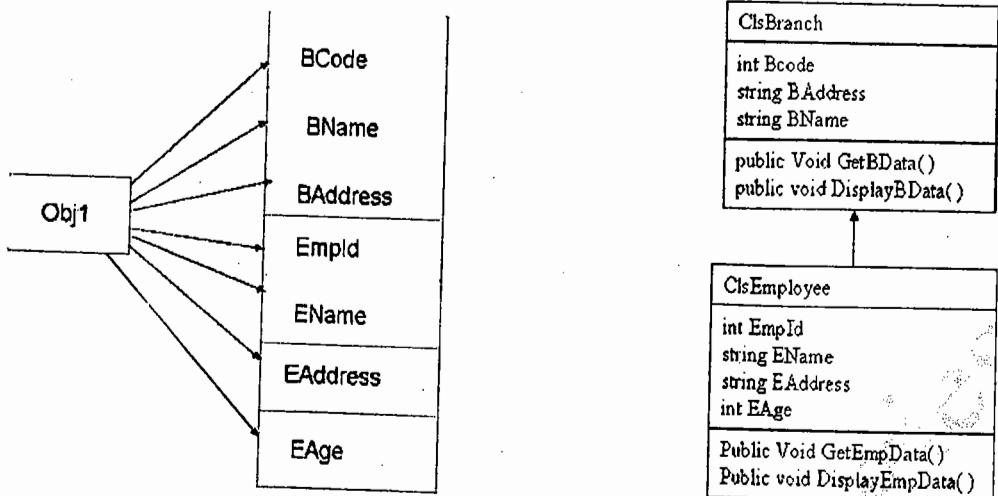
- Assume that a company has N-no. of branches and asked to computerize Branches Details of the company, then we create a class like...
- Late after, some period of time, company also asked to computerize Employees details of each Branch.
- If we create two classes individually with out inheritance we need to create objects for every class individually and obects will have maintained separate references to the members of the classes like...



ClsBranch
int Bcode
string BAddress
string BName
public Void GetBData()
public void DisplayBData()

ClsEmployee
int EmpId
string EName
string EAddress
int EAge
Public Void GetEmpData()
Public void DisplayEmpData()

- Where Obj1 is object to ClsBranch class and Obj5 is object to ClsEmployee class.
- So it becomes difficult to identify which employee belongs to which branch and integrate the ClsBranch class objects with ClsEmployee class objects.
- So if we derive the ClsEmployee class from ClsBranch class we create object to the Derived class ClsEmployee then it will represent both the classes and will maintain reference to the members of both base and derived classes like...



Where Obj1 is object to the derived class ClsEmployee

- As ClsEmployee is inherited from ClsBranch class, ClsEmployee class will get all the features of ClsBranch class.

Example 6.2

Program to Implement Single Inheritance:-

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

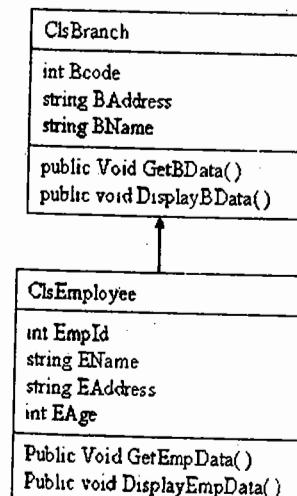
namespace CASInheritance
{
    class ClsBranch
    {
        int BCode; string BName, BAddress;
        public void GetBData()
        {
            Console.WriteLine("Enter Branch Details:- ");
            BCode = Convert.ToInt32(Console.ReadLine());
            BName = Console.ReadLine();
            BAddress = Console.ReadLine();
        }

        public void DisplayBData()
        {
            Console.WriteLine("Branch Code is:- " + BCode);
            Console.WriteLine("Branch Name is:- " + BName);
            Console.WriteLine("Branch Address is:- " + BAddress);
        }
    }

    class ClsEmployee : ClsBranch
    {
        int EmpId, EAge; string EName, EAddress;
        public void GetEmpData()
    }
}

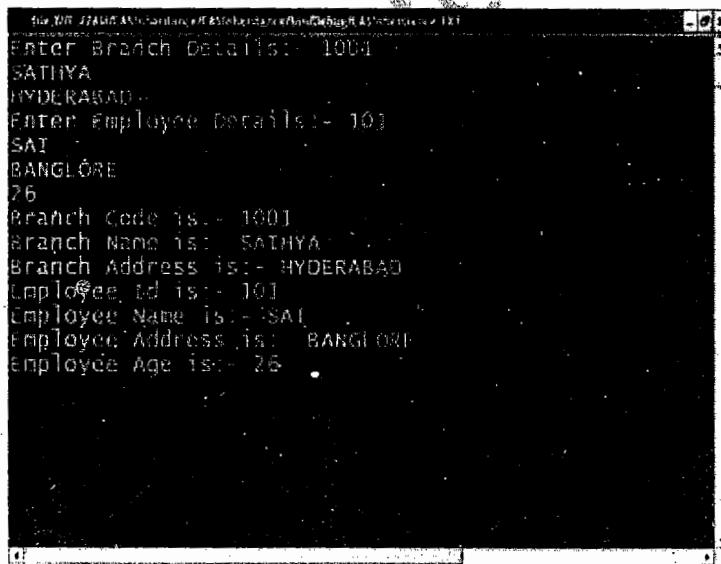
```

Class Diagram:



```
{  
    Console.WriteLine("Enter Employee Details:- ");  
    this.EmpId = Convert.ToInt32(Console.ReadLine());  
    this.EName = Console.ReadLine();  
    this.EAddress = Console.ReadLine();  
    this.EAge = Convert.ToInt32(Console.ReadLine());  
}  
public void DisplayEmpData()  
{  
    Console.WriteLine("Employee Id is:- " + this.EmpId);  
    Console.WriteLine("Employee Name is:- " + this.EName);  
    Console.WriteLine("Employee Address is:- " + this.EAddress);  
    Console.WriteLine("Employee Age is:- " + this.EAge);  
}  
}  
class ClsSInheritance  
{  
    static void Main(string[] args)  
    {  
        ClsEmployee Obj1 = new ClsEmployee();  
        Obj1.GetBData();  
        Obj1.GetEmpData();  
        Obj1.DisplayBData();  
        Obj1.DisplayEmpData();  
        Console.ReadLine();  
    }  
}
```

Output:



- In the above example we made the functions GetEmpData() and DisplayEmpData() of ClsBranch class as public, because to access from out side the class i.e. from ClsSInheritance.
 - And the Data Fields BCode, Bname and BAddress are private by default, so they are accessible with in the same class only.
 - But if we do not want to give accessibility of the base class members to the non derived class (ClsSInheritance) and would like to give to the Derived class(ClsEmployee) then we use protected to the members.
 - And our code will be like...

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CASInheritance
{
    class ClsBranch
    {
        int BCode; string BName, BAddress;
        protected void GetBData()
        {
            Console.Write("Enter Branch Details:- ");
            BCode = Convert.ToInt32(Console.ReadLine());
            BName = Console.ReadLine();
            BAddress = Console.ReadLine();
        }
        protected void DisplayBData()
        {
            Console.WriteLine("Branch Code is:- " + BCode);
            Console.WriteLine("Branch Name is:- " + BName);
            Console.WriteLine("Branch Address is:- " + BAddress);
        }
    }
    class ClsEmployee : ClsBranch
    {
        int EmpId, EAge; string EName, EAddress;
        public void GetEmpData()
        {
            base.GetBData();
            Console.Write("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }
        public void DisplayEmpData()
        {
            base.DisplayBData();
            Console.WriteLine("Employee Id is:- " + this.EmpId);
            Console.WriteLine("Employee Name is:- " + this.EName);
            Console.WriteLine("Employee Address is:- " + this.EAddress);
            Console.WriteLine("Employee Age is:- " + this.EAge);
        }
    }
    class ClsSInheritance
    {
        static void Main(string[] args)
        {
            ClsEmployee Obj1 = new ClsEmployee();
            Obj1.GetEmpData();
            Obj1.DisplayEmpData();
            Console.ReadLine();
        }
    }
}
```

}

6.4.2 Access Modifiers:

- These are used to provide restriction to the members of a class or class to access in same class or in derived class in same Assembly or different Assembly etc.
- .NET supports Five types of Access Modifiers.
 - **private:** Private Members are accessible only with in the same class.
 - **protected:** Protected Members are accessible with in the same class and also in the derived class, derived class might be in same assembly or in different assembly.
 - **Internal:** Internal Members are accessible in any class with in the same assembly but nor accessible in any clas out side the assembly.
 - **protectedinternal:** This is combination of both Protected and Internal i.e protectedinternal members are accessible in any class with in the same assembly and also from the derived classes out side the assembly, but not accessible from the Non – Derived classes out side the assembly.
 - **public:** Public members ar accessible in any class in any assembly.
- We can summerise all these in a table like given below, in his table 'X' represents not accessible and ✓ represents accessible.

Class Access Modifier	Same Assembly			Other Assemblies	
	Same class	Derive d class	Non-Derived Class	Derived class	Non-Derived class
Private	✓	X	X	X	X
Protected	✓	✓	X	✓	X
Internal	✓	✓	✓	X	X
Protected Internal	✓	✓	✓	✓	X
Public	✓	✓	✓	✓	✓

6.4.3 Implementing Multi Level Inheritance:

- In single Inheritance we have derived a new class ClsEmployee from ClsBranch class.
- Later during period of time, let us say company asked to computerize salary details of an employee. Then, for every salary details we need to have respective employee details and for every employee we need to have respective branch details.
- Then we derive a new class say ClsSalary from the ClsEmployee class like...

Example 6.3 Program to Implement Multi Level Inheritance:-

Code:

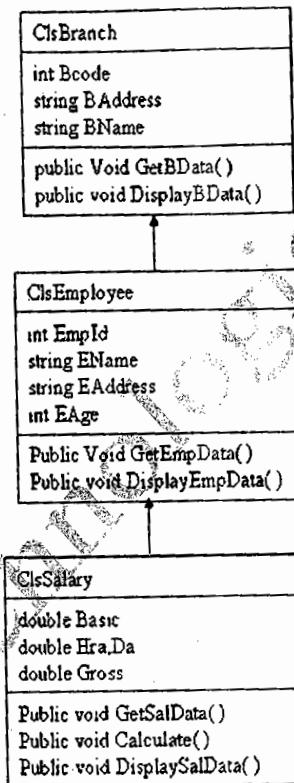
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAMLIInheritance
{
    class ClsBranch
    {
        int BCode; string BName, BAddress;
        public void GetBData()
        {
            Console.WriteLine("Enter Branch Details:- ");
            BCode = Convert.ToInt32(Console.ReadLine());
            BName = Console.ReadLine();
            BAddress = Console.ReadLine();
        }
        public void DisplayBData()
        {
            Console.WriteLine("Branch Code is:- " + BCode);
            Console.WriteLine("Branch Name is:- " + BName);
            Console.WriteLine("Branch Address is:- " + BAddress);
        }
    }
    class ClsEmployee : ClsBranch
    {
        int EmpId, EAge; string EName, EAddress;
        public void GetEmpData()
        {
            Console.WriteLine("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }
        public void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is:- " + this.EmpId);
            Console.WriteLine("Employee Name is:- " + this.EName);
            Console.WriteLine("Employee Address is:- " + this.EAddress);
            Console.WriteLine("Employee Age is:- " + this.EAge);
        }
    }
    class ClsSalary : ClsEmployee
    {
        double Basic, DA, HRA, Gross;
        public void GetSalData()
        {
            Console.WriteLine("Enter Basic Salary:- ");
            Basic = Convert.ToDouble(Console.ReadLine());
        }
    }
}

```

Class Diagram

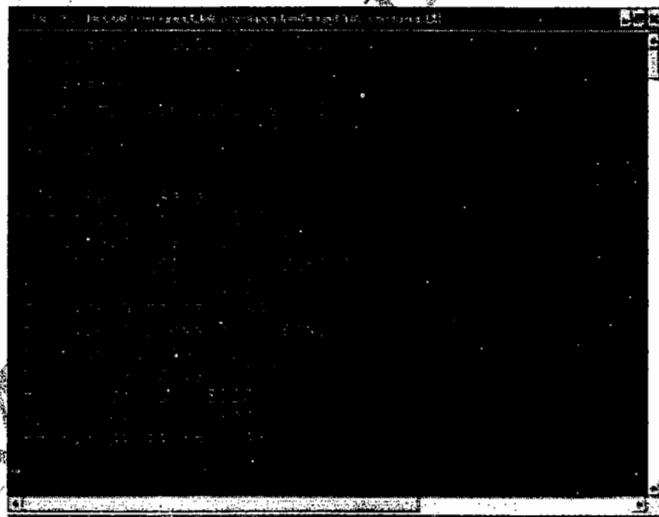


```

public void Calculate()
{
    DA = 0.4 * Basic;
    HRA = 0.3 * Basic;
    Gross = Basic + DA + HRA;
}
public void DisplaySalData()
{
    Console.WriteLine("Employe Basic is:- " + Basic);
    Console.WriteLine("Employe DA is:- " + DA);
    Console.WriteLine("Employe HRA is:- " + HRA);
    Console.WriteLine("Employe GROSS is:- " + Gross);
}
class ClsMInheritance
{
    static void Main(string[] args)
    {
        ClsSalary Obj1 = new ClsSalary();
        Obj1.GetBData();
        Obj1.GetEmpData();
        Obj1.GetSalData();
        Obj1.Calculate();
        Obj1.DisplayBData();
        Obj1.DisplayEmpData();
        Obj1.DisplaySalData();
        Console.ReadLine();
    }
}

```

Output:-



6.4.4 Implementing Dynamic Polymorphism i.e Function Overiding In C#.Net:

- Function overriding is providing new Implementation to a function with same signature.
- In general function overriding will be implemented with in the derived class.
- Let us consider a company asked to computerize Employee details, then we design a class ie, ClsEmployee Like

- During the period of Time, Company has expanded and no. of employees in the company are expanded to large, then company decided to categories the employees as Managers and Non-Managers.
- Further decided not to accept EAddress and EAge for Managers and to provide additionally Car Allowance and Bonus.
- For non-managers we can use ClsEmployee class as it is, but can not be used for Managers.
- So, we will derive a new class from ClsEmployee, as ClsManager and provide new functionality for GetEmpData() and DisplayEmpData() functions.
- As we are providing new functionality for base class functions in derived class and base class functions are not useful now so base class functions are known as Virtual functions and derived class are known as overriding functions.
- To make any function as virtual use **virtual** keyword.
- To override any virtual function in derived class use **override** keyword.

```

ClsEmployee
protected int EmpId
protected string EName
string EAddress
int EAge
Public void GetEmpData()
Public void DisplayEmpData()

```

Example 6.4

Program for Function Overriding:-

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAFOVERRIDE
{
    class ClsEmployee
    {
        protected int EmpId, EAge;
        protected string EName, EAddress;
        public virtual void GetEmpData()
        {
            Console.WriteLine("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }
        public virtual void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is:- " + this.EmpId);
            Console.WriteLine("Employee Name is:- " + this.EName);
            Console.WriteLine("Employee Address is:- " + this.EAddress);
            Console.WriteLine("Employee Age is:- " + this.EAge);
        }
    }
    class ClsManager : ClsEmployee
    {
        double Bonus, CA;
        public override void GetEmpData()

```

Class Diagram

```

ClsEmployee
protected int EmpId
protected string EName
string EAddress
int EAge
public virtual void GetEmpData()
public virtual void DisplayEmpData()

ClsManager
double Bonus
double CA
public override void GetEmpData()
public override void DisplayEmpData()

```

Out

➤ C
➤ I
➤ f
➤ W
➤ C
➤ O

```

        {
            Console.WriteLine("Enter Manager Details:- ");
            EmpId = Convert.ToInt32(Console.ReadLine());
            EName = Console.ReadLine();
            Bonus = Convert.ToDouble(Console.ReadLine());
            CA = Convert.ToDouble(Console.ReadLine());
        }
        public override void DisplayEmpData()
        {
            Console.WriteLine("Manager Id is:- " + EmpId);
            Console.WriteLine("Manager Name is:- " + EName);
            Console.WriteLine("Manager Bonus is:- " + Bonus);
            Console.WriteLine("Manager CA is:- " + CA);
        }
    }
    class ClsOverride
    {
        static void Main(string[] args)
        {
            ClsManager Obj1 = new ClsManager();
            Obj1.GetEmpData();
            Obj1.DisplayEmpData();
            Console.ReadLine();
        }
    }
}

```

Output:-

```

Title: BHU-JAVA\override\bin\Debug\A1 Override
Enter Manager Details:- SAI
SAI
20000
10000
Manager Id is:- 100
Manager Name is:- SAI
Manager Bonus is:- 20000
Manager CA is:- 10000

```

- Overriding of virtual functions is not compulsory, optional only.
- If base class virtual function is not overridden with in the derived class then base class virtual function is called.
- We can instantiate / create an object to a class which contains one (or) more virtual functions.
- Creating a new class from a class which contains virtual functions is not compulsory, optional.

Differences between function overloading and function overriding

Srno	Function Overloading	Function Overriding
01	Providing New Implementation to a function with same Name and Different Signature is known as Function Overloading	Providing New Implementation to a function with same Name and Same Signature is known as Function Overriding
02	Function Overloading will be done in the Same Class	Function Overriding will be done in the Base and Derived Classes
03	This is Code refinement Technique	This is Code Replacement Technique
04	No separate keywords are used to implement function Overloading	Use virtual keyword for base class function and override keyword in derived class function to implement function overriding
05	Used to implement static polymorphism	Used to implement dynamic polymorphism

6.4.5 Abstract Functions and Abstract Classes:

- **Abstract Function:** - A function which contains only Declaration / Signature and doesn't contain Implementation / Body / Definition, is known as **Abstract function**.
- To make any function as abstract use **abstract** keyword.
- An abstract Function should be terminated.
- Overriding of an abstract function is compulsory.
- **Abstract Class:** A class which contains one or more abstract functions is known as an abstract class.
- To make any class as abstract use **abstract** keyword.
- An abstract class can't be instantiated directly.
- It's compulsory to create / derive a new class from an abstract class in order to provide functionality to its abstract functions.
- An abstract class can contain non- abstract functions.
- An abstract class can contain all members of a class.
- By default abstract class functions are not treated as public and abstract.

Example 6.5

Program for Abstract Class:-

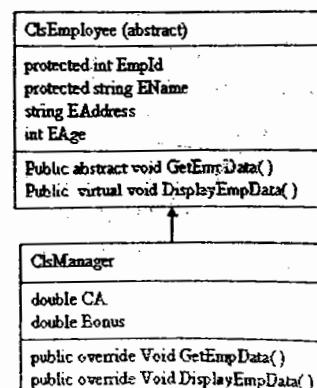
(For making it easy we modify same example of function overriding)

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAAbstract
{
    abstract class ClsEmployee
    {
        protected int EmpId, EAge;
    }
}
```

Class Diagram



```

protected string EName, EAddress;
public abstract void GetEmpData();
public virtual void DisplayEmpData()
{
    Console.WriteLine("Employee Id is:- " + this.EmpId);
    Console.WriteLine("Employee Name is:- " + this.EName);
    Console.WriteLine("Employee Address is:- " + this.EAddress);
    Console.WriteLine("Employee Age is:- " + this.EAge);
}
}

class ClsManager : ClsEmployee
{
    double Bonus, CA;
    public override void GetEmpData()
    {
        Console.Write("Enter Manager Details:- ");
        EmpId = Convert.ToInt32(Console.ReadLine());
        EName = Console.ReadLine();
        Bonus = Convert.ToDouble(Console.ReadLine());
        CA = Convert.ToDouble(Console.ReadLine());
    }
    public override void DisplayEmpData()
    {
        Console.WriteLine("Manager Id is:- " + EmpId);
        Console.WriteLine("Manager Name is:- " + EName);
        Console.WriteLine("Manager Bonus is:- " + Bonus);
        Console.WriteLine("Manager CA is:- " + CA);
    }
}

class ClsAbstract
{
    static void Main(string[] args)
    {
        ClsManager Obj1 = new ClsManager();
        //ClsEmployee Obj2 = new ClsEmployee();
        Obj1.GetEmpData();
        Obj1.DisplayEmpData();
        Console.Read();
    }
}

```

Output:-

File:///E:/TAMCAT/Overriding/PLAT/Overrides/bin/Debug/C#OOverride.exe

Enter Manager Data Id:- 101
SAI
20000
10000
Manager Id is:- 101
Manager Name is:- SAI
Manager Bonus Id:- 10000
Manager CA is:- 10000

6.5 Working with Interfaces

- If a class contains all abstract functions then it is known as an Interface.
- Interfaces do not provide implementation. They are implemented by classes, and defined as separate entities from classes.
- An Interface represents a **contract**, in that a class that implements an interface must implement every aspect of that interface exactly as it is defined.
- In general an Interface provides control over the classes.
- We can define features as small groups of closely related members and can develop enhanced implementations for your interfaces without jeopardizing existing code, thus minimizing compatibility problems.
- It is also possible to add new features at any time by developing additional interfaces and implementations.
- Although interface implementations can evolve, interfaces themselves cannot be changed once published.
- Changes to a published interface may break existing code.
- If you think of an interface as a contract, it is clear that both sides of the contract have a role to play.
- The publisher of an interface agrees never to change that interface, and the implementer agrees to implement the interface exactly as it was designed.
- To create an Interface use **interface** Keyword.
- An Interface can contain
 - Abstract function
 - Properties
 - Indexes
 - Events.
- An Interface can not contain
 - Non-Abstract Functions
 - Data Fields
 - Constructors
 - Destructors.
- By default Interface functions are treated as Public and abstract.
- An Interface can't be instantiated directly.
- Creating / deriving a new class or other Interface from an Interface is compulsory/mandatory in order to provide implementation to its abstract function.
- An interface can itself inherit from multiple interfaces.
- An Interface is used to Implementation Multiple Inheritance in C#.NET

Example 6.6

Program to implement Interface:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

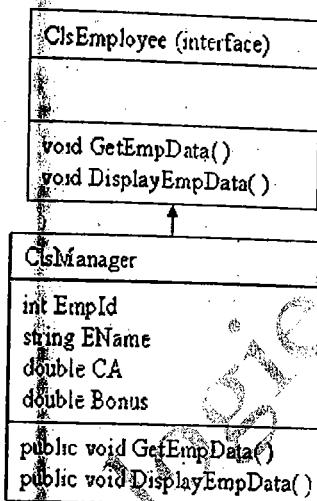
namespace CAInterface
{
    interface ClsEmployee
```

```

    {
        void GetEmpData();
        void DisplayEmpData();
    }
    class ClsManager : ClsEmployee
    {
        int EmpId; string EName;
        double Bonus, CA;
        public void GetEmpData()
        {
            Console.WriteLine("Enter Manager Details:- ");
            EmpId = Convert.ToInt32(Console.ReadLine());
            EName = Console.ReadLine();
            Bonus = Convert.ToDouble(Console.ReadLine());
            CA = Convert.ToDouble(Console.ReadLine());
        }
        public void DisplayEmpData()
        {
            Console.WriteLine("Manager id is:- " + EmpId);
            Console.WriteLine("Manager Name is:- " + EName);
            Console.WriteLine("Manager Bonus is:- " + Bonus);
            Console.WriteLine("Manager CA is:- " + CA);
        }
    }
    class ClsInterface
    {
        static void Main(string[] args)
        {
            //ClsEmployee Obj2 = new ClsEmployee();
            ClsManager Obj1 = new ClsManager();
            Obj1.GetEmpData();
            Obj1.DisplayEmpData();
            Console.ReadLine();
        }
    }
}

```

Class Diagram



Output:-

```

C:\file:///E:/MCA/C#Override/CAFOverride/bin/Debug/CAFOverride.exe
Enter Manager Details:- 101
SAI
20000
10000
Manager Id is:- 101
Manager Name is:- SAI
Manager Bonus is:- 20000
Manager CA is:- 10000

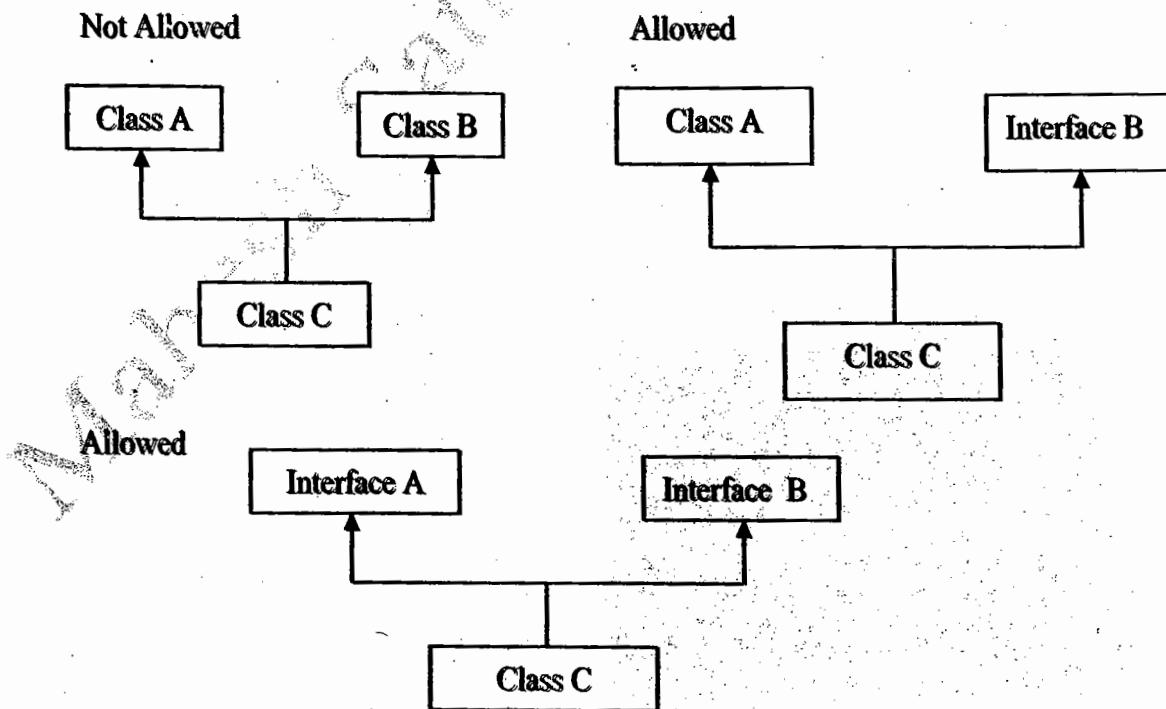
```

6.5.1 Comparisons between an Abstract class and an Interface

Srno	Abstract Class	Interface
01	A class Which contains one or more Abstract Functions is known as an Abstract Class	A class which contains all Abstract functions is known as an Interface
02	An Abstract Class can contain non Abstract functions	An Interface can not contain non Abstract functions
03	An Abstract Class can contain all members of class	An Interface can contain only Abstract functions, Properties, Indexers, Events and can not contain Non abstract functions, Data Fields, Constructors and Destructors.
04	Use abstract keyword to create an Abstract Class	Use interface keyword to create an Interface
05	An abstract class can not be used to Implement Multiple Inheritance	An Interface can be used to Implement Multiple Inheritance
06	By Default Abstract Class Members are Not Public and not Abstract	By Default Interface Members are Public and Abstract
07	An abstract class can not be instantiated directly	An Interface can not be instantiated directly
08	Creating a new class from an abstract class is must to Consume it	Creating a new class from an Interface is must to Consume it

6.5.2 Implementing Multiple Inheritance In C#.Net Using Interfaces:

- C#.Net doesn't support multiple inheritance, by using tow or more base classes, like.



- To implement multiple inheritance, there should be one or none base classes and should be one or more Interfaces

- When we implement multiple inheritance using a class and an interface then, first mention class name then interface name like...

```
class C : A,B //A is Class Name and B is Interface name
{
    Code..
}
```

- But do not write Like

```
class C : B,A //A is Class Name and B is Interface name
{
    Code..
}
```

This will raise compilation error.

Example 6.7 Program to implement Multiple Inheritance:-

Code:

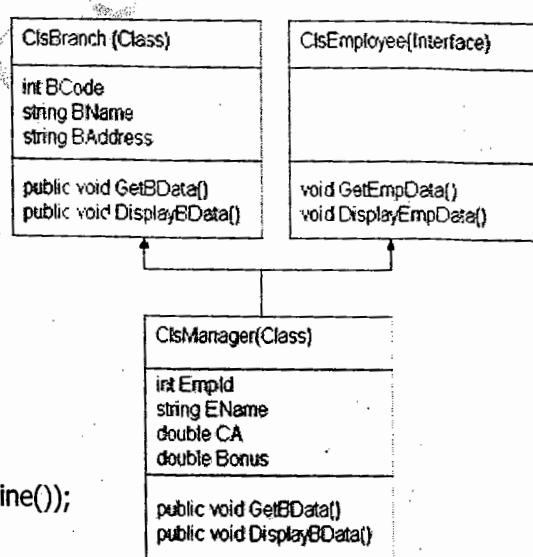
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAMPLInheritance
{
    class ClsBranch
    {
        int BCode; string BName, BAddress;
        public void GetBData()
        {
            Console.WriteLine("Enter Branch Details:- ");
            BCode = Convert.ToInt32(Console.ReadLine());
            BName = Console.ReadLine();
            BAddress = Console.ReadLine();
        }

        public void DisplayBData()
        {
            Console.WriteLine("Branch Code is:- " + BCode);
            Console.WriteLine("Branch Name is:- " + BName);
            Console.WriteLine("Branch Address is:- " + BAddress);
        }
    }

    interface ClsEmployee
    {
        void GetEmpData();
        void DisplayEmpData();
    }

    class ClsManager : ClsBranch, ClsEmployee
    {
        int EmpId; string EName;
```



```

        double Bonus, CA;
        public void GetEmpData()
        {
            Console.WriteLine("Enter Manager Details:- ");
            EmpId = Convert.ToInt32(Console.ReadLine());
            EName = Console.ReadLine();
            Bonus = Convert.ToDouble(Console.ReadLine());
            CA = Convert.ToDouble(Console.ReadLine());
        }
        public void DisplayEmpData()
        {
            Console.WriteLine("Manager Id is:- " + EmpId);
            Console.WriteLine("Manager Name is:- " + EName);
            Console.WriteLine("Manager Bonus is:- " + Bonus);
            Console.WriteLine("Manager CA is:- " + CA);
        }
    }
    class ClsMPLInheritance
    {
        static void Main(string[] args)
        {
            ClsManager Obj1 = new ClsManager();
            Obj1.GetBData();
            Obj1.GetEmpData();
            Obj1.DisplayBData();
            Obj1.DisplayEmpData();
            Console.Read();
        }
    }
}

```

Output:-

```

Branch Details:- 1001
SATHYA
HYDRAVAD
Enter Manager Details:- 101
SAI
20000
10000
Branch Code is:- 1001
Branch Name is:- SATHYA
Branch Address is:- HYDRAVAD
Manager Id is:- 101
Manager Name is:- SAI
Manager Bonus is:- 20000
Manager CA is:- 10000

```

- C#.NET doesn't support Multiple Inheritance using classes because:

If we consider two classes C1, C2 like...

class C1

{

 public void F1()

 {

 Code....

}

Exa

Coc

```
}
```



```
class C2
{
    public void F1()
    {
        Code....
    }
}
```

And create a new class C3 like

```
class C3:C1,C2
{
    Code....
}
```

Then create object for class C3 like

```
C3 obj1 = new C3();
```

And if we call function F1() like.,,

```
Obj1.F1();
```

Then runtime will get confusion to call function F1() of class C1 or C2

- This situation is known as Ambiguity situation, So C#.Net doesn't support multiple inheritance using 2 or more classes at base level.

Example 6.8

Program to implement Multiple Inheritance Using Two Interfaces at Base Level:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAInterface1
{
    interface I1
    {
        void F1();
    }

    interface I2
    {
        void F1();
    }

    class C3 : I1, I2
    {
        public void F1()
        {
            Console.WriteLine("This is Overriding Function of I1 and I2 Interfaces");
        }
    }

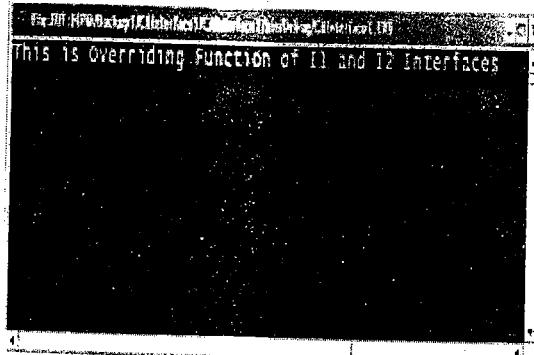
    class clsInterface1
    {
        static void Main(string[] args)
```

```

    {
        C3 Obj1 = new C3();
        Obj1.F1();
        Console.Read();
    }
}
}

```

Output:-



6.5.3 Explicit Interface Implementation:

- In the above example function F1() of class C3 is providing same implementation to the function F1() of I1 interface and also to the function F1() of I2 interface.
- But we want to provide different implementations to the function F1() inherited from I1 interface and to the function F1() inherited from I2 interface within the derived class C3. This is known as **Explicit Interface Implementation**.

Example 6.9

Program for Explicit Interface Implementation:-

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAInterface2
{
    interface I1
    {
        void F1();
    }
    interface I2
    {
        void F1();
    }
    class C3 : I1, I2
    {
        void I1.F1()
        {
            Console.WriteLine("This is Overriding Interface I1 function");
        }
    }
}

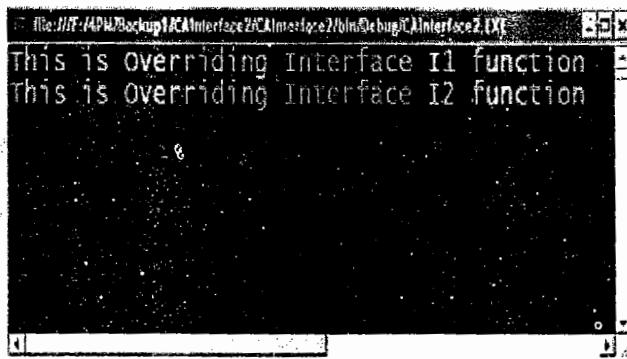
```

```

void I2.F1()
{
    Console.WriteLine("This is Overriding Interface I2 function");
}
}
class clsInterface2
{
    static void Main(string[] args)
    {
        C3 Obj1 = new C3();
        I1 Obj2 = (I1)Obj1; //Creating Object for I1 Interface
        I2 Obj3 = (I2)Obj1; //Creating Object for I2 Interface
        Obj2.F1();
        Obj3.F1();
        Console.Read();
    }
}

```

Output:-



Interview Questions

1. What are the features of Object Oriented Programming?
2. What is a Class Explain?
3. What is an Object Explain?
4. What syntax is used to create a Class?
5. What is the default accessibility of a class?
6. What syntax is used to create an Object for the class?
7. What happens when an Object to the class is created?
8. What is the Purpose of new keyword while creating object?
9. What members a class can contain?
10. What are Data Fields explain?
11. What is the default accessibility of the class members?
12. What is Abstraction Explain?
13. List out the types of Abstraction?
14. Explain what is Data Abstraction?
15. Explain what is Functional Abstraction?
16. Explain what is Encapsulation?
17. Explain what is Polymorphism?
18. List out the types of Polymorphism?
19. Explain what is static / Compile Time Polymorphism or Early Binding?

20. How do you achieve static polymorphism?
21. Explain about Function Over Loading?
22. Explain about Operator Over Loading?
23. What are the situations that a function can be over loaded?
24. What is function signature explain?
25. Explain what is Dynamic / Run Time Polymorphism or Late Binding?
26. How do you achieve Dynamic Polymorphism?
27. What is Function Overriding explain?
28. What is a Virtual Function explain?
29. What key word is used to make a function as Virtual?
30. What key word is used to override a virtual function?
31. Is overriding of a virtual function mandatory?
32. Differentiate between Function Over Loading and Function Overriding?
33. What is Inheritance explain?
34. List out the types of Inheritance?
35. What is Single Inheritance explain?
36. What is Multiple Inheritance explain?
37. What is Multi Level Inheritance explain?
38. What is Hybrid Inheritance explain?
39. How do you implement Inheritance in C#.NET explain?
40. What is "this" keyword explain?
41. What is "base" keyword explain?
42. What is the purpose of Member Access Operator (.)?
43. What is an Access Modifier explain?
44. List out the types of Access Modifiers?
45. Explain the difference among Private, Protected, Internal, ProtectedInternal and Public Access Modifiers?
46. What is an Abstract Function explain?
47. What key word is used to make a function as Abstract?
48. Is overriding of an abstract function mandatory?
49. Can an abstract function contain function body?
50. What is an Abstract Class explain?
51. What key word is used to make a Class as Abstract?
52. Can an abstract class contain Non – Abstract functions?
53. Can we instantiate an abstract class directly?
54. Is creating / Deriving a new class from an abstract class Mandatory?
55. What members an abstract class can contain?
56. Differentiate between a Virtual Function and an Abstract Function?
57. What is an Interface explain?
58. What key word is used to create an Interface?
59. What members an Interface can contain?
60. What members an Interface can not contain?
61. What is the purpose of an Interface explain?
62. By default how Interface members are treated?
63. Is it Compulsory to create a new class from an Interface?
64. Can an Interface be Instantiated Directly?
65. Differentiate and Compare between an Abstract class and Interface?
66. Explain about the Explicit Interface Implementation?
67. What happens if a Virtual Function is not overridden?

68. What happens if an Abstract Function is not overridden?
69. What happens if an Object is created to an Abstract Class?
70. What happens if an Object is created to an Interface?
71. Give the syntax to over load an operator?
72. Explain the situation where do you Implement Inheritance?
73. Whether C#.Net supports procedures?

Chapter 7 Properties

7.1 Introduction

- A property is a member of a class, used to write the data in the data field and read the data from the data field of a class.
- A property can never store the data, just used to transfer the data.
- Properties are members that provide a flexible mechanism to read, write, or compute the values of private fields.
- Properties enable a class to expose a public way of getting and setting values, while hiding implementation or verification code.
- To perform read and write operations, property can contain two assessors / methods.
- Properties provide the convenience of public data members without the risks that come with unprotected, uncontrolled, and unverified access to an object's data.
- This is accomplished through accessors: special methods that assign and retrieve values from the underlying data member.

7.2 Accessors with in the property

- Accessors enable data to be accessed easily while still providing the safety and flexibility of methods.
 - a) Set Accessor
 - b) Get Accessor

7.2.1 set Accessor

- Set Accessor is used to write the data into the data field.
- This will contain a default and fixed variable named as "value".
- Whenever we call the property to write the data, any data we supply will come and store in value variable by default.
- Syntax:

```
set
{
    Data File Name=value;
}
```

7.2.2 get Accessor

- Get Accessor is used to read the data from the data field, using this accessor we can't write the data.
- Syntax:

```
get
{
    return Data File Name;
}
```

7.3 Types of Properties

- C#.NET will support three types of properties.,
 - Read only Property
 - Write only Property
 - Read write Property

7.3.1 Read only Property:

- This property is used to read the data from the Data field.
- We can't write the Data into data field using this property.
- This property will contain only one accessor ie, get Accessor

Syntax:

```
Access Modifier Data Type Property Name  
{  
    set  
    {  
        Data Filed Name=value;  
    }  
}
```

7.3.2 Write only property:

- This is used to write the data into Data field of a class.
- Using this property, we can't read the data from the data field.
- This property will contain only one accessor ie set accessor.

Syntax :

```
Access Modifier Data Type Property Name  
{  
    get  
    {  
        return Data Filed Name;  
    }  
}
```

7.3.3 Read Write Property:

- This is used to read the data from the Data Field and to write the Data in to the Data Field.
- This property will contain two accessors ie get and set.

Syntax:

```
Access Modifier Data Type Property Name  
{  
    set  
    {  
        Data Filed Name=value;  
    }  
}
```

```

get
{
    return Data Filed Name;
}
}

```

- Whenever, we create property, the Data type of a property must and should be same as the Data type of the Data field into which we want to transfer the data.
- A property can never accept Arguments.
- Declaration of variables or writing the code out side the accessors with in the property is prohibited.

Example 7.1

Program for Read Write Property: -

Code:

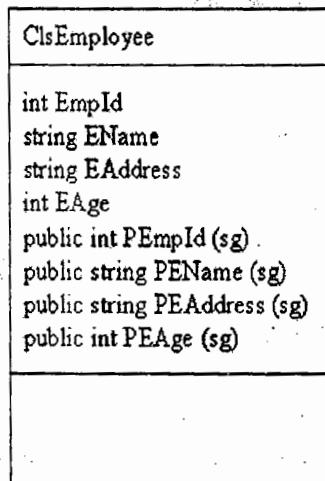
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAProperties
{
    class clsEmployee
    {
        int EmpId, EAge; string EName, EAddress;
        public int PEmpId
        {
            set
            {
                EmpId = value;
            }
            get
            {
                return EmpId;
            }
        }
        public string PENAME
        {
            set
            {
                EName= value;
            }
            get
            {
                return EName;
            }
        }
        public string PEAddress
        {
            set
            {
                EAddress= value;
            }
        }
    }
}

```

Class Diagram



Out

EN
SA
HY
26
EM
EM
EM

```

        }
        get
        {
            return EAddress;
        }
    }
    public int PEAge
    {
        set
        {
            EAge= value;
        }
        get
        {
            return EAge;
        }
    }
}
class ClsProperty1
{
    static void Main(string[] args)
    {
        clsEmployee Obj1 = new clsEmployee();
        Console.Write("Enter Employee Details:- ");
        Obj1.PEmpId = Convert.ToInt32(Console.ReadLine());
        Obj1.PENName = Console.ReadLine();
        Obj1.PEAddress = Console.ReadLine();
        Obj1.PEAge = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Employee Id is:- " + Obj1.PEmpId);
        Console.WriteLine("Employee Name is:- " + Obj1.PENName);
        Console.WriteLine("Employee EAddress is:- " + Obj1.PEAddress);
        Console.WriteLine("Employee EAge is:- " + Obj1.PEAge);
        Console.ReadLine();
    }
}

```

Output:-

```

file:///F:/TAM/CAProperties/CAProperties/bin/Debug/CAProperties.EXE
Enter Employee Details:- 101
SAI
HYDERABAD
26
Employee Id is:- 101
Employee Name is:- SAI
Employee EAddress is:- HYDERABAD
Employee EAge is:- 26

```

- In the above example Data fields of ClsEmployee are private (default). Though they are not accessible from Clsproperty1 class, we transferred the Data into Data fields with the help of

properties and properties code is hidden from Clsproperty1 class, but we are able to serve the result finally.

- As, we providing abstraction to the Data fields here, this is known as Data abstraction. So, properties will provide Data abstraction.

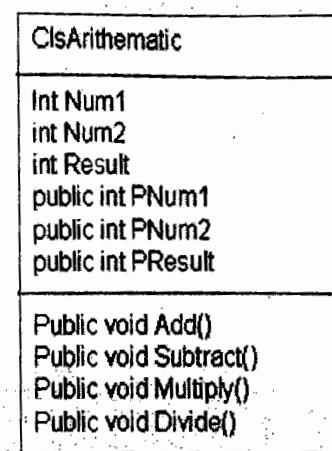
Example 7.2

Program for Read only and Write Only Properties: - Class Diagram

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAProperties
{
    class ClsArithematic
    {
        int Num1, Num2, Result;
        public int PNum1
        {
            set { Num1 = value; }
        }
        public int PNum2
        {
            set { Num2 = value; }
        }
        public int PResult
        {
            get { return Result; }
        }
        public void Add()
        {
            Result = Num1 + Num2;
        }
        public void Subtract()
        {
            Result = Num1 - Num2;
        }
        public void Multiply()
        {
            Result = Num1 * Num2;
        }
        public void Divide()
        {
            Result = Num1 / Num2;
        }
    }
    class ClsProperty2
    {
        static void Main()
        {
            ClsArithematic Obj1 = new ClsArithematic();
            Console.Write("Enter Any Two Numbers:- ");
            Obj1.PNum1 = Convert.ToInt32(Console.ReadLine());
            Obj1.PNum2 = Convert.ToInt32(Console.ReadLine());
            Obj1.Add();
            Console.WriteLine("Sum is:- " + Obj1.PResult);
            Obj1.Subtract();
        }
    }
}
```



```

        Console.WriteLine("Difference is:- " + Obj1.PResult);
        Obj1.Multiply();
        Console.WriteLine("Prouct is:- " + Obj1.PResult);
        Obj1.Divide();
        Console.WriteLine("Quotient is:- " + Obj1.PResult);
        Console.ReadLine();
    }
}
}

```

Output:-

```

file:///F:/TAM/CAProperties/CAProperties/bin/Debug/CAProperties.EXE
Enter Any Two Numbers:- 40
10
Sum is:- 50
Difference is:- 30
Prouct is:- 400
Quotient is:- 4

```

- In the above example, the Result Data field, which is calculated in ClsArithmatic class can't be modified from outside the class, because property PResult is read only, using which we can't write the data into data field.
- If there is no property we need to "set" our result Data field as public, then result field doesn't have any security from outside the class.

7.4 Advantages of Properties:

- Properties will provide abstraction to the Data fields.
- Properties will provide security to the Data fields.
- Properties can be used to validate the data before allowing a change.

7.5 Symmetric and Asymmetric Accessors:

- By default Accessibility of Assessors is same as the Accessibility of the property.

Ex: public int PEmpId

```

{
    set
    {
        EmpId = value;
    }
    get
    {
        return EmpId;
    }
}

```

- In the above property, property PEmpId is declared as public, so set and get will be public.
- If property is private, then set and get will be private.

- **Symmetric Assessors:** If the accessibility of the Accessors are same, then accessors are known as Symmetric Accessors.
- **Asymmetric Assessors:** If the Accessibility of the Accessors is not same, then Assessors are known as Assymmetric.

Example:

```
public int PEmpId
{
    protected set
    {
        EmpId = value;
    }
    get
    {
        Return EmpId;
    }
}
```

- In the above example set is protected and get is public, so they are known as **Asymmetric**.
- In general Asymmetric accessors are used in Inheritance process.
- We can also write the Read Only Property using two accessors like:

```
public int PEmpId
{
    set
    {
        EmpId = value;
    }
    private get
    {
        Return EmpId;
    }
}
```

- We can also write the Write Only Property using two accessors like:

```
public int PEmpId
{
    private set
    {
        EmpId = value;
    }
    get
    {
        Return EmpId;
    }
}
```

Interview Questions

1. What is a property explain?
2. What is an accessor explain?
3. How many accessors can a property contain?
4. Explain the purpose of set accessor?
5. Explain the purpose of get accessor?
6. What is the syntax of set accessor?
7. What is the syntax of get accessor?
8. What is the default and fixed variable that set accessor contains?

8.

➤➤➤➤➤➤➤➤➤➤

Ex

Co

9. What should be the Data Type of the property?
10. List out the Types of Properties?
11. Explain about Read Only Property?
12. What accessor does Read Only Property contain?
13. Explain about Write Only Property?
14. What accessor does Write Only Property contain?
15. Explain about Read Write Property?
16. What accessors does Read Write Property contain?
17. What is the syntax of Read Only Property?
18. What is the syntax of Write Only Property?
19. What is the syntax of Read Write Property?
20. Can a Property store the data?
21. Can a Property accept arguments?
22. List out the types of accessors?
23. What is the default accessibility of the accessors?
24. What are Symmetric Accessors explain?
25. What are Asymmetric Accessors explain?
26. What is the purpose of property?
27. What are the advantages of properties? (or) Why Properties are used?
28. Explain how do you provide Data Abstraction using Properties?
29. Explain how do you provide security to the Data Fields using Properties?
30. Explain how do you validate the data using Properties?
31. Can a property be overridden?
32. How do you make a property as Read only by writing both the accessors?
33. How do you make a property as Write only by writing both the accessors?

Maheshwari

Chapter 8

Sealed Classes and Partial Classes

8.1 Sealed Class

- A class from which it is not possible to Create / Derive a new class is known as **Sealed Class**
- To make any class as sealed we use sealed keyword.
- A sealed class is completely opposite to an abstract class.
- A sealed class can't contain abstract functions.
- A sealed class can not contain virtual functions.
- A sealed class should be the bottom most class with in the inheritance hierarchy.
- A sealed class can never be used as a base class.
- A sealed class is especially used to avoid further Inheritance.
- The sealed keyword can be used with classes, instance methods and properties.

Example 8.1

Program for Sealed Class: -

Class Diagram

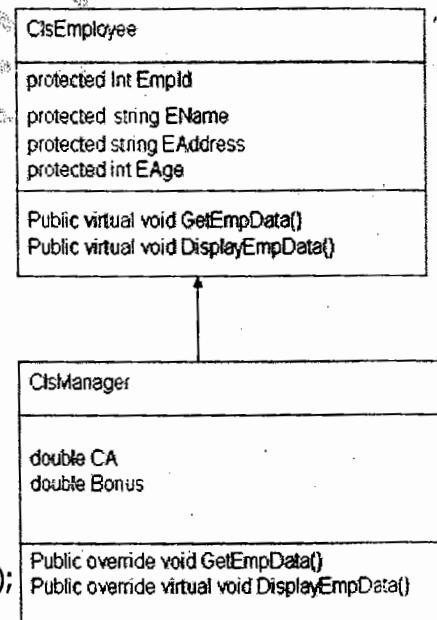
Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CASealed
{
    class ClsEmployee
    {
        protected int EmpId; int EAge;
        protected string EName; string EAddress;
        public virtual void GetEmpData()
        {
            Console.WriteLine("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }

        public virtual void DisplayEmpData()
        {
            Console.WriteLine("Employee Id is:- " + this.EmpId);
            Console.WriteLine("Employee Name is:- " + this.EName);
            Console.WriteLine("Employee Address is:- " + this.EAddress);
            Console.WriteLine("Employee Age is:- " + this.EAge);
        }
    }

    sealed class ClsManager : ClsEmployee
    {
        double Bonus, CA;
        public override void GetEmpData()
        {
            Console.WriteLine("Enter Manager Details:- ");
            EmpId = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

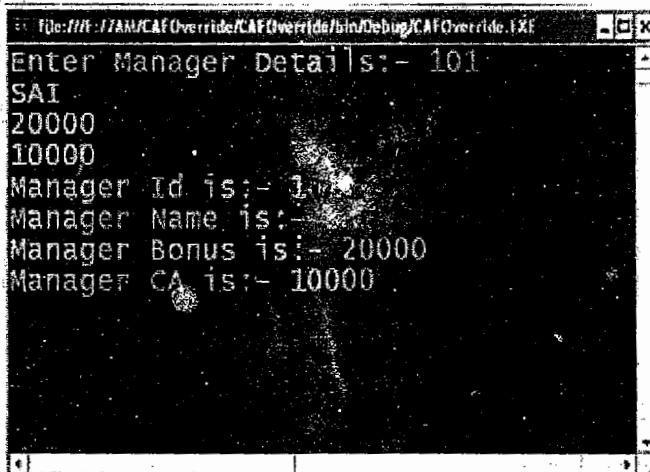


```

        EName = Console.ReadLine();
        Bonus = Convert.ToDouble(Console.ReadLine());
        CA = Convert.ToDouble(Console.ReadLine());
    }
    public override void DisplayEmpData()
    {
        Console.WriteLine("Manager Id is:- " + EmpId);
        Console.WriteLine("Manager Name is:- " + EName);
        Console.WriteLine("Manager Bonus is:- " + Bonus);
        Console.WriteLine("Manager CA is:- " + CA);
    }
}
class ClsSealed
{
    static void Main(string[] args)
    {
        ClsManager Obj1 = new ClsManager();
        Obj1.GetEmpData();
        Obj1.DisplayEmpData();
        Console.Read();
    }
}

```

Output:-



(Following points are from MSDN as it is)

8.1.1 Do not seal classes without having a good reason to do so.

- Do not assume that because you do not see a scenario in which extending a class would be desirable, that it is appropriate to seal the class. You should seal classes that meet certain conditions:
 - The class is static.
 - The class contains inherited protected members with security-sensitive information.
 - The class inherits many virtual members and the development and testing costs for sealing each member are significantly more costly than sealing the entire class.
 - The class is an attribute that requires fast searching using reflection. Sealing an attribute improves reflection's performance when retrieving attributes.

8.1.2 Do not declare protected or virtual members on sealed types.

- If a type is sealed, it cannot have derived classes. Protected members can be accessed only from a derived class and virtual members can be overridden only in a derived class.

8.1.3 Differences between an Abstract class and a sealed class:

Srno	Abstract Class	Sealed Class
01	A class which contains one or more abstract functions is known as an abstract class	A class from which it is not possible to derive a new class is known as a sealed class
02	Abstract class can contain Non-Abstract and abstract functions	Sealed class can contain Non-Abstract functions, can not contain abstract / virtual functions
03	Creating a new class from an abstract class is compulsory to consume it	It is not possible to create a new class from sealed class
04	An abstract class can not be instantiated directly, we need to create object for derived class to consume an abstract class	We should create an object for sealed class only to consume it
05	Use abstract keyword to make any class as abstract	Use sealed keyword to make any class as sealed
06	An abstract class can not be the bottom most class with in the inheritance hierarchy	Sealed class should be the bottom most class with in the inheritance hierarchy
07	An abstract class should be used as base class only	A sealed class can not be used as base class

8.2 Partial Class

- A Class which code can be written in two or more files is known as Partial Class.
- To make any class as partial use partial keyword.
- Partial classes will provide flexibility in Application Development.
- It is possible to split the definition of a class or a struct, or an interface over two or more source files. Each source file contains a section of the class definition, and all parts are combined when the application is compiled. There are several situations when splitting a class definition is desirable:
 - When working on large projects, spreading a class over separate files allows multiple programmers to work on it simultaneously.
 - When working with automatically generated source, code can be added to the class without having to recreate the source file. Visual Studio uses this approach when creating Windows Forms, Web Service wrapper code, and so on. You can create code that uses these classes without having to edit the file created by Visual Studio.

Example 8.2

Program for Partial Class: -

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAPartial
{
    partial class ClsEmployee
    {
        int EmpId, EAge; string EName, EAddress;
        public void GetEmpData()
        {
            Console.Write("Enter Employee Details:- ");
            this.EmpId = Convert.ToInt32(Console.ReadLine());
            this.EName = Console.ReadLine();
            this.EAddress = Console.ReadLine();
            this.EAge = Convert.ToInt32(Console.ReadLine());
        }
    }
    class ClsPartial
    {
        static void Main(string[] args)
        {
            ClsEmployee Obj1 = new ClsEmployee();
            Obj1.GetEmpData();
            Obj1.DisplayEmpData();
            Console.Read();
        }
    }
}
```

Create a new class with the name ClsSample and write the following code in that

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAPartial
{
 class ClsSample
 {
 }
 partial class ClsEmployee
 {
 public void DisplayEmpData()
 {
 Console.WriteLine("Employee Id is:- " + EmpId);
 Console.WriteLine("Employee Name is:- " + this.EName);
 Console.WriteLine("Employee Address is:- " + this.EAddress);
 Console.WriteLine("Employee Age is:- " + this.EAge);
 }
 }
}

Output:-

```
file:///C:/AM/CAPartial/CAPartial/bin/Debug/CAPartial.exe
Enter Employee Details:- 1001
SAI
HYDERABAD
26
Employee Id is:- 1001
Employee Name is:- SAI
Employee Address is:- HYDERABAD
Employee Age is:- 26
```

Chapter 9

POINTERS

- Pointer is a variable which stores address of another variable.
- The code that has been written using pointers is known as Unsafe code.
- Unsafe code means CLR doesn't provide any security for the code.
- Unsafe code should be written using "unsafe" keyword, within the unsafe block (or) unsafe function (or) unsafe class.
- Unsafe code must and should be compiled using unsafe option only.
- To declare any pointer variable we use following
- **Syntax:** Data Type* Variable Name;
- **Examples:**
 - int* x → x is a ptr variable stores address of an Integer variable.
 - String* x → x is a ptr variable stores address of an string variable.
 - Double* x → x is a ptr variable stores address of a double variable
 - Void* x → x is a ptr variable stores address of an unknown variable.
- Difference between reference types and pointers is, though reference types are maintained by pointers only CLR will provide complete security for the reference types. Whereas for pointers CLR doesn't provide any security.

Example 9.1

Program to work with Pointers: -

Code:

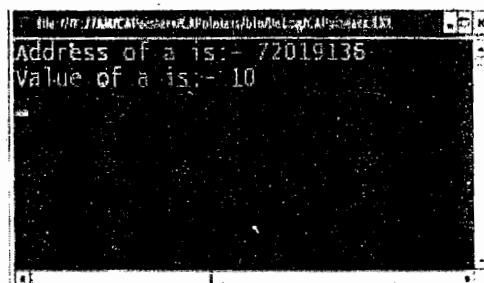
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAPointers
{
    class ClsPointer1
    {
        static void Main(string[] args)
        {
            int a = 10;
            unsafe
            {
                int* x = &a;
                Console.WriteLine("Address of a is:- " + (int)x);
                Console.WriteLine("Value of a is:- " + *x);
            }
            Console.Read();
        }
    }
}
```

- To Compile and Run this code use the following steps:
 - Go to solution explorer.
 - Double click on properties.
 - Click on build.
 - Activate allow unsafe code option.

- Save and close the properties.
- Run the application.

Output:-



Example 9.2

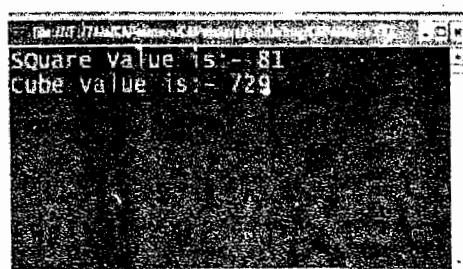
Program to work with Pointers:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAPointers
{
    unsafe class ClsPointer2
    {
        static void Square(int* x)
        {
            Console.WriteLine("Square Value is:- " + (*x * *x));
        }
        static void Cube(int* x)
        {
            Console.WriteLine("Cube Value is:- " + (*x * *x * *x));
        }
        static void Main()
        {
            int a=9;
            Square(&a);
            Cube(&a);
            Console.Read();
        }
    }
}
```

Output:-



Chapter 10 GENERICS

E

- These are known as General Data types.
- Using generics, we will write generic functions and generic classes.
- Generics are introduced in .NET framework version 2.0.
- Namespaces related to Generics are System.Collections.Generic, System.Collections.ObjectModel, these namespaces contain generic collection classes that will allow working with Generic Types.
- Generic interfaces that will provide sorting and equality comparisons are present in System namespace.
- Generics can be implemented with classes, structures, interfaces, methods, Events and Delegates.
- The data types used in Generics can be obtained with the help of Reflection at runtime.
- Generics will maximize Code Reusability, Type Safety and Performance.
- Generics are used to avoid function overloading, when Data types of arguments are changed.
- To work with generics, we use two things,
 - Place holder represented by <>
 - Type parameter
- Always type parameter should be enclosed within the place holder, like <type parameter>.
-

C

Non - Generic Method

```
public Display(string S)
{
    Code...
}
```

Calling: Display("Welcome");

Generic Method

```
public Display<G>(string S)
{
    Code...
}
```

Calling: Display<string>("Welcome");
Calling: Display<int>(10);
Calling: Display<double>(10.5);

Ou

Passing Multiple Data Types to a Generic Function:

- In the above examples, we passed only single data type to the generic function.
- If at all we want to pass multiple data types to a generic function at once, we declare the function like...

```
public void display <TP1, TP2>(TP1a, TP2b)
{
    Code....
}
```

Calling:

Display <int, int> (10, 20);
Display <int, string> (10, "Sai")
Display < string, string> ("Welcome", "Hello");

Example 10.1

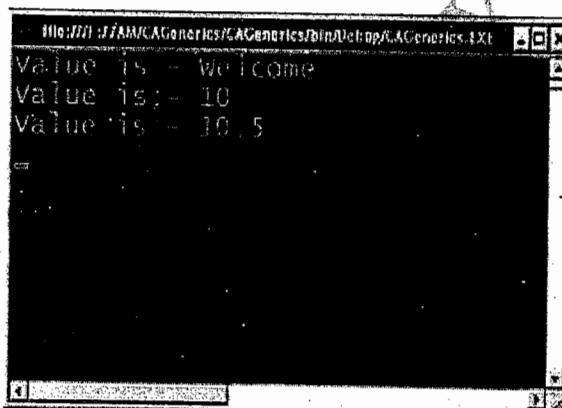
Program for Generic Method:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAGenerics
{
    class ClsGeneric1
    {
        static void Display<G>(G S)
        {
            Console.WriteLine("Value is:- " + S);
        }
        static void Main(string[] args)
        {
            Display<string>("Welcome");
            Display<int>(10);
            Display<double>(10.5);
            Console.Read();
        }
    }
}
```

Output:-



Example 10.2

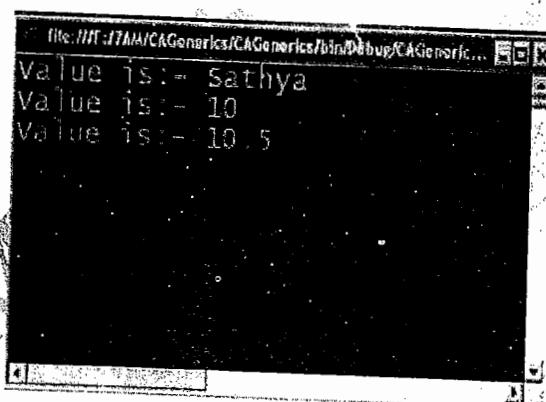
Program for Generic Class:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAGenerics
{
    class ClsSample<TP>
    {
        public void Display(TP S)
        {
            Console.WriteLine("Value is:- " + S);
        }
    }
    class ClsGeneric2
    {
        static void Main()
        {
            ClsSample<string> ObjS = new ClsSample<string>();
            ClsSample<int> ObjI = new ClsSample<int>();
            ClsSample<double> ObjD = new ClsSample<double>();
            ObjS.Display("Sathya");
            ObjI.Display(10);
            ObjD.Display(10.5);
            Console.Read();
        }
    }
}
```

Output:-



Chapter 11 DELEGATES

11.1 Introduction

- It's used to represent or refer one or more functions.
- Delegates are user defined types in C#.NET.
- Delegates in C#.NET are similar to function pointers in C++.
- It's not a member of a class, but similar to a class.
- To consume any delegate, we need to create an object to delegate.
- A delegate is a type that references a method.
- Once a delegate is assigned a method, it behaves exactly like that method.
- The delegate method can be invoked like any other method with parameters and written value.
- These can be used to define callback methods.
- These are the backbone for events.
- These can be chained together, ie, multiple methods can be called on a single event.

11.2 Types Of Delegates

- There are of two types od Delegates available
 - a) Single cast delegate b) Multi cast delegate
- A delegate that represents only a single function is known as **Single Cast Delegate**.
- A delegate that represents only a more than one functions is known as **Multi Cast Delegate**.
- Delegates will support Generics.
- In C#.NET 2.0 Version, a new feature of Delegate is introduced ie Anomous Delegate.
- Delegates are obj.oriented, types are and secure.

11.3 To Work With Delegate Use The Following Steps:

- Creating a Delegate
- Instantiating a Delegate,
- Invoking a Delegate
- Ex: If we consider a function, like..,

```
public void Add (int a, int b)
{
    Code.
}
```
- To refer this function, if we want to use the Delegate, we use the above steps like...
- **STEP1: Creating A Delegate**
 - **Syntax:** Access modifier delegate return type Delgete name([arguments list]);
 - **Ex:** public delegate void SampleDelegate(int x, int y);
 - When we create a delegate, Access modifier, return type, number of arguments and their data types of the delegate must and should be same as Access modifier, return type, number of arguments and their data types of the function that we want to refer.
- **STEP2: Instantiating the Delegate**
 - **Syntax:** Delegate name object name = new Delegate name(Target function name);
 - **Ex:** SampleDelegate obj = new SampleDelegate(Add);

- At this step a reference will be maintained from the delegate object to the function that we want to refer.
- **STEP3: Invoking the Delegate**
- **Syntax:** Object Name([Arguments Values])
 - **Ex:** Obj(10, 20).
 - At this step the function that is referred by the delegate will be called for the execution.

Example 11.1

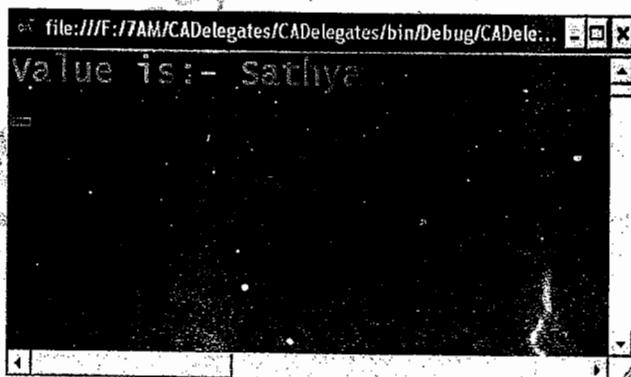
Program for Single Cast Delegate To Refer a Function in Same Class:-

Code:-

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CADelegates
{
    class ClsDelegate1
    {
        static void Display(string S)
        {
            Console.WriteLine("Value is:- " + S);
        }
        delegate void X(string a);
        static void Main(string[] args)
        {
            X ObjD = new X(Display);
            ObjD("Sathya");
            Console.Read();
        }
    }
}
```

Output:-



that

Example 11.2

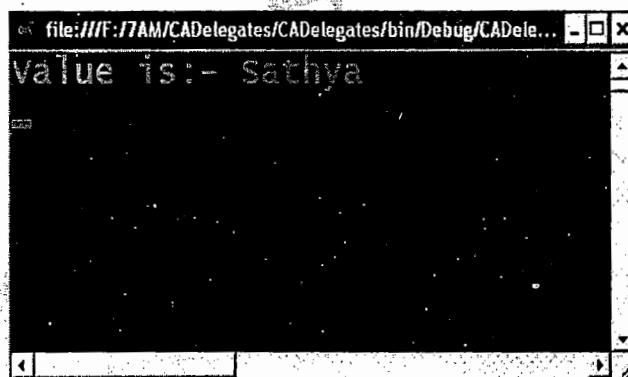
Program for Single Cast Delegate To Refer a Function in Different Class: -

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CADelegates
{
    class ClsSample
    {
        public void Display(string S)
        {
            Console.WriteLine("Value is:- " + S);
        }
    }
    public delegate void SampleDelegate(string a);
    class ClsDelegate2
    {
        static void Main()
        {
            ClsSample Obj1=new ClsSample();
            SampleDelegate ObjD = new SampleDelegate(Obj1.Display);
            ObjD("Sathya");
            Console.Read();
        }
    }
}
```

Output:-



Example 11.3

Program for Multi Cast Delegate: -

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CADelegates
{
    class ClsArithematic
    {
        public void Add(int x, int y)
        {
            Console.WriteLine("Sum is:- " + (x + y));
        }
        public void Subtract(int x, int y)
        {
            Console.WriteLine("Difference is:- " + (x - y));
        }
        public void Multiply(int x, int y)
        {
            Console.WriteLine("Product is:- " + (x * y));
        }
        public void Divide(int x, int y)
        {
            Console.WriteLine("Quotient is:- " + (x / y));
        }
    }
    public delegate void MCDelegate(int a,int b);
    class ClsMCDelegate
    {
        static void Main()
        {
            ClsArithematic Obj1 = new ClsArithematic();
            MCDelegate ObjD = new MCDelegate(Obj1.Multiply);
            ObjD += Obj1.Add;
            ObjD += Obj1.Subtract;
            ObjD += Obj1.Divide;
            ObjD(40, 10);
            ObjD -= Obj1.Add;
            ObjD -= Obj1.Divide;
            ObjD(50, 10);
            Console.Read();
        }
    }
}
```

Output:-

```
Product is:- 400
Sum is:- 50
Difference is:- 30
Quotient is:- 4
Product is:- 500
Difference is:- 40
```

- To add reference of more functions to a delegate object, use += operator like...,
objD+=obj1.subtract;
objD+=obj1.Add;
- To delete the reference of any function from Delegate object use -= operator like...,
objD-=Obj1.subtract;
objD-=Obj1.Add;

Chapter 12

Exception Handling Mechanism

12.1 Introduction

- When we write and execute in our code in .NET, there is possibility of three types of error occurrence.
 - Syntactical errors
 - Compilation errors
 - Runtime errors

12.1.1 Syntactical errors:

- These errors occur by typing wrong syntax like, missing double quotes, terminators, typing wrong spelling for keywords etc.
- Programmer can identify these errors, while writing the code and can be rectified.
- These errors do not cause any harm to the program execution.

12.1.2 Compilation errors:

- These errors occur when the program is compiled.
- These errors are like, assigning wrong data to a variable, trying to create object for abstract class or Interface etc.
- These errors can be identified by the programmer and can be rectified, before execution of the program only.
- These errors do not cause any harm to program execution.

12.1.3 Run time errors:

- These errors will occur at the time of executing the program.
- These errors are like.. entering wrong data into a variable trying to open a file for which there is no permission, trying to connect to database with wrong user id and password etc.

12.1.4 Exception

- A runtime error is known as an Exception.
- Exception can't be identified and rectified by the programmer.
- Exceptions will cause abnormal termination of the program execution.
- So, to avoid abnormal termination of program execution, we need to handle the exceptions.

12.2 Methods to Handle Exception:

- There are three methods to handle the except.
 - Logical Implementation
 - Try catch Implementation
 - On error go to implementation.

12.2.1 Logical Implementation

- In this method we handle the exceptions by using logical statements.
- In real time programming first and fore most importance should be given for logical implementation only.
- If it's not possible to handle any exception using logical implementation then we use , try catch implementation.

Example 12.1

Program to handle an Exception using Logical Implementation: -

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAExceptions
{
    class ClsException1
    {
        static void Main(string[] args)
        {
            int a, b, c;
            Console.Write("Enter Any Two Numbers:- ");
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            if (b == 0)
            {
                Console.WriteLine("Second Number Can not be Zero");
            }
            else
            {
                c = a / b;
                Console.WriteLine("Quotient value is:- " + c);
            }
            Console.Read();
        }
    }
}
```

Output:-

```
file:///F:/7AM/CAExceptions/CAExceptions/bin/Debug/CAExceptions.EXE
Enter Any Two Numbers:- 40
0
Second Number Can not be zero.
```

- In the above example, when user enters second number as zero, exception will be raised, this is handle using logical implementation.

12.2.2 Try Catch Implementation:

- Syntax :

```
try
{
    Code...
}
catch[Exception Class object)]
{
    Code...
}
:
:
finally
{
    Code...
}
```

- A try block can be followed by any number of catch blocks, writing finally block is optional.
- Writing an exception class with in the catch block is optional.
- If catch block is used with out an exception class then it is known as **generic catch block**.
- If catch block is used with an exception class then it is known as **specific catch block**.
- **Try:** This block contains all the statements in which there is possibility of exception occurrence.
- **Catch:** This block contains all the statements to handle the exception that is raised with in the try block.
- **Finally:** This contains the statements to be executed compulsory, though try block is executed or catch block is executed.
- **Execution:**
 - Execution starts from try block, if there any exception occurs in any statement of try block then following lines of exception statement are ignored and control jumps to catch block, catch block is executed and then finally block.

- If no exception occurs in any statement of try block then all statements of try block are executed and catch block is ignored then finally block is executed.
- We can also write Nested try, catches like...,

```
try
{
    try
    {
        Statements
    }
    catch[(ExceptionClass Object)]
    {
        Statements
    }
    finally
    {
        Statements
    }
}
catch[(ExceptionClass Object)]
{
    try
    {
        Statements
    }
    catch[(ExceptionClass Object)]
    {
        Statements
    }
    finally
    {
        Statements
    }
}
```

Example 12.2

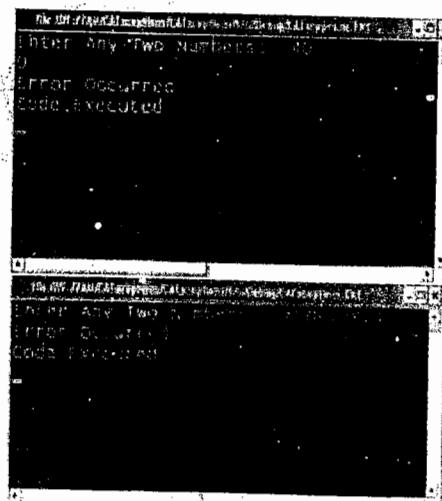
Program to handle an Exception using try – catch Implementation with generic catch block:-

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAExceptions
{
    class ClsException2
    {
        static void Main(string[] args)
        {
            int a, b, c;
            Console.WriteLine("Enter Any Two Numbers:- ");
            try
            {
                a = Convert.ToInt32(Console.ReadLine());
                b = Convert.ToInt32(Console.ReadLine());
                c = a / b;
                Console.WriteLine("Quotient value is:- " + c);
            }
            catch
            {
                Console.WriteLine("Error Occurred");
            }
            finally
            {
                Console.WriteLine("Code Executed");
            }
            Console.Read();
        }
    }
}
```

Output:-



12.

12.2

Exa

Cod

- In the above example there is no exception class used in catch block, so it is known as generic catch block.
- But in the above example any kind of exception may occur same message will be displayed to the user and user can not understand why error has occurred, to over come this specific catch blocks are used.
- Using specific catch blocks it is possible to know more information about the exceptions.

12.2.3 Some important exception classes

- ArgumentException
- ArgumentNullException
- ArgumentOutOfRangeException
- ArithmeticException
- ArrayTypeMismatchException
- BadImageFormatException
- DivideByZeroException
- DllNotFoundException
- EntryPointNotFoundException
- Exception (Super class of all other exception classes)
- FormatException
- IndexOutOfRangeException
- InsufficientMemoryException
- InvalidCastException
- MemberAccessException
- MethodAccessException
- NullReferenceException
- UnauthorizedAccessException

12.2.4 Properties With Exception Classes:

- **Message:** This property will store about the reason, why exception has occurred.
- **Source:** This stores name of the application from which exception has been raised.
- **HelpLink:** This is used to provide link to any file / URL to give help information to the user, when exception is raised.

Example 12.3

Program to handle an Exception using try - catch Implementation with specific catch block:-

Code:

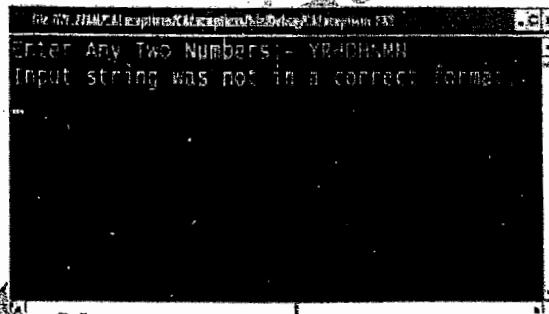
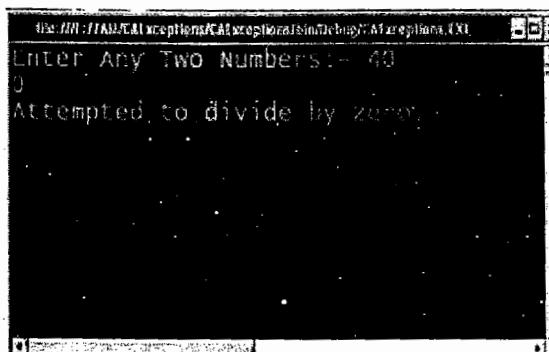
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAExceptions
{
    class ClsException3
```

```

static void Main(string[] args)
{
    int a, b, c;
    Console.WriteLine("Enter Any Two Numbers:- ");
    try
    {
        a = Convert.ToInt32(Console.ReadLine());
        b = Convert.ToInt32(Console.ReadLine());
        c = a / b;
        Console.WriteLine("Quotient value is:- " + c);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    Console.Read();
}
}

```

Output:-**Out**

- In the above example, super class exception is used to handle the exceptions, But, if we use super class of any exception class when there is any relevant class is available, it will kill the execution performance of the program.
- So any time don't use super class of any exception class to handle any exception when there is relevant exception class is available.

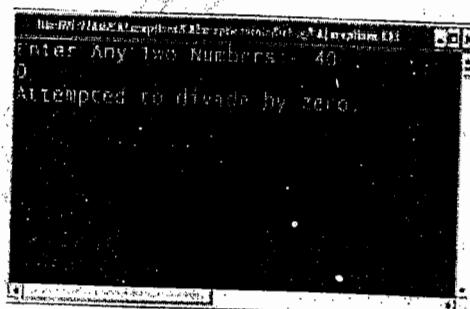
Example 12.4

Program with multiple catch blocks:-
Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CAExceptions
{
    class ClsException4
    {
        static void Main(string[] args)
        {
            int a, b, c;
            Console.Write("Enter Any Two Numbers:- ");
            try
            {
                a = Convert.ToInt32(Console.ReadLine());
                b = Convert.ToInt32(Console.ReadLine());
                c = a / b;
                Console.WriteLine("Quotient value is:- " + c);
            }
            catch (DivideByZeroException ex)
            {
                Console.WriteLine(ex.Message);
            }
            catch (FormatException ex)
            {
                Console.WriteLine(ex.Message);
            }
            Console.Read();
        }
    }
}
```

Output:-



- Whenever we write multiple catch blocks, at any point of time only one catch block will be executed and other catch block(s) will be ignored.
- Whenever multiple catch blocks are used, it is not possible to write the catch blocks in the following way and raises compilation error because first catch block Exception class only handles all the exceptions.

```

try
{
    Code...
}
catch (Exception ex)
{
    Code...
}
catch (DivideByZeroException ex)
{
    Code...
}

```

De

13.



FAQs or IQs on Exception Handling

1. List out the types of errors those occur during the program development and Execution?
2. Explain about Syntactical Errors?
3. Explain about Logical Errors?
4. Explain about Runtime Errors?
5. What is an Exception?
6. What happens when an Exception is raised?
7. Why to handle an Exception?
8. List out in what ways an Exception can be handled?
9. Explain about how do you handle an Exception using Logical implementation?
10. Explain about try – catch implementation?
11. What is the syntax of try – catch?
12. Explain the execution process of try – catch implementation?
13. What statements can be written in try block?
14. What is the significance of catch block?
15. What is the significance of finally block?
16. Can a catch block be written without an Exception class?
17. What is the purpose of Exception class in catch block?
18. List out the properties of an Exception class?
19. List out some important Exception classes and explain their purpose?
20. What is Structured Exception Handling explain?
21. Explain how Structured Exception Handling does not kill the execution performance?
22. What is an Unstructured Exception Handling explain?
23. What happens if we write any exception class catch block followed by its Super Class of exception class catch block?
24. How many catch blocks can be followed by a try block?
25. Explain about nested try – catches implementation?

Exa

I

Step

Click

→ se

Type

→ cl

Char

Writ

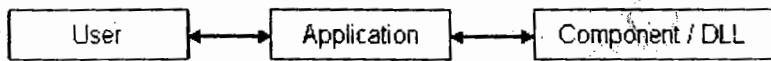
Code

Chapter 13

Designing .NET components or Assemblies or User Defined Class Libraries or DLLS

13.1 What is a Component

- A component is reusable piece of code and is in the form of DLL.
- Once component is designed it can be reused from any kind of application like in console application or windows application or web application or device application etc.
- Once a component is designed in any programming language of .Net that can be reused from any other programming languages of .Net i.e. .Net components are or assemblies will provide language interoperability.
- To create computer in .NET, we use class library templates.
- End user will never interact with DLL or component directly.
- All ways end user will interact with some application and the application will interact with component or DLL.



Example 13.1

Program for Creating a Component / Assembly:-

Steps to Create an Assembly:

Click on file → new → project select visual C# →

→ select class library template →

Type the class library or assembly name (CLibMaths)

→ click on ok.

Change the class name to ClsArithmetric →

Write the following code

Code:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
namespace CLibMaths
```

```
{
```

```
    public class ClsArithmetric
```

```
{
```

```
    int Num1, Num2, Result;
```

```
    public int PNum1
```

```
{
```

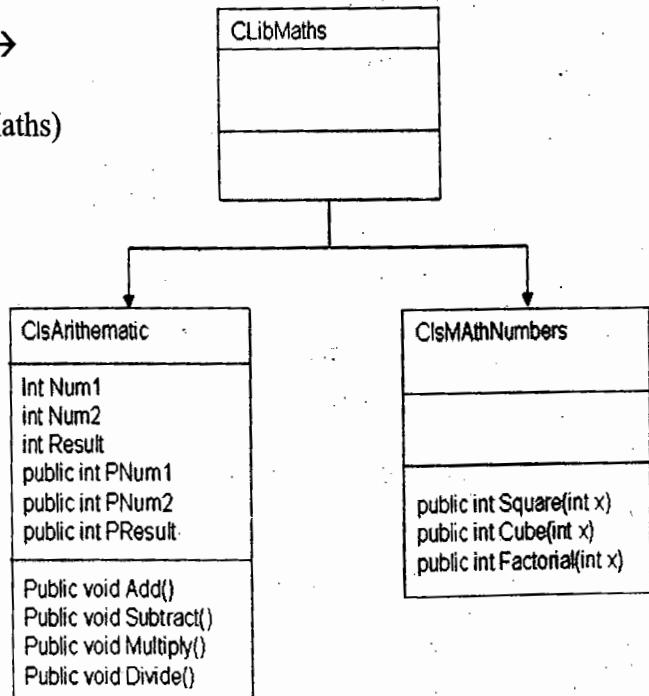
```
        set { Num1 = value; }
```

```
}
```

```
    public int PNum2
```

```
{
```

Class Diagram



```
    set { Num2 = value; }
}
public int PResult
{
    get {return Result; }
}
public void Add()
{
    Result = Num1 + Num2;
}
public void Subtract()
{
    Result = Num1 - Num2;
}
public void Multiply()
{
    Result = Num1 * Num2;
}
public void Divide()
{
    Result = Num1 / Num2;
}
```

13.2

- Add a new class with name ClsMathNumbers.
- Change the accessibility of the class to public and write following code.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
namespace CLibMaths
{
    public class ClsMathNumbers
    {
        public int Square(int x)
        {
            return x * x;
        }
        public int Cube(int x)
        {
            return x * Square(x);
        }
        public int Factorial(int x)
        {
            int R = 1;
            for (int i = 1; i <= x; i++)
            {
                R = R * i;
            }
            return R;
        }
    }
}
```

Code

- Build the solution, this will create a DLL. This dll is known as .NET component.
- To see the dll go the \bin\Debug folder of the application, yo find the CLibMaths.dll

13.2 Consuming the Component / Assembly as Private:

- Create a console Application with the name CACheckCLibMathPrivate.
- Change class name to ClsSample.
- To consume any DLL / Component first we need to add reference to the DLL.
- **Steps to Add the Reference:**
 - Go to solution Explorer.
 - Select the solution
 - Click with right mouse button
 - Click on Add Reference.
 - Click on Browse
 - Go to the location, where dll is saved
 - Select the CLibMaths.dll and click on OK
 - Write the following code.

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using CLibMaths;

namespace CACheckCLibPrivate
{
    class ClsSample
    {
        static void Main(string[] args)
        {
            ClsArithematic Obj1 = new ClsArithematic();
            ClsMathNumbers Obj2 = new ClsMathNumbers();
            Console.WriteLine("Enter Any two Numbers:- ");
            Obj1.PNum1 = Convert.ToInt32(Console.ReadLine());
            Obj1.PNum2 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Any Number:- ");
            int a = Convert.ToInt32(Console.ReadLine());
            Obj1.Add();
            Console.WriteLine("Sum is:- " + Obj1.PResult);
            Obj1.Subtract();
            Console.WriteLine("Difference is:- " + Obj1.PResult);
            Obj1.Multiply();
            Console.WriteLine("Product is:- " + Obj1.PResult);
            Obj1.Divide();
            Console.WriteLine("Quotient is:- " + Obj1.PResult);
            Console.WriteLine("Square Value is:- " + Obj2.Square(a));
            Console.WriteLine("Cube Value is:- " + Obj2.Cube(a));
            Console.WriteLine("Factorial Value is:- " + Obj2.Factorial(a));
            Console.ReadLine();
        }
    }
}

```

Output:-

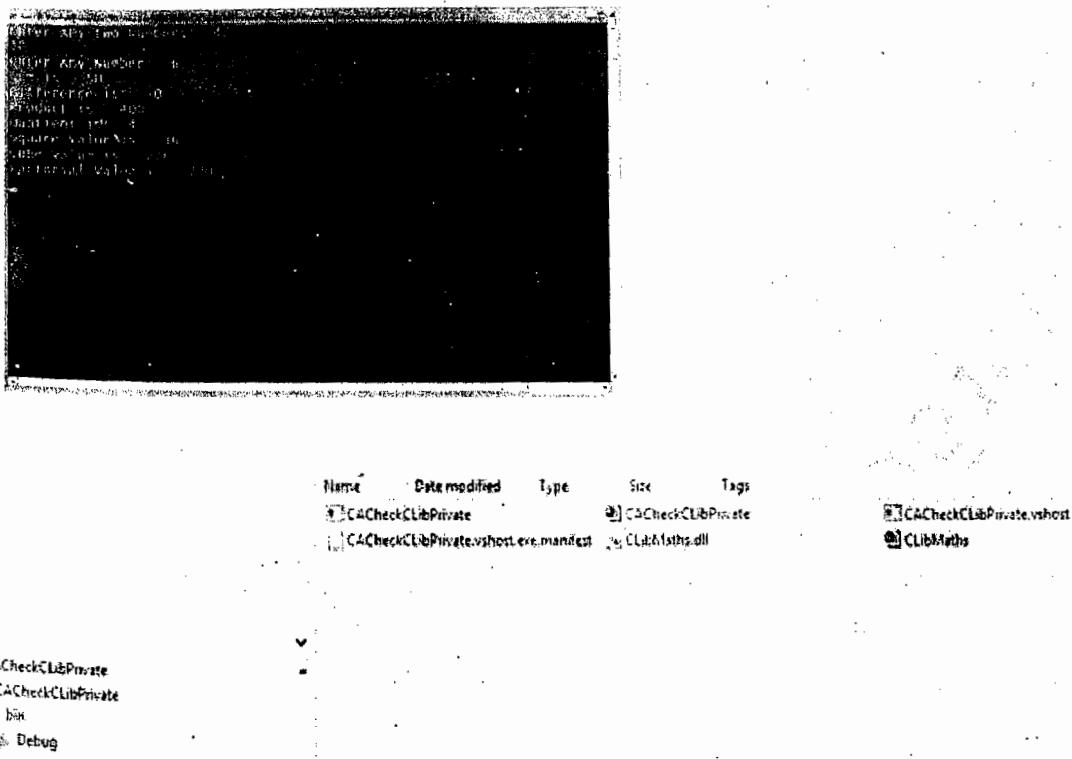


Fig 13.1

13.3 Making Assembly as Public

- To make any Assembly as public, use following steps:
 - Build the strong name.
 - Signing of Assembly
 - Copy the Assembly into GAC.

13.3.1 Building Strong Name:

- To build the strong name, we use a toll sn.exe.
 - To build the strong name, use following steps:
 - Click on start → Click on programs → Click on Microsoft Visual studio 2008 → Click on Visual Studio Tools → Click on Visual Studio 2008 Command prompt.
 - Change the disk location to the CLibMaths.dll location.
 - Type Sn.exe -k CLibKey.snk
 - This will generate the key pair and wil' write into CLibKey.snk file.

13.3.2 Signing of the Assembly:

- Signing of assembly means writing key file path into the Assembly. To perform this use the following steps :
 - Open CLibMaths application in VS.NET
 - Go to Solution Explorer → click on ‘+’ symbol of properties.

- Double click on AssemblyInfo.cs file and add the following attribute to mention the key path. Ie.,
- [assembly:AssemblyKeyFile("D:\\CLibMaths\\CLibMaths\\bin\\Debug\\CLibKey.snk")] //Assuming that application is saved in D:\\
- Build the application.

13.3.3 Copying the Assembly into GAC.

- To Copy the Assembly into GAC use the following steps
 - Go to Visual Studio.Net command prompt.
 - Change the disk location to CLibMaths.dll location
 - Type the following code.
 - GACUTIL.exe -i CLibMaths.dll
- Message appears that “assembly successfully added to the cache”.

Consuming Public Assembly

- Create a console application with a name CACheckCLibPublic.
- Change the class name to ClsSample.
- Add Reference to CLibMaths.dll (Note: Don't add the reference from GAC, rather add the reference from the original disk location of the dll).
- Write the following code.

Code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using CLibMaths;

namespace CACheckCLibPrivate
{
    class ClsSample
    {
        static void Main(string[] args)
        {
            CLibMaths.Obj Obj1 = new CLibMaths.Obj();
            CLibMaths.Obj Obj2 = new CLibMaths.Obj();
            Console.WriteLine("Enter Any two Numbers:-");
            Obj1.PNum1 = Convert.ToInt32(Console.ReadLine());
            Obj1.PNum2 = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Enter Any Number:-");
            int a = Convert.ToInt32(Console.ReadLine());
            Obj1.Add();
            Console.WriteLine("Sum is:- " + Obj1.PResult);
            Obj1.Subtract();
            Console.WriteLine("Difference is:- " + Obj1.PResult);
            Obj1.Multiply();
            Console.WriteLine("Product is:- " + Obj1.PResult);
            Obj1.Divide();
            Console.WriteLine("Quotient is:- " + Obj1.PResult);
            Console.WriteLine("Square Value is:- " + Obj1.Square(a));
        }
    }
}

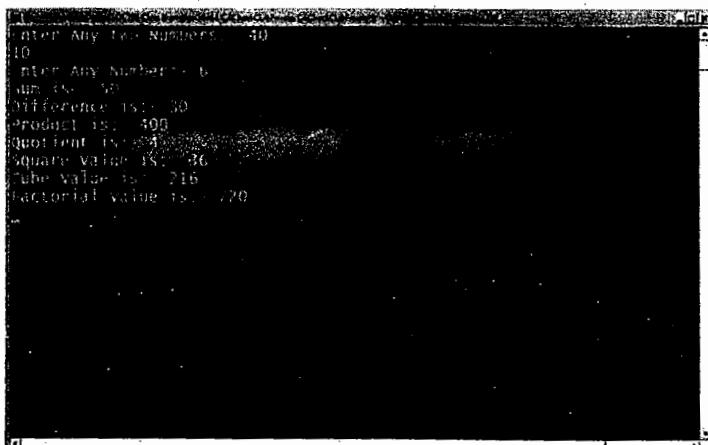
```

```

        Console.WriteLine("Cube Value is:- " + Obj2.Cube(a));
        Console.WriteLine("Factorial Value is:- " + Obj2.Factorial(a));
        Console.ReadLine();
    }
}
}

```

Output:-



```

Enter Any Two Numbers: 10
Enter Any Number: 6
sum is: 16
difference is: 4
product is: 60
quotient is: 1.6666666666666667
square value is: 36
cube value is: 216
factorial value is: 720

```

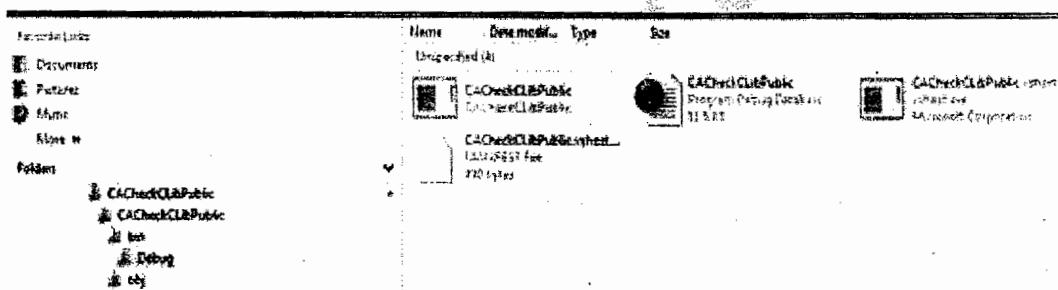


Fig 13.2

- There is no difference in writing code in private and public assemblies.
- In case of private assemblies the dll will be copied in to the application bin\Debug folder (observe the Fig 12.1), And also where ever the application exe is copied the assembly dll also should be copied.
- In case of public assemblies the dll will not be copied in to the application bin\Debug folder (observe the Fig 12.2), And also where ever the application exe is copied the assembly dll is not required be copied rather reference will be taken from the GAC.

Consuming the Assembly / Component from ASP.NET:

Chapter 14

Windows Forms Applications

14.1 Introduction

- Windows Forms Applications will provide complete GUI facilities.
- In Windows Forms Applications we can work with all controls like Text Box, Button, Radio Button, List Box etc.
- Using Windows Forms Applications we can design very good, easy and user friendly applications.

14.2 Creating A Windows Forms Application:

Click on File → Click on new → Click on project → Select Visual C# → Select Windows Forms Application Template → Type the application name → Choose the location to save → Click on ok.

14.2.1 IDE (Integrated Development Environment):

- This contains various components like,
 - Solution Explorer.
 - Tool Box.
 - Server Explorer
 - Properties Window
 - Output window
 - Error List Window etc.

14.2.2 Form

- A form is a container, which can hold all other controls with in it.
- In C#.NET every form is treated as a class and is derived from System.Windows.Forms.Form class
- In C#.NET Windows Form we can place various controls like Text Box, Button, List Box, Radio Button, Check Box etc.
- In C#.NET Windows Forms Applications every control has a special class and is present in System.Windows.Forms namespace observe the following table.

Srno	Control Name	Class Name
01	TextBox	System.Windows.Forms.TextBox
02	Label	System.Windows.Forms.Label
03	Button	System.Windows.Forms.Button
04	RadioButton	System.Windows.Forms.RadioButton
05	CheckBox	System.Windows.Forms.CheckBox
06	GroupBox	System.Windows.Forms.GroupBox
07	Panel	System.Windows.Forms.Panel
08	TabControl	System.Windows.Forms.TabControl
09	OpenFileDialog	System.Windows.Forms.OpenFileDialog
10	SaveFileDialog	System.Windows.Forms.SaveFileDialog

14.3
➤ t

14.3

Srn
01
02

03

04

11	FontDialog	System.Windows.Forms.FontDialog
12	ColorDialog	System.Windows.Forms.ColorDialog
13	PageSetupDialog	System.Windows.Forms.PageSetupDialog
14	PrintDialog	System.Windows.Forms.PrintDialog
15	Timer	System.Windows.Forms.Timer
16	ImageList	System.Windows.Forms.ImageList
17	PictureBox	System.Windows.Forms.PictureBox
18	ToolTip	System.Windows.Forms.ToolTip
19	NumericUpDown	System.Windows.Forms.NumericUpDown
20	Calender	System.Windows.Forms.Calender
21	DateTimePicker	System.Windows.Forms.DateTimePicker
22	NotifyIcon	System.Windows.Forms.NotifyIcon
23	ListBox	System.Windows.Forms.ListBox
24	ComboBox	System.Windows.Forms.ComboBox
25	ListView	System.Windows.Forms.ListView
26	TreeView	System.Windows.Forms.TreeView
27	RichTextBox	System.Windows.Forms.RichTextBox
28	LinkLabel	System.Windows.Forms.LinkLabel
29	ProgresBar	System.Windows.Forms.ProgressBar
30	CheckedListBox	System.Windows.Forms.CheckedListBox
31	MenuStrip	System.Windows.Forms.MenuStrip
32	ContextMenuStrip	System.Windows.Forms.ContextMenuStrip

14.3 TextBox Control

- This control is used to accept the data from the user and also to display the data dynamically to the user

14.3.1 Properties With TextBox Class:

Srno	Property and options	Type	Description
01	BackColor	struct	Used to set or get Back ground color of the control
02	BorderStyle None Single Fixed3D(Default)	enum	None: No Border appears to the Textbox Single: Single Line Border appears to the Textbox Fixed3D: 3D Border appears to the Textbox
03	CharacterCasing Normal(Default) Lower Upper	enum	Normal: Data in the Textbox appears in the case it is typed. Lower: Data in the Textbox appears in lower case letters Upper: Data in the Textbox appears in upper case letters
04	Dock Top Left Bottom Right Fill	enum	Top: Control is arranged on the top border of the form Left: Control is arranged on the left border of the form Bottom: Control is arranged on the bottom border of the form Right: Control is arranged on the right border of the form

	None(Default)		Fill: Control is arranged with in the complete form None: Control can be placed anywhere in the form	19
05	Enabled True(Default) False	bool	True: Control can be accessed at run time False: Control can not be accessed at run time	
06	Font		Used to set or get font attributes like Size, Style, Name etc. of the control	20
07	ForeColor	enum	Used to set or get foreground ground color of the control	
08	Location X (Left) Y (Left)	struct	Used to set or get the distance of the control from the top and left borders of the form	
09	MaxLength	int	Used to set or get the maximum number of characters that user can be allowed to enter in the Textbox, Default value is 32767, When set to 0 user can be allowed to enter 1GB characters for single line textbox and 4GB characters for multi line textbox	
10	Modifiers Private(Default) Protected Internal ProtectedInternal Public		Used to set or get the required accessibility for the control. (Refer the table in Inheritance chapter for the detailed description of the Modifiers)	
11	MultiLine True False(Default)	bool	True: Data in the textbox appears in more than one lines False: Data in the textbox appears one line	
12	PasswordChar	char	Used to set or get an alternate character to be appeared in textbox like password field. It is possible to set any character from the keyboard or Space. But 2 or more characters are not allowed. PasswordChar will work when MultiLine is false only.	
13	ReadOnly True False(Default)	bool	True: Data in the textbox is editable False: Data in the textbox is not editable	
14	ScrollBars None(Default) Horizontal Vertical Both	enum	None: No scroll bar appears to the control Horizontal: Horizontal scroll bar appears to the control Vertical: Vertical scroll bar appears to the control Both: Both scroll bar appears to the control	
15	Size Height Width	struct	Used to set or get height and width of the control	
16	TabIndex		Determines index value of the control in the TAB order	
17	Text	int	Used to set or get the data in to the textbox control	
18	TextAlign Left(Default) Right		Left: Data in the Textbox appears with left Alignment Right: Data in the Textbox appears with right Alignment	

	Center		Center: Data in the Textbox appears with center Alignment
19	Visible True(Default) False	bool	True: Control appears at design time and also at run time False: Control appears at design time but not at run time
20	WordWrap True(Default) False	bool	True: Data is wrapped automatically to the next line for multi line textbox False: Data is not wrapped automatically to the next line for multi line textbox

In the above properties most of the properties are same for all the other controls and few properties are different, so for other controls the different properties are discussed.

14.3.2 Events

- An event is a member of a class, used to perform required action.
- Event is similar to function but is pointed by a Delegate or function pointer.
- Every event will have particular period of time that is to be called known as Event raising/Firing.
- Delegates are completely backbone for the events.

Events Associated with TextBox Class

Srno	Name	Description
01	<u>Click</u>	Occurs when the text box is clicked.
02	<u>DoubleClick</u>	Occurs when the control is double-clicked.
03	<u>Enter</u>	Occurs when the control is entered.
04	<u>FontChanged</u>	Occurs when the <u>Font</u> property value changes.
05	<u>ForeColorChanged</u>	Occurs when the <u>ForeColor</u> property value changes.
06	<u>Invalidated</u>	Occurs when a control's display requires redrawing.
07	<u>KeyDown</u>	Occurs when a key is pressed while the control has focus.
08	<u>KeyPress</u>	Occurs when a key is pressed while the control has focus.
09	<u>KeyUp</u>	Occurs when a key is released while the control has focus.
10	<u>Leave</u>	Occurs when the input focus leaves the control.
11	<u>LocationChanged</u>	Occurs when the <u>Location</u> property value has changed.

12	<u>MouseClick</u>	Occurs when the control is clicked by the mouse.
13	<u>MouseDoubleClick</u>	Occurs when the control is double clicked by the mouse.
14	<u>MouseDown</u>	Occurs when the mouse pointer is over the control and a mouse button is pressed.
15	<u>MouseEnter</u>	Occurs when the mouse pointer enters the control.
16	<u>MouseHover</u>	Occurs when the mouse pointer rests on the control.
17	<u>MouseLeave</u>	Occurs when the mouse pointer leaves the control.
18	<u>MouseMove</u>	Occurs when the mouse pointer is moved over the control.
19	<u>MouseUp</u>	Occurs when the mouse pointer is over the control and a mouse button is released.
20	<u>MouseWheel</u>	Occurs when the mouse wheel moves while the control has focus.
21	<u>Move</u>	Occurs when the control is moved.
22	<u>TextAlignChanged</u>	Occurs when the value of the <u> TextAlign</u> property has changed.
23	<u>TextChanged</u>	Occurs when the <u> Text</u> property value changes.
24	<u>Validated</u>	Occurs when the control is finished validating.
25	<u>Validating</u>	Occurs when the control is validating.

Example 14.1

Program with TextBox: -

Purpose: To get the Idea about, how events will work.

What to do:

1) When focus leaves from TextBox1

Action: Display “welcome” in TextBox4 and Change the back ground color of textbox3.

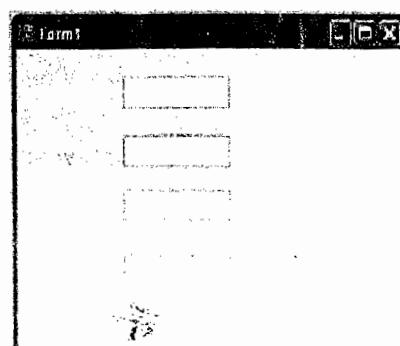
2) When user clicks on TextBox1

Action: Change the Back Ground color of Form.

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

Design of Form



```

using System.Text;
using System.Windows.Forms;

namespace WABASICS
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void textBox1_Leave(object sender, EventArgs e)
        {
            textBox4.Text = "Welcome";
            textBox3.BackColor = Color.Blue;
        }

        private void textBox1_Click(object sender, EventArgs e)
        {
            this.BackColor = Color.Green;
        }
    }
}

```

Example 14.2

Program with TextBox, Label and Button Controls :-

Purpose:

- 1) To understand real time naming conventions
 - 2) To understand and implement functional or field level validations.
- In general in real time programming programmer is not supposed to assign the names of the controls rather Project Leader / System Analyst will design the Forms, Reports and the controls to be present in those, What names to be given to the Controls with in the Design Document.
 - Programmers should use the same Names given in the Design Documents.
 - In general control name can be divided in to two parts:
 - Control Identification
 - Purpose of the control

Control Identification

TextBox	→ txt
Button	→ btn / cmd
Label	→ lbl
ListBox	→ lst
Combobox	→ cmb
RadioButton	→ opt
CheckBox	→ chk

Purpose of the Control

For Accepting Ac No in Bank Applications → Acno

For Accepting Ac Holder Name in Bank Applications → ACHName

Examples of Some Sample Control Names are Like

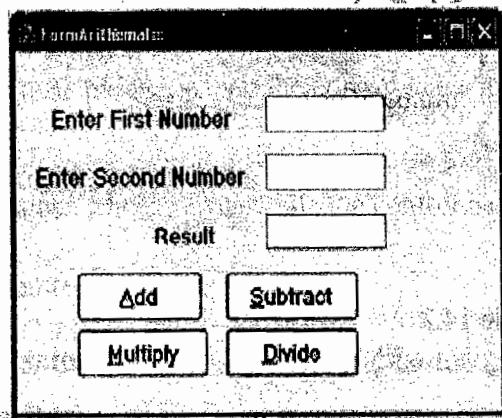
txtAcno
txtACHName
txtPinNum
btnAccept
btnSearch

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WABASICS
{
    public partial class FormArithematic : Form
    {
        int a, b, c;
        public FormArithematic()
        {
            InitializeComponent();
        }
        private void btnAdd_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a + b;
            txtResult.Text = c.ToString();
        }
        private void btnSubtract_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a - b;
            txtResult.Text = c.ToString();
        }
        private void btnMultiply_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a * b;
            txtResult.Text = c.ToString();
        }
        private void btnDivide_Click(object sender, EventArgs e)
```

Design of Form



```

    {
        a = Convert.ToInt32(txtNum1.Text);
        b = Convert.ToInt32(txtNum2.Text);
        c = a / b;
        txtResult.Text = c.ToString();
    }
}

```

14.4 Validations

- In the above Example code will work fine with out any problem but there are some bugs available to be solved these are called **Functional / Field Level Validations**.

14.4.1 List of Validations to be Implemented:

1. With txtNum1
 - User should be allowed to enter only digits
 - Entering data into txtNum1 is mandatory.
 2. With txtNum2
 - User should be allowed to enter only digits.
 - Entering date into txtNum2 is mandatory.
 3. With btnDivide:
 - txtNum2 Data should not be zero.
- To identify whether user typing Digit or Not suitable Event is KeyPress Event.
- **Key press event:**

```

private void txtNum1_Leave(object sender, EventArgs e)
{
}

```

- This Event has two arguments
 - object sender
 - EvenArgs e
- **object sender**
 - object is the data type and sender is variable for object type, stores the source of the Event
- **KeyPressEventArgs e**
 - KeyPressEventArgs is the class and e is an Object for this class.
 - This class has two important properties.
 - KeyChar
 - Handled
 - **KeyChar:** This property stores the character value that is typed by the user with in the textbox control.
 - **Handled:** This is a boolean property.
When set to **true** typed character doesn't appear within textbox control, **False:** Typed character will appear within the textbox control.

14.5 Char Structure

Co

- This structure provides various functions that allow the programmer to work with single character.

14.5.1 Functions in Char Structure

- `Char.IsLetter("Character")` → Returns true if the character is an alphabet otherwise returns false
 - `Char.IsLetter("A")` → true
 - `Char.IsLetter("@")` → false
 - `Char.IsLetter("9")` → false
- `Char.IsDigit("Character")` → Returns true if the character is a Digit otherwise returns false
 - `Char.IsDigit("A")` → false
 - `Char.IsDigit("@")` → false
 - `Char.IsDigit("9")` → true
- `Char.IsSymbol("Character")` → Returns true if the character is a special Symbol otherwise returns false
 - `Char.IsSymbol("A")` → false
 - `Char.IsSymbol("4")` → false
 - `Char.IsSymbol("@")` → true
- `Char.IsLetterOrDigit("Character")` → Returns true if the character is an alphabet or digit otherwise returns false
 - `Char.IsLetterOrDigit("A")` → true
 - `Char.IsLetterOrDigit("9")` → true
 - `Char.IsLetterOrDigit("@")` → false
- `Char.ToLower("Character")` → Returns true if the character is in Lower case letter otherwise returns false
 - `Char.ToLower("A")` → false
 - `Char.ToLower("a")` → true
 - `Char.ToLower("9")` → false
- `Char.ToUpper("Character")` → Returns true if the character is in Upper case letter otherwise returns false
 - `Char.ToUpper("A")` → true
 - `Char.ToUpper("a")` → false
 - `Char.ToUpper("9")` → false
- `Char.ToLower("Character")` → Converts the given character is in to Lower case letter
 - `Char.ToLower("A")` → a
 - `Char.ToLower("a")` → a
- `Char.ToUpper("Character")` → Converts the given character is in to Upper case letter
 - `Char.ToUpper("A")` → A
 - `Char.ToUpper("a")` → A
- Using the above functions we write the code for the validations mentioned in the Example 14.2

Code with Validations for the Example 14.2

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WABASICS
{
    public partial class FormArithematic : Form
    {
        int a, b, c;
        public FormArithematic()
        {
            InitializeComponent();
        }
        private void btnAdd_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a + b;
            txtResult.Text = c.ToString();
        }
        private void btnSubtract_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a - b;
            txtResult.Text = c.ToString();
        }
        private void btnMultiply_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            c = a * b;
            txtResult.Text = c.ToString();
        }
        private void btnDivide_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(txtNum1.Text);
            b = Convert.ToInt32(txtNum2.Text);
            if (b == 0)
            {
                MessageBox.Show("Second Number Can not be Zero");
                txtNum2.Focus();
            }
            else
            {
                c = a / b;
                txtResult.Text = c.ToString();
            }
        }
    }
}
```

```
private void txtNum1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (Char.IsDigit(e.KeyChar) == false)
    {
        MessageBox.Show("Enter Digits Only");
        e.Handled = true;
    }
}

private void txtNum1_Leave(object sender, EventArgs e)
{
    if (txtNum1.Text == "")
    {
        MessageBox.Show("Enter Any Data");
        txtNum1.Focus();
    }
}

private void txtNum2_KeyPress(object sender, KeyPressEventArgs e)
{
    if (Char.IsDigit(e.KeyChar) == false)
    {
        MessageBox.Show("Enter Digits Only");
        e.Handled = true;
    }
}

private void txtNum2_Leave(object sender, EventArgs e)
{
    if (txtNum2.Text == "")
    {
        MessageBox.Show("Enter Any Data");
        txtNum2.Focus();
    }
}
```

14.0

14.6

Sr n

01

02

03

Even

> C

14.6 RadioButton Control

- This control is used to provide selection of only one option from the given group of options. And the name is given like if take Radio we can tune only one channel at once in the same way it is possible to select only one RadioButton from the group.
- The RadioButton control can display text, an Image, or both.

14.6.1 Additional Properties With RadioButton Class:

Srno	Property and options	Type	Description
01	AutoSize True False(Default)	bool	True: Control is resized according to contents False: Control can not be resized according to contents
02	CheckAlign TopLeft TopCenter TopRight MiddleLeft MiddleCenter MiddleRight BottomLeft BottomCenter BottomRight	enum	TopLeft: Content is vertically aligned at the top, and horizontally aligned on the left. TopCenter: Content is vertically aligned at the top, and horizontally aligned on the center. TopRight: Content is vertically aligned at the top, and horizontally aligned on the right. MiddleLeft: Content is vertically aligned at the middle, and horizontally aligned on the left. MiddleCenter: Content is vertically aligned at the middle, and horizontally aligned on the center. MiddleRight: Content is vertically aligned at the middle, and horizontally aligned on the right. BottomLeft: Content is vertically aligned at the bottom, and horizontally aligned on the left. BottomCenter: Content is vertically aligned at the bottom, and horizontally aligned on the center. BottomRight: Content is vertically aligned at the bottom, and horizontally aligned on the right.
03	Checked True False(Default)	bool	True: Radio Button is Activated / Selected False: Radio Button is not Activated / Selected

Events Associated with RadioButton Class

- CheckedChanged is the default Event of the RadioButton.
 - This Event Will be fired when radio button is selected or deselected.

Example 14.3

Program with RadioButton: -

Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WABASICS
{
    public partial class FormRadio : Form
    {
        public FormRadio()
        {
            InitializeComponent();
        }

        private void optRed_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Red;
        }

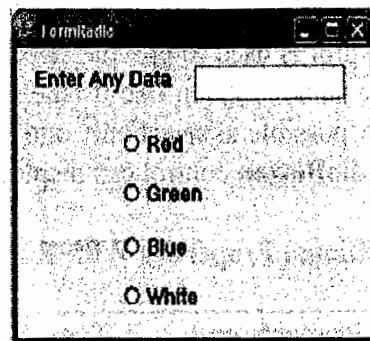
        private void optGreen_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Green;
        }

        private void optBlue_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Blue;
        }

        private void optWhite_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.White;
        }
    }
}

```

Design of Form



14.7 Grouping Controls:

In the above example user can be able to select only one radio button but if user would like to select more than one radio button then we use grouping controls.

14.7.1 Types of Grouping Controls:

- Groupbox
- Panel
- SplitContainer
- Tab Control

14.7.2 Differences between Groupbox and Panel

SRNO	Group Box	Panel
01	Has Text Area	Has no Text Area
02	Does not provide scrolling facilities	Provides scrolling facilities (set AutoScroll→true)
03	Occupies more design space	Occupies less design space
04	Occupies less memory	Occupies more memory

Example 14.4

Program with GroupBox and Panel: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WABASICS
{
    public partial class FormGroup : Form
    {
        public FormGroup()
        {
            InitializeComponent();
        }

        private void optRed_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Red;
        }

        private void optGreen_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Green;
        }

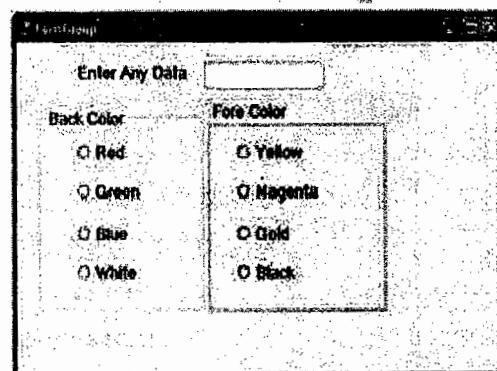
        private void optBlue_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Blue;
        }

        private void optWhite_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.White;
        }

        private void optYellow_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.ForeColor = Color.Yellow;
        }

        private void optMagenta_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.ForeColor = Color.Magenta;
        }
    }
}
```

Design of Form



```
{  
    txtSample.ForeColor = Color.Magenta;  
}  
private void optGold_CheckedChanged(object sender, EventArgs e)  
{  
    txtSample.ForeColor = Color.Gold;  
}  
private void optBlack_CheckedChanged(object sender, EventArgs e)  
{  
    txtSample.ForeColor = Color.Black;  
}  
}  
}
```

1.

>

>

St

01

Ex

De

Cod

+ - L L L L L

14.7.2 TabControl

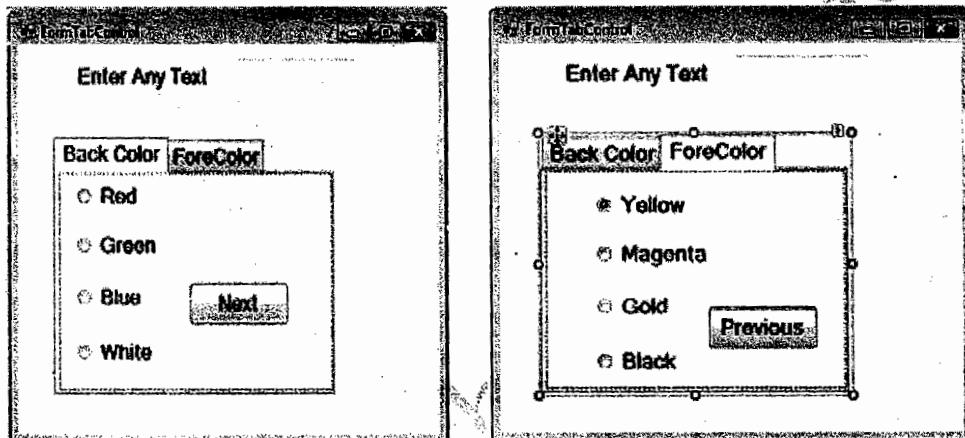
- This control is used to create required tab pages so that each tab page can contain required number of controls to group together.
- Properties With TabControl Class:

Srno	Property and options	Type	Description
01	TabPages	Collection	Used to set or get the tab pages with in it in the form of collection

Example 14.5

Program with TabControl: -

Design of Form



Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormTabControl : Form
    {
        public FormTabControl()
        {
            InitializeComponent();
        }
        private void optRed_CheckedChanged(object sender, EventArgs e)
        {
            txtSample.BackColor = Color.Red;
        }
        private void optGreen_CheckedChanged(object sender, EventArgs e)
        {
```

```

txtSample.BackColor = Color.Green;
}
private void optBlue_CheckedChanged(object sender, EventArgs e)
{
    txtSample.BackColor = Color.Blue;
}
private void optWhite_CheckedChanged(object sender, EventArgs e)
{
    txtSample.BackColor = Color.White;
}
private void optYellow_CheckedChanged(object sender, EventArgs e)
{
    txtSample.ForeColor = Color.Yellow;
}
private void optMagenta_CheckedChanged(object sender, EventArgs e)
{
    txtSample.ForeColor = Color.Magenta;
}
private void optGold_CheckedChanged(object sender, EventArgs e)
{
    txtSample.ForeColor = Color.Gold;
}
private void optBlack_CheckedChanged(object sender, EventArgs e)
{
    txtSample.ForeColor = Color.Black;
}
private void btnNext_Click(object sender, EventArgs e)
{
    tabControl1.SelectedIndex = tabControl1.SelectedIndex + 1;
}
private void btnPrevious_Click(object sender, EventArgs e)
{
    tabControl1.SelectedIndex = tabControl1.SelectedIndex - 1;
}
}

```

14.8 Collection

- The Collection object provides a convenient way to refer to a related group of items as a single object.
- The items, or *elements*, in a collection need only be related by the fact that they exist in the collection.
- Elements of a collection do not have to share the same data type.
- Closely related data can be handled more efficiently when grouped together into a collection.
- Instead of writing separate code to handle each individual object, you can use the same code to process all the elements of a collection.
- To manage a collection, use the Array class and the System.Collections classes to add, remove, and modify either individual elements of the collection or a range of elements. An entire collection can even be copied to another collection.
- A collection is similar to an Array but provides additional flexibilities.

14.8.1 Differences between Arrays and Collections

SRNO	ARRAYS	COLLECTIONS
01	We can not insert a new element in to the Array	We can Insert a New Element into the collection anywhere, then following elements will flush down automatically
02	We can not delete any element from the Array	We can delete any element from the Collection, then following elements will flush up automatically
03	Array size is Fixed	Collection size is not Fixed as long as Memory is available we can add the Elements
04	We get either Insufficient Memory or Out of Memory problem with Array	We do not get either Insufficient Memory or Out of Memory problem with Collection
05	Array Supports only Single Data Type	Collection can store all Data type values

14.8.2 Methods with Collection

Add(Object Item)

Insert(int Index, object Item)

Remove(object Item)

RemoveAt(int Index)

Clear()

14.8.3 Properties with Collection

Count

14.9 ListBox Control:

This control is used to display group of items to user so that user can select a single item or group of items randomly or Range of items.

14.9.1 Additional Properties and Methods with Listbox Class

Srno	Property and Options	Type	Description
01	Items (Collection)	Object Collection Class	Stores all the elements in the form of collection and each element is identified using index value
02	SelectionMode None One (Default) MultiSimple MultiExtended	enum	None: User can't select any item from the list. One: User can select single item from the list. MultiSimple: User can select single Item or group of items randomly without holding any key. MultiExtended: User can select single item without hold any key or multiple items by holding Ctrl key or range of items by holding shift key.
03	Sorted True False(Default)	bool	True: Items are arranged in the alphabetical order. False: Items are not arranged in the alphabetical order.

Properties at Run Time

01	SelectedItem	object	Sets or gets the item value selected
02	SelectedIndex	object	Sets or gets index value of the item selected
03	SelectedItems	ObjectCollection Class	Gets the collection of items selected
04	SelectedIndices	ObjectCollection Class	Gets the collection of indices of the items selected

Properties with Items Collection

01	Count	int	Returns the count of items present in the items collection
----	-------	-----	--

Methods with Item Collection of the Listbox

01	Add(object Item)		Used to add a new item to the items collection of the listbox, by default the item is added at the end of the collection provided sorted is false
02	Insert(int Index,object Item)		Used to add a new item to the items collection of the listbox at the given index, the item is added at the given index of the collection provided sorted is false
03	Remove(object Item)		Used to delete an item from the items collection of the listbox using the item value
04	RemoveAt(int Index)		Used to delete an item from the items collection of the listbox using the index value

05	Clear()		Used to delete all items from the items collection of the listbox
----	---------	--	---

Consider the Listbox given right (Fig14.1) here and assume name as lstSample then

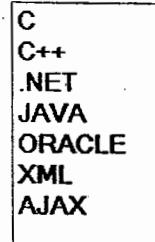
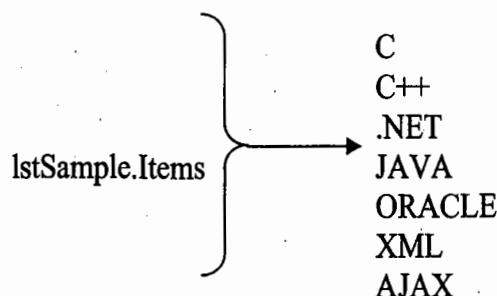


Fig 14.1

To identify the individual item we use index value like

- lstSample.Items[0] → C
- lstSample.Items[1] → C++
- lstSample.Items[2] → .NET
- lstSample.Items[3] → ORACLE
- lstSample.Items[4] → JAVA
- lstSample.Items[5] → XML
- lstSample.Items[6] → AJAX

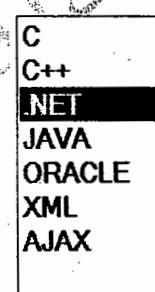


Fig 14.2

Consider the Fig 14.2 if user selected the item .NET then

- lstSample.SelectedItem → .NET
- lstSample.SelectedIndex → 2

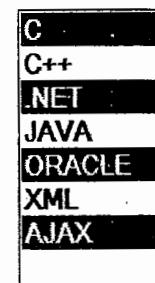
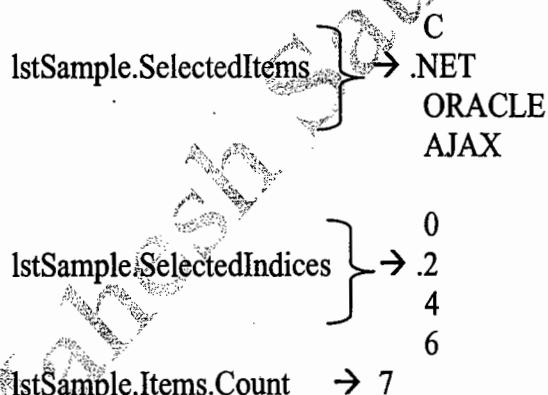


Fig 14.3

Consider the Fig 14.3 if user selected multiple items then



lstSample.Items.Add("C#.ET") →



```
C  
C++  
.NET  
JAVA  
ORACLE  
XML  
AJAX
```

`lstSample.Items.Insert(3, "C#.ET") →`

```
C  
C++  
.NET  
C#.NET  
JAVA  
ORACLE  
XML  
AJAX
```

```
C  
C++  
.NET  
JAVA  
ORACLE  
XML  
AJAX
```

`lstSample.Items.Remove("JAVA") →`

```
C  
C++  
.NET  
ORACLE  
XML  
AJAX
```

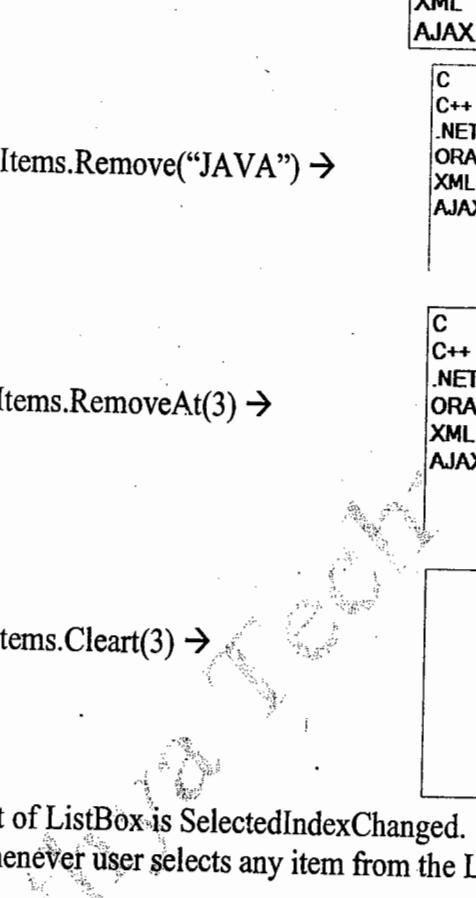
```
C  
C++  
.NET  
JAVA  
ORACLE  
XML  
AJAX
```

`lstSample.Items.RemoveAt(3) →`

```
C  
C++  
.NET  
ORACLE  
XML  
AJAX
```

```
C  
C++  
.NET  
JAVA  
ORACLE  
XML  
AJAX
```

`lstSample.Items.Clear() →`



```
C  
C++  
.NET  
ORACLE  
XML  
AJAX
```

Event For ListBox: Default event of ListBox is SelectedIndexChanged.

- This event will be fired whenever user selects any item from the ListBox.

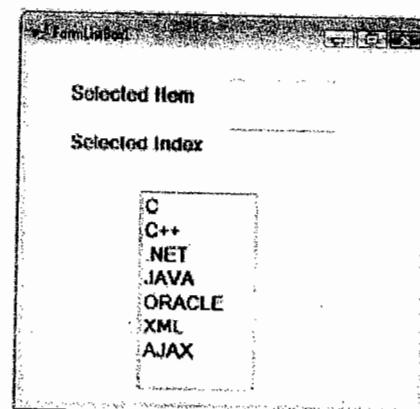
Example 14.6

Program with Listbox: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormListBox1 : Form
    {
        public FormListBox1()
        {
            InitializeComponent();
        }
        private void lstSample_SelectedIndexChanged(object sender, EventArgs e)
```

Design of Form



```

        txtItem.Text = lstSample.SelectedItem.ToString();
        txtIndex.Text = lstSample.SelectedIndex.ToString();
    }
}

```

Example 14.7

Program with Listbox: -
Code (Without any Validations):

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

```

```
namespace WABASICS
```

```

{
    public partial class FormListBox2 : Form
    {
        public FormListBox2()
        {
            InitializeComponent();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            lstSample.Items.Add(txtItem.Text);
        }

        private void btnInsert_Click(object sender, EventArgs e)
        {
            lstSample.Items.Insert(Convert.ToInt32(txtIndex.Text), txtItem.Text);
        }

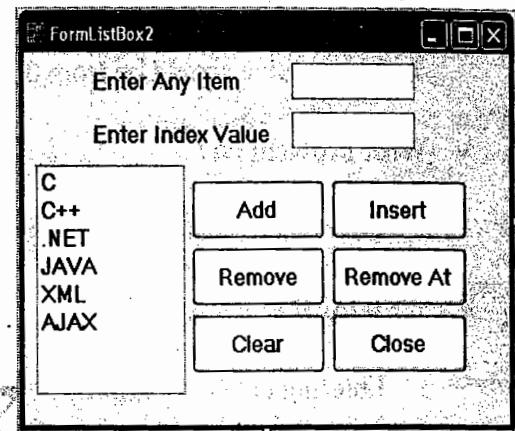
        private void btnRemove_Click(object sender, EventArgs e)
        {
            lstSample.Items.Remove(txtItem.Text);
        }

        private void btnRemoveAt_Click(object sender, EventArgs e)
        {
            lstSample.Items.RemoveAt(Convert.ToInt32(txtIndex.Text));
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            lstSample.Items.Clear();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Design of Form



List of validations to be Implemented of Above Example:

➤ With btnAdd:

1. Entering data into txtItem TextBox is mandatory.
2. Duplicate item should not be allowed to add to the items collections.

➤ With btnInsert:

1. Entering data into txtItem TextBox is mandatory.
2. Duplicate item should not be allowed to insert in to the items collection
3. Entering data into txtIndex TextBox is mandatory.
4. Index value should be from 0 to count of items only.

➤ With btnRemove:

1. User entered data in txtitem should be present in the listobx ctr

➤ With btnRemoveAt:

1. Entering data into txtIndex is mandatory.
2. Index value should be from 0 to count -1 of items only.

Code (With Validations):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormListBox2 : Form
    {
        public FormListBox2()
        {
            InitializeComponent();
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            //Code for Validation1 (btnAdd)
            if (txtItem.Text == "")
            {
                MessageBox.Show("Enter Any Item");
                txtItem.Focus();
                return;
            }
            //Code for Validation2 (btnAdd)
            for (int i = 0; i < lstSample.Items.Count; i++)
            {
                if (lstSample.Items[i].ToString() == txtItem.Text)
                {
                    MessageBox.Show("Item already exists");
                    txtItem.Focus();
                    return;
                }
            }
            lstSample.Items.Add(txtItem.Text);
            txtItem.Clear();
        }
    }
}
```

```

    {
        if (txtItem.Text == lstSample.Items[i].ToString())
        {
            MessageBox.Show("Duplicate Item Not Allowed");
            txtItem.Focus();
            return;
        }
    }
    lstSample.Items.Add(txtItem.Text);
}

private void btnInsert_Click(object sender, EventArgs e)
{
    //Code for Validation1 (btnInsert)
    if (txtItem.Text == "")
    {
        MessageBox.Show("Enter Any Item");
        txtItem.Focus();
        return;
    }
    //Code for Validation2 (btnInsert)
    for (int i = 0; i < lstSample.Items.Count; i++)
    {
        if (txtItem.Text == lstSample.Items[i].ToString())
        {
            MessageBox.Show("Duplicate Item Not Allowed");
            txtItem.Focus();
            return;
        }
    }
    //Code for Validation3 (btnInsert)
    if (txtIndex.Text == "")
    {
        MessageBox.Show("Enter Index Value");
        txtIndex.Focus();
        return;
    }
    //Code for Validation4 (btnInsert)
    if (Convert.ToInt32(txtIndex.Text) < 0 || Convert.ToInt32(txtIndex.Text) >
        lstSample.Items.Count)
    {
        MessageBox.Show("Invalid Index");
        txtIndex.Focus();
        return;
    }
    lstSample.Items.Insert(Convert.ToInt32(txtIndex.Text), txtItem.Text);
}

private void btnRemove_Click(object sender, EventArgs e)
{
    //Code for Validation1 (btnRemove)
    for (int i = 0; i < lstSample.Items.Count; i++)
    {
        if (txtItem.Text == lstSample.Items[i].ToString())
        {
            lstSample.Items.Remove(txtItem.Text);
        }
    }
}

```

Ex:
Co

```
        return;
    }
}
MessageBox.Show("Item is not present in the List Enter a Different Item");
txtItem.Focus();
}

private void btnRemoveAt_Click(object sender, EventArgs e)
{
    //Code for Validation1 (btnRemoveAt)
    if (txtIndex.Text == "")
    {
        MessageBox.Show("Enter Index Value");
        txtIndex.Focus();
        return;
    }
    //Code for Validation2 (btnRemoveAt)
    if (Convert.ToInt32(txtIndex.Text) < 0 || Convert.ToInt32(txtIndex.Text) >=
        lstSample.Items.Count)
    {
        MessageBox.Show("Invalid Index");
        txtIndex.Focus();
        return;
    }
    lstSample.Items.RemoveAt(Convert.ToInt32(txtIndex.Text));
}

private void btnClear_Click(object sender, EventArgs e)
{
    //Writing code to btnClear without Clear() Method
    for (int i = 0 ; i < lstSample.Items.Count;)
    {
        lstSample.Items.RemoveAt(i);
    }
    //lstSample.Items.Clear();
}

private void btnClose_Click(object sender, EventArgs e)
{
    this.Close();
}
```

There is another way to write the code to btnClear button see below

```
private void btnClear_Click(object sender, EventArgs e)
{
    //Writing code to btnClear without Clear() Method
    for (int i = lstSample.Items.Count ; i > 0; i--)
    {
        lstSample.Items.RemoveAt(i);
    }
}
```

Example 14.7

Program with Listbox: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormListBox3 : Form
    {
        public FormListBox3()
        {
            InitializeComponent();
        }

        private void btnLROne_Click(object sender, EventArgs e)
        {
            lstHotDrinks.Items.Add(lstCoolDrinks.SelectedItem);
            lstCoolDrinks.Items.Remove(lstCoolDrinks.SelectedItem);
        }

        private void btnLRAll_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < lstCoolDrinks.Items.Count; i++)
            {
                lstHotDrinks.Items.Add(lstCoolDrinks.Items[i]);
            }
            lstCoolDrinks.Items.Clear();
        }

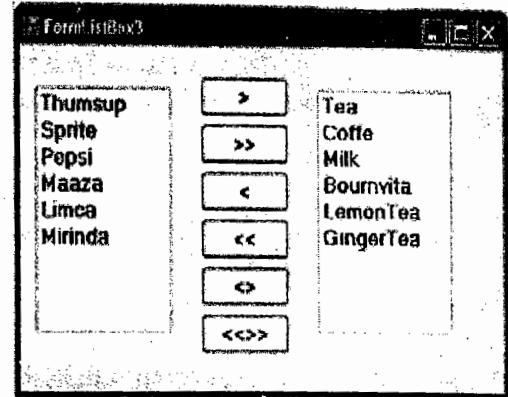
        private void btnRLOne_Click(object sender, EventArgs e)
        {
            lstCoolDrinks.Items.Add(lstHotDrinks.SelectedItem);
            lstHotDrinks.Items.Remove(lstHotDrinks.SelectedItem);
        }

        private void btnRLAll_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < lstHotDrinks.Items.Count; i++)
            {
                lstCoolDrinks.Items.Add(lstHotDrinks.Items[i]);
            }
            lstHotDrinks.Items.Clear();
        }

        private void cmdSwapOne_Click(object sender, EventArgs e)
        {
            if (lstCoolDrinks.SelectedIndex >= 0 && lstHotDrinks.SelectedIndex >= 0)
            {

```

Design of Form



```
int LIndex = lstCoolDrinks.SelectedIndex;
int RIndex = lstHotDrinks.SelectedIndex;
object S1 = lstCoolDrinks.SelectedItem;
object S2 = lstHotDrinks.SelectedItem;
lstCoolDrinks.Items.Remove(S1);
lstHotDrinks.Items.Remove(S2);
lstCoolDrinks.Items.Insert(LIndex, S2);
lstHotDrinks.Items.Insert(RIndex, S1);
}
else
{
    MessageBox.Show("Select one Item each from both Listboxes");
}
}

private void btnSwapAll_Click(object sender, EventArgs e)
{
    ListBox lstTemp = new ListBox();
    for (int i = 0; i < lstHotDrinks.Items.Count; i++)
    {
        lstTemp.Items.Add(lstHotDrinks.Items[i]);
    }
    lstHotDrinks.Items.Clear();
    for (int i = 0; i < lstCoolDrinks.Items.Count; i++)
    {
        lstHotDrinks.Items.Add(lstCoolDrinks.Items[i]);
    }
    lstCoolDrinks.Items.Clear();
    for (int i = 0; i < lstTemp.Items.Count; i++)
    {
        lstCoolDrinks.Items.Add(lstTemp.Items[i]);
    }
    lstTemp.Items.Clear();
}
}
```

14.10 Creating Controls Dynamically (i.e. at Runtime)

To create any control dynamically we need to perform three Steps

Step1:- Creating object for respective control class

Syntax:- ClassName ObjectName=new ClassName();

Example:- TextBox T1=new TextBox();

Step2:- Setting the required properties to the object.

Syntax:- ObjectName.PropertyName=Value;

Example:- T1.BackColor=Color.Red;

T1.ForeColor=Color.Yellow;

T1.Left=100;

T1.Top=100;

Step3:- Adding this object to the controls collection of Form class.

Syntax:- this.Controls.Add(ObjectName);

Example:- this.Controls.Add(T1);

Example 14.8

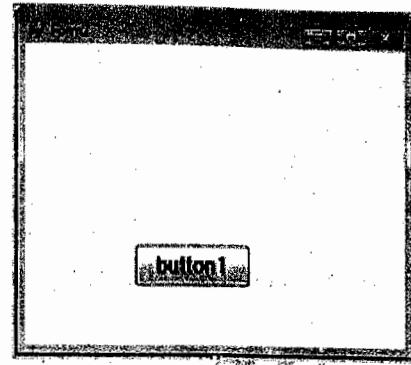
Program to create controls at run time: -

Code:

Design of Form

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.Win32;
namespace WABasics
{
    public partial class Form2 : Form
    {
        bool Flag;
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            TextBox T1 = new TextBox();
            T1.Left = 200;
            T1.Top = 50;
            this.Controls.Add(T1);
            Label L1 = new Label();
            L1.Left = 10;
            L1.Top = 50;
            L1.AutoSize = true;
            L1.Text = "Enter Any Text";
            this.Controls.Add(L1);
        }
    }
}
```



14.11 Checkbox Control

- This control is used to provide selection of more than one options from the given group of options.
- The **Checkbox** control can display text, an Image, or both.

14.11.1 Additional Properties With Checkbox Class:

Srno	Property and options	Type	Description
01	AutoSize True False(Default)	bool	True: Control is resized according to contents False: Control can not be resized according to contents
02	CheckAlign TopLeft TopCenter TopRight MiddleLeft MiddleCenter MiddleRight BottomLeft BottomCenter BottomRight	enum	TopLeft: Content is vertically aligned at the top, and horizontally aligned on the left. TopCenter: Content is vertically aligned at the top, and horizontally aligned on the center. TopRight: Content is vertically aligned at the top, and horizontally aligned on the right. MiddleLeft: Content is vertically aligned at the middle, and horizontally aligned on the left. MiddleCenter: Content is vertically aligned at the middle, and horizontally aligned on the center. MiddleRight: Content is vertically aligned at the middle, and horizontally aligned on the right. BottomLeft: Content is vertically aligned at the bottom, and horizontally aligned on the left. BottomCenter: Content is vertically aligned at the bottom, and horizontally aligned on the center. BottomRight: Content is vertically aligned at the bottom, and horizontally aligned on the right.
03	Checked True False(Default)	bool	True: Checkbox is Activated / Selected False: Checkbox is not Activated / Selected
04	CheckState UnChecked(Default) Checked Indeterminate	enum	Indicates the state of the Checkbox
05	ThreeState True False(Default)	bool	True: Checkbox will have three states False: Checkbox will have two states

Events Associated with CheckBox Class

- CheckedChanged is the default Event of the Checkbox.
 - This Event will be fired when Checkbox is selected or deselected.

Example 14.9

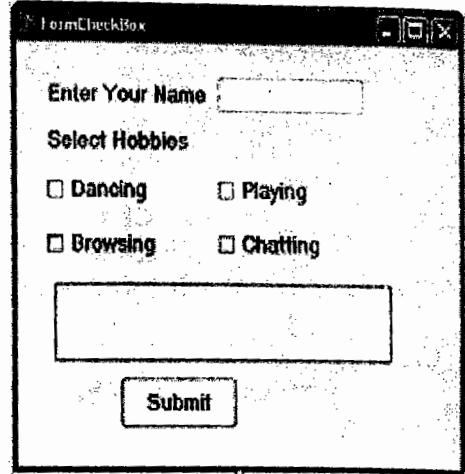
Program with Checkbox Control: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormCheckBox : Form
    {
        public FormCheckBox()
        {
            InitializeComponent();
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            string S = txtName.Text + " Hobbies are ";
            if (chkDancing.Checked == true)
                S = S + chkDancing.Text + " ";
            if (chkPlaying.Checked == true)
                S = S + chkPlaying.Text + " ";
            if (chkBrowsing.Checked == true)
                S = S + chkBrowsing.Text + " ";
            if (chkChatting.Checked == true)
                S = S + chkChatting.Text + " ";
            lblDisplay.Text = S;
        }
    }
}
```

Design of Form



In the above example the code is not generic code if number of check boxes increase or decrease then code should be modified which is not acceptable in real time programming, so we write the code like given below.

Generic Code for the above Example:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormCheckBox : Form
    {
        public FormCheckBox()
    }
```

```

    {
        InitializeComponent();
    }
    private void btnSubmit_Click(object sender, EventArgs e)
    {
        string S = txtName.Text + " Hobbies are ";
        foreach (Control X in this.Controls)
        {
            if (X is CheckBox)
            {
                CheckBox C1 = (CheckBox)X;
                if (C1.Checked == true)
                    S = S + C1.Text + " ";
            }
        }
        lblDisplay.Text = S;
    }
}

```

14.12 ComboBox Control :-

When we take listBox is TextBox controls both have their own Advantages and Disadvantages.

TextBox :-

Advantage: - User can enter the data at Runtime

DisAdvantage: - It's not possible to display group of items to the user.

ListBox :-

Advantage: - We can display group of items to user

DisAdvantage: - User can't enter any data at run time.

Combo box is the combination of both TextBox & ListBox controls

In combo Box control, user can enter the data at run time and also we can display group of items to the user.

14.12.1 Additional Properties and Methods with Combobox Class:

Srno	Property and Options	Type	Description
01	Items (Collection)	ObjectCollection Class	Stores all the elements in the form of collection and each element is identified using index value
02	DropDownStyle Simple DropDown(Default) DropDownList	enum	<p>Simple: ComboBox looks like Textbox and ListBox, User can enter any data at runtime and also can select any item in the list.</p> <p>DropDown: ComboBox appearance does not change, User can enter any data at runtime and also can select any item in the list.</p> <p>DropDownList: ComboBox appearance does not change, User can not enter any data at runtime but can select any item in</p>

			the list.
03	MaxDropDownItems	int	Used to set a value that how many items should be displayed to the user in drop down list
04	Sorted True False(Default)	bool	True: Items are arranged in the alphabetical order. False: Items are not arranged in the alphabetical order.
Properties at Run Time			
01	SelectedItem	object	Sets or gets the item value selected
02	SelectedIndex	object	Sets or gets index value of the item selected
01	Count	int	Returns the count of items present in the items collection
Methods with Item Collection of the Listbox			
01	Add(object Item)		Used to add a new item to the items collection of the listbox, by default the item is added at the end of the collection provided sorted is false
02	Insert(int Index, object Item)		Used to add a new item to the items collection of the listbox at the given index, the item is added at the given index of the collection provided sorted is false
03	Remove(object Item)		Used to delete an item from the items collection of the listbox using the item value
04	RemoveAt(int Index)		Used to delete an item from the items collection of the listbox using the index value
05	Clear()		Used to delete all items from the items collection of the listbox

Event of DropDownListBox: Default event for dropdown list box is SelectedIndexChanged.

- This event will be fired whenever user selects any item from the combo box.

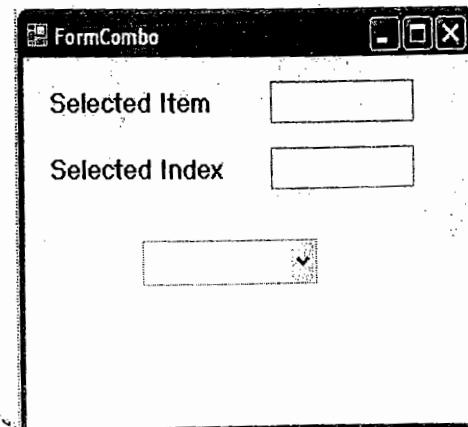
Example 14.10

Program to create controls at run time: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABASICS
{
    public partial class FormCombo : Form
```

Design of Form



```

public FormCombo()
{
    InitializeComponent();
}
private void cmbSample_SelectedIndexChanged(object sender, EventArgs e)
{
    txtItem.Text = cmbSample.SelectedItem.ToString();
    txtIndex.Text = cmbSample.SelectedIndex.ToString();
}
}

```

14.13 Common Dialog Controls: These controls are used to perform some tasks like opening a file, saving a file, applying font attributes, color attributes, performing page settings, Print Settings etc.

Types of Common Dialog Controls:

1. OpenFileDialog control
2. SaveFileDialog control
3. FontDialog control
4. ColorDialog control
5. FolderBrowserDialog control
6. PageSetupDialog control
7. PrintDialog control

14.13.1 OpenFileDialog control

➤ This control is used to display, open file dialogue box to the user, so that user can select any file to open.

14.13.1.1 Properties With OpenFileDialog:

Srno	Property and options	Type	Description
01	FileName	string	Sets or gets the name of the file including the path selected by the user with in the OpenFileDialog box
02	Filter	string	used to filter the files like text files, word files, excel files, all files etc.

Using Filter Property is like: Open file dialog1.filter = "Filter expression".

Filter Expression Contains 2 parts separated by the | (pipe) symbol

Display Text | Filter criteria

Text files	*.txt
Word files	*.doc
Excel files	*.xls
All files	*.*

Ex: Open file dialog1.filter="Text Files | *.txt" | Word Files | *.doc | Excel Files | *.xls | All Files | *.*

14.13.2 SaveFileDialog control

- This control is used to display, save file dialog box to the user, so that user can enter any file name to save.

14.13.2.1 Properties With SaveFileDialog:

Srno	Property and options	Type	Description
01	FileName	string	Sets or gets the name of the file including the path selected by the user with in the OpenFileDialog box
02	Filter	string	used to filter the files like text files, word files, excel files, all files etc.

14.13.3 FontDialog control

- This control is used to display, font dialog box to the user, so that user can select required font attributes to apply to the required data or others.

14.13.3.1 Properties With FontDialog:

Srno	Property and options	Type	Description
01	Font	object	Sets or gets the font attributes selected with in the FontFileDialog control
02	MaxSize	int	Used to set the maximum font size, that is to be displayed to the user with in the font dialog control.
03	MinSize	int	Used to set the minimum font size, that is to be displayed to the user with in the font dialog control.
04	ShowEffects	bool	True: User can access show effects options with in the font dialog control. False: User can not access show effects options with in the font dialog control.

14.13.4 ColorDialog control

- This control is used to display, color dialog box to the user, so that user can select required color to apply to the required data or other controls etc.

14.13.4.1 Properties With ColorDialog:

Srno	Property and options	Type	Description
01	Color	struct	Sets or gets the color value selected with in the ColorDialog control
02	AllowFullOpen	Int	True: User can access define custom colors button with in the color dialog control. False: User can not access define custom colors button with in the color dialog control.
03	FullOpen	int	True: Color dialog box is opened in full mode. False: Color dialog box is opened in partial mode.

14.13.5 FolderBrowserDialog control

- This control is used to display, folder browse dialog box to the user, so that user can select the required folder.

14.13.5.1 Properties With FolderBrowserDialog:

Srno	Property and options	Type	Description
01	SelectedPath	string	Sets or gets the path of the folder selected by the user

14.13.6 PageSetupDialog control

- This control is used to display, Page setup dialog box to the user, so that user can perform required page settings to apply to the required file pages.

14.13.6.1 Properties With PageSetupDialog:

Srno	Property and options	Type	Description
01	Document	PrintDocument class Object	Gets or sets a value indicating the PrintDocument to get page settings from.

14.13.7 PrintDialog control

- This control is used to display, Print dialog box to the user, so that user can set required options for printing any file.

14.13.7.1 Properties With PrintDialog:

Srno	Property and options	Type	Description
01	Document	PrintDocument class Object	Gets or sets a value indicating the PrintDocument used to obtain PrinterSettings.

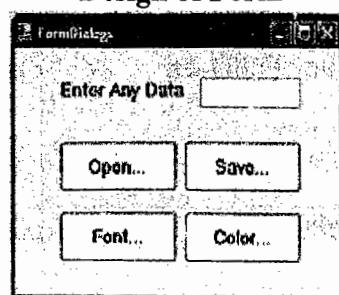
Example 14.11

Program to create controls at run time: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
```

Design of Form



```

public partial class FormDialogs : Form
{
    public FormDialogs()
    {
        InitializeComponent();
    }

    private void btnOpen_Click(object sender, EventArgs e)
    {
        openFileDialog1.Filter = "Text Files|*.txt|Word Document|*.doc|All Files|*.*";
        openFileDialog1.ShowDialog();
        txtSample.Text = openFileDialog1.FileName;
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        saveFileDialog1.ShowDialog();
        txtSample.Text = saveFileDialog1.FileName;
    }

    private void btnFont_Click(object sender, EventArgs e)
    {
        fontDialog1.ShowDialog();
        txtSample.Font = fontDialog1.Font;
    }

    private void btnColor_Click(object sender, EventArgs e)
    {
        colorDialog1.ShowDialog();
        txtSample.BackColor = colorDialog1.Color;
    }
}

```

14.14 Working with Menus:

- Any windows based application will provide two types of menus.
- Main menu
- Context / Shortcut menu

14.14.1 Main menu

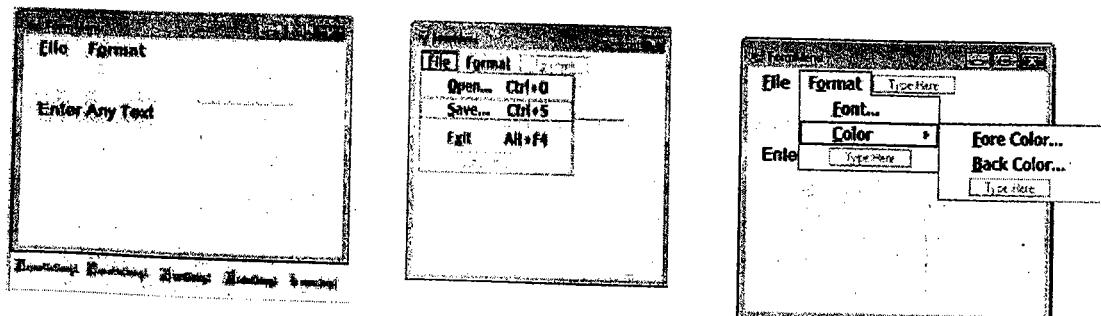
- This will be available at the top of the application
- Every application can contain only one main menu.
- To create main menu, we have menu strip control.

14.14.2 Context menu

- This menu is displayed when users clicks right mouse button with in the application
- An application can contain any number of context menus baed on requirement.
- To create main menu, we have ContextMenuStrip control.

Example 14.12

Program with Main Menu Control: -
Design of Form



Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormMenu : Form
    {
        public FormMenu()
        {
            InitializeComponent();
        }

        private void smnuopen_Click(object sender, EventArgs e)
        {
            openFileDialog1.ShowDialog();
            txtSample.Text = openFileDialog1.FileName;
        }

        private void smnusave_Click(object sender, EventArgs e)
        {
            saveFileDialog1.ShowDialog();
            txtSample.Text = saveFileDialog1.FileName;
        }

        private void smnuexit_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void smnufont_Click(object sender, EventArgs e)
        {
            fontDialog1.ShowDialog();
            txtSample.Font = fontDialog1.Font;
        }

        private void smnuforeColor_Click(object sender, EventArgs e)
```

```

        colorDialog1.ShowDialog();
        txtSample.ForeColor = colorDialog1.Color;
    }

    private void smnubackColor_Click(object sender, EventArgs e)
    {
        colorDialog1.ShowDialog();
        txtSample.BackColor = colorDialog1.Color;
    }
}
}

```

Assigning Shortcut Keys to Menus:

- Select open menu → go to property window → go to shortcut keys property → Activate ctrl check box → Select the key 'O', similarly assign respective shortcut keys for the remaining menus.

Working With Shortcut/Context Menus:

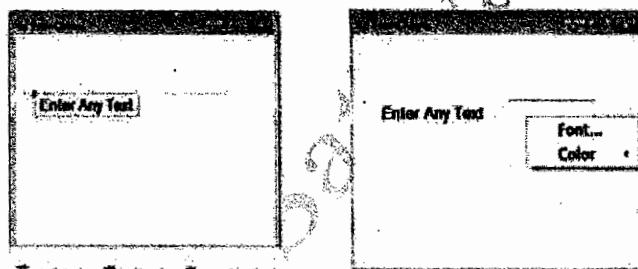
- To create context menu, we have a separate control known as context menu strip control.
- We can create any number of shortcut menus within a single application.

Example 14.13

Program with Shortcut Menu Control: -

Design of Form

Code:



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

```

namespace WABasics

```

{
    public partial class FormCMenu : Form
    {
        public FormCMenu()
        {
            InitializeComponent();
        }
    }
}
```

```
private void mnufont_Click(object sender, EventArgs e)
```

```
{  
    fontDialog1.ShowDialog();  
    txtSample.Font = fontDialog1.Font;  
}  
  
private void smnuforeColor_Click(object sender, EventArgs e)  
{  
    colorDialog1.ShowDialog();  
    txtSample.ForeColor = colorDialog1.Color;  
}  
  
private void smnubackColor_Click(object sender, EventArgs e)  
{  
    colorDialog1.ShowDialog();  
    txtSample.BackColor = colorDialog1.Color;  
}  
}
```

Assigning ContextMenuStrip1 to the txtSample Text Box:

Select txtSample → go to properties windows

→ go to context menu strip property → select contextMenuStrip1 → run the application and check.

14.15 Timer Control:

- This Control is used to execute the required code repeatedly again and again with out any user intervention.

14.15.1 Properties and Methods With Timer Control:

Srno	Property and options	Type	Description
01	Enable True False(Default)	bool	True: As long as Enabled is true timer will be working. False: Timer stops working.
02	Interval	int	Time period in Milli seconds after which the timer tick event should be fired, default is 100 1000 Milli Seconds=1 Second
Methods with Timer Control			
01	Start()	void	Starts the Timer Control
02	Stop()	void	Stops the Timer Control

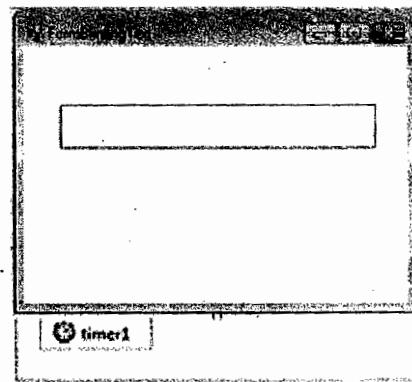
Example 14.14

Program with Timer Control (Blinking Text): -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormBlinlingText : Form
    {
        int i = 0;
        public FormBlinlingText()
        {
            InitializeComponent();
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            if (i == 0)
            {
                lblDisplay.Text = "Sathya Technologies";
                lblDisplay.ForeColor = Color.Blue;
                lblDisplay.BackColor = Color.Yellow;
                i = 1;
            }
            else
            {
```

Design of Form



```

        lblDisplay.Text = "Hyderabad";
        lblDisplay.BackColor = Color.White;
        lblDisplay.ForeColor = Color.Red;
        i = 0;
    }
}
}
}

```

Example 14.15

Program with Timer Control (Scrolling Marquee) :-

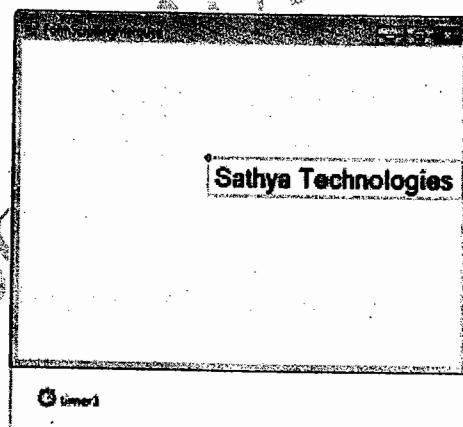
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormScrollingMarquee : Form
    {
        public FormScrollingMarquee()
        {
            InitializeComponent();
        }
        private void timer1_Tick(object sender, EventArgs e)
        {
            label1.Left = label1.Left - 20;
            if (label1.Left + label1.Width <= 0)
            {
                label1.Left = this.Width;
            }
        }
    }
}

```

Design of Form



1.

>

14

Si

01

02

03

Ex

Co

14.16 PictureBox Control:-

- This control is used to display the images to the user on the form, We can display the images like JPEG, BMP, GIFF, ICON images etc.

14.16.1 Properties With PictureBox Class:

Srno	Property and options	Type	Description
01	Image	ImageClass	Sets or gets the image to be displayed in the picturebox control.
02	ImageLocation	string	Sets or gets the file path (Disk Location) of the image to be displayed in the picturebox control.
03	SizeMode Normal(Default) StretchImage AutoSize CenterImage	enum	Normal: Picturebox and Image will have their own sizes, there do not be any change in image size when picturebox size changes. StretchImage: Image will be scaled to picturebox size, Image size will increase or decrease according to picturebox size. AutoSize: Picturebox will be scaled to Image size, it is not possible to increase or decrease either picturebox size or Image size. CenterImage: Image will be displayed with in the center location of picturebox.

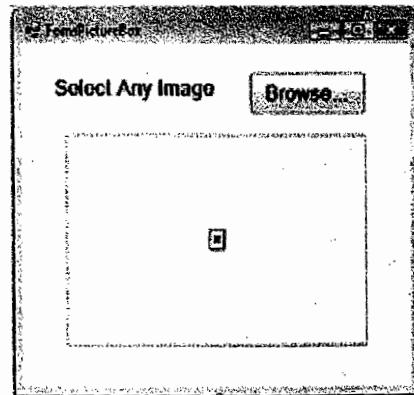
Example 14.16

Program with Picturebox Control:

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormPictureBox : Form
    {
        public FormPictureBox()
        {
            InitializeComponent();
        }
        private void cmdBrowse_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "Jpeg Files|*.Jpg|Giff Files|*.Gif|Bitmap Files|*.Bmp|All
Files|*.*";
            openFileDialog1.ShowDialog();
            //pictureBox1.ImageLocation = openFileDialog1.FileName;
        }
    }
}
```

Design of Form



```
    pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
```

```
}
```

14.17 ImageList Control:-

- This control is used to store the images required to be used with in the application so that used to attach to ToolStrip or ListView or other controls, We can store the images like JPEG, BMP, GIFF, ICON images etc.

14.17.1 Properties With ImageList Class:

Srno	Property and options	Type	Description
01	Images	Collection of Image Objects	Used to set or get group of images with in the Image List Control
02	ImageSize	enum	Sets or gets the size of the images are to be stored with in the ImageList control
03	ColorDepth Depth4Bit Depth8Bit(Default) Depth16Bit Depth24Bit Depth32Bit	enum	As the more the color depth the clarity of the image will be more. As the less the color depth the clarity of the image will be less.

Example 14.17

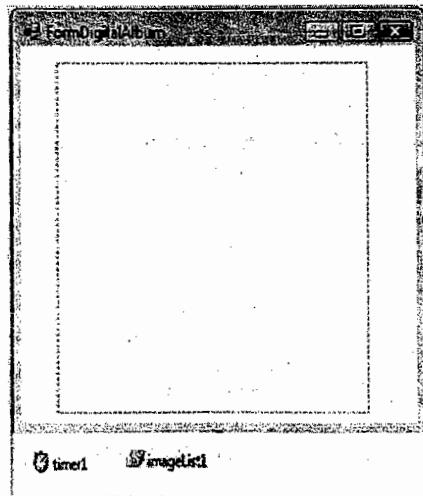
Program with Picturebox Control (Digital Album): -

Note: Add the required images to the ImageList Control at Design time

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormDigitalAlbum : Form
    {
        int i = 0;
        public FormDigitalAlbum()
        {
            InitializeComponent();
        }
        private void timer1_Tick(object sender, EventArgs e)
```

Design of Form

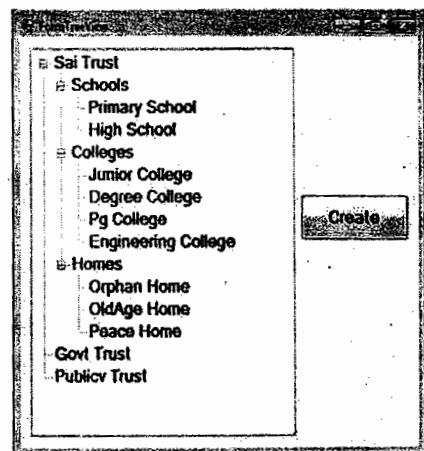
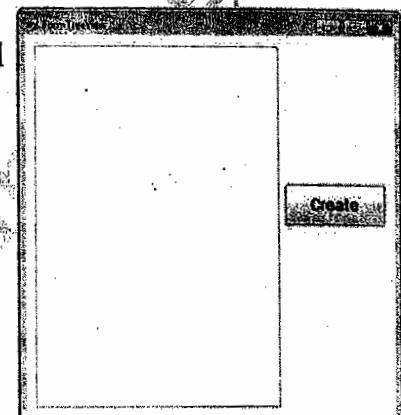


```
{  
    pictureBox1.Image = imageList1.Images[i];  
    i++;  
    if (i == imageList1.Images.Count)  
    {  
        i = 0;  
    }  
}
```

14.18 TreeView Control

- This control is used to display the required item / data in hierarchical Manner

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WABasics
{
    public partial class FormTreeView : Form
    {
        public FormTreeView()
        {
            InitializeComponent();
        }
        private void cmdCreate_Click(object sender, EventArgs e)
        {
            TVSample.Nodes.Add("Sai Trust");
            TVSample.Nodes.Add("Govt Trust");
            TVSample.Nodes.Add("Publicv Trust");
            TVSample.Nodes[0].Nodes.Add("Schools");
            TVSample.Nodes[0].Nodes.Add("Colleges");
            TVSample.Nodes[0].Nodes.Add("Homes");
            TVSample.Nodes[0].Nodes[0].Nodes.Add("Primary Sc");
            TVSample.Nodes[0].Nodes[0].Nodes.Add("High Scho");
            TVSample.Nodes[0].Nodes[1].Nodes.Add("Junior Col");
            TVSample.Nodes[0].Nodes[1].Nodes.Add("Degree Co");
            TVSample.Nodes[0].Nodes[1].Nodes.Add("Pg College");
            TVSample.Nodes[0].Nodes[1].Nodes.Add("Engineering");
            TVSample.Nodes[0].Nodes[2].Nodes.Add("Orphan Ho");
            TVSample.Nodes[0].Nodes[2].Nodes.Add("OldAge Ho");
            TVSample.Nodes[0].Nodes[2].Nodes.Add("Peace Ho");
        }
    }
}
```



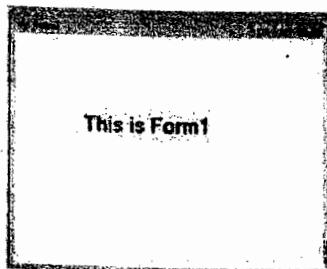
Chapter 15

MDI Applications

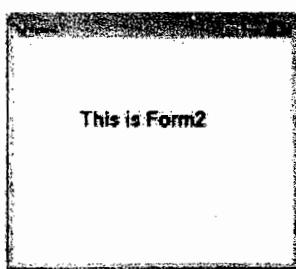
- Any windows based environment supports two types of application.
 - SDI Application (Single Document Interface Application)
 - MDI Application (Multiple Document Interface Application)
- **SDI Applications:** The applications that will allow the user to work with only one file / document at once are known as SDI Applications.
 - Ex: Notepad, Paintbrush, Wordpad etc.
- **MDI Applications:** The applications that will allow the user to work with more than one file / Document at once are known as MDI Applications.
 - Ex: MS- Word, MS – Excel, MS - Powerpoint etc.
- To create any MDI App., we need to include MDI form with in the application.
- MDI form will be treated as parent form.
- Remaining forms will be treated as child forms.

Creating MDI Application:

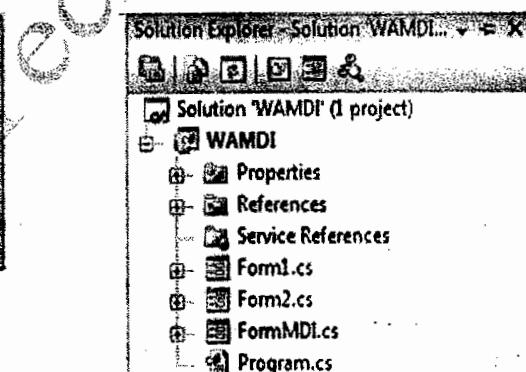
Form1 Design



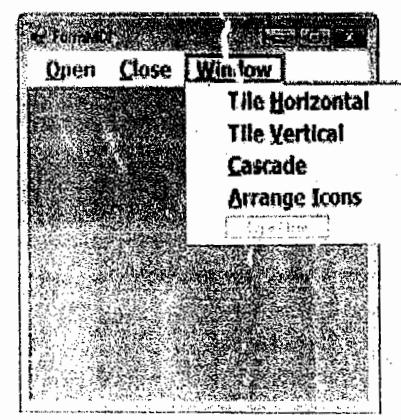
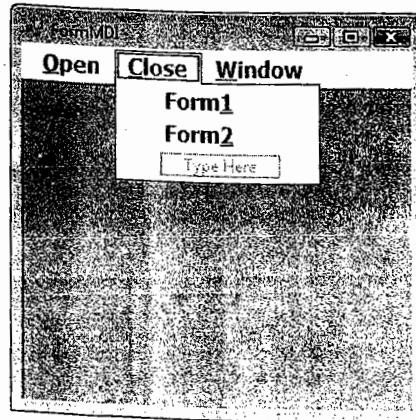
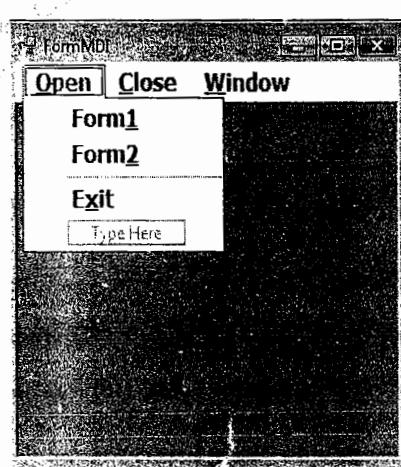
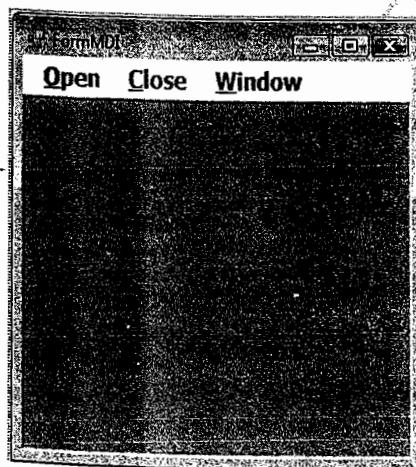
Form2 Design



Application Structure



FormMDI Design



Code in FormMDI:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WAMDI
{
    public partial class FormMDI : Form
    {
        public static Form1 Obj1 = new Form1();
        public static Form2 Obj2 = new Form2();
        public FormMDI()
        {
            InitializeComponent();
        }

        private void smnuOform1_Click(object sender, EventArgs e)
        {
            if (Obj1 == null)
            {
                Obj1 = new Form1();
            }
            Obj1.MdiParent = this;
            Obj1.Show();
        }

        private void smnuOform2_Click(object sender, EventArgs e)
        {
            if (Obj2 == null)
            {
                Obj2 = new Form2();
            }
            Obj2.MdiParent = this;
            Obj2.Show();
        }

        private void smnuCform1_Click(object sender, EventArgs e)
        {
            Obj1.Hide();
        }

        private void smnuCform2_Click(object sender, EventArgs e)
        {
            Obj2.Hide();
        }

        private void smnutilHorizontal_Click(object sender, EventArgs e)
        {
            this.LayoutMdi(MdiLayout.TileHorizontal);
        }
    }
}
```

```

private void smnutileVertical_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}

private void smnucascade_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.Cascade);
}

private void smnuarrangeIcons_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.ArrangeIcons);
}

private void smnuexit_Click(object sender, EventArgs e)
{
    this.Close();
}
}

```

Code in Form1:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WAMDI
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            FormMDI.Obj1 = null;
        }
    }
}

```

Code in Form2:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WAMDI
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        private void Form2_FormClosing(object sender, FormClosingEventArgs e)
        {
            FormMDI.Obj2 = null;
        }
    }
}
```

Where MDI Forms Are Used:

In general MDI forms are used at the end phase of the project.

- Initially individual modules are computerized by the teams. Then after completion of all modules. All modules will be integrated together using MDI FORM.

16.1 Introduction

- Here to understand the role of ADO.NET first we look into some basic terms like Front end Application, Front end Tool Back end Application, Back end Tool etc.
- **Front end Application:** - The application with which end user will interact like screens, forms, reports is known as Front end Application
- **Back end Application:** - The application where we store the end user data is known as Backend Application
- **Front end Tool:** - The software / Tool used to design the front end application is known as Front end Tool.
 - **Ex:** - Dbase, FoxPro, MS Access, Power Builder, D2K/ Oracle forms, Visual Basics, .Net, Java etc.
- **Back end Tool:** - The software used to design the back end application is known as Backend Tool.
 - **Ex:** -Dbase, FoxPro, MS Access, Sql server, Oracle, DB2, MySql, Sybase etc.

16.2 Layered Approach

- Now we look into an approach called layered approach (This is not N – Tier Architecture)

16.2.1 I layered Approach

- In this method both front end & back end applications are designed using same tool & can be present only in single machine.

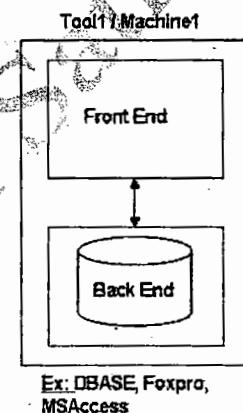


Figure 16.1

Advantages of I Layered Approach:-

- As front end & backend applications are designed using same tool no separate translatory mediums are required.

Disadvantages of I Layered Approach:-

- Doesn't support centralized database.
- Cannot withstand for large amount of data.
- To overcome the disadvantages of I layered approach we use II layered approach.

16.2.2 II layered Approach

- In this method both front end & back end applications are designed using two different tools & can be present in two different machines.

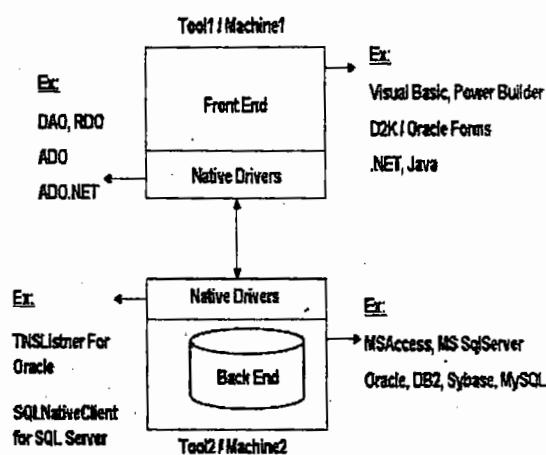


Figure 16.2

Role of Native Drivers:-

- As front end & backend applications are designed using 2 different tools. So, front end tool cannot understand backend tool & backend tool cannot understand front end tool. Native drivers are used as translators between front end & back end tools.
- Front end tools native driver will take the request from front end tool, converts into back end tool native driver understandable format & sends to front end.
- Backend tool native driver will take the request from backend tool & converts into front end tool native driver understandable format & sends to front end.

Advantages of II Layered Approach:-

- Supports centralized database.
- Most of the databases can withstand for large amount of data.

Disadvantages of I Layered Approach:-

- In any front end tool if one native driver is designed to interact with any backend tool & that native driver cannot be used to interact with other backend tool that means every front end tool should have separate set of native driver to interact with backend tool like

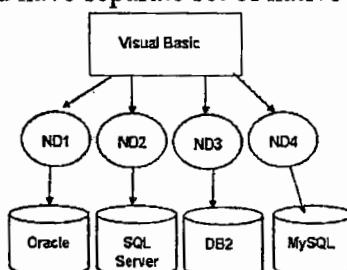


Figure 16.3

- If one native driver is designed for any backend tool to interact with some front end tool that native driver cannot be used to interact with other front end tools. So every

backend tool need to have one separate set of native drivers to interact with each front end tool like

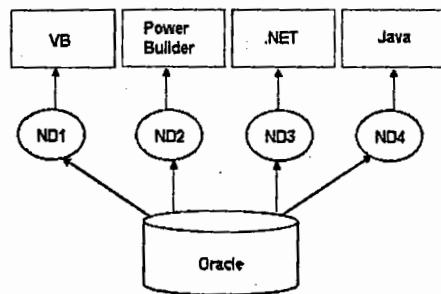


Figure 16.4

- To overcome this disadvantage we use III layered approach

16.2.3 III layered Approach

- In this method Front end and Backend Tool will have only one set of native drivers.
- These native Drivers do not interact with each other, rather interact with 3rd component known as "Middleware"
- Middleware is designed in such a way that it can understand any front end native drivers and any back end native drivers

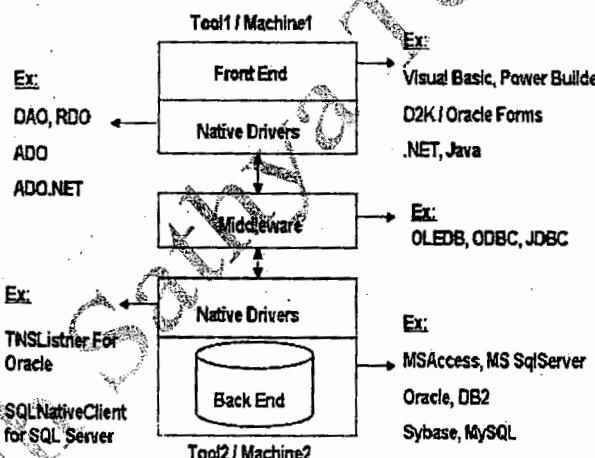


Figure 16.5

Abbreviations of the terms used

DAO	→ Data Access Object
RDO	→ Remote Data Object
ADO	→ ActiveX Data Object
ADO.Net	→ ActiveX Data Object for .Net
OLEDB	→ Object Linking Embedding Data Base
ODBC	→ Open Data Base Connectivity
JDBC	→ Java Data Base Connectivity

16.3 ADO.Net

Note: Most of the following points are as it is from MSDN

- ADO.Net is the native driver designed by Microsoft for .Net technology used to interact with middle wave or database.
- Without ADO.Net we cannot write database programming code in .Net
- ADO.Net is a set of classes that expose data access service to .Net programmer.
- ADO.Net provides set of components for creating distributed, data sharing applications.
- ADO.Net is an integral part of the .Net framework which provides access to relational XML and application data.
- ADO.Net supports connection oriented architecture & also disconnected architecture.
- ADO.NET supports a variety of development needs, including the creation of front-end database clients and middle-tier business objects used by applications, tools, languages, or Internet browsers
- ADO.NET provides consistent access to data sources such as SQL Server and XML, and to data sources exposed through OLE DB and ODBC.
- Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, handle, and update the data that they contain.
- ADO.NET separates data access from data manipulation into discrete components that can be used separately or in tandem.
- ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.
- Those results are either processed directly, placed in an ADO.NET DataSet object in order to be exposed to the user in an ad hoc manner, combined with data from multiple sources, or passed between tiers. The **DataSet** object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.
- The ADO.NET classes are found in System.Data.dll, and are integrated with the XML classes found in System.Xml.dll.
- ADO.NET provides functionality to developers who write managed code similar to the functionality provided to native component object model (COM) developers by ActiveX Data Objects (ADO).

16.3.1 Benefits of ADO.NET

- ADO.NET offers several advantages over previous versions of ADO and over other data access components. These benefits fall into the following categories:
 - Interoperability
 - Maintainability
 - Programmability
 - Performance
 - Scalability

Interoperability

- ADO.NET applications can take advantage of the flexibility and broad acceptance of XML. Because XML is the format for transmitting datasets across the network, any component that can read the XML format can process data.

- In fact, the receiving component need not be an ADO.NET component at all: The transmitting component can simply transmit the dataset to its destination without regard to how the receiving component is implemented.
- The destination component might be a Visual Studio application or any other application implemented with any tool whatsoever.
- The only requirement is that the receiving component be able to read XML. As an industry standard, XML was designed with exactly this kind of interoperability in mind.

Maintainability

- In the life of a deployed system, modest changes are possible, but substantial architectural changes are rarely attempted because they are so difficult.
- That is unfortunate, because in a natural course of events, such substantial changes can become necessary.
- For example, as a deployed application becomes popular with users, the increased performance load might require architectural changes.
- As the performance load on a deployed application server grows, system resources can become scarce and response time or throughput can suffer.
- Faced with this problem, software architects can choose to divide the server's business-logic processing and user-interface processing onto separate tiers on separate machines. In effect, the application server tier is replaced with two tiers, alleviating the shortage of system resources.
- The problem is not designing a three-tiered application. Rather, it is increasing the number of tiers after an application is deployed.
- If the original application is implemented in ADO.NET using datasets, this transformation is made easier.
- Remember, when you replace a single tier with two tiers, you arrange for those two tiers to trade information. Because the tiers can transmit data through XML-formatted datasets, the communication is relatively easy.

Programmability

- ADO.NET data components in Visual Studio encapsulate data access functionality in various ways that help you program more quickly and with fewer mistakes. For example, data commands abstract the task of building and executing SQL statements or stored procedures.
- Similarly, ADO.NET data classes generated by the designer tools result in typed datasets. This in turn allows you to access data through typed programming. For example, consider the following line of code that accesses a data member in an untyped dataset.

Performance

- For disconnected applications, ADO.NET datasets offer performance advantages over ADO disconnected recordsets.
- When using COM marshalling to transmit a disconnected recordset among tiers, a significant processing cost can result from converting the values in the recordset to data types recognized by COM. In ADO.NET, such data-type conversion is not necessary.

Scalability

- Because the Web can vastly increase the demands on your data, scalability has become critical. Internet applications have a limitless supply of potential users.
- Although an application might serve a dozen users well, it might not serve hundreds —or hundreds of thousands — equally well.
- An application that consumes resources such as database locks and database connections will not serve high numbers of users well, because the user demand for those limited resources will eventually exceed their supply.
- ADO.NET accommodates scalability by encouraging programmers to conserve limited resources. Because any ADO.NET application employs disconnected access to data, it does not retain database locks or active database connections for long durations.

16.3.2 Connection Oriented Architecture: -

- In this architecture we have 2 methods supported
- **1st Method:** - In this method first we establish connection to the database, performs the required operations like inserting the records, updating, deleting the records etc & then closes the connection

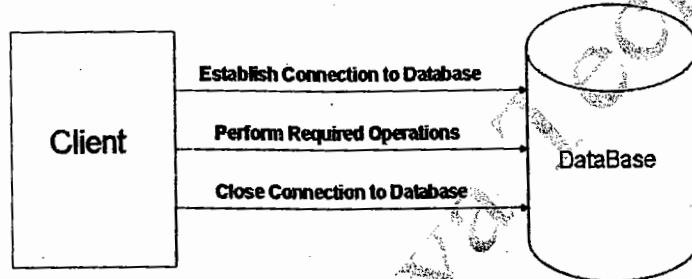


Figure 16.6

- **2nd Method:** - This method is used just to read the data from the database & using this method we cannot perform the operations like inserting the records, updating the records, deleting the records etc.

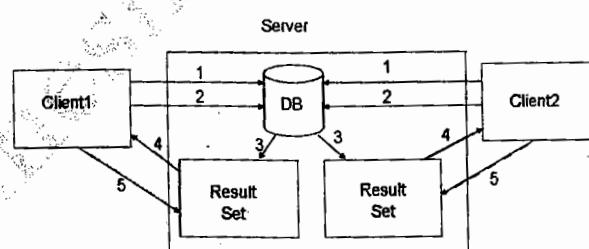


Figure 16.7

- **Step1:** Makes a connection from client to database
- **Step2:** Client sends request for required data to the server
- **Step3:** Client request is processed & requested data is dumped into separate memory at server, allotted for the client this memory is known as Network Buffer or Result set.
- **Step4:** Link is created from Network Buffer / Result set to client
- **Step5:** Client starts reading the data from Network Buffer / Result set.
- After the completion of reading the data connection to the database can be closed

16.3.3 Disconnected Architecture: -

- In this method it is not required to establish prior connection to the database. When request is sent for data connection to the database is established automatically.
- Once requested data is dumped into client machine memory, no more connection is maintained to the database.

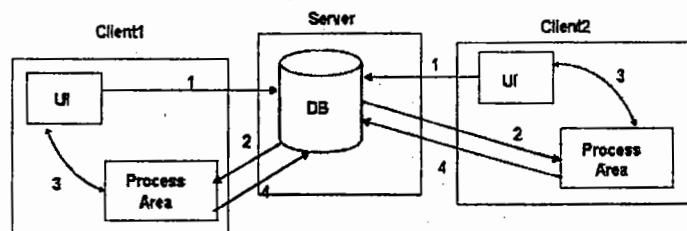


Figure 16.8

- Step1: Client sends request for required data from the server, at the time of request only connection to database is automatically opened.
- Step2: Requested data is dumped into client machines memory as Process Area (PA). Once data is dumped into Process Area, no more connection is maintained to the database and the connection is closed automatically.
- Step3: User starts interacting with process area data.
- Step4: After completion of the user interaction, Process Area data is updated to the database. While updating the data again connection is automatically opened & closed after updating.

Disadvantages of Disconnected Architecture: -

- Cannot be used where immediate updating of data is required.
- Not suitable where security of the data is most important

Disconnected Data Cycle Figure from MSDN:

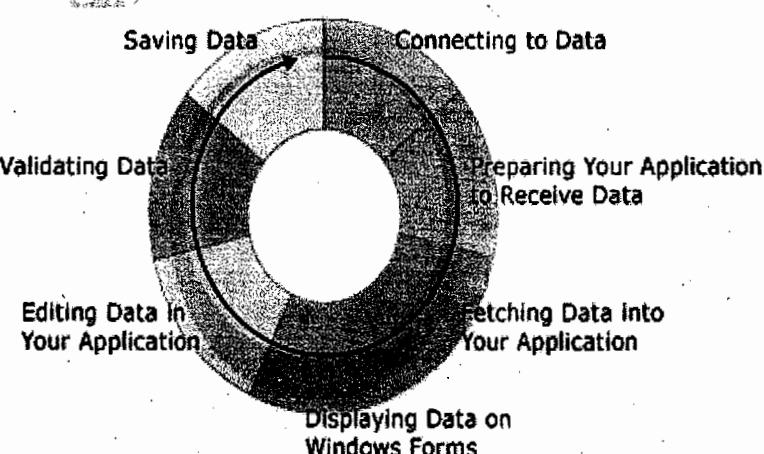
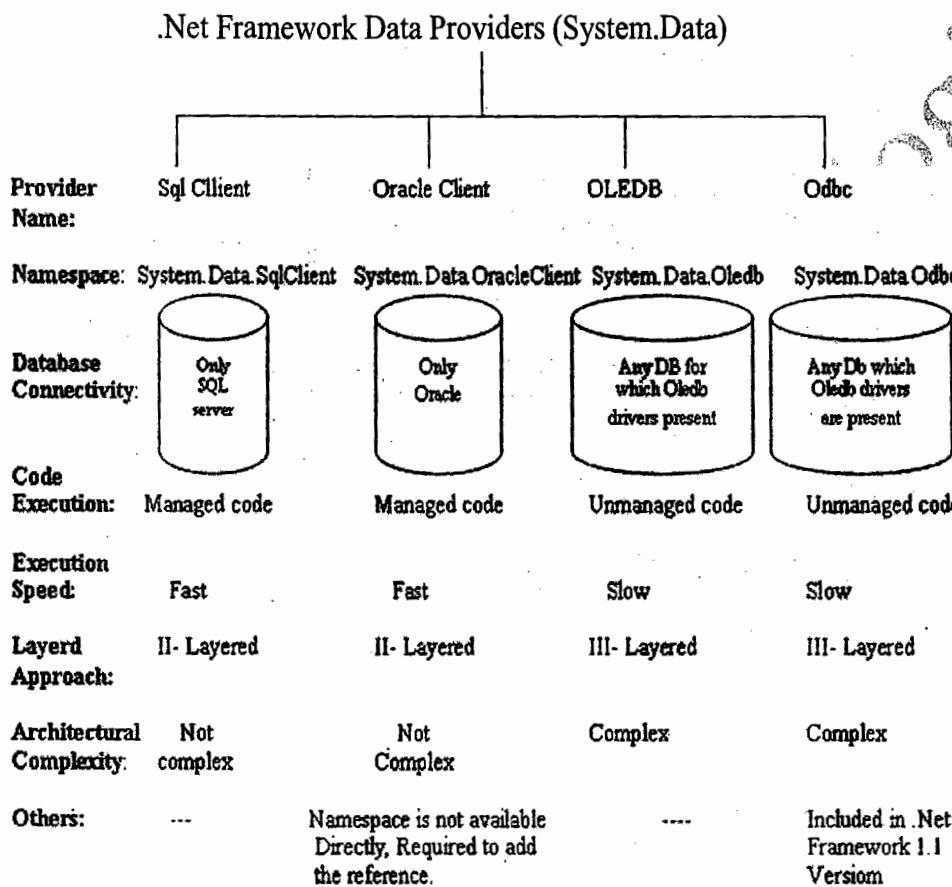


Figure 15.8

16.3.4 Data Providers in ADO.NET: -

- Data providers in ADO.Net will help for connecting to database, executing commands & retrieving results from the database.
- .NET Framework data providers are lightweight, creating a minimal layer between the data source and code, increasing performance without sacrificing functionality.
- All data providers or ADO.Net class library is available in system.data name space.
- ADO.Net supports following data providers.

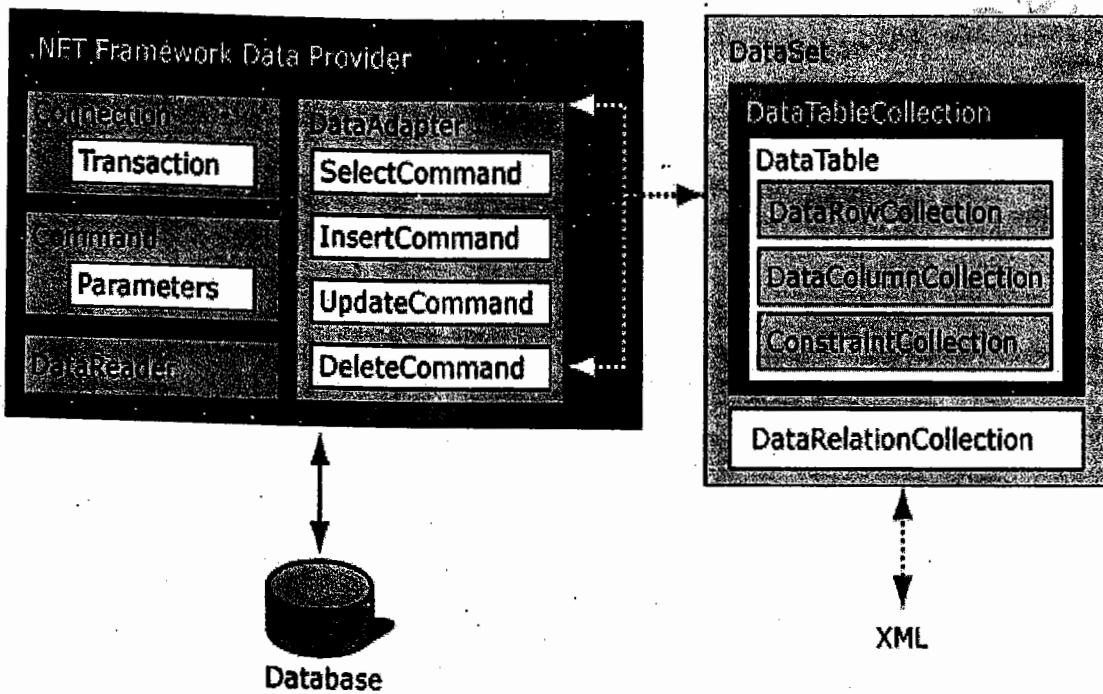


16.3.5 Objects in ADO.NET

Srno	Object Name	Class(es) Name(s)
01	Connection	SqlConnection / OracleConnection / OLEDBConnection / ODBCConnection
02	Transaction	SqlTransaction / OracleTransaction / OLEDBTransaction / ODBCTransaction
03	Command	SqlCommand / OracleCommand / OLEDBCommand / ODBCCommand
04	Parameter	SqlParameter / OracleParameter / OLEDBParameter / ODBCParameter
05	DataReader	SqlDataReader / OracleDataReader / OLEDBDataReader / ODBCDataReader
06	DataAdapter	SqlDataAdapter / OracleDataAdapter / OLEDBDataAdapter / ODBCDataAdapter

07	CommandBuilder	SqlCommandBuilder / OracleCommandBuilder / OLEDBCommandBuilder / ODBCCommandBuilder
08	Dataset	Dataset
09	DataTable	DataTable
10	DataRelation	DataRelation
11	DataRow	DataRow
12	DataColumn	DataColumn
13	Constraint	Constraint
14	DataView	DataView

16.3.6 ADO.NET Architecture



Chapter 17

Working with Connection Object

17.1 Introduction

- Connection object is used to establish connection to the database.
- Using connection object it is not possible to perform any kind of operations like inserting, updating, deleting the records on database objects like Tables, Views, Stored Procedures etc,

17.2 Properties and Methods with Connection Class:

Srno	Property and options	Type	Description
01	ConnectionString	string	Gets or sets the connection string value used to open connection to the database.
02	ConnectionTimeout	int	Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error. Default time is 15 seconds
03	DataSource	string	Gets the name of the instance of the Database Server to which to connect.
04	PacketSize	int	Gets the size (in bytes) of network packets used to communicate with an instance of the Database Server
05	State Closed Open Connecting Executing Fetching Broken	enum	Indicates the state of the Connection Object, all the connection state values are available in ConnectionState enumeration
06	ServerVersion	string	Gets a string that contains the version of the instance of the Database to which the client is connected.

Methods with Connection Class

01	BeginTransaction()	Starts a database transaction
02	Close()	Used to Close an opened connection to the database. This is the preferred method of closing any open connection.
03	CreateCommand()	Creates and returns a Command object associated with the respective Connection class
04	GetSchema()	Returns schema information for the data source of this Connection.
05	Open()	Opens a database connection with the property settings specified by the ConnectionString.

17.3 Steps to work with Connection Object

- **Step1:** Declare Connection Object
- **Syntax:** Class Name Object Name;
- **Ex:** SqlConnection Con;

- **Step2:** Define Connection Object
- **Syntax:** Object Name=new Class Name("Connection String");
- **Ex:** Con=new SqlConnection("Connection String");
- **Step3:** Open the Connection
- **Syntax:** Connection Object Name.Open();
- **Ex:** Con.Open();

17.4

➤
➤Exa
Pro

Co

17.4 Connection Strings to connect to the various Databases using Different Data Providers

17.4.1 For SqlServer Using SqlClient Provider:

- **Syntax:** Server=ServerName;User Id=
UserName;Password=UserPassword;Database=DBName
- **Ex:** Server=SaiServer;User Id=sa;Password=sai;Database=Employee

17.4.2 For Oracle Using OracleClient Provider:

- **Syntax:** User Id=UserName;Password=UserPassword;Data Source=HostStringName
- **Ex:** User Id=scott;Password=tiger;Data Source=sai

Steps to find the Host String Name:

Search for the File TnsNames.Ora in your System → Select the File → Click with Right Mouse Button → Click on Open → Select the Notepad Application → Click on OK → Check for the Service_Name=sai (Service_Name value is the Host String Name).

17.4.3 For SqlServer Using OLEDB Provider:

- **Syntax:** Provider=ProviderName;Server=ServerName;User Id=
UserName;Password=UserPassword;Database=DBName
- **Ex:** Provider=SQLOLEDB;Server=SaiServer;User Id=sa;Password=sai;Database=Employee

17.4.4 For Oracle Using OLEDB Provider:

- **Syntax:** Provider=ProviderName;User Id= UserName;Password=UserPassword;Data
Source=HostStringName
- **Ex:** Provider=MSDAORA.1;User Id=scott;Password=tiger;Data Source=sai

17.4.5 For MSAccess Using OLEDB Provider:

- **Syntax:** Provider=ProviderName;Data Source=DBFileNameUsingPath
- **Ex:** Provider=MicroSoft.JET.OLEDB.4.0;Data Source=sD:\\Sample.Mdb

17.4.6 For SqlServer Using ODBC Provider:

- **Syntax:** DSN=UserDataSourceName;User Id=UserName;Password=UserPassword
- **Ex:** DSN=abc;User Id=sa;Password=sai

Exa
Pro

Co

Firs
Step
Rig
Tab
Cli

17.4.7 For Oracle Using ODBC Provider:

- Syntax: DSN=UserDataSourceName;User Id=UserName;Password=UserPassword
- Ex: DSN=abc;User Id=scott;Password=tiger

Example 17.1

Program to connect to SQLServer using SqlClient Provider: -

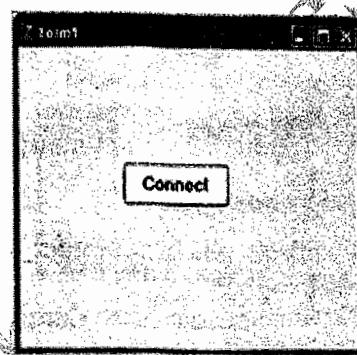
Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            SqlConnection Con;
            Con = new OdbcConnection("Server=SaiServer;User Id=sa;Password=sai; Database=
Employee");
            Con.Open();
            MessageBox.Show("Connection Successful");
        }
    }
}
```

Design of Form



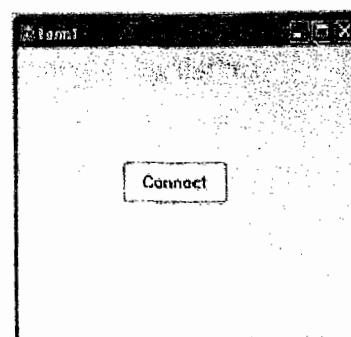
Example 17.2

Program to connect to Oracle using OracleClient Provider: -

Code:

First add the reference to System.Data.OracleClient Namespace
Steps: Go to Solution Explorer → Select Solution → Click with Right Mouse Button → Click on Add Reference → Select .NET Tab page → Select System.Data.OracleClient Namespace → Click on OK → Write the following code

Design of Form



```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OracleClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            OracleConnection Con;
            Con = new OracleConnection("User Id=scott;Password=tiger;Data Source=Sai");
            Con.Open();
            MessageBox.Show("Connection Successful");
        }
    }
}

```

Example 17.3
Program to connect to SQLServer using OleDbProvider: -

Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

```

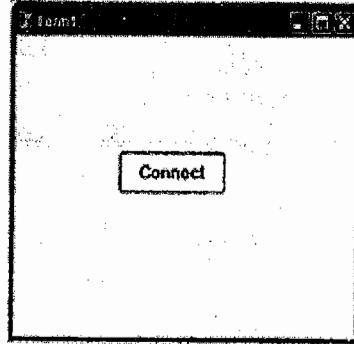
```

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {

```

Design of Form



```

        OleDbConnection Con;
        Con = new OleDbConnection("Provider=SQLOLEDB;Server=SaiServer;User Id=sa;
        Password=sai;Database=Employee");
        Con.Open();
        MessageBox.Show("Connection Successful");
    }
}
}

```

Example 17.4

Program to connect to Oracle using OleDbProvider: -

Code:

```

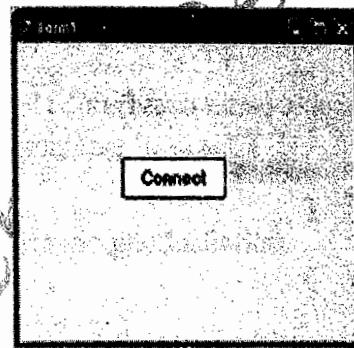
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            OleDbConnection Con;
            Con = new OleDbConnection("Provider=MSDAORA.1; User Id=scott;Password=tiger;
            Data Source=Sai ");
            Con.Open();
            MessageBox.Show("Connection Successful");
        }
    }
}

```

Design of Form



Example 17.5

Program to connect to MSAccess using OleDbProvider: -

Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            OleDbConnection Con;
            Con = new OleDbConnection("Provider=Microsoft.JET.OLEDB.4.0;Data Source=" +
                "D:\\Sample.mdb ");
            Con.Open();
            MessageBox.Show("Connection Successful");
        }
    }
}

```

Example 17.6

Program to connect to SqlServer using OdbcProvider: -

Code:

First Create User Data Sorce Name(DSN)

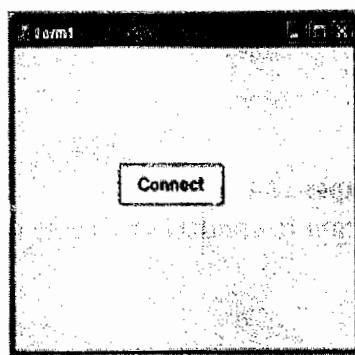
Steps: Go to Control Panel → Double Click on Administrative Tools → Double Click on Data Sources (ODBC) → Select User DSN → Click on Add... Button → Select SQL Native Client → Click on finish → Type Data Source Name (abc) Type Server Name (SaiServer) → Select the option “With SQL Server Authentication using a login ID and Password entered by the user → Click on Next → Activate “Change the default databse to” option → Select the Required Database (Employee) → Click on Next → Click on finish → Click on OK → Write the following code

```

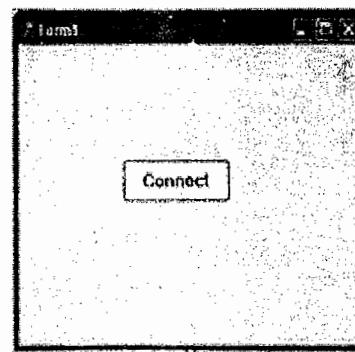
using System;
using System.Collections.Generic;

```

Design of Form



Design of Form



```
using System.ComponentModel;
using System.Data;
using System.Data.Odbc;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            OdbcConnection Con;
            Con = new OdbcConnection("DSN=abc;User Id=sa;Password=abc");
            Con.Open();
            MessageBox.Show("Connection Successful");
        }
    }
}
```

Chapter 18

Command Object

18.1 Introduction

- Command object is used to perform the required operations on the database objects (Tables, Stored Procedure, Views etc) like Inserting, Deleting, Updating records, Creating, Altering, Dropping Tables, Stored Procedures, Views etc after establishing the connection.
- Command object will work with connection oriented architecture.

18.2 Properties and Methods with Command Class

Srno	Property and options	Type	Description
01	CommandText	string	Used to set or get the required command text value, CommandText can be any one from SQL Query or Stored Procedure Name or Table Name.
02	CommandType Text (Default) StoredProcedure TableDirect	enum	Indicates (sets or gets) the value of the command text. Use CommandType like Text: When CommandText is Sql Query StoreProcedure: When CommandText is stored procedure name Text: When CommandText is table name
03	CommandTimeOut	Int	Used to set or get the wait time in seconds before terminating the attempt to execute a command and generating an error. Default time is 30 seconds
04	Connection	Connection class	Used to set or get the Connection object name to be used with the Command object.
05	Parameters	collection	Used to set or get the parameters names, their values required to be passed to the stored procedures of the database or with the parameterized queries.
06	Transaction	Truncation class	Used to set or get group of SQL queries to be executed with the help of T-SQL transaction at database.

Execution Methods with Command Class

01	ExecuteNonQuery()	Check the following table for description of the three methods
02	ExecuteReader()	
03	ExecuteScalar()	

Srno
01

02

03

04

18.3

Exa
Pro

Co

18.2.1 Comparision among Execution Methods of Command Class

Srno	ExecuteNonQuery	ExecuteReader	ExecuteScalar
01	Will work with Action Queries only (Create, Alter, Drop, Insert, Update, Delete)	Will work with Action and Non - Action Queries (Select)	Will work with Non - Action Queries that contain aggregate functions
02	Returns the count of rows effected by the Query	Returns the collection of rows selected by the Query	Returns the first row and first column value of the query result
03	Return type is int	Return type is DataReader	Return type is object
04	Return value is optional and can be assigned to an integer variable	Return value is compulsory and should be assigned to an another object DataReader	Return value is compulsory and should be assigned to a variable of required type

18.3 Steps to work with Command Object

- **Step1:** Declare Command Object
- **Syntax:** Class Name Object Name;
- **Ex:** SqlCommand Cmd;
- **Step2:** Define Command Object
- **Syntax:** Object Name=new Class Name("CommandText", Connection Object);
- **Ex:** Cmd=new SqlCommand("Delete EmpDetails where EmpId=105",Con);
- **Step3:** Mention the Command Type
- **Syntax:** CommandObjectName.CommandType=CommandType.value;
- **Ex:** Cmd.CommandType=CommandType.Text;
- **Step4:** Open the Connection
- **Syntax:** Connection Object Name.Open();
- **Ex:** Con.Open();
- **Step5:** Execute the Command Object
- **Syntax:** Command Object Name.ExecutionMethod();
- **Ex:** Cmd.ExecuteNonQuery();

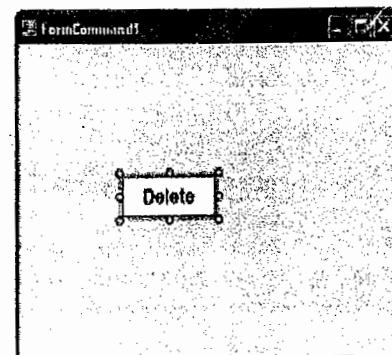
Example 18.01

Program to connect to Delete a Record using SqlCommand Class

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand1 : Form
    {
        public FormCommand1()
        {
            InitializeComponent();
        }
    }
}
```

Design of Form



```

    {
        InitializeComponent();
    }
    private void btnDelete_Click(object sender, EventArgs e)
    {
        SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        string S = "Delete EmpDetails Where EmpId=109";
        SqlCommand Cmd;
        Cmd = new SqlCommand(S, Con);
        Cmd.CommandType = CommandType.Text;
        Con.Open();
        Cmd.ExecuteNonQuery();
        Con.Close();
    }
}

```

- In the above example we are not displaying any message to the user and user does not know whether record is deleted from database or not.
- To display the message to the user we can write the code like:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand1 : Form
    {
        public FormCommand1()
        {
            InitializeComponent();
        }
        private void btnDelete_Click(object sender, EventArgs e)
        {
            SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
            DataBase=Employee");
            string S = "Delete EmpDetails Where EmpId=109";
            SqlCommand Cmd;
            Cmd = new SqlCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            int i=Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}

```

- We can also write the above code using command class properties like:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand1 : Form
    {
        public FormCommand1()
        {
            InitializeComponent();
        }
        private void btnDelete_Click(object sender, EventArgs e)
        {
            SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
DataBases=Employee");
            string S = "Delete EmpDetails Where EmpId=109";
            SqlCommand Cmd = new SqlCommand();
            Cmd.CommandText = S;
            Cmd.CommandType = CommandType.Text;
            Cmd.Connection = Con;
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}

```

Example 18.2

Program to Delete a Record by Using the Non – Parameterized Query: -

Code:

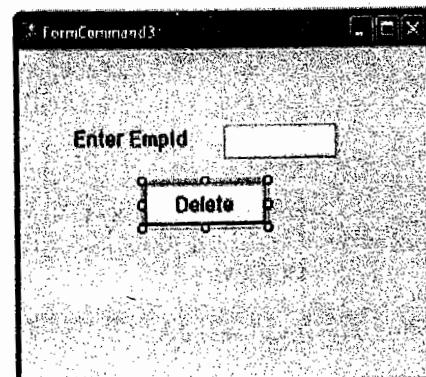
```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class FormCommand3 : Form
    {
        public FormCommand3()
        {

```

Design of Form



```

    {
        InitializeComponent();
    }
    private void cmdDelete_Click(object sender, EventArgs e)
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
        Id=sa;Password=abc;DataBase=Employee");
        string S = "Delete EmpDetails Where EmpId=@X";
        SqlCommand Cmd;
        Cmd = new SqlCommand(S, Con);
        Cmd.CommandType = CommandType.Text;
        Cmd.Parameters.AddWithValue("@X", txtEmpId.Text);
        Con.Open();
        int i = Cmd.ExecuteNonQuery();
        Con.Close();
        MessageBox.Show(i + " Record(s) Deleted");
    }
}

```

Example 18.3

Program to Delete a Record by Using the Parameterized Query: -

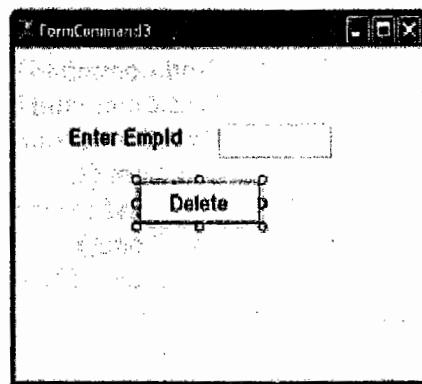
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand3 : Form
    {
        public FormCommand3()
        {
            InitializeComponent();
        }
        private void cmdDelete_Click(object sender, EventArgs e)
        {
            SqlConnection Con = new SqlConnection("Server=satya;User
            Id=sa;Password=abc;DataBase=Employee");
            string S = "Delete EmpDetails Where EmpId=@X";
            SqlCommand Cmd;
            Cmd = new SqlCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Cmd.Parameters.AddWithValue("@X", txtEmpId.Text);
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}

```

Design of Form



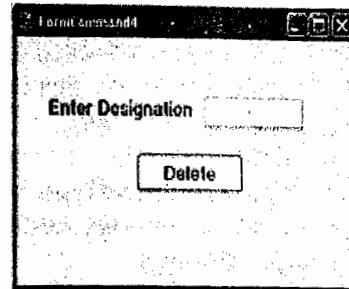
Example 18.04

Program to Delete a Record by Using the Non - Parameterized Query: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand4 : Form
    {
        public FormCommand4()
        {
            InitializeComponent();
        }
        private void cmdDelete_Click(object sender, EventArgs e)
        {
            SqlConnection Con = new SqlConnection("Server=satya;User
                Id=sa;Password=abc;DataBase=Employee");
            string S = "Delete EmpDetails Where Designation=" + txtDesignation.Text + "";
            SqlCommand Cmd;
            Cmd = new SqlCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}
```

Design of Form



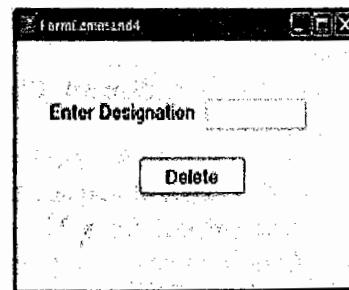
Example 18.5

Program to Delete a Record by Using the Parameterized Query: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand4 : Form
    {
        public FormCommand4()
    }
```

Design of Form



```

    {
        InitializeComponent();
    }
    private void cmdDelete_Click(object sender, EventArgs e)
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
        Id=sa;Password=abc;DataBase=Employee");
        string S = "Delete EmpDetails Where Designation=@X";
        SqlCommand Cmd;
        Cmd = new SqlCommand(S, Con);
        Cmd.CommandType = CommandType.Text;
        Cmd.Parameters.AddWithValue("@X", txtDesignation.Text);
        Con.Open();
        int i = Cmd.ExecuteNonQuery();
        Con.Close();
        MessageBox.Show(i + " Record(s) Deleted");
    }
}
}

```

Example 18.06

Program to Insert, Update and Delete Records by Using the Non -Parameterized Query: -

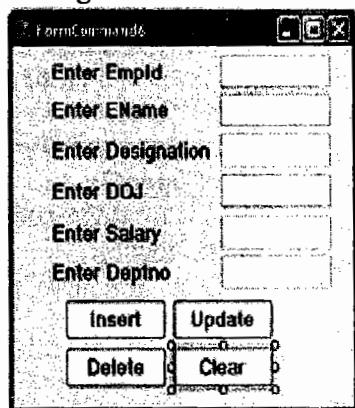
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand6 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
        Id=sa;Password=abc;DataBase=Employee");
        SqlCommand Cmd;
        public FormCommand6()
        {
            InitializeComponent();
        }
        private void cmdInsert_Click(object sender, EventArgs e)
        {
            String S = "Insert into EmpDetails Values(" + txtEmpId.Text + "," + txtEName.Text +
            "," + txtDesignation.Text + "," + txtDOJ.Text + "," + txtSalary.Text + "," +
            txtDeptno.Text + ")";
            Cmd = new SqlCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Inserted");
        }
}

```

Design of Form



Ex:
Pr

Co

```

}

private void cmdUpdate_Click(object sender, EventArgs e)
{
    string S = "Update EmpDetails set EName=" + txtEName.Text + ",Designation=" +
    txtDesignation.Text + ",DOJ=" + txtDOJ.Text + ",Salary=" + txtSalary.Text +
    ",Deptno=" + txtDeptno.Text + " Where EmpId=" + txtEmpId.Text;
    Cmd = new SqlCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Updated");
}

private void cmdDelete_Click(object sender, EventArgs e)
{
    string S = "Delete EmpDetails Where EmpId=" + txtEmpId.Text;
    Cmd = new SqlCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Deleted");
}

private void cmdClear_Click(object sender, EventArgs e)
{
    foreach (Control X in this.Controls)
    {
        if (X is TextBox)
            X.Text = "";
    }
}
}
}

```

Example 18.7

Program to Insert, Update and Delete Records by Using the Parameterized Query: -

Code:

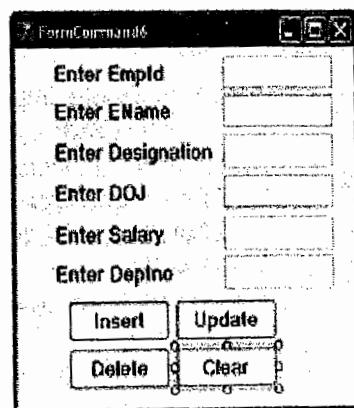
```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class FormCommand7 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
Id=sa;Password=abc;DataBase=Employee");

```

Design of Form



```

SqlCommand Cmd;
public FormCommand7()
{
    InitializeComponent();
}
private void cmdInsert_Click(object sender, EventArgs e)
{
    String S = "Insert into EmpDetails Values(@P1,@P2,@P3,@P4,@P5,@P6)";
    Cmd = new SqlCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Cmd.Parameters.AddWithValue("@P1", txtEmpId.Text);
    Cmd.Parameters.AddWithValue("@P2", txtEName.Text);
    Cmd.Parameters.AddWithValue("@P3", txtDesignation.Text);
    Cmd.Parameters.AddWithValue("@P4", txtDOJ.Text);
    Cmd.Parameters.AddWithValue("@P5", txtSalary.Text);
    Cmd.Parameters.AddWithValue("@P6", txtDeptno.Text);
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Inserted");
}

private void cmdUpdate_Click(object sender, EventArgs e)
{
    String S = "Update EmpDetails Set
    EName=@P2,Designation=@P3,DOJ=@P4,Salary=@P5,Deptno=@P6 Where EmpId=@P1";
    Cmd = new SqlCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Cmd.Parameters.AddWithValue("@P1", txtEmpId.Text);
    Cmd.Parameters.AddWithValue("@P2", txtEName.Text);
    Cmd.Parameters.AddWithValue("@P3", txtDesignation.Text);
    Cmd.Parameters.AddWithValue("@P4", txtDOJ.Text);
    Cmd.Parameters.AddWithValue("@P5", txtSalary.Text);
    Cmd.Parameters.AddWithValue("@P6", txtDeptno.Text);
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Updated");
}

private void cmdDelete_Click(object sender, EventArgs e)
{
    string S = "Delete EmpDetails Where EmpId=@X";
    SqlCommand Cmd;
    Cmd = new SqlCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Cmd.Parameters.AddWithValue("@X", txtEmpId.Text);
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Deleted");
}

private void cmdClear_Click(object sender, EventArgs e)
{
    foreach (Control X in this.Controls)
    {
}

```

Exa
Wo
Pro
Co

```
if (X is TextBox)  
    X.Text = "";  
}  
}  
}
```

Example 18.8

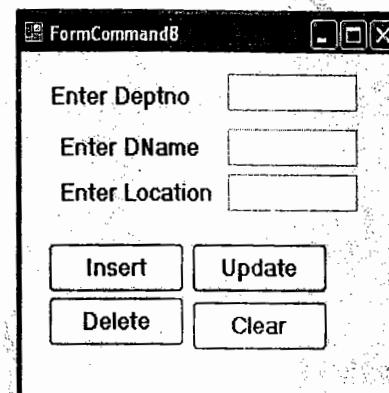
Working with OracleClient Provider

Program to Insert, Update and Delete Records by Using the Non- Parameterized Query:-

Code:

Design of Form

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OracleClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand8 : Form
    {
        OracleConnection Con = new OracleConnection("User Id=scott;Password=tiger;Data
Source=satya");
        OracleCommand Cmd;
        public FormCommand8()
        {
            InitializeComponent();
        }
        private void btnInsert_Click(object sender, EventArgs e)
        {
            string S = "Insert into Dept1 values(" + txtDeptno.Text + "," + txtDName.Text + ","
            + txtLocation.Text + ")";
            Cmd = new OracleCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Inserted");
        }
        private void btnDelete_Click(object sender, EventArgs e)
        {
            string S = "Delete Dept1 Where Deptno=&X";
            Cmd = new OracleCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Cmd.Parameters.AddWithValue("&X", txtDeptno.Text);
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}
```



```

private void btnUpdate_Click(object sender, EventArgs e)
{
    string S = "Update Dept1 set DName='" + txtDName.Text + "',Loc='" +
    txtLocation.Text + "' Where Deptno=" + txtDeptno.Text;
    Cmd = new OracleCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Updated");
}
private void btnClear_Click(object sender, EventArgs e)
{
    foreach (Control C in this.Controls)
    {
        if (C is TextBox)
            this.Text = "";
    }
}
}

```

Example 18.9

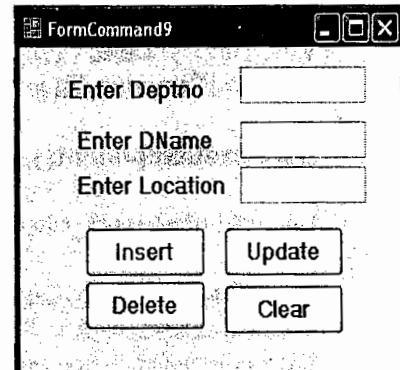
Working with Oledb Provider

Program to Insert, Update and Delete Records by Using the Non- Parameterized Query:-
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormCommand9 : Form
    {
        OleDbConnection Con = new OleDbConnection("Provider=MSDAORA.1;User Id=scott;
        Password=tiger;Data Source=satya");
        OleDbCommand Cmd;
        public FormCommand9()
        {
            InitializeComponent();
        }
        private void btnInsert_Click(object sender, EventArgs e)
        {
            string S = "Insert into Dept1 values(" + txtDeptno.Text + "," + txtDName.Text + ","
            + txtLocation.Text + ")";
            Cmd = new OleDbCommand(S, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
        }
    }
}

```



```
int i = Cmd.ExecuteNonQuery();
Con.Close();
MessageBox.Show(i + " Record(s) Inserted");
}
private void btnUpdate_Click(object sender, EventArgs e)
{
    string S = "Update Dept1 set DName=" + txtDName.Text + ",Loc=" + 
    txtLocation.Text + " Where Deptno=" + txtDeptno.Text;
    Cmd = new OleDbCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Updated");
}
private void btnDelete_Click(object sender, EventArgs e)
{
    string S = "Delete Dept1 Where Deptno=" + txtDeptno.Text;
    Cmd = new OleDbCommand(S, Con);
    Cmd.CommandType = CommandType.Text;
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Deleted");
}
private void btnClear_Click(object sender, EventArgs e)
{
    foreach (Control C in this.Controls)
    {
        if (C is TextBox)
            this.Text = "";
    }
}
```

Chapter 19

Working with Stored Procedures

19.1 Why Stored Procedures

- In general when we work with queries from front end like:


```
String S = "Insert into EmpDetails Values(102,'Raju','Programmer','02/02/2009',15000,10);  
Cmd = new SqlCommand(S, Con);  
Cmd.CommandType = CommandType.Text;  
Con.Open();  
int i = Cmd.ExecuteNonQuery();  
Con.Close();  
MessageBox.Show(i + " Record(s) Inserted");
```
- When front end encounters Cmd.ExecuteNonQuery() front end sends request to the database.
- At database side 2 actions will be performed.
 - Compilation of the query.
 - Running of the query.
- At the time of compilation database engine will check for the syntactical errors, and if there any error(s) occur that error(s) will be sent as it is to the front end.
- If there are no errors query will be run and the result will be sent to front end.
- How many times the request may be sent these two actions will be performed.
- But if there are no syntactical errors compilation of the query is not required and will increase burden on the database so performance will be reduced.

19.2 Disadvantages of Working with Queries:

1. Unnecessary compilation will increase burden on the database.
 2. Application performance is reduced.
 3. User will get delayed response.
 4. Possibility of SQL injection attacks.
 5. No code reusability of queries
- To overcome these disadvantages Stored Procedures are used.

19.3 What is a Stored Procedure:

- A stored procedure is a database object which contains precompiled queries.
- Whenever stored procedure is called from front end queries present in stored procedure will not be compiled rather will run directly.

19.4 Advantages of using Stored Procedures:

1. As there is no unnecessary compilation of queries, this will reduce burden on Dbase.
2. Application performance will be improved.
3. User will get quick response.
4. SQL Injection attacks are avoided.
5. Code reusability facility.

19.5 Working With Stored Procedure with out Parameters:

- Syntax to create a stored procedure with out any parameters in SQL Server:

```
Create procedure ProcedureName
```

```
As
```

```
Begin
```

```
    Statements to be executed
```

```
End
```

- Example to create a stored procedure with out any parameters in SQL Server to delete a record using EmpId from EmpDetails table:

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code

```
Create Procedure DelRec
```

```
As
```

```
Begin
```

```
    Delete EmpDetails Where EmpId=105
```

```
End
```

- Click on Execute Tool Button
- You get the message "Command(s) Executed Successfully".

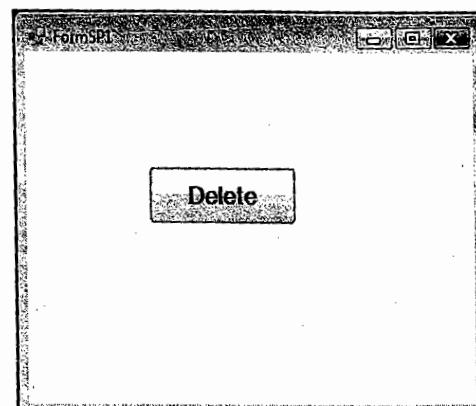
Example 19.1

Program to call a Stored Procedure with out any Parameters from Front End: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADATABASE
{
    public partial class FormSP1 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
Id=sa;Password=abc;DataBase=Employee");
        public FormSP1()
        {
            InitializeComponent();
        }
        private void cmdDelete_Click(object sender, EventArgs e)
        {
            SqlCommand Cmd;
            Cmd = new SqlCommand("DelRec", Con);
            Cmd.CommandType = CommandType.StoredProcedure;
```

Design of Form



```

        Con.Open();
        int i=Cmd.ExecuteNonQuery();
        Con.Close();
        MessageBox.Show(i + " Record(s) Deleted");
    }
}
}
}

```

19.6 Working with Stored Procedure with One Parameter:

- Syntax to create a stored procedure with parameters in SQL Server:

```

Create procedure ProcedureName
    @P1Name Data type(Size), @P2Name Data type(Size),...
    As
    Begin
        Statements to be executed
    End

```

- Example to create a stored procedure with parameters in SQL Server to delete a record using EmpId from EmpDetails table.

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code

```

Create Procedure DelRecP
    @PEmpId int
    As
    Begin
        Delete EmpDetails Where EmpId=@PEmpId
    End

```

- Click on Execute Tool Button
- You get the message “Command(s) Executed Successfully”.

Steps to Work with Parameter Object:

- Step1: Declare command object
 - Ex: SqlCommand cmd.
- Step2: Define command object
 - Ex: Cmd= new SqlCommand("DelRecP", con);
- Step3: Mention CMD Type
 - Ex: Cmd.CommandType = CommandType.StoredProcedure
- Step4: Declare parameter object
 - Syntax: ClassName ObjectName
 - Ex: SqlParameter P1.
- Step5: Define Parameter object
 - Syntax: ObjectName = New ClassName("Parameter name in Database Stored procedure", Data type);
 - Ex: P1=new SqlParameter("@PEMPID", SqlDbType.Int);
- Step6: Store value into parameter

- Syntax: ParameterObjectName.Value=Any data;
 - Ex: P1.Value=102;
- Step7: Add Parameter Object to the Parameters Collections of Command Object
- Syntax: ObjectName.Parameters.Add(ParameterObjectName);
 - Ex: Cmd.Parameters.Add (P1);
- Step8: Open the Connection
- Syntax: ConnectionObject.Open();
 - Ex: Con.Open()
- Step9: Execute the Command Object
- Ex: Cmd.ExecuteNonQuery ();

Example 19.2

Program to call a Stored Procedure with One Parameter from Front End: -

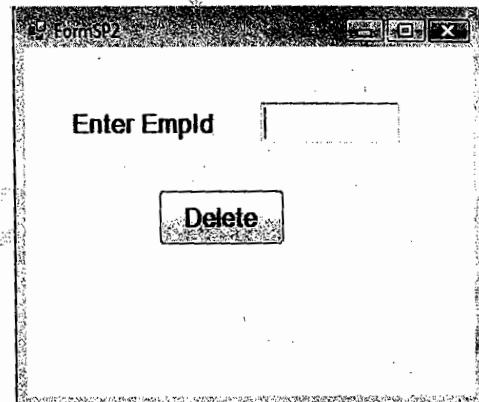
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADATABASE
{
    public partial class FormSP2 : Form
    {
        SqlConnection Con = new SqlConnection("Server=.;User
Id=sa;Password=abc;DataBase=Employee");
        public FormSP2()
        {
            InitializeComponent();
        }
        private void cmdDelete_Click(object sender, EventArgs e)
        {
            SqlCommand Cmd= new SqlCommand("DelRecp", Con);
            Cmd.CommandType = CommandType.StoredProcedure;
            SqlParameter P1;
            P1 = new SqlParameter("@PEmpId", SqlDbType.Int);
            P1.Value = Convert.ToInt32(txtEmpId.Text);
            Cmd.Parameters.Add(P1);
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Deleted");
        }
    }
}

```

Design of Form



19.7 Working with Stored Procedure with Multiple Parameters:

- Example to create a stored procedure with parameters in SQL Server to Insert a record into EmpDetails table.

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code
Create Procedure InsertRecP

```
@PEmpId int, @PName VarChar(20),@PDesignation VarChar(20), @P DOJ Date,  
@PSalary Money,@PDeptNo int
```

As

Begin

```
Insert into EmpDetails values(@PEmpId,@PName,@PDesignation,@P DOJ,  
@PSalary,@PDeptNo)
```

End.

- Click on Execute Tool Button
- You get the message "Command(s) Executed Successfully".

- Example to create a stored procedure with parameters in SQL Server to Update a record into EmpDetails table using EmpId.

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code
Create Procedure UpdateRecP

```
@PEmpId int, @PName VarChar(20),@PDesignation VarChar(20), @P DOJ Date,  
@PSalary Money,@PDeptNo int
```

As

Begin

```
Update EmpDetails set EName=@PName,Designation=@PDesignation,DOJ=@  
P DOJ,Salary=@PSalary,Deptno=@PDeptno where EmpId=@PEmpId
```

End.

- Click on Execute Tool Button
- You get the message "Command(s) Executed Successfully".

Example 19.3

Program to call a Stored Procedure with Multiple Parameters from Front End: -

Code:

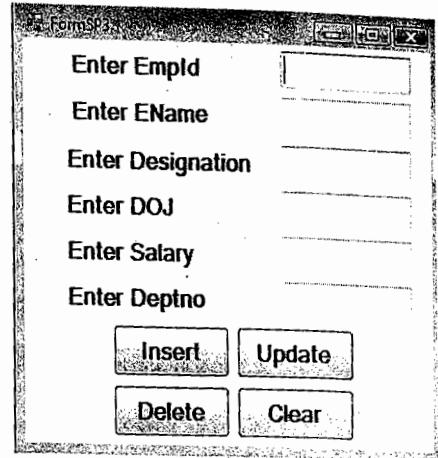
```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Data.SqlClient;
```

Design of Form

```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADATABASE
{
    public partial class FormSP3 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;
User Id=sa; Password=abc; DataBase=Employee");
        SqlCommand Cmd; SqlParameter P1;
        public FormSP3()
        {
            InitializeComponent();
        }
        private void cmdInsert_Click(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("InsertRecP", Con);
            Cmd.CommandType = CommandType.StoredProcedure;
            P1 = new SqlParameter("@PEmpId", SqlDbType.Int);
            P1.Value = Convert.ToInt32(txtEmpId.Text);
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PENName", SqlDbType.VarChar),
            P1.Value = txtEName.Text;
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PDesignation", SqlDbType.VarChar);
            P1.Value = txtDesignation.Text;
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PDOJ", SqlDbType.DateTime);
            P1.Value = Convert.ToDateTime(txtDOJ.Text);
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PSalary", SqlDbType.Money);
            P1.Value = Convert.ToInt32(txtSalary.Text);
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PDeptno", SqlDbType.Int);
            P1.Value = Convert.ToInt32(txtDeptno.Text);
            Cmd.Parameters.Add(P1);
            Con.Open();
            int i = Cmd.ExecuteNonQuery();
            Con.Close();
            MessageBox.Show(i + " Record(s) Inserted");
        }
        private void cmdUpdate_Click(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("UpdateRecP", Con);
            Cmd.CommandType = CommandType.StoredProcedure;
            P1 = new SqlParameter("@PEmpId", SqlDbType.Int);
            P1.Value = Convert.ToInt32(txtEmpId.Text);
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PENName", SqlDbType.VarChar);
            P1.Value = txtEName.Text;
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PDesignation", SqlDbType.VarChar);
            P1.Value = txtDesignation.Text;
            Cmd.Parameters.Add(P1);
            P1 = new SqlParameter("@PDOJ", SqlDbType.DateTime);

```



```
P1.Value = Convert.ToDateTime(txtDOJ.Text);
Cmd.Parameters.Add(P1);
P1 = new SqlParameter("@PSalary", SqlDbType.Money);
P1.Value = Convert.ToInt32(txtSalary.Text);
Cmd.Parameters.Add(P1);
P1 = new SqlParameter("@PDeptno", SqlDbType.Int);
P1.Value = Convert.ToInt32(txtDeptno.Text);
Cmd.Parameters.Add(P1);
Con.Open();
int i = Cmd.ExecuteNonQuery();
Con.Close();
MessageBox.Show(i + " Record(s) Updated");
}

private void cmdDelete_Click(object sender, EventArgs e)
{
    Cmd = new SqlCommand("DelRecp", Con);
    Cmd.CommandType = CommandType.StoredProcedure;
    P1 = new SqlParameter("@PEmpId", SqlDbType.Int);
    P1.Value = Convert.ToInt32(txtEmpId.Text);
    Cmd.Parameters.Add(P1);
    Con.Open();
    int i = Cmd.ExecuteNonQuery();
    Con.Close();
    MessageBox.Show(i + " Record(s) Deleted");
}

private void cmdClear_Click(object sender, EventArgs e)
{
    foreach (Control Ctrl in this.Controls)
    {
        if (Ctrl is TextBox)
        {
            Ctrl.Text = "";
        }
    }
}
```

- Whenever we use `Cmd.Parameters.Add(P1)` internally system will create a collection table with name Parameters like

Parameter Name	Data Type	Value
@PEmpId	Int	100
@PENAME	Varchar	Sai
@PDESIGNATION	Varchar	PL
@PDOJ	Datetime	01/01/2008
@PSALARY	Money	80000
@PDEPTNO	Int	10

- This collection table will be sent to database stored procedure, values will be replaced in parameters and then queries will be executed.

19.8 Working with Stored Procedure with Output Parameters:

- Example to create a stored procedure with parameters in SQL Server to find Salary of an Employee from EmpDetails table by passing EmpId value

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code

Create Procedure GetSal

@PEmpId int,@PSalary Money Output

As

Begin

Select @PSalary=Salary from EmpDetails Where EmpId=@PEmpId

End

- Click on Execute Tool Button
- You get the message "Command(s) Executed Successfully".

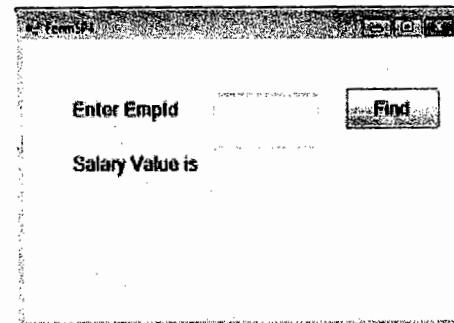
Example 19.4

Program to Find Salary of an Employee When EmpId is given using Output Parameter in Stored Procedure:-

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADATABASE
{
    public partial class FormSP4 : Form
    {
        SqlConnection Con = new SqlConnection("Server=.;User Id=sa;Password=abc;
DataBAsE=Employee");
        SqlCommand Cmd;
        public FormSP4()
        {
            InitializeComponent();
        }
        private void cmdFind_Click(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("GetSal", Con);
            Cmd.CommandType = CommandType.StoredProcedure;
            SqlParameter P1 = new SqlParameter("@PEmpId", SqlDbType.Int);
            P1.Value = Convert.ToInt32(txtEmpId.Text);
            Cmd.Parameters.Add(P1);
            P1=new SqlParameter("@PSalary",SqlDbType.Money);
```

Design of Form



```

P1.Direction = ParameterDirection.Output;
Cmd.Parameters.Add(P1);
Con.Open();
Cmd.ExecuteNonQuery();
Con.Close();
if (Cmd.Parameters["@PSalary"].Value.ToString() == "")
    MessageBox.Show("Record Not Found");
else
    txtSalary.Text=Cmd.Parameters["@PSalary"].Value.ToString();
}
}
}

```

- Example to create a stored procedure with parameters in SQL Server to find a record data from EmpDetails table by passing EmpId value.

- Go to Sql Server DataBase
- Select the Database Employee
- Double Click on Programmability
- Select Stored Procedures option
- Click on NewQuery option (Tool Button)
- A New Query window will Open, Type the following Code

Create Procedure GetData

```

@PEmpId int,@PENName VarChar(20) Output,@PDesignation VarChar(20)
Output,@PDOJ DateTime Output,@PSalary Money Output,@PDeptno int Output
As
Begin
```

Select
 @PENName=EName,@PDesignation=Designation,@PDOJ=DOJ,@PSalary=Salary,@PD
 eptno=Deptno from EmpDetails Where EmpId=@PEmpId

End

- Click on Execute Tool Button
- You get the message “Command(s) Executed Successfully”.

Example 19.5

Program to Find a Record of an Employee Where EmpId is given using Output Parameters in Stored Procedure: -

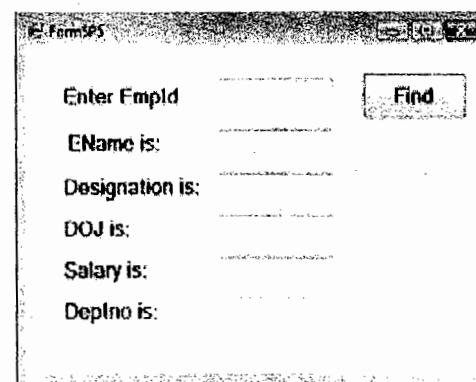
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADATABASE
{
    public partial class FormSP5 : Form
    {

```

Design of Form



```
SqlConnection Con = new SqlConnection("Server=.;User Id=sa;Password=abc;  
DataBase=Employee");  
SqlCommand Cmd; SqlParameter P1;  
public FormSP5()  
{  
    InitializeComponent();  
}  
private void cmdFind_Click(object sender, EventArgs e)  
{  
    Cmd = new SqlCommand("GetData", Con);  
    Cmd.CommandType = CommandType.StoredProcedure;  
    P1 = new SqlParameter("@PEmpId", SqlDbType.Int);  
    P1.Value = Convert.ToInt32(txtEmpId.Text);  
    Cmd.Parameters.Add(P1);  
    P1 = new SqlParameter("@PENAME", SqlDbType.VarChar, 20);  
    P1.Direction = ParameterDirection.Output;  
    Cmd.Parameters.Add(P1);  
    P1 = new SqlParameter("@PDESIGNATION", SqlDbType.VarChar, 20);  
    P1.Direction = ParameterDirection.Output;  
    Cmd.Parameters.Add(P1);  
    P1 = new SqlParameter("@PDOJ", SqlDbType.DateTime);  
    P1.Direction = ParameterDirection.Output;  
    Cmd.Parameters.Add(P1);  
    P1 = new SqlParameter("@PSalary", SqlDbType.Money);  
    P1.Direction = ParameterDirection.Output;  
    Cmd.Parameters.Add(P1);  
    P1 = new SqlParameter("@PDeptno", SqlDbType.Int);  
    P1.Direction = ParameterDirection.Output;  
    Cmd.Parameters.Add(P1);  
    Con.Open();  
    Cmd.ExecuteNonQuery();  
    Con.Close();  
    txtEName.Text = Cmd.Parameters["@PENAME"].Value.ToString();  
    txtDesignation.Text = Cmd.Parameters["@PDESIGNATION"].Value.ToString();  
    txtDOJ.Text = Cmd.Parameters["@PDOJ"].Value.ToString();  
    txtSalary.Text = Cmd.Parameters["@PSalary"].Value.ToString();  
    txtDeptno.Text = Cmd.Parameters["@PDeptno"].Value.ToString();  
}
```

19.9 Temporary Stored Procedures:

- There are two types of Temporary Stored Procedures.
 - Private / Local Temporary Stored Procedures
 - Public / Global Temporary Stored Procedures
 - **Private / Local Temporary Stored Procedures**
 - These are created with the # prefix added to the procedure name.
 - These are executed by the connection that created it.
 - These are automatically deleted when the connection created it is closed.

➤ Public / Global Temporary Stored Procedures

- These are created with the ## prefix added to the procedure name.
 - Any connection can execute the global temporary stored procedure.
 - A global temporary stored procedure exists until the connection used by the user who created the procedure is closed and any currently executing versions of the procedure by any other connections are completed.
 - Once the connection that was used to create the procedure is closed, no further execution of the global temporary stored procedure is allowed. Only those connections that have already started executing the stored procedure are allowed to complete.

➤ Use of Temporary Stored Procedures

- Temporary stored procedures are useful when connecting to earlier versions of SQL Server that do not support the reuse of execution plans for Transact-SQL statements or batches.

➤ **Check yourself:**

1. What is a Stored Procedure?
 2. What are the advantages of Stored Procedure?
 3. How do you create a stored procedure?
 4. How do you call a stored procedure from front end?
 5. How do you create a stored procedure with parameters?
 6. How do you pass parameters to stored procedure from front end?
 7. What are the various data type class enumerations available in respect to parameters?
 8. What symbol is used to precede a parameter in SQLServer?
 9. What are Temporary stored procedures Explain?
 10. List out the types of Temporary stored procedures?
 11. What are Local Temporary stored procedures explain?
 12. What are Global Temporary stored procedures explain?
 13. How do you create Local Temporary stored procedures?
 14. How do you create Global Temporary stored procedures?
 15. When a Local Temporary stored procedure is deleted?
 16. When a Global Temporary stored procedure is deleted?
 17. Where Temporary Stored Procedures are stored?
 18. What Tool is used to debug a Stored Procedure in SQL Server?
 19. Explain about in, out and inout parameters of Stored Procedures?
 20. What are SQL Injection attacks and how can we prevent using stored procedures?
 21. Why stored procedures are used?

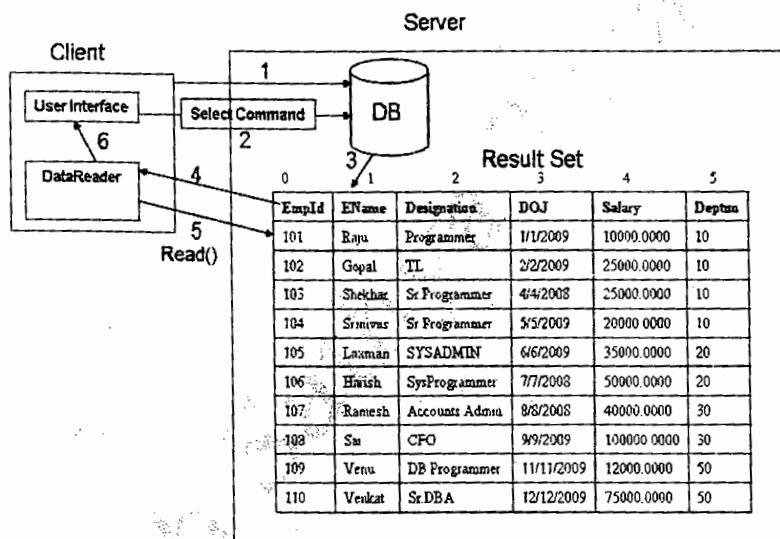
Chapter 20

Working with DataReader

20.1 Introduction

- DataReader will work with connection oriented architecture.
- DataReader is read only object ie, using DataReader we can just read the data from the database, and we can't perform any operations like Inserting, Updating and Deleting records etc.
- DataReader is sequential only object, i.e can read the records one by one only.
- There is no mechanism to move to N^{th} record directly.
- DataReader is Forward only object i,e it is possible to read current record to next record only and is not possible to read the previous record from current record.
- DataReader will use the Command object to fetch the data from database.
- DataReader will reserve the connection object ie, as long as we are working with DataReader connection object can't be used for other purpose.

20.2 How DataReader will work (Architecture of DataReader)



- Step1: Establish Connection to the Server.
- Step2: Client sends request to the Server using Command Object for the required data.
- Step3: Client request is processed at the Server and the Query Execution Result is transferred to the separate memory allotted to the client at server known as ResultSet.
- Step4: Link is created from the header row of ResultSet to DataReader.
- Step5: DataReader starts the data using the Read() method.
- Step6: The data Read by using the DataReader can be displayed in the user interface elements or used for calculations or for any other purpose required.
- After completion of the total reading of the data either DataReader can be closed or Connection is closed.

- When Result Set is Destroyed
 - Result Set is destroyed in following situations.
 - When DataReader is Closed.
 - When Connection is Closed

➤ How to Identify the Data Using DataReader

- In the above figure if DataReader is reading the first record then we identify the filed data using the index value or field name like.
 DR[0] or DR["EmpId"] → 101
 DR[1] or DR["EName"] → Raju
 DR[2] or DR["Designation"] → Programmer
 DR[3] or DR["DOJ"] → 01/01/2009
 DR[4] or DR["Salary"] → 10000
 DR[5] or DR["Deptno"] → 10
- DataReader will return the data with object type.

20.3 Advantages of DataReader

- Using the DataReader can increase application performance both by retrieving data as soon as it is available, and (by default) storing only one row at a time in memory, reducing system overhead.
- The DataReader is a good choice when retrieving large amounts of data because the data is not cached in memory.

20.4 Properties and with DataReader Class:

Srno	Property and options	Type	Description
01	FieldCount	Int	Returns the count of fields present in the result set
02	HasRows	Bool	Returns true if result set contains one or more rows otherwise returns false if there are no rows in the result set
03	IsClosed	Bool	Returns true if DataReader is closed otherwise returns false
04	RecordsAffected	Int	Returns the count of records Inserted, Deleted, Updated when worked with Action Queries, if used with select query returns -1

20.5 Methods with DataReader Class

Srno	Method	Type	Description
01	Close()	Void	Closes the SqlDataReader object.
02	GetName(int i)	String	Gets the name of the specified column.
03	GetOrdinal(string FieldName)	Int	Gets the column index of the given column name.
04	GetSchemaTable()	DataTable	Returns a DataTable that describes the column metadata of the SqlDataReader.
05	NextResult()	Bool	Advances the data reader to the next result, when reading the results of batch Transact-SQL statements. Returns true if there is another result set otherwise returns false
06	Read()	Bool	Advances the DataReader to the next record. Returns true if there is another record otherwise returns false if EOF is encountered

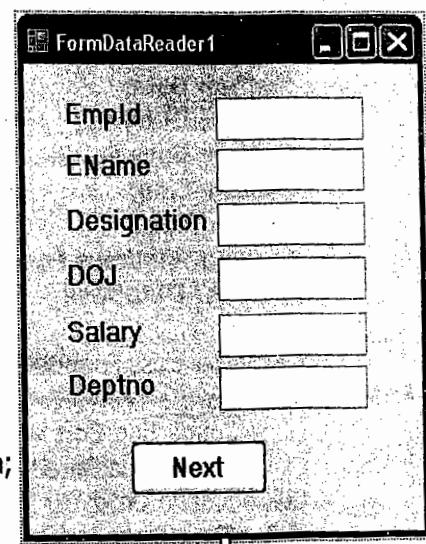
Example 20.1

Program Navigate to Next Record Using DataReader: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDataReader1 : Form
    {
        SqlConnection Con = new SqlConnection("Server=Satya;
User Id=sa;Password=abc;DataBase=Employee");
        SqlCommand Cmd; SqlDataReader DR;
        public FormDataReader1()
        {
            InitializeComponent();
        }
        private void FormDataReader1_Load(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("Select * from EmpDetails", Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            DR = Cmd.ExecuteReader();
            MessageBox.Show("Data is Available in Network Buffer");
        }
        private void btnNext_Click(object sender, EventArgs e)
        {
            if (DR.Read() == true)
            {
                txtEmpId.Text = DR[0].ToString();
                txtEName.Text = DR[1].ToString();
                txtDesignation.Text = DR[2].ToString();
                txtDOJ.Text = DR[3].ToString();
                txtSalary.Text = DR[4].ToString();
                txtDeptno.Text = DR[5].ToString();
            }
            else
            {
                MessageBox.Show("There are No Records");
                DR.Close();
                btnNext.Enabled = false;
            }
        }
    }
}
```

Design of Form



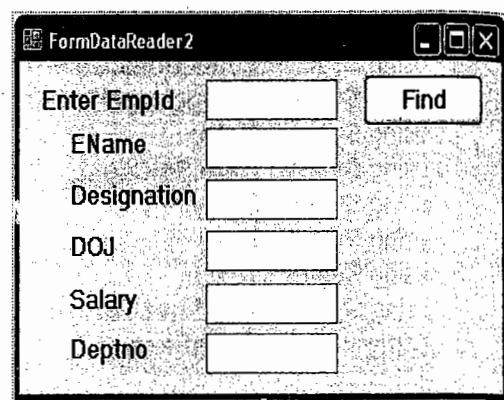
Example 20.2

Program to Find a Record Using DataReader: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDataReader2 : Form
    {
        SqlConnection Con = new SqlConnection("Server=Satya;User Id=sa;
        Password=abc;DataBase=Employee");
        SqlCommand Cmd; SqlDataReader DR;
        public FormDataReader2()
        {
            InitializeComponent();
        }
        private void btnFind_Click(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("Select * from EmpDetails Where EmpId=" + txtEmpId.Text,
            Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            DR = Cmd.ExecuteReader();
            if (DR.Read() == true)
            {
                txtEName.Text = DR[1].ToString();
                txtDesignation.Text = DR[2].ToString();
                txtDOJ.Text = DR[3].ToString();
                txtSalary.Text = DR[4].ToString();
                txtDeptno.Text = DR[5].ToString();
            }
            else
            {
                MessageBox.Show("Record Not Found");
            }
            Con.Close();
        }
    }
}
```

Design of Form

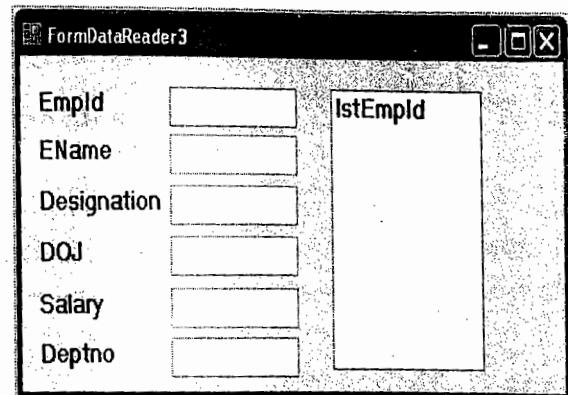


Example 20.3
Program to Get a Record of Selected EmpId: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDataReader3 : Form
    {
        SqlConnection Con = new SqlConnection("Server=Satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlCommand Cmd; SqlDataReader DR;
        public FormDataReader3()
        {
            InitializeComponent();
        }
        private void FormDataReader3_Load(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("Select EmpId from EmpDetails", Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            DR = Cmd.ExecuteReader();
            while (DR.Read() == true)
            {
                lstEmpId.Items.Add(DR[0]);
            }
            DR.Close();
        }
        private void lstEmpId_SelectedIndexChanged(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("Select * from EmpDetails where EmpId=" +
            lstEmpId.SelectedItem.ToString(), Con);
            Cmd.CommandType = CommandType.Text;
            DR = Cmd.ExecuteReader();
            DR.Read();
            txtEmpId.Text = DR[0].ToString();
            txtEName.Text = DR[1].ToString();
            txtDesignation.Text = DR[2].ToString();
            txtDOJ.Text = DR[3].ToString();
            txtSalary.Text = DR[4].ToString();
            txtDeptno.Text = DR[5].ToString();
            DR.Close();
        }
    }
}
```

Design of Form



Result Screen:

The screenshot shows a Windows application window titled "FormDataReader3". On the left, there are six text input fields with the following data:

Empld	110
EName	Sai
Designation	PL
DOJ	10/10/2009
Salary	80000.000
Deptno	10

To the right of these fields is a vertical list box containing a series of employee IDs: 101, 102, 103, 104, 105, 106, 108, 109, and 110. The item "110" is highlighted.

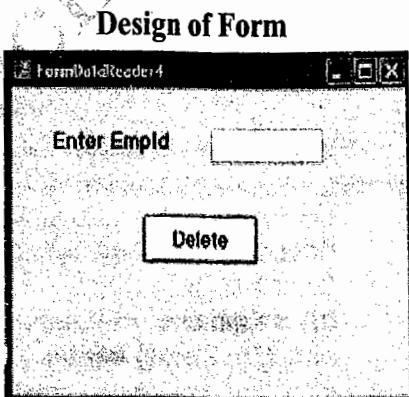
21.1

Example 20.4

Program to Delete a Record Using DataReader: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDataReader4 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User
Id=sa;Password=abc;Database=Employee");
        SqlCommand Cmd; SqlDataReader DR;
        public FormDataReader4()
        {
            InitializeComponent();
        }
        private void btnDelete_Click(object sender, EventArgs e)
        {
            Cmd = new SqlCommand("Delete EmpDetails Where EmpId=" + txtEmpId.Text, Con);
            Cmd.CommandType = CommandType.Text;
            Con.Open();
            DR = Cmd.ExecuteReader();
            MessageBox.Show(DR.RecordsAffected + " Record(s) Deleted");
            Con.Close();
        }
    }
}
```



21.1

➤➤➤➤➤

➤➤➤➤➤

21.1

➤➤➤➤➤

Chapter 21

Working with Dataset

21.1 What is Dataset

- A Dataset is an in-memory object or we can say that memory representation of the data.
- Dataset will work with disconnected architecture
- Dataset stores the data temporarily within the application so that data can be used within the application.
- A Dataset can be either Typed or Untyped.
- When working with dataset it is not required to establish the prior connection to the database.

21.2 External Architecture of Dataset

- Dataset does not know how to interact with database so dataset will use DataAdapter object to interact with database.
- DataAdapter will perform following tasks
 - Gets the data from database and fills into dataset.
 - Updates the dataset data back to database.
- DataAdapter will use command object internally to interact with database.

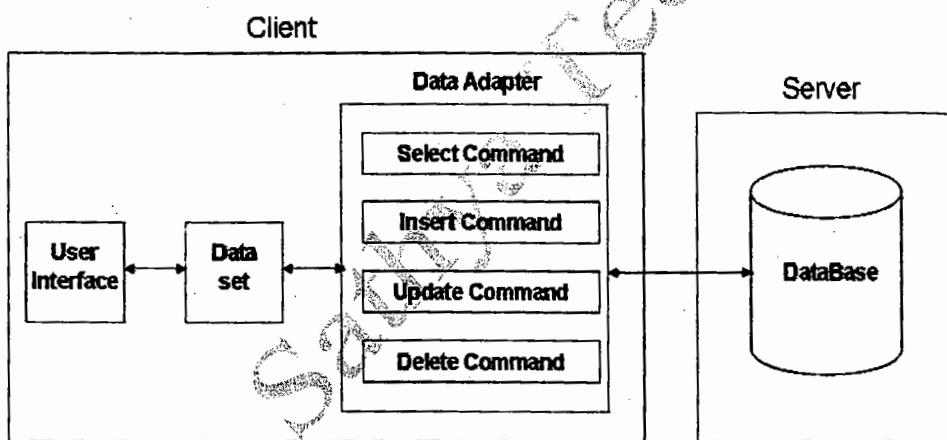


Fig21.1

- DataAdapter will use Select Command (a Command Object with Select Query) to fetch the data from the database and to fill into dataset, at the time of request connection to the database is automatically established and once data is filled into the dataset connection is automatically closed.
- User interacts with dataset data, after completion of the user interaction dataset data can be updated to the database using Insert, Update and Delete commands of the DataAdapter.

21.3 Internal Architecture of Dataset

- Internally Dataset will store the data in the form of collections.
- Dataset will maintain two Collections internally
 - Tables Collection.
 - Relations Collection.
- Every table will have three collections again

- Rows Collection.
 - Columns Collection.
 - Constraints Collection.
- Single Dataset contain any number of tables with in the Tables Collection and any number of relations with in Relations Collection.
- There is no limit for number of tables and relations with in te single dataset as long as memory is available we can store the number of tables and relations.
- The structure of a DataSet is similar to that of a relational database; it exposes a hierarchical object model of tables, rows, columns, constraints, and relationships.

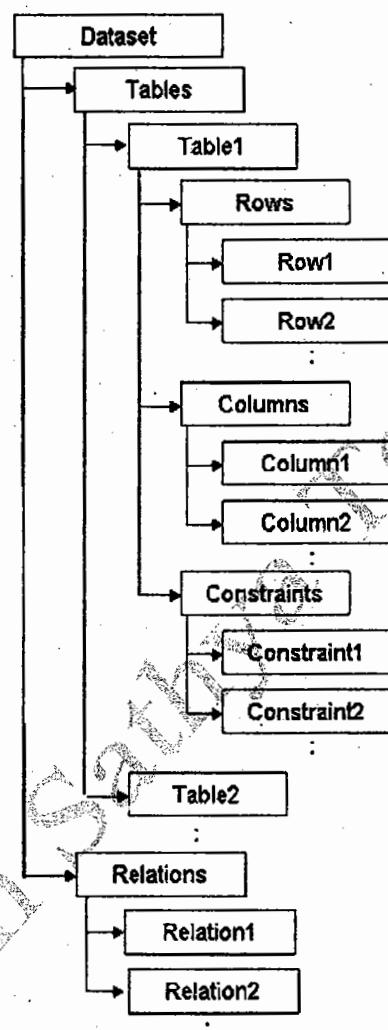


Fig 21.2

If we consider that there are 2 tables available in Dataset named DS like in the fig given below

DataSet (DS)						
EmpDetails	0	1	2	3	4	5
	EmpId	EName	Designation	DOJ	Salary	Deptno
	101	Raju	Programmer	1/1/2007	10000.0000	10
	102	Gopal	TL	2/2/2007	25000.0000	10
	103	Shekhar	Sr.Programmer	4/4/2008	25000.0000	10
	104	Srinivas	Sr.Programmer	5/5/2007	20000.0000	10
	105	Laxman	SYSADMIN	6/6/2007	35000.0000	20
	106	Harish	SysProgrammer	7/7/2008	50000.0000	20
Dept	0	1	2			
	Deptno	DName	Location			
	10	Development	Hyderabad			
	20	Networks	Bangalore			
	30	HR	Mumbai			
	40	Database	Delhi			
	50	Accounts	Chennai			
	60	Operations	Pune			

Fig 21.3

21.4 .1 Identification of Tables in Dataset:

Ds.Tables →

EmpId	EName	Designation	DOJ	Salary	Deptno
101	Raju	Programmer	1/1/2007	10000.0000	10
102	Gopal	TL	2/2/2007	25000.0000	10
103	Shekhar	Sr.Programmer	4/4/2008	25000.0000	10
104	Srinivas	Sr.Programmer	5/5/2007	20000.0000	10
105	Laxman	SYSADMIN	6/6/2007	35000.0000	20
106	Harish	SysProgrammer	7/7/2008	50000.0000	20
107	Ramesh	Accounts Admin	8/8/2008	40000.0000	30
108	Sai	CFO	9/9/2007	100000.0000	30

Deptno	DName	Location
10	Development	Hyderabad
20	Networks	Bangalore
30	HR	Mumbai
40	Database	Delhi
50	Accounts	Chennai
60	Operations	Pune

- As tables are stored in the form of collection each table can be identified using index value or table name like

Ds.Tables[0]

Or

Ds.Tables["EmpDetails"] →

EmpId	EName	Designation	DOJ	Salary	Deptno
101	Raju	Programmer	1/1/2007	10000.0000	10
102	Gopal	TL	2/2/2007	25000.0000	10
103	Shekhar	Sr.Programmer	4/4/2008	25000.0000	10
104	Srinivas	Sr.Programmer	5/5/2007	20000.0000	10
105	Laxman	SYSADMIN	6/6/2007	35000.0000	20
106	Harish	SysProgrammer	7/7/2008	50000.0000	20
107	Ramesh	Accounts Admin	8/8/2008	40000.0000	30
108	Sai	CFO	9/9/2007	100000.0000	30

Deptno	DName	Location
10	Development	Hyderabad
20	Networks	Bangalore
30	HR	Mumbai
40	Database	Delhi
50	Accounts	Chennai
60	Operations	Pune

Ds.Tables[1]

Or

Ds.Tables["Dept"] →

21.5

21.4.2 Identification of Rows in Dataset Table

- As every table will contain collection of rows each row can be identified using index value like

Ds.Tables[0].Rows[1] →

102	Gopal	TL	2/2/2007	25000.0000	10
-----	-------	----	----------	------------	----

Ds.Tables[1].Rows[3] →

40	Database	Delhi
----	----------	-------

21.4.3 Identification of Columns in Dataset Table

- As every table will contain collection of Columns each Column can be identified using index value or column name like

Ds.Tables[0].Columns[2] →
Or

Ds.Tables[0].Columns["Designation"] →

Designation
Programmer
TL
Sr.Programmer
Sr.Programmer
SYSADMIN
SysProgrammer
Accounts Admin
CFO

Ds.Tables[1].Columns[1] →
Or

Or
Ds.Tables[1].Columns["DName"]

DName
Development
Networks
HR
Database
Accounts
Operations

21.4.4 Identification of Cells or Cell Data in Dataset Table

- A cell or cell data with in the dataset table is identified using Matrix notation like

DS.Tables[0].Rows[7][1] →
Or

DS.Tables[0].Columns[1][7]

Sat

DS.Tables[1].Rows[3][2] →
Or

DS.Tables[1].Columns[2][3]

Delhi

21.6

01

02

03

22.

Sr.

01

02

03

04

05

21.5 Steps to Work with Dataset

- **Step1:** Declare DataAdapter object
 - **Syntax:** ClassName ObjectName;
 - **Ex:** SqlDataAdapter Da;
- **Step2:** Declare DataSet object
 - **Syntax:** ClassName ObjectName;
 - **Ex:** Dataset DS;
- **Step3:** Define DataAdapter Object
 - **Syntax:** ObjectName =new ClassName("Select Query", ConnectionObjectName);
 - **Ex:** Da=new SqlDataAdapter("Select * from EmpDetails", Con);
- **Step4:** Define Dataset Object
 - **Syntax:** ObjectName =new ClassName();
 - **Ex:** Ds=new Dataset();
- **Step5:** Fill the data into Dataset using Fill() Method of DataAdapter
 - **Syntax:** DataAdapterObject.Fill(DataSetObject, "Temp / Src Table Name");
 - **Ex:** Da.Fill(Ds,"Table1");

Here table name can be same as in database or different.

21.6 Properties with Dataset Class:

Srno	Property and options	Type	Description
01	CaseSensitive	bool	Gets or sets a value indicating whether string comparisons within DataTable objects are case-sensitive.
02	Relations	Collection	Stores all the relations in the form of collection that link tables and allow navigation from parent tables to child tables.
03	Tables	Collection	Stores all the relations in the form of collection

22. Methods with Dataset Class

Srno	Method	Return Type	Description
01	AcceptChanges()	void	Commits all the changes made to this DataSet since it was loaded or since the last time AcceptChanges was called.
02	Clear()	void	Clears the DataSet of any data by removing all rows in all tables.
03	Clone()	void	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
04	GetSchemaTable()	DataTable	Returns a DataTable that describes the column metadata of the SqlDataReader.
05	Copy()	Dataset	Advances the data reader to the next result, when reading the results of batch Transact-SQL statements. Returns true if there is another result

			set otherwise returns false
06	GetChanges ()	Dataset	Gets a copy of the DataSet containing all changes made to it since it was last loaded, or since AcceptChanges was called.
07	GetXml()	string	Returns the XML representation of the data stored in the DataSet.
08	GetXmlSchema()	string	Returns the XML Schema for the XML representation of the data stored in the DataSet.
09	HasChanges()	bool	Returns true if there are modifications in Dataset otherwise returns false.
10	Merge(DataSet Dataset)	void	Merges a specified DataSet object into the current DataSet.

21.8 Properties with DataAdapter Class:

Srno	Property and options	Type	Description
01	AcceptChangesDuringFill	bool	Gets or sets a value indicating whether AcceptChanges is called on a DataRow after it is added to the DataTable during any of the Fill operations.
02	AcceptChangesDuringUpdate	bool	Gets or sets whether AcceptChanges is called during a Update.
03	DeleteCommand	SqlCommand	Gets or sets a SQL statement or stored procedure name to Delete records from Dataset
04	FillLoadOption		Gets or sets the LoadOption that determines how the adapter fills the DataTable from the DbDataReader.
05	InsertCommand	SqlCommand	Gets or sets a SQL statement or stored procedure name to Insert records from Dataset
06	UpdateCommand	SqlCommand	Gets or sets a SQL statement or stored procedure name to Update records from Dataset

21.9 Methods with DataAdapter Class

Srno	Method	Return Type	Description
01	Fill(DatasetName, "Src / TempTable Name") Or Fill(DatasetName, int StartRecord, int MaxRecords, "Src / TempTable Name")	int	Used to fill the given table data into the Dataset. Returns the The number of rows successfully added to or refreshed in the DataSet. This does not include rows affected by statements that do not return rows. This Method does not fill any constraints of database table into dataset
02	FillSchema(DatasetName, SchemaType, "Src /	DataTable[]	This method will fill the complete table schema including the constraints into dataset

	TempTable Name")		and does not fill the data.
03	Update(DatasetName, "Src / TempTable Name")	int	Calls the respective INSERT, UPDATE, or DELETE statements for each inserted, updated, or deleted row in the specified DataSet from a DataTable named "Table." Returns the number of rows successfully updated from the DataSet.

21.10 DataGridView Control

- The **DataGridView** control provides a powerful and flexible way to display data in a tabular format.
- You can use the **DataGridView** control to show read-only views of a small amount of data, or you can scale it to show editable views of very large sets of data.
- You can extend the **DataGridView** control in a number of ways to build custom behaviors into your applications.
- For example, you can programmatically specify your own sorting algorithms, and you can create your own types of cells.
- You can easily customize the appearance of the **DataGridView** control by choosing among several properties.
- Many types of data stores can be used as a data source, or the **DataGridView** control can operate with no data source bound to it.
- With the **DataGridView** control, you can display and edit tabular data from many different kinds of data sources.
- Binding data to the **DataGridView** control is straightforward and intuitive, and in many cases it is as simple as setting the **DataSource** property.
- When you bind to a data source that contains multiple lists or tables, set the **DataMember** property to a string that specifies the list or table to bind to.
- The **DataGridView** control supports the standard Windows Forms data binding model, so it will bind to instances of classes described in the following list:
 - Any class that implements the **IList** interface, including one-dimensional arrays.
 - Any class that implements the **IListSource** interface, such as the **DataTable** and **DataSet** classes.
 - Any class that implements the **IBindingList** interface, such as the **BindingList(T)** class.
 - Any class that implements the **IBindingListView** interface, such as the **BindingSource** class.
- The **DataGridView** control supports data binding to the public properties of the objects returned by these interfaces or to the properties collection returned by an **ICustomTypeDescriptor** interface, if implemented on the returned objects.

21.11 Properties With DataAdapter Class:

Srno	Property and options	Type	Description
01	AllowUserToAddRows	bool	
02	AllowUserToDeleteRows	bool	
03	AllowUserToOrderColumns	bool	

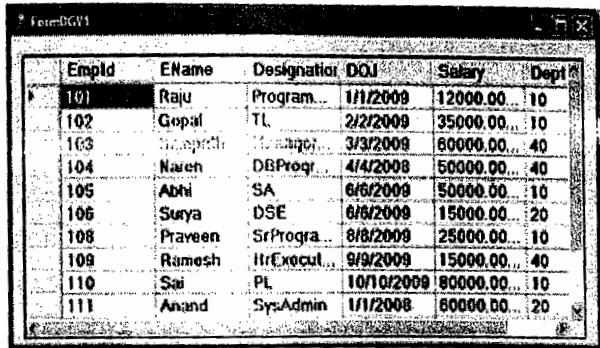
04	AllowUserToResizeColumns	bool	
05	AllowUserToResizeRows	bool	
06	AutoGenerateColumns	bool	
07	ColumnCount	int	
08	Columns(ReadOnly)	DataGridView ColumnCollection	
09	Controls	ControlCollection	
10	CurrentCell	DataGridViewCell	
11	CurrentCellAddress	Point	
12	CurrentRow	DataGridViewRow	
13	DataMember	string	
14	DataSource	object	
15	EditMode	enum	
16	MultiSelect	bool	
17	NewRowIndex	int	
18	RowCount	int	
19	Rows	DataGridViewRow Collection	
20	SelectedCells	DataGridView SelectedCell Collection	
21	SelectedColumns	DataGridView SelectedColumn Collection	
22	SortedColumn	DataGridView Column	
23	SortOrder	Enum	

Example 21.1
Program to Display the Data with in the DataGridView: -

Code:

Design of Form

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDGV1 : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        public FormDGV1()
        {
            InitializeComponent();
        }
        private void FormDGV1_Load(object sender, EventArgs e)
        {
            SqlDataAdapter Da;
            Da = new SqlDataAdapter("Select * from EmpDetails", Con);
            DataSet Ds = new DataSet();
            Da.Fill(Ds, "Table1");
            DGVSample.DataSource = Ds;
            DGVSample.DataMember = "Table1";
        }
    }
}
```



EmpId	EName	Designation	DOJ	Salary	Dept
101	Raju	Program	1/1/2009	12000.00	10
102	Gopal	TL	2/2/2009	35000.00	10
103	Manoj	Manager	3/3/2009	60000.00	40
104	Naren	DBProg	4/4/2009	50000.00	40
105	Abhi	SA	5/5/2009	50000.00	10
106	Surya	DSE	6/6/2009	15000.00	20
108	Praveen	SrProg	8/8/2009	25000.00	10
109	Ramesh	HR Execut	9/9/2009	15000.00	40
110	Sai	PL	10/10/2009	80000.00	10
111	Anand	Sys Admin	1/1/2009	60000.00	20

We can also write the above code like:

Method1 using Command Object:

```
private void FormDGV1_Load(object sender, EventArgs e)
{
    SqlDataAdapter Da;
    Da = new SqlDataAdapter("Select * from EmpDetails", Con);
    Da.Fill(Ds, "Table1");
    DGVSample.DataSource = Ds;
    DGVSample.DataMember = "Table1";
}
```

Method2 with out using Connection Object:

```
private void FormDGV1_Load(object sender, EventArgs e)
{
    SqlCommand Cmd = new SqlCommand("Select * from EmpDetails", ("Server=satya;User Id
    =sa;Password=abc; DataBase=Employee"));
    SqlDataAdapter Da;
```

```

Da = new SqlDataAdapter(Cmd);
Da.Fill(Ds, "Table1");
DGVSample.DataSource = Ds;
DGVSample.DataMember = "Table1";
}

```

Method3 with out using DataMember Property of DataGridView:

```

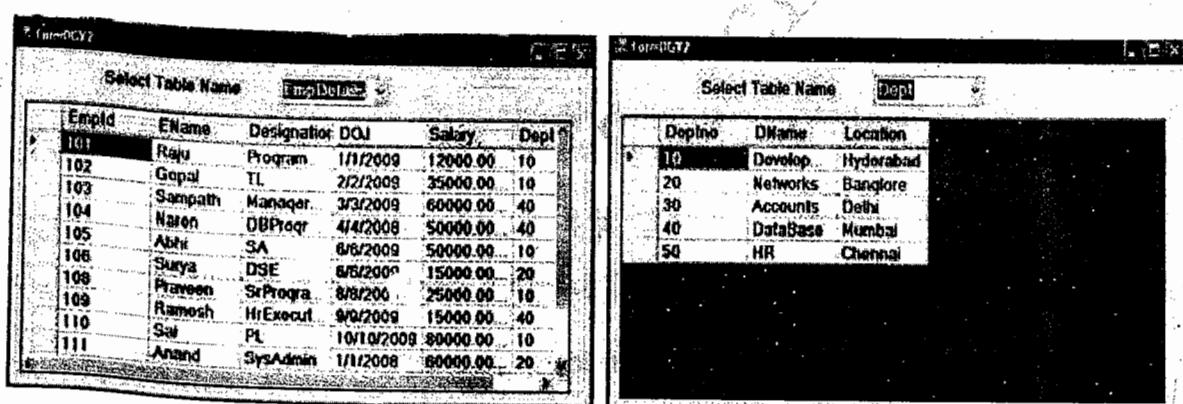
private void FormDGV1_Load(object sender, EventArgs e)
{
    SqlDataAdapter Da;
    Da = new SqlDataAdapter("Select * from EmpDetails", Con);
    DataSet Ds = new DataSet();
    Da.Fill(Ds, "Table1");
    DGVSample.DataSource = Ds.Tables[0];
}

```

Example 21,2

Program to Display the Selected Table Data with in the DataGridView: -

Design of Form



Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WADatabase
{
    public partial class FormDGV2 : Form
    {
        SqlConnection Con = new SqlConnection("server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlDataAdapter Da; DataSet Ds;
        public FormDGV2()
        {

```

```

    InitializeComponent();
}

private void FormDGV2_Load(object sender, EventArgs e)
{
    Da = new SqlDataAdapter("Select * from Dept", Con);
    Ds = new DataSet();
    Da.Fill(Ds, "Dept");
    Da = new SqlDataAdapter("select * from EmpDetails", Con);
    Da.Fill(Ds, "EmpDetails");
}

private void cmbTable_SelectedIndexChanged(object sender, EventArgs e)
{
    DGVSample.DataSource = Ds;
    DGVSampleDataMember = cmbTable.SelectedItem.ToString();
}
}
}

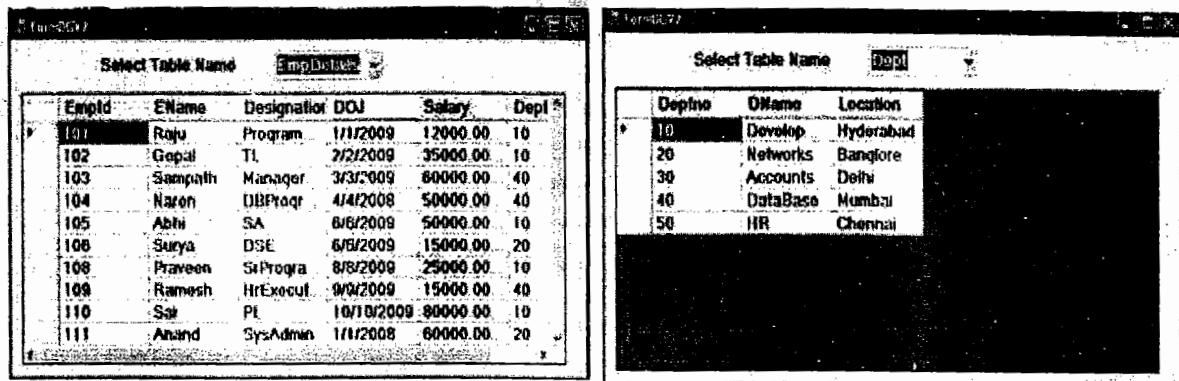
```

- In the above example table names are stored with in the Combobox cmbTable at design time but there may be any number of tables can be present with in the database then to get all the table names and store in the Combobox cmbTable at runtime we use the following example.

Example 21.4

Program to Display the Selected Table Data with in the DataGridView (Retrieving the tables list from the database directly)

Design of Form



Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

```

```
namespace WADATABASE
```

```
{
    public partial class FormDGV3 : Form
    {

```

```
SqlConnection Con = new SqlConnection("server=satya;user id=sa;password=abc;  
DataBase=Employee");  
SqlCommand Cmd; SqlDataAdapter Da; DataSet Ds = new DataSet();  
SqlDataReader Dr;  
public FormDGV3()  
{  
    InitializeComponent();  
}  
private void FormDGV3_Load(object sender, EventArgs e)  
{  
    Cmd = new SqlCommand("select name from sysobjects where xtype='u'", Con);  
    Cmd.CommandType = CommandType.Text;  
    Con.Open();  
    Dr = Cmd.ExecuteReader();  
    while (Dr.Read())  
    {  
        cmbTable.Items.Add(Dr[0]);  
    }  
    Con.Close();  
    for (int i = 0; i < cmbTable.Items.Count; i++)  
    {  
        string S = "Select * from " + cmbTable.Items[i].ToString();  
        Da = new SqlDataAdapter(S, Con);  
        Da.Fill(Ds, cmbTable.Items[i].ToString());  
    }  
    MessageBox.Show("Tables are added to combobox and Data is stored in Dataset");  
}  
private void cmbTable_SelectedIndexChanged(object sender, EventArgs e)  
{  
    DGVSample.DataSource = Ds.Tables[cmbTable.SelectedIndex];  
}  
}
```

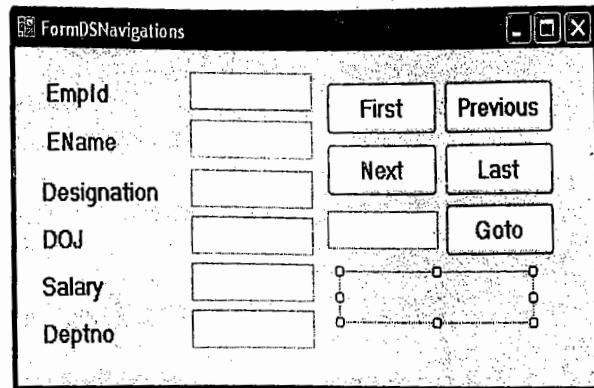
Example 21.5

Program to Navigate through the records in Dataset table: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDSNavigations : Form
    {
        SqlConnection Con = new SqlConnection("server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlDataAdapter Da; DataSet Ds; int R;
        public FormDSNavigations()
        {
            InitializeComponent();
        }
        private void FormDSNavigations_Load(object sender, EventArgs e)
        {
            Da = new SqlDataAdapter("Select * from EmpDetails", Con);
            Ds = new DataSet();
            Da.Fill(Ds, "EmpDetails");
        }
        private void GetData()
        {
            txtEmpId.Text = Ds.Tables[0].Rows[R][0].ToString();
            txtEName.Text = Ds.Tables[0].Rows[R][1].ToString();
            txtDesignation.Text = Ds.Tables[0].Rows[R][2].ToString();
            txtDOJ.Text = Ds.Tables[0].Rows[R][3].ToString();
            txtSalary.Text = Ds.Tables[0].Rows[R][4].ToString();
            txtDeptno.Text = Ds.Tables[0].Rows[R][5].ToString();
            lblDisplay.Text = (R + 1) + " of " + Ds.Tables[0].Rows.Count;
        }
        private void cmdFirst_Click(object sender, EventArgs e)
        {
            R = 0;
            GetData();
        }
        private void cmdPrevious_Click(object sender, EventArgs e)
        {
            if (R == 0)
                MessageBox.Show("This is First Record");
            else
            {
                R = R - 1;
                GetData();
            }
        }
        private void cmdNext_Click(object sender, EventArgs e)
```

Design of Form



```
{  
    if (R == Ds.Tables[0].Rows.Count - 1)  
        MessageBox.Show("This is First Record");  
    else  
    {  
        R = R + 1;  
        GetData();  
    }  
}  
  
private void cmdLast_Click(object sender, EventArgs e)  
{  
    R = Ds.Tables[0].Rows.Count - 1;  
    GetData();  
}  
  
private void cmdGoto_Click(object sender, EventArgs e)  
{  
    if (Convert.ToInt32(txtGoto.Text) < 1 || Convert.ToInt32(txtGoto.Text) >  
        Ds.Tables[0].Rows.Count)  
        MessageBox.Show("Invalid Index");  
    else  
    {  
        R = Convert.ToInt32(txtGoto.Text) - 1;  
        GetData();  
    }  
}  
}
```

Sample Result Form:

Empld	110	First	Previous
EName	Sai	Next	Last
Designation	PL	Goto	
DOJ	10/10/2009	9 of 13	
Salary	80000.000		
Deptno	10		

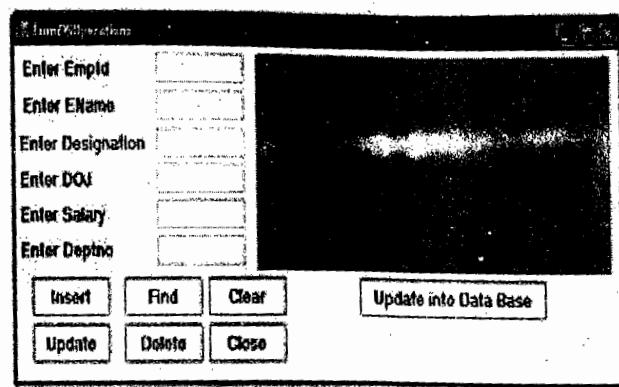
Example 21.6

Program to Perform the Operations with in Dataset table: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDSOperations : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlDataAdapter Da; DataSet Ds; DataRow Rec; SqlCommandBuilder Bldr;
        public FormDSOperations()
        {
            InitializeComponent();
        }
        private void FormDSOperations_Load(object sender, EventArgs e)
        {
            Da = new SqlDataAdapter("Select * from EmpDetails", Con);
            Ds = new DataSet();
            Da.Fill(Ds, "EmpDetails");
            DGVSample.DataSource = Ds.Tables[0];
            Da.FillSchema(Ds, SchemaType.Source, "EmpDetails");
            Bldr = new SqlCommandBuilder(Da);
        }
        private void cmdInsert_Click(object sender, EventArgs e)
        {
            try
            {
                Rec = Ds.Tables[0].NewRow();
                Rec[0] = txtEmpId.Text;
                Rec[1] = txtEName.Text;
                Rec[2] = txtDesignation.Text;
                Rec[3] = txtDOJ.Text;
                Rec[4] = txtSalary.Text;
                Rec[5] = txtDeptno.Text;
                Ds.Tables[0].Rows.Add(Rec);
                MessageBox.Show("Record is Inserted into Dataset Table");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        private void cmdFind_Click(object sender, EventArgs e)
        {
            try
            {
                Rec = Ds.Tables[0].Select("EmpId=" + txtEmpId.Text)[0];
```

Design of Form



```
txtEName.Text = Rec[1].ToString();
txtDesignation.Text = Rec[2].ToString();
txtDOJ.Text = Rec[3].ToString();
txtSalary.Text = Rec[4].ToString();
txtDeptno.Text = Rec[5].ToString();
cmdUpdate.Enabled = true;
}
catch
{
    MessageBox.Show("Record Not Found");
}
}
private void cmdUpdate_Click(object sender, EventArgs e)
{
    try
    {
        Rec[1] = txtEName.Text;
        Rec[2] = txtDesignation.Text;
        Rec[3] = txtDOJ.Text;
        Rec[4] = txtSalary.Text;
        Rec[5] = txtDeptno.Text;
        cmdUpdate.Enabled = false;
        MessageBox.Show("Record is Updated into Dataset Table");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void cmdDelete_Click(object sender, EventArgs e)
{
    try
    {
        Rec = Ds.Tables[0].Select("EmpId=" + txtEmpId.Text)[0];
        Rec.Delete();
        MessageBox.Show("Record is Deleted From Dataset Table");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
private void cmdUpdateDB_Click(object sender, EventArgs e)
{
    try
    {
        if (Ds.HasChanges() == true)
        {
            Da.Update(Ds, "EmpDetails");
            MessageBox.Show("Dataset Data is Updated into DataBase");
        }
        else
            MessageBox.Show("No Modifications in Dataset");
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message);
    }
}

private void cmdClear_Click(object sender, EventArgs e)
{
    foreach (Control X in this.Controls)
    {
        if (X is TextBox)
            X.Text = "";
    }
}

private void cmdClose_Click(object sender, EventArgs e)
{
    this.Close();
}
}
}

```

Finding the Records From Dataset Table

- There are two methods available to find the records from the Dataset table
 - Find(object Key) Method with Rows Collection of table.
 - Select(string FilterExpression) with any tableof Dataset
- **Find(object Key):** This methos is used to find the record using the primary key filed data. Return type of this method is DataRow type.
- **Syntax:** DataRow object=DatasetName.Tables[Index].Rows.Find(object key);
- **Ex:** DataRow Rec= Ds.Tables[0].Rows.Find(103);
- **Select(string FilterExpression):** This methos is used to find the record using the primary key filed data and also using Non – Primary key filed data . Return type of this method is an array DataRow type.
- **Syntax:** DataRow []object=DatasetName.Tables[Index].Select("FilterExpression");
- **Ex:** DataRow[] Rec= Ds.Tables[0].Select("EmpId=103");
 - DataRow[] Rec= Ds.Tables[0].Select("EName='Sai'");
 - DataRow[] Rec= Ds.Tables[0].Select("DOJ>=#01/01/2009#");
 - DataRow[] Rec= Ds.Tables[0].Select("Salary>=20000");
 - DataRow[] Rec= Ds.Tables[0].Select("Salary>15000 OR Deptno=10");
 - DataRow[] Rec= Ds.Tables[0].Select("Salary>15000 AND Deptno=10");
 - DataRow[] Rec= Ds.Tables[0].Select("EName Like 'S*' ");
- To Find the Single Record we can also use like
 - DataRow Rec= Ds.Tables[0].Select("EmpId=103")[0];
- When ever any record(s) are found and transferred toDataRow(s) internally link ill be created from datset tale Row(s) to DataRow(s), any modifications made into DataRow(s) will be updated intodataset tabl rows(s) and vice - versa

21.12 CommandBuilder Object

- CommandBuilder object is to bind with DataAdapter object at the time of fetching the data from the database and to fill into dataset.
- CommandBuilder will perform the foolowing tasks

- Keeps track all the modifications getting done into dataset table
- Prepares all necessary DML statements to send updations to database.

Example 21.7

Program to with Dataset Methods Like AcceptChanges(), RejectChanges() etc: -

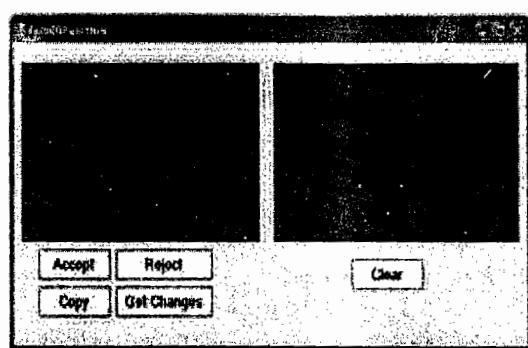
Code:

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WADatabase
{
    public partial class FormDSFunctions : Form
    {
        SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlDataAdapter Da; DataSet Ds1, Ds2;
        public FormDSFunctions()
        {
            InitializeComponent();
        }
        private void FormDSFunctions_Load(object sender, EventArgs e)
        {
            Da = new SqlDataAdapter("Select * from EmpDetails", Con);
            Ds1 = new DataSet();
            Da.Fill(Ds1, "EmpDetails");
            DGVSample1.DataSource = Ds1.Tables[0];
        }
        private void btnAccept_Click(object sender, EventArgs e)
        {
            Ds1.AcceptChanges();
            MessageBox.Show("Modifications into Dataset are Committed");
        }
        private void btnReject_Click(object sender, EventArgs e)
        {
            Ds1.RejectChanges();
            MessageBox.Show("Modifications into Dataset are Rolled Back");
        }
        private void btnCopy_Click(object sender, EventArgs e)
        {
            Ds2 = new DataSet();
            Ds2 = Ds1.Copy();
            DGVSample2.DataSource = Ds2.Tables[0];
        }
        private void btnGetChanges_Click(object sender, EventArgs e)
        {
            Ds2 = new DataSet();
            Ds2 = Ds1.GetChanges();
            DGVSample2.DataSource = Ds2.Tables[0];
        }
    }
}

```

Design of Form



Rest

```

private void btnClear_Click(object sender, EventArgs e)
{
    Ds2.Clear();
}
}

```

Result Form after Clicking on Copy:

FormDSFunction

Empld	Ename	Design.
101	Raju	Program
102	Gopal	TL
103	Sampath	Manager
104	Haren	DBProg
105	Abhi	SA
106	Sunya	DSE
108	Prawan	Suprvsor

Empld	Ename	Design.
101	Raju	Program
102	Gopal	TL
103	Sampath	Manager
104	Haren	DBProg
105	Abhi	SA
106	Sunya	DSE
108	Prawan	Suprvsor

Accept **Reject** **Clear**

Copy **Get Changes**

Chapter 22

Building N – Tier Architecture Application

22.1 Introduction

- Any application we develop related to the database will contain three types of Code
 1. UI Design and Validations code
 2. Business Logic Code
 3. Data Access Code
- **UI Design and Validations Code:**
 - This code is related to creating User Interface elements like Text Boxes, Buttons, Labels, Grid views, etc
 - Validations code is like user should enter only digits in EmpId Textbox, only characters in EName Textbox etc.
 - This code will change based on the type application we use like Console Application, Windows Forms Application, Web Application and Mobile Application etc.
- **Business Logic Code:**
 - This code is related to Business Logic of the application that is being developed.
 - This code is like find the records using one or more fields, Deleting the records using one or more fields combination etc.
 - This code will change based on the business logic rules of the domain that is being computerized like Telecom Domain, Banking Domain etc.
- **Data Access Code:**
 - This code is related to interacting with the Database, i.e writing code to get the data from the database and to update the data back to data base.
 - This code will change based on the Data Provider and Database used with in the application.

22.2 1 – Tier Architecture:

- In this method complete front end application code is developed using only part or application or will be present in single layer only.
- In general this layer will be developed using Console / Windows Forms / Web / Mobile Applications.

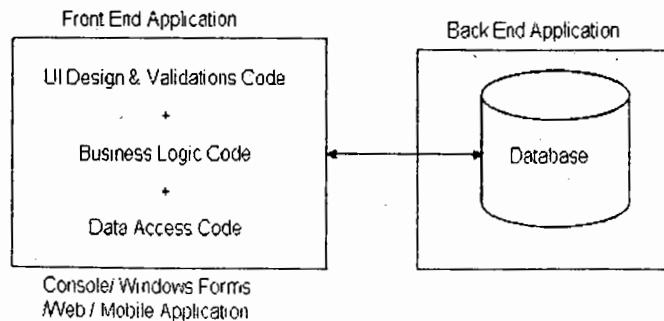


Figure 22.1 1 – Tier Architecture

22.2 2 – Tier Architecture:

- In this method complete front end application code is divided into two parts (Layer).
- First Part (Layer) will contain UI Design & Validations Code
- In general First layer will be developed using Console / Windows Forms / Web / Mobile Applications.
- Second Part (Layer) will contain Business Logic and Data Access Code.
- In general Second layer will be developed using .NET Components / Class Libraries which are in the form Dlls.
- In this method always First Layer will interact with Second Layer and Second Layer will interact with Database and First Layer never interacts with Database.

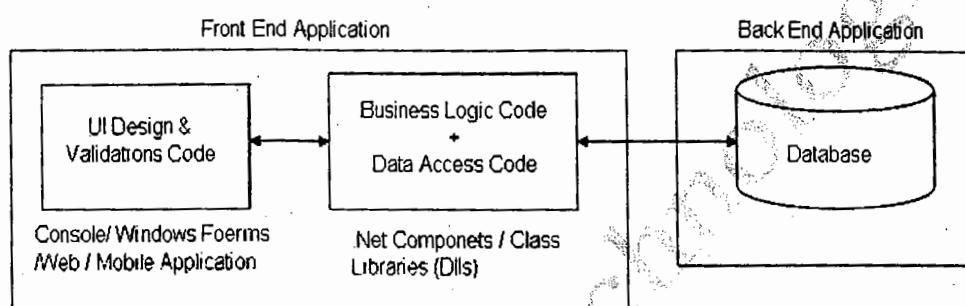


Figure 22.2 2 – Tier Architecture

22.3 3 – Tier Architecture:

- In this method complete front end application code is divided into three parts (Layers).
- First Part (Layer) will contain UI Design & Validations Code
- In general First layer will be developed using Console / Windows Forms / Web / Mobile Applications.
- Second Part (Layer) will contain Business Logic Code.
- In general Second layer will be developed using .NET Components / Class Libraries which are in the form Dlls.
- Third Part (Layer) will contain Data Access Code.
- In general Third layer will be developed using .NET Components / Class Libraries which are in the form Dlls.
- The Layers are named as
 1. First Layer → Presentation Layer
 2. Second Layer → Business Logic Layer
 3. Third Layer → Data Access Layer
- In this method always Presentation Layer will interact with Business Logic Layer, Business Logic Layer will interact with Data Access Layer and Data Access Layer will interact with Database.
- Presentation Layer will never interacts with Business Logic Layer or Database.
- Business Logic Layer will never interact with Database.

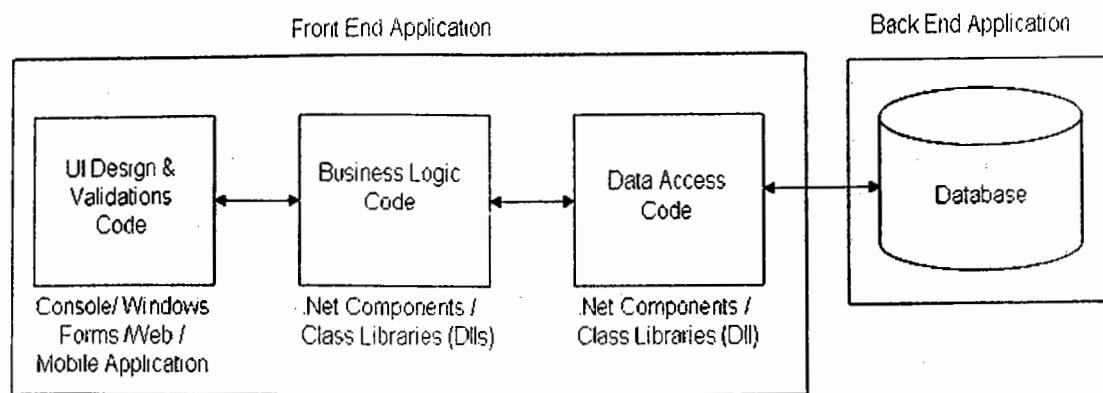


Figure 22.3 3 – Tier Architecture

22.4 N – Tier Architecture:

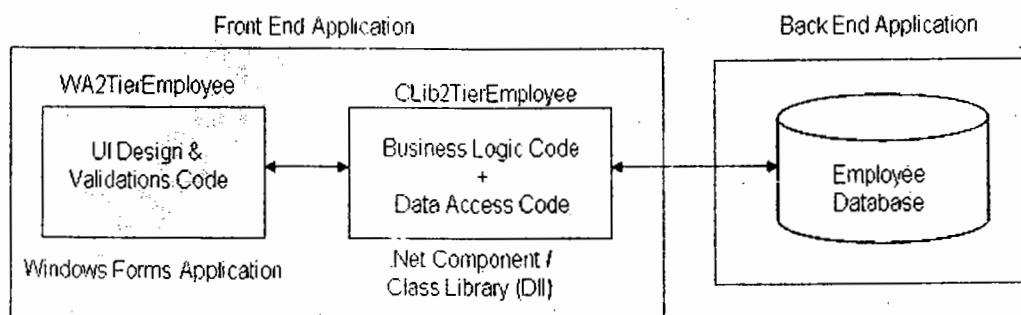
- If front end application code is divided into three or more parts (Layers) then it is known as N – Tier Architecture.

In Chapters 15, 16 and 17 all Examples are 1 – Tier Architecture only now we take Example 17.05 (Program to Perform the Operations with in Dataset table) and convert this to 2 - Tier and 3 – Tier Architectures.

Example 22.01

Building 2 – Tier Architecture Application: -

Architectural Diagram



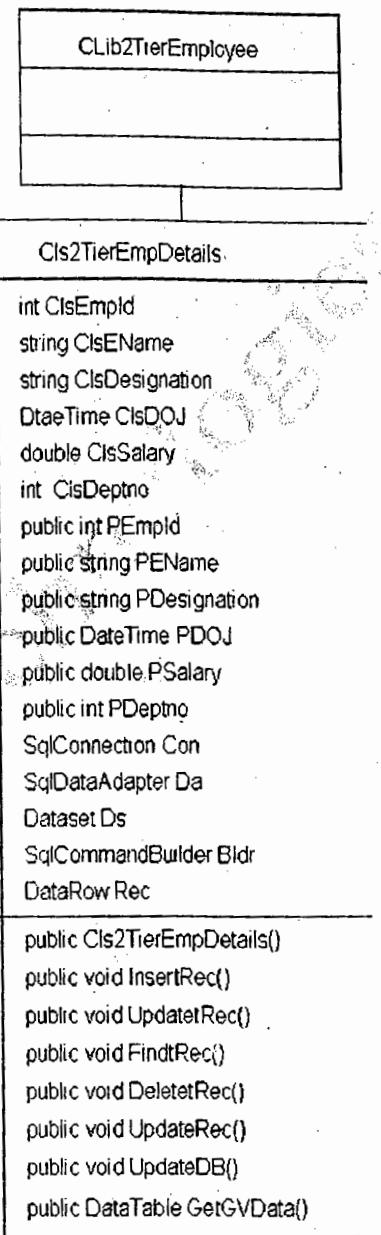
Design and code for Second Layer

Code:

Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select Class Library Template → Type the Application Name (CLib2TierEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Class Name to Cls2TierEmpDetails → Write the following code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;
namespace CLib2TierEmployee
{
    public class Cls2TierEmpDetails
    {
        int ClsEmpId, ClsSalary, ClsDeptno;
        string ClsEName, ClsDesignation; DateTime ClsDOJ;
        SqlConnection Con; SqlDataAdapter Da; DataSet Ds;
        DataRow Rec; SqlCommandBuilder Bldr;
        public int PEmpId
        {
            set { ClsEmpId = value; }
        }
        public string PName
        {
            set { ClsEName = value; }
            get { return ClsEName; }
        }
        public string PDesignation
        {
            set { ClsDesignation = value; }
            get { return ClsDesignation; }
        }
        public DateTime PDOJ
        {
            set { ClsDOJ = value; }
            get { return ClsDOJ; }
        }
        public int PSalary
        {
            set { ClsSalary = value; }
            get { return ClsSalary; }
        }
        public int PDeptno
        {
            set { ClsDeptno = value; }
            get { return ClsDeptno; }
        }
        public Cls2TierEmpDetails()
        {
        }
    }
}
```

Design of CLib2TierEmployee



```

Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;DataBase=Employee");
Da = new SqlDataAdapter("Select * from EmpDetails", Con);
Ds = new DataSet();
Da.Fill(Ds, "EmpDetails");
Da.FillSchema(Ds, SchemaType.Source, "EmpDetails");
Bldr = new SqlCommandBuilder(Da);
}

public void InsertRec()
{
    Rec = Ds.Tables[0].NewRow();
    Rec[0] = ClsEmpId;
    Rec[1] = ClsEName;
    Rec[2] = ClsDesignation;
    Rec[3] = ClsDOJ;
    Rec[4] = ClsSalary;
    Rec[5] = ClsDeptno;
    Ds.Tables[0].Rows.Add(Rec);
}

public void FindRec()
{
    Rec = Ds.Tables[0].Select("EmpId=" + ClsEmpId)[0];
    ClsEName = Rec[1].ToString();
    ClsDesignation = Rec[2].ToString();
    ClsDOJ = Convert.ToDateTime(Rec[3]);
    ClsSalary = Convert.ToInt32(Rec[4]);
    ClsDeptno = Convert.ToInt32(Rec[5]);
}

public void UpdateRec()
{
    Rec[1] = ClsEName;
    Rec[2] = ClsDesignation;
    Rec[3] = ClsDOJ;
    Rec[4] = ClsSalary;
    Rec[5] = ClsDeptno;
}

public void DelRec()
{
    Rec = Ds.Tables[0].Select("EmpId=" + ClsEmpId)[0];
    Rec.Delete();
}

public void UpDateDB()
{
    Da.Update(Ds, "EmpDetails");
}

public DataTable GetGVData()
{
    return Ds.Tables[0];
}
}
}

```

Save the Application → Build the Application (Ctrl + Shift + B) → This will generate the Dll for CLib2TierEmployee → This Dll is .Net Component and is the Second Layer of this Example.

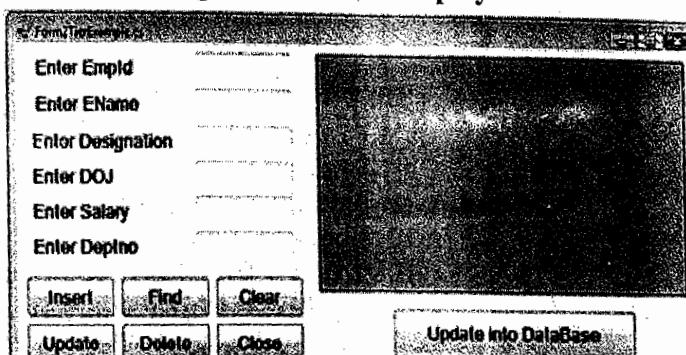
Now we design and write the code for first layer (WA2TierEmployee)

Code:

Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select WindowsFormsApplication Template → Type the Application Name (WA2TierEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Form Name to Form2TierEmployee → Go to Solution Explorer → Click with Right Mouse button → Click on Add Reference → Add Reference dialog box will appear → Click on Browse tab page Go to the location where CLib2TierEmployee.Dll is saved (D:\CLib2TierEmployee\CLib2TierEmployee\Bin\Debug) → Select the CLib2TierEmployee.Dll → Click on Ok → Write the following code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using CLib2TierEmployee;
namespace WA2TierEmployee
{
    public partial class Form2TierExample : Form
    {
        Cls2TierEmpDetails Obj1 = new Cls2TierEmpDetails();
        public Form2TierExample()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            GVSample.DataSource = Obj1.GetGVData();
        }
        private void cmdInsert_Click(object sender, EventArgs e)
        {
            try
            {
                Obj1.PEmpId = Convert.ToInt32(txtEmpId.Text);
                Obj1.PENAME = txtEName.Text;
                Obj1.PDesignation = txtDesignation.Text;
                Obj1.PDOJ = Convert.ToDateTime(txtDOJ.Text);
                Obj1.PSalary = Convert.ToInt32(txtSalary.Text);
                Obj1.PDeptno = Convert.ToInt32(txtDeptno.Text);
                Obj1.InsertRec();
                MessageBox.Show("Record is Inserted into Dataset Table");
            }
        }
}
```

Design of WA2TierEmployee



```
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

private void cmdFind_Click(object sender, EventArgs e)
{
try
{
    Obj1.PEmpId = Convert.ToInt32(txtEmpId.Text);
    Obj1.FindRec();
    txtEName.Text = Obj1.PENName;
    txtDesignation.Text = Obj1.PDesignation;
    txtDOJ.Text = Obj1.PDOJ.ToString();
    txtSalary.Text = Obj1.PSalary.ToString();
    txtDeptno.Text = Obj1.PDeptno.ToString();
    cmdUpdate.Enabled = true;
    MessageBox.Show("Record Found");
}
catch
{
    MessageBox.Show("Record Not Found");
}
}

private void cmdUpdate_Click(object sender, EventArgs e)
{
try
{
    Obj1.PENName = txtEName.Text;
    Obj1.PDesignation = txtDesignation.Text;
    Obj1.PDOJ = Convert.ToDateTime(txtDOJ.Text);
    Obj1.PSalary = Convert.ToInt32(txtSalary.Text);
    Obj1.PDeptno = Convert.ToInt32(txtDeptno.Text);
    Obj1.UpdateRec();
    MessageBox.Show("Record is Updated into Dataset Table");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

private void cmdDelete_Click(object sender, EventArgs e)
{
try
{
    Obj1.PEmpId = Convert.ToInt32(txtEmpId.Text);
    Obj1.DelRec();
    MessageBox.Show("Record is Deleted from Dataset Table");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

private void cmdUpdateDB_Click(object sender, EventArgs e)
{
```

```
try
{
    Obj1.UpdateDB();
    MessageBox.Show("Data is Updated into DataBase");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
```

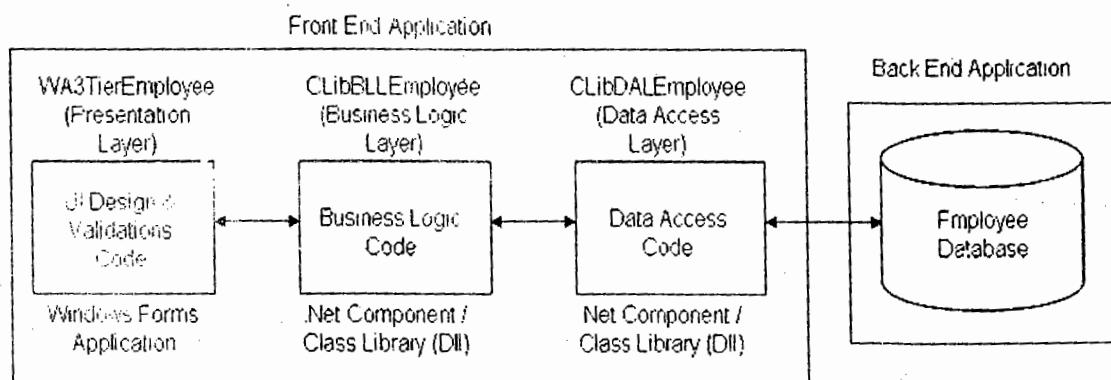
Save the Application → Build the Application (Ctrl + Shift + B) → This will generate the Exe for WA2TierEmployee → Run the Application and check.

- Go to the Location D:\WA2TierEmployee\WA2TierEmployee\Bin\Debug you will find WA2TierEmployee.exe and also a copy of CLib2TierEmployee.Dll.
 - When we give the exe to the client we should also give this dll otherwise the application does not work.

Example 22.2

Building 3 – Tier Architecture Application:

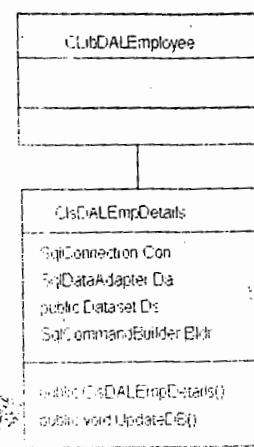
Architectural Diagram



Design and code for Data Access Layer Code:

Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select Class Library Template → Type the Application Name (CLibDALEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Class Name to Cls2DALEmpDetails → Write the following code

```
using System;
using System.Data;
using System.Data.SqlClient;
```



```

namespace CLibDALEmployee
{
    public class ClsDALEmpDetails
    {
        SqlConnection Con = new SqlConnection("Server=satya;User Id=sa;Password=abc;
        DataBase=Employee");
        SqlDataAdapter Da; public DataSet Ds; SqlCommandBuilder Bldr;
        public ClsDALEmpDetails()
        {
            Da = new SqlDataAdapter("Select * from EmpDetails", Con);
            Ds = new DataSet();
            Da.Fill(Ds, "EmpDetails");
            Da.FillSchema(Ds, SchemaType.Source, "EmpDetails");
            Bldr = new SqlCommandBuilder(Da);
        }
        public void UpdateDB()
        {
            Da.Update(Ds, "EmpDetails");
        }
    }
}

```

Save the Application → Build the Application (Ctrl + Shift + B) → This will generate the Dll for CLibDALEmployee → This Dll is .Net Component for the Data Access Layer.

Design and code for Business Logic Layer

Code:

Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select Class Library Template → Type the Application Name (CLib2BLLEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Class Name to ClsBLLEmpDetails → Go to Solution Explorer → Click with Right Mouse button → Click on Add Reference → Add Reference dialog box will appear → Click on Browse tab page Go to the location where CLibDALEmployee.Dll is saved (D:\CLibDALEmployee\CLibDALEmployee\Bin\Debug) → Select the CLibDALEmployee.Dll → Click on Ok → Write the following code

```

using System;
using System.Data;
using CLibDALEmployee;

namespace CLibBLLEmployee
{
    public class ClsBLLEmpDetails
    {
        int ClsEmpId;
        string ClsEName;
        string ClsDesignation;
        DateTime ClsDOJ;
        double ClsSalary;
        int ClsDeptno;
        public int PEmpld;
        public string PName;
        public string PDesignation;
        public DateTime PDOJ;
        public double PSalary;
        public int PDeptno;
        DataRow Rec;
    }
}

```

Design of CLibBLLEmployee

CLibBLLEmployee
ClsBLLEmpDetails
int ClsEmpId
string ClsEName
string ClsDesignation
DateTime ClsDOJ
double ClsSalary
int ClsDeptno
public int PEmpld
public string PName
public string PDesignation
public DateTime PDOJ
public double PSalary
public int PDeptno
DataRow Rec
public void InsertRec()
public void FindtRec()
public void UpdatetRec()
public void DeletetRec()
public void UpdateDB()
public DataTable GetGVData()

```

ClsDALEmpDetails ObjDAL = new ClsDALEmpDetails();
public int PEmpId
{
    set { ClxEmpId = value; }
}
public string PName
{
    set { ClxEName = value; }
    get { return ClxEName; }
}
public string PDDesignation
{
    set { ClxDesignation = value; }
    get { return ClxDesignation; }
}
public DateTime PDOJ
{
    set { ClxDOJ = value; }
    get { return ClxDOJ; }
}
public int PSalary
{
    set { ClxSalary = value; }
    get { return ClxSalary; }
}
public int PDeptno
{
    set { ClxDeptno = value; }
    get { return ClxDeptno; }
}
public void InsertRec()
{
    Rec = ObjDAL.Ds.Tables[0].NewRow();
    Rec[0] = ClxEmpId;
    Rec[1] = ClxEName;
    Rec[2] = ClxDesignation;
    Rec[3] = ClxDOJ;
    Rec[4] = ClxSalary;
    Rec[5] = ClxDeptno;
    ObjDAL.Ds.Tables[0].Rows.Add(Rec);
}
public void FindRec()
{
    Rec = ObjDAL.Ds.Tables[0].Select("EmpId=" + ClxEmpId)[0];
    ClxEName = Rec[1].ToString();
    ClxDesignation = Rec[2].ToString();
    ClxDOJ = Convert.ToDateTime(Rec[3]);
    ClxSalary = Convert.ToInt32(Rec[4]);
    ClxDeptno = Convert.ToInt32(Rec[5]);
}
public void UpdateRec()
{
    Rec[1] = ClxEName;
    Rec[2] = ClxDesignation;
    Rec[3] = ClxDOJ;
    Rec[4] = ClxSalary;
}

```

```

Rec[5] = ClsDeptno;
}
public void DelRec()
{
    Rec = ObjDAL.Ds.Tables[0].Select("EmpId=" + ClsEmpId)[0];
    Rec.Delete();
}
public void UpDateDB()
{
    ObjDAL.UpdateDB();
}
public DataTable GetGVData()
{
    return ObjDAL.Ds.Tables[0];
}
}
}

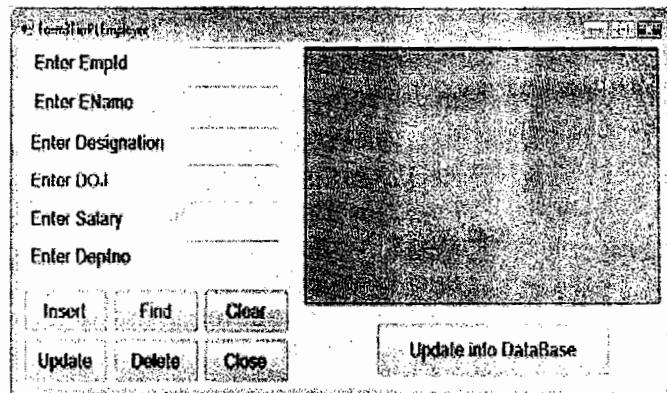
```

Save the Application → Build the Application (Ctrl + Shift + B) → This will generate the Dll for CLibBLLEmployee → This Dll is .Net Component for the Business Access Layer → Go to the Location D:\CLibBLLEmployee\CLibBLLEmployee\Bin\Debug and observe you find 2 Dlls one for Business Logic Layer (CLibBLLEmployee.Dll) and the another is copy of Data Access Layer (CLibDALEmployee.Dll).

Design and code for Presentation Layer

Code:

Design of WA3TierPLEmployee



Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select WindowsFormsApplication Template → Type the Application Name (WA3TierPLEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Form Name to Form3TierPLEmployee → Go to Solution Explorer → Click with Right Mouse button → Click on Add Reference → Add Reference dialog box will appear → Click on Browse tab page Go to the location where CLibBLLEmployee.Dll is saved (D:\CLibBLLEmployee\CLibBLLEmployee\Bin\Debug) → Select the CLibBLLEmployee.Dll → Click on Ok → Write the following code

```

using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;
using CLibBLLEmployee;

namespace WA3TierEmployee
{
    public partial class Form3TierPLEmployee : Form

```

```

{
    ClsBLLEmpDetails ObjBLL = new ClsBLLEmpDetails();
    public Form3TierPLEmployee()
    {
        InitializeComponent();
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        GVSample.DataSource = ObjBLL.GetGVData();
    }

    private void cmdInsert_Click(object sender, EventArgs e)
    {
        try
        {
            ObjBLL.PEmpId = Convert.ToInt32(txtEmpId.Text);
            ObjBLL.PName = txtEName.Text;
            ObjBLL.PDesignation = txtDesignation.Text;
            ObjBLL.PDOJ = Convert.ToDateTime(txtDOJ.Text);
            ObjBLL.PSalary = Convert.ToInt32(txtSalary.Text);
            ObjBLL.PDeptno = Convert.ToInt32(txtDeptno.Text);
            ObjBLL.InsertRec();
            MessageBox.Show("Record is Inserted into Dataset Table");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    private void cmdFind_Click(object sender, EventArgs e)
    {
        try
        {
            ObjBLL.PEmpId = Convert.ToInt32(txtEmpId.Text);
            ObjBLL.FindRec();
            txtEName.Text = ObjBLL.PName;
            txtDesignation.Text = ObjBLL.PDesignation;
            txtDOJ.Text = ObjBLL.PDOJ.ToString();
            txtSalary.Text = ObjBLL.PSalary.ToString();
            txtDeptno.Text = ObjBLL.PDeptno.ToString();
            cmdUpdate.Enabled = true;
            MessageBox.Show("Record Found");
        }
        catch
        {
            MessageBox.Show("Record Not Found");
        }
    }

    private void cmdUpdate_Click(object sender, EventArgs e)
    {
        try
        {
            ObjBLL.PName = txtEName.Text;
            ObjBLL.PDesignation = txtDesignation.Text;

```

```
ObjBLL.PDOJ = Convert.ToDateTime(txtDOJ.Text);
ObjBLL.PSalary = Convert.ToInt32(txtSalary.Text);
ObjBLL.PDeptno = Convert.ToInt32(txtDeptno.Text);
ObjBLL.UpdateRec();
MessageBox.Show("Record is Updated into Dataset Table");
}

catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

private void cmdDelete_Click(object sender, EventArgs e)
{
    try
    {
        ObjBLL.PEmpId = Convert.ToInt32(txtEmpId.Text);
        ObjBLL.DelRec();
        MessageBox.Show("Record is Deleted from Dataset Table");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void cmdUpdateDB_Click(object sender, EventArgs e)
{
    try
    {
        ObjBLL.UpDateDB();
        MessageBox.Show("Data is Updated into DataBase");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Save the Application → Build the Application (Ctrl + Shift + B) → This will generate the Exe for WA3TierPLEmployee → Run the Application and check.

- Go to the Location D:\WA3TierPLEmployee\WA3TierPLEmployee\Bin\Debug you will find WA3TierPLEmployee.exe and also a copy of CLibBLLEmployee.Dll as well as a copy of CLibDALEmployee.Dll .
 - When we give the exe to the client we should also give these two dlls otherwise the application does not work.

Chapter 23

Typed Dataset and Crystal Reports

23.1 Typed Dataset

- The Dataset that is created at design time is known as **Typed Dataset** and the dataset that is created at runtime is known as **Untyped Dataset**. So far in earlier chapters the Dataset created is Untyped Dataset.
- Differences between Typed Dataset and Untyped Dataset

Srno	Typed Dataset	Un Typed Dataset
01	Dataset is created at Design Time	Dataset is created at Run Time
02	Knows Database, Table is to be connected, Field Names and their data types are to be bind at design time only	Knows Database, Table is to be connected, Field Names and their data types are to be bind at run time only
03	Data & the structure of the tables is stored in the form of XML & XSD	Data and the structure of the tables is stored in the form of Collections
04	Execution is faster	Execution is slow
05	Not flexible to use when client requirements change	Flexible to use when client requirements change

- To work with Typed Dataset we need to follow the three steps
 - Creating typed dataset
 - Attaching to required data bound controls
 - Providing navigation and operational facilities

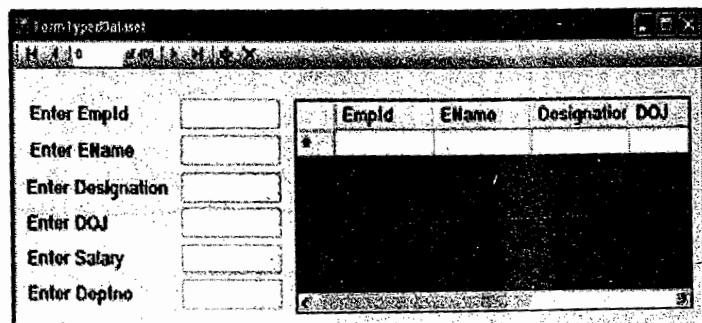
Example 23.1

Working with Typed Dataset: -

Step1: Creating typed dataset

Enter into VS2008 → Click on File → Click on New → Click on Project → Select Visual C# from Project Types → Select WindowsFormsApplication Template → Type the Application Name (WA3TierPLEmployee) → Select the Location to Save (D:\) → Click on Ok → Change the Form Name to FormTypedDataset → Design the Form → (Place Labels, TextBoxes, DataGridView and set Text and Name Properties) → Go to Tool Box → Click on Data Tool tab → Double Click on Binding Source → This will create bindingSource1 → Select bindingSource1 → Go to

Design of Form



properties window → Go to DataSource property → Click on Add Project DataSource → Select DataBase option → Click on Next → Click on NewConnection → Click on change → Select SQLServer Data Source → Select .NET Framework provider for SqlSerever → Click on OK → Type Server Name (Satya) → Select use SqlServer authentication option → Type User Id (sa) → Type Password (abc) → Activate Save my Password option → Select the required Database → Click on Ok → Activate the option "Include sensitive data in Connection string" → Click on Next → Change the Name (Sqlcon) → Click on Next → This will retrieve the Employee database information → Select the all fields from EmpDetails table → Click on Finish → This will create employeeDataset → Select bindingSource1 → Go to properties window → Go toDataMember property → Select the table EmpDetails → This will create empDetailsTableAdapter

23

23

>

>

>

23

In

Step2: Displaying the data in data bound controls

Select the DataGridView DGVSample → Go to properties window → Go to DataSource → property → Select bindingSource1 → Select the Text Box txtEmpId → Go to properties window Go to DataBindings property → Click on + symbol → Go to Text property in that → Select the field EmpId → Similarly attach EName, Designation, DOJ, Salary, Deptno → Fields to the txtEName, txtDesignation,txtDOJ,txtSalary, txtDeptno Text Boxes respectively.

Step3: Providing navigation and operational facilities

Go to tool box → Go to Data tool tab → Double click on BindingNavigator → This will create bindingNavigator1 → Select bindingNavigator1 → Go to properties window → Go to BindingSource property → Select bindingSource1 → Run the application and check

Result Form

Empid	EName	Designation	DOJ
101	Raju	Program..	1/1/2
102	Gopal	TL	2/2/2
103	Sampath	Manager..	3/3/2
104	Naren	DBProg..	4/4/2
105	Abhi	SA	5/5/2
106	Surya	DSE	6/6/2
108	Pravann	SrPravann	8/8/2

23.2 Reports

23.2.1 What is a Report

- A report is an output tool used to display the required data or summary information from the data base to the user in user required format.
- In general in real time reports are very much vital for any organization, complete top level management will depend only on reports to take appropriate decisions related to their business needs.
- In real time projects reports are treated as a separate module.

23.2.2 Elements of a Report

In general any report will contain following elements

1. Report Header
2. Page Header
3. Group Header
4. Details Section
5. Group Footer
6. Page Footer
7. Report Footer

Report Header:

This part will contain the information like Company Name, Address, Quality Levels of the company, Contact Numbers, Web site address etc.

Page Header:

This part will contain the information like Report Title, Field Headings to be displayed etc.

Group Header:

This part will contain the information of the field name on which the data is grouped on.

Details Section:

This part will contain the original records information that is to be displayed on the report.

Group Footer:

This part will contain the summary information of the filed that is grouped like Sum of the Salary, Count of Employees, Min of salary in a Dept etc.

Page Footer:

This part will contain the summary information of the page like Page Totals, Current Page Number, Total no of Pages etc.

Report Footer:

This part will contain the complete summary information of the report like Grand Totals, Prepared by, Date, Place, Authorized signature etc.

- A report can be created with grouping and without grouping.
- If a report is created without grouping it contains five elements Report Header, Page Header, Details Section, Page Footer and Report Footer.
- If a report is created with grouping it contains seven elements Report Header, Page Header, Group Header, Details Section, Group Footer, Page Footer and Report Footer.

23.3 Reporting Tools:

- The software used to develop the reports is known as a reporting tool, there are following reporting tools available
 1. Crystal Reports
 2. MS Data Reports
 3. Hyperion

23.3.1 Crystal Reports

- These are developed by the Seagate Company, and is most efficient reporting tool compared to other tools.
- Early to .NET it was required to purchase crystal reports software separately from Seagate Company, install separately then required to integrate with other software like VB etc.
- But from .NET onwards Microsoft itself purchased rights of Crystal reports from Seagate Company and integrated with VS.NET and .NET Framework.
- When VS.Net is installed we also get crystal reports software with it.

23.3.2 Working with Crystal Reports

- To work with crystal reports use the following steps.
 - Designing the report
 - Displaying the designed report to the user.

Example 23.1

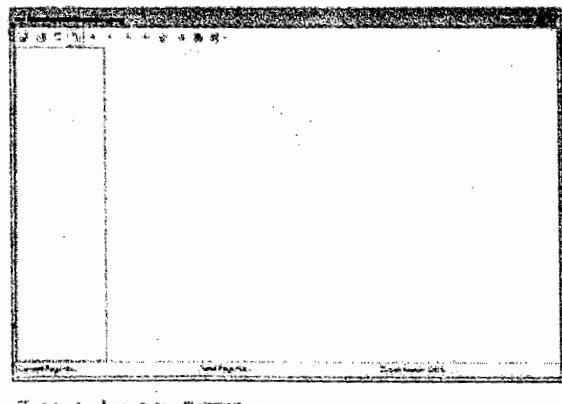
Creating a Report with out Grouping: -

Step1 Designing of Report

Step1.1 Creating a Typed Dataset

Enter into VS2008 → Click on File →
Click on New → Click on Project →
Select Visual C# from Project Types →
Select WindowsFormsApplication
Template → Type the Application Name
(WA3Reports) → Select the

Design of Form



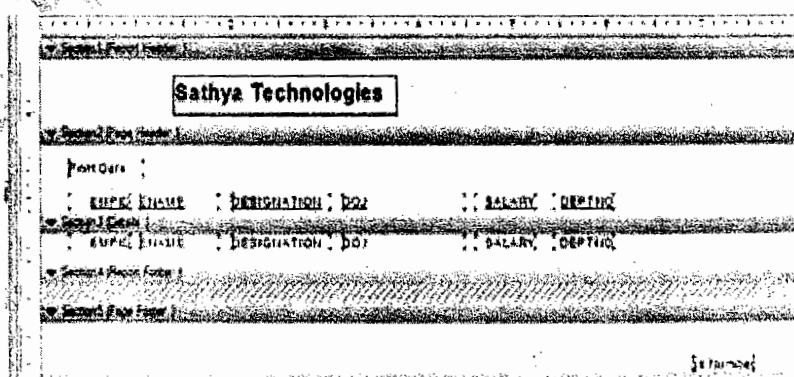
Location to Save (D:\) → Click on Ok →

Change the Form Name to FormReportWithoutGrouping → Go to Tool Box → Click on Data Tool tab → Double Click on Binding Source → This will create bindingSource1 → Select bindingSource1 → Go to properties window → Go to DataSource property → Click on Add Project DataSource → Select DataBase option → Click on Next → Click on NewConnection → Click on change → Select SQLServer Data Source → Select .NET Framework provider for SqlSerever → Click on OK → Type Server Name (Satya) → Select use SqlServer authentication option → Type User Id (sa) → Type Password (abc) → Activate Save my Password option → Select the required Database → Click on Ok → Activate the option "Include sensitive data in Connection string" → Click on Next → Change the Name (Sqlcon) → Click on Next → This will retrieve the Employee database information → Select the all fields from EmpDetails table → Click on Finish → This will create employeeDataset → Select bindingSource1 → Go to properties window → Go to DataMember property → Select the table EmpDetails → This will create empDetailsTableAdapter

Step1.2 Creating a Report

Go to Solution Explorer → Select the Solution → Click with Right Mouse Button → Click on Add → Click on NewItem... → Select Reporting from Categories → Select CrystalReport Template → Type the Report Name (crystalReport1) → Click on Add (A wizard will start now) → Select the Report type as "Standard" → Click on Ok → Double Click on Project Data → Double Click on ADO.Net Datasets → Select typed dataset (employeeDataset) → Add to "Selected Tables" option → Click on Next → Select the Required fields → Add to "Fields To Display" option → Click on Next → Click on Next → Click on Next → Select the required report style (Standard) → Click on Finish → A Report is created Now → Select Report Header → Click with Right Mouse Button → Click on Don't Suppress option → Click with Right Mouse Button in Report Header → Click on Insert → Click on Text Object → A text Object is created → Type Sathya Technologies → Select Text object → Click with Right Mouse Button → Click on Format Text Object option → set the required Font attributes borders etc. → Click on Ok

You find the report like



Step2 Displaying the Designed Report to the user

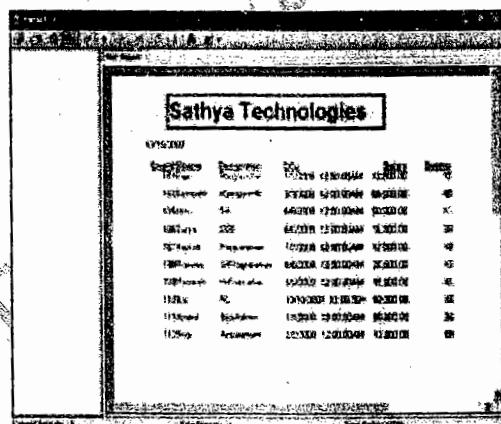
Go to the design part of FormReportWithoutGrouping → Go to the tool box → Go to Reporting tool tab → Double click on CrystalReportViewer tool → This will create → crystalReportViewer1 → Go to code window of the form and write the following code.

E
C
St
Cl
Cl
→
Na
wi
Cl
Cl
Ta
op
Ne
Cl
→
St
St

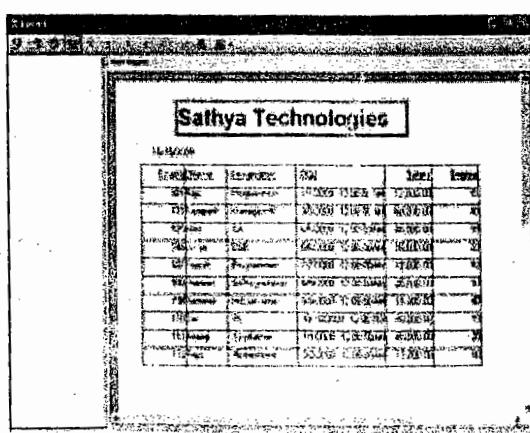
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace WAReports
{
    public partial class Form1 : Form
    {
        CrystalReport1 ObjRpt = new CrystalReport1();
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'employeeDataSet.EMPTDETAILS' table. You
            // can move, or remove it, as needed.
            this.eMPDETAILSTableAdapter.Fill(this.employeeDataSet.EMPTDETAILS);
            ObjRpt.SetDataSource(employeeDataSet.Tables[0]);
            crystalReportViewer1.ReportSource = ObjRpt;
        }
    }
}
```

Run the Application and check you find the output like:



To display the report in table format like the shown in below draw the Horizontal and Vertical lines in the design part of the report



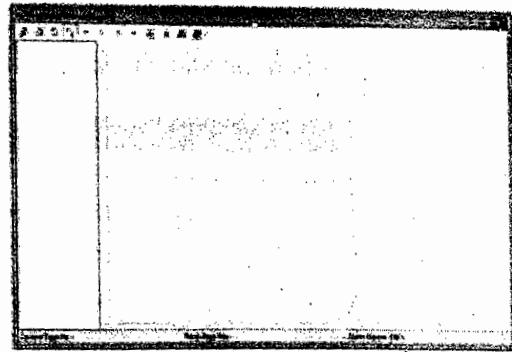
Example 23.2

Creating a Report with Grouping: -

Design of Form

Step1 Designing of Report

Create Typed Dataset with the name employeeDataset →
Go to Solution Explorer → Select the Solution →
Click with Right Mouse Button → Click on Add →
Click on NewItem... → Select Reporting from Categories
→ Select CrystalReport Template → Type the Report
Name (crystalReport1) → Click on Add (A wizard
will start now) → Select the Report type as "Standard" →
Click on Ok → Double Click on Project Data → Double
Click on ADO.Net Datasets → Select typed dataset (employeeDataset) → Add to "Selected
Tables" option → Click on Next → Select the Required fields → Add to "Fields To Display"
option → Click on Next → Select the Field Deptno and add to "Group By" option → Click on
Next → Delete the fields EmpId, Deptno from Summarized Fields option → Click on Next →
Click on Next → Click on Next → Click on Next → Select the required report style (Standard)
→ Click on Finish



Step2 To display the designed report we use a tool known as CrystalReportViewer

Steps:

- Go to the Form in which typed dataset is created
- Go to the tool box
- Go to Reporting tool tab
- Double click on CrystalReportViewer tool
- This will create crystalReportViewer1
- Go to code window of the form and write the following code.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
namespace WReports
{
    public partial class Form1 : Form
    {
        CrystalReport2 ObjRpt = new CrystalReport2();
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'employeeDataSet.EMPDETAILS' table. You
            // can move, or remove it, as needed.
            this.eMPDETAILTableAdapter.Fill(this.employeeDataSet.EMPDETAILS);
            ObjRpt.SetDataSource(employeeDataSet.Tables[0]);
        }
    }
}
```

```
        crystalReportViewer1.ReportSource = ObjRpt;
```

```
}
```

Run the Application and check you find the output like:

Sathya Technologies					
DeptID	EmployeeID	EmployeeName	Occupation	Dob	Salary
10	101 Raju	Programmer	10/12/09	12:00:00AM	12,000.00
10	105 Akshay	SA	10/12/09	12:00:00AM	10,000.00
10	107 Raghav	Programmer	10/12/09	12:00:00AM	12,000.00
10	108 Praveen	SrProgrammer	10/12/09	12:00:00AM	25,000.00
10	110 Ema	PL	10/12/09	12:00:00AM	10,000.00
				10	<u>175,000.00</u>
20					
20	106 Sunil	OSE	06/02/09	12:00:00AM	15,000.00
20	111 Anind	SysAdmin	10/12/09	12:00:00AM	10,000.00
				70	<u>25,000.00</u>
30					
30	103 Sampath	ManagerHR	30/03/09	12:00:00AM	60,000.00
30	109 Ramya	REExecutive	26/03/09	12:00:00AM	15,000.00
				40	<u>75,000.00</u>
40					
40	112 Raju	Accountant	30/03/09	12:00:00AM	12,000.00
				50	<u>12,000.00</u>
					<u>Grand Total:</u> <u>341,000.00</u>

Example 23.3

Creating a Report with Parameters: -

Create the report with out grouping normally as in Example 19.01.

Write the following Code

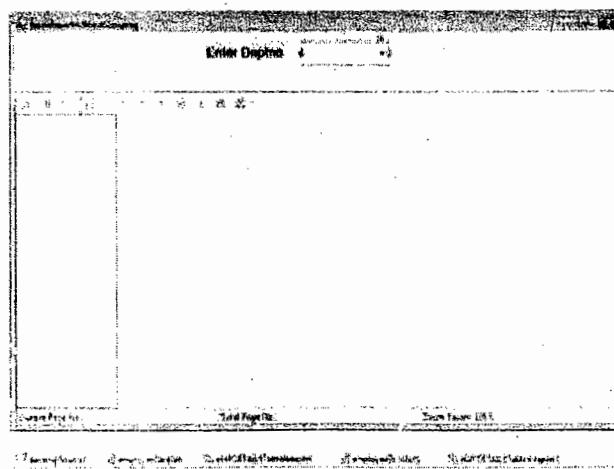
Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace WARReports
```

```
{
```

Design of Form

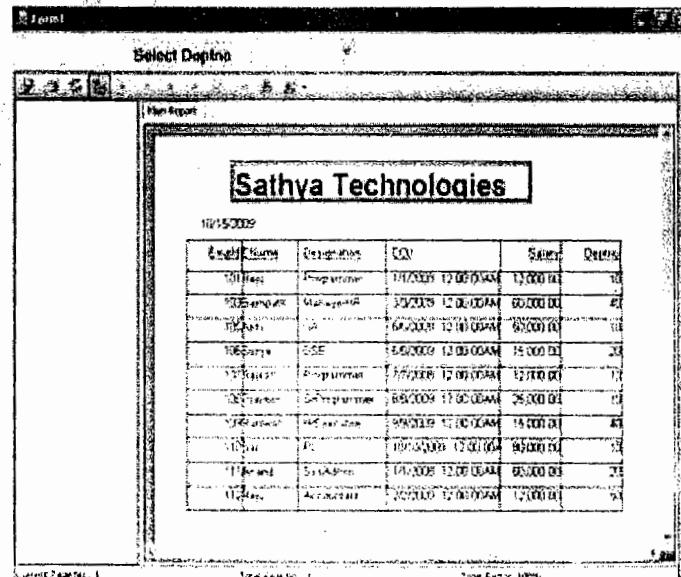


```

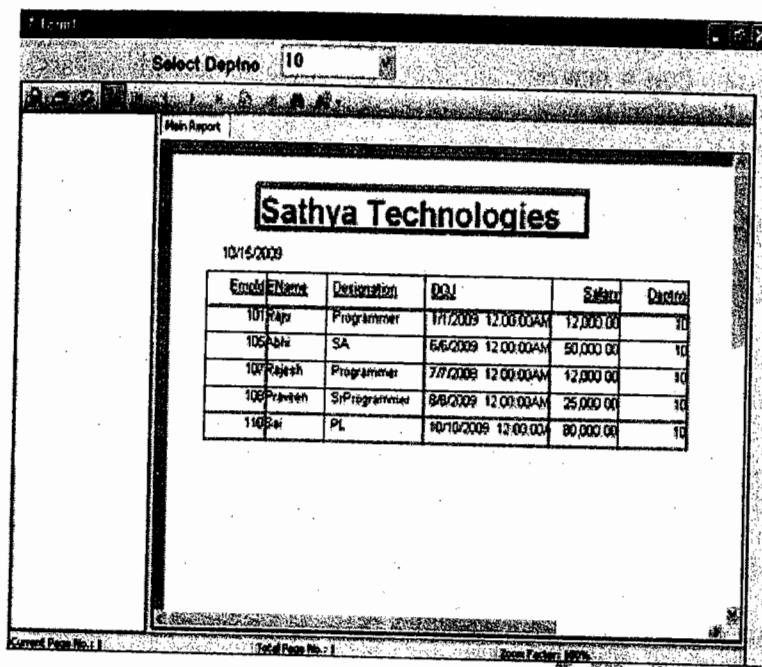
public partial class Form1 : Form
{
    CrystalReport1 ObjRpt = new CrystalReport1();
    SqlConnection Con = new SqlConnection("server=satya;User Id=sa;Password=abc;
    DataBase=Employee");
    public Form1()
    {
        InitializeComponent();
    }
    private void Form1_Load(object sender, EventArgs e)
    {
        // TODO: This line of code loads data into the 'employeeDataSet.EmpDetails' table. You can
        move, or remove it, as needed.
        this.empDetailsTableAdapter.Fill(this.employeeDataSet.EmpDetails);
        ObjRpt.SetDataSource(employeeDataSet.Tables[0]);
        crystalReportViewer1.ReportSource = ObjRpt;
        SqlCommand Cmd = new SqlCommand("Select Distinct(Deptno) from EmpDetails", Con);
        Con.Open();
        SqlDataReader DR = Cmd.ExecuteReader();
        while (DR.Read() == true)
            cmbDeptno.Items.Add(DR[0]);
        Con.Close();
    }
    private void cmbDeptno_SelectedIndexChanged(object sender, EventArgs e)
    {
        SqlDataAdapter Da = new SqlDataAdapter("Select * from EmpDetails Where Deptno=" +
        cmbDeptno.SelectedItem.ToString(), Con);
        DataSet Ds = new DataSet();
        Da.Fill(Ds, "EmpDetails");
        ObjRpt.SetDataSource(Ds.Tables[0]);
        crystalReportViewer1.ReportSource = ObjRpt;
    }
}

```

Run the Application and check you find the output like:



When user selects the Deptno 10 from Combobox cmbDeptno then output will be like:

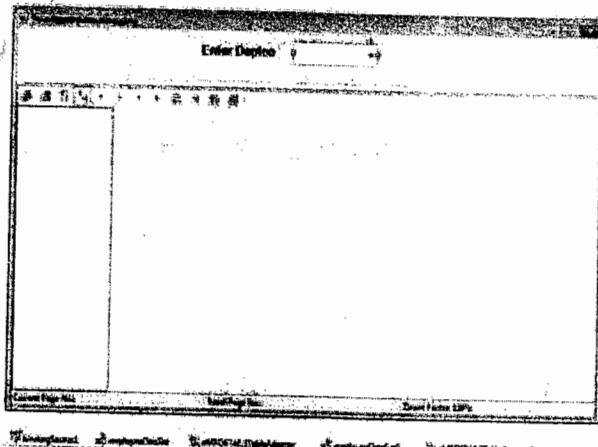


Example 23.4

Including a Chart with in the Report: -

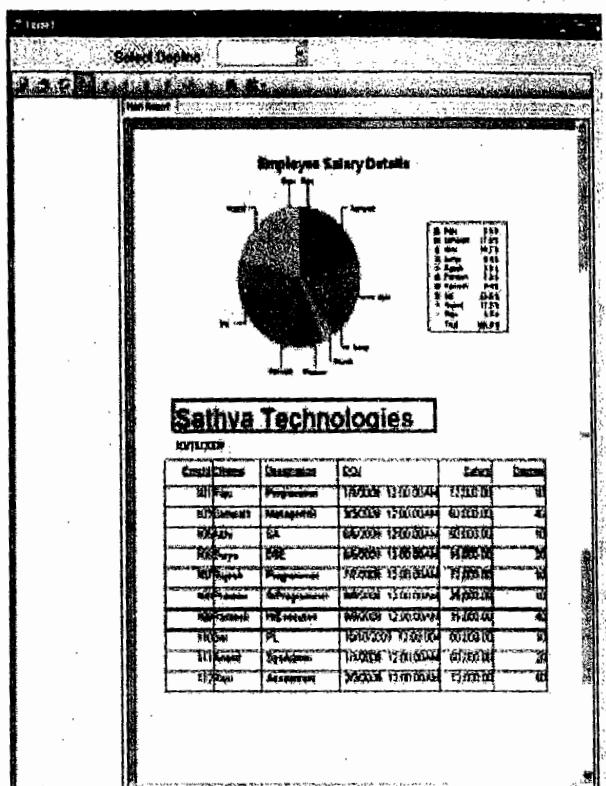
Create a Report Normally with out any grouping as in Example19.01 → Go to Report Header → Click with Right Mouse Button → Click on Insert → Click on Chart → Select Type of Chart as Bar Chart → Click on Data Tab Page → Select “For each Record” Option in place of “On Change of” Option → Select the Field EName and add to “For Each Record” option → Select the Field Salary and Add to → “Show Value” option → Click on Text tab Page Type the Required Chart Titles → Click on Ok

Design of Form



Code is same as in Example 19.03

Run the Application and Check



Chapter24

Working with XML

24.1 Why Xml

- In general when applications designed in various technologies can not communicate with each other, because every technology will have its own proprietary format.
- But all companies can not be computerized with the same technology and which is impossible.
- So if we want to communicate among the applications designed in various technologies we use XML.

24.2 What Is Xml

- XML is an open standard format proposed by the W3 org.
- XML stands for EXtensible Markup Language.
- XML is a tag based Programming Language.
- XML is just used to describe the data.
- Pure XML can not be used for designing purpose.
- XML tags are case sensitive.
- XML code can be written in any Text Editor.
- To execute the XML code it is required to have the XML Parser.
- By default all the Browsers contain XML Parser, so XML file can be executed in any browser.
- XML file should start with a line known as start document and is like <?xml version="1.0" encoding="utf-8"?>
- XML file will store the data in hierarchical manner.
- XML file will contain only one Root tag.
- Every root tag will contain one or more parent tags
- Every parent tags will contain one or more child tags
- In general Root tag will represent the Database name, Parent tag will represent the Table name, child tag(s) will represent the field(s) name(s).
- With in XML data is stored in a file with an extension of .Xml, structure is stored in a separate file with an extension .Xsd (XML Schema Document).

24.3 Structure of XML File

```
<Root Tag>
  <Parent Tag1>
    <ChildTag1> Data </ChildTag1>
    <ChildTag2> Data </ChildTag2>
    :
    :
  </Parent Tag1>
  <Parent Tag1>
    <ChildTag1> Data </ChildTag1>
    <ChildTag2> Data </ChildTag2>
    :
    :
```

```
</Parent Tag1>
:
:
<Parent Tag2>
<ChildTag1> Data </ChildTag1>
<ChildTag2> Data </ChildTag2>
:
:
</Parent Tag2>
<Parent Tag2>
<ChildTag1> Data </ChildTag1>
<ChildTag2> Data </ChildTag2>
:
:
</Parent Tag2>
:
:
</Root Tag>
```

24.4 Creating an Xml File using Notepad

Open the Notepad Application → Type the following code

```
<?xml version="1.0" encoding="utf-8" ?>
<Employee>
<EmpDetails>
<EmpId>101</EmpId>
<EName>Raju</EName>
<Designation>Programmer</Designation>
<DOJ>01/01/2008</DOJ>
<Salary>15000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
<EmpId>102</EmpId>
<EName>Gopal</EName>
<Designation>TL</Designation>
<DOJ>02/02/2008</DOJ>
<Salary>35000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
<EmpId>103</EmpId>
<EName>Abhi</EName>
<Designation>SA</Designation>
<DOJ>03/03/2008</DOJ>
<Salary>50000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
<EmpId>104</EmpId>
<EName>Sai</EName>
```

```
<Designation>PL</Designation>
<DOJ>04/04/2008</DOJ>
<Salary>80000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<Dept>
<Deptno>10</Deptno>
<DName>Development</DName>
<Location>Hyderabad</Location>
</Dept>
<Dept>
<Deptno>20</Deptno>
<DName>Networks</DName>
<Location>Delhi</Location>
</Dept>
<Deptno>30</Deptno>
<DName>Database</DName>
<Location>Mumbai</Location>
</Dept>
<Deptno>40</Deptno>
<DName>Accounts</DName>
<Location>Bangalore</Location>
</Dept>
<Deptno>50</Deptno>
<DName>Accounts</DName>
<Location>Chennai</Location>
</Dept>
</Employee>
```

- Save the file with an extension of .Xml.
- Open it in Internet Explorer (or in any other browser) and check it.

24.5 Creating an Xml File in VS.Net

In VS.Net Microsoft integrated both Xml editor and Xml parser together. These are installed along with VS.Net only.

Steps to Create an XML File

Go to Solution Explorer → Select the Solution → Click with Right Mouse Button → Click on Add → Click on New Item → Select XML File template → Type the File Name (Sample.Xml) → Click on Add → Write the following code and check

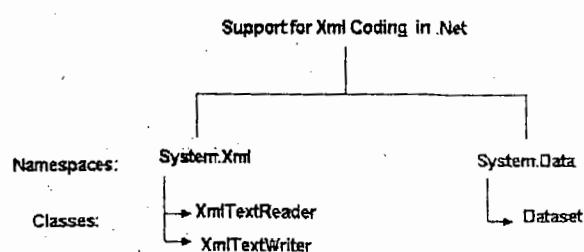
```
<?xml version="1.0" encoding="utf-8" ?>
<Employee>
<EmpDetails>
<EmpId>101</EmpId>
<EName>Raju</EName>
<Designation>Programmer</Designation>
<DOJ>01/01/2008</DOJ>
<Salary>15000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
```

```

<EmpId>102</EmpId>
<EName>Gopal</EName>
<Designation>TL</Designation>
<DOJ>02/02/2008</DOJ>
<Salary>35000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
<EmpId>103</EmpId>
<EName>Abhi</EName>
<Designation>SA</Designation>
<DOJ>03/03/2008</DOJ>
<Salary>50000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<EmpDetails>
<EmpId>104</EmpId>
<EName>Sai</EName>
<Designation>PL</Designation>
<DOJ>04/04/2008</DOJ>
<Salary>80000</Salary>
<Deptno>10</Deptno>
</EmpDetails>
<Dept>
<Deptno>10</Deptno>
<DName>Development</DName>
<Location>Hyderabad</Location>
</Dept>
<Dept>
<Deptno>20</Deptno>
<DName>Networks</DName>
<Location>Delhi</Location>
</Dept>
<Dept>
<Deptno>30</Deptno>
<DName>Database</DName>
<Location>Mumbai</Location>
</Dept>
<Dept>
<Deptno>40</Deptno>
<DName>Accounts</DName>
<Location>Banglore</Location>
</Dept>
<Dept>
<Deptno>50</Deptno>
<DName>Accounts</DName>
<Location>Chennai</Location>
</Dept>
</Employee>

```

24.5 Coding of Xml Files in .Net



24.5.1 Working with XmlTextReader Class

- This class will contain necessary properties and functions that help to read the data from any xml file

Exa

➤ Properties With XmlTextReader Class:

Srno	Property and options	Type	Description
01	EOF	Bool	Returns true when encounters EOF otherwise returns false
02	Name	String	Returns the opening or closing tag name
03	NodeType	XmlNodeType	Used to Identify the type of node and all types of nodes are present in an enumeration XmlNodeType
04	Value	String	Returns the data present in child nodes
05	ValueType	Type	Retuns the data type of the node that is being read in CLRs Data type format

➤ Methods with DataAdapter Class

Srno	Method	Return Type	Description
01	Read()	Bool	Reads the data present in an XML file Returns true as long as non EOF is encountered and returns false when EOF is encountered Read() will read the data in Sequential only and forward only manner.
02	Close()	Void	Used to close an xml file that has been opened for reading.

24.5.2 steps to with XmlTextReader Class

- **Step1:** Include NameSpace
 - **Ex:** using System.Xml;
- **Step2:** Declare XmlTextReader class Object
 - **Syntax:** ClassName ObjectName;
 - **Ex:** XmlTextReader XR;
- **Step3:** Define XmlTextReader class Object
 - **Syntax:** ObjectName=new ClassName("Xml File Path");
 - **Ex:** XR=new XmlTextReader("D:\\Sample.Xml");
- **Step4:** Raed the data from Xml File using Read() Method
 - **Syntax:** ObjectName.Read();
 - **Ex:** XR.Read();
- **Step4:** Close the Xml File after completion of Reading.
 - **Syntax:** ObjectName.Close();
 - **Ex:** XR.Close();

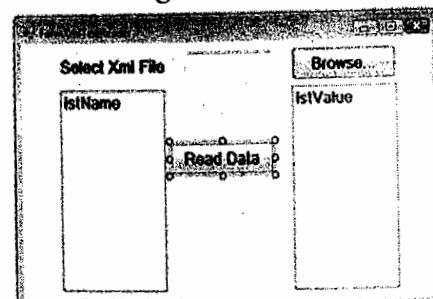
Example 24.1

Example with XmlTextReader Class: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;
namespace WAXML
{
    public partial class FormXmlTextReader : Form
    {
        public FormXmlTextReader()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void cmdBrowse_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "XML Files (*.xml)|All Files (*.*)";
            openFileDialog1.ShowDialog();
            txtFName.Text = openFileDialog1.FileName;
        }
        private void cmdReadData_Click(object sender, EventArgs e)
        {
            XmlTextReader Xr;
            Xr = new XmlTextReader(txtFName.Text);
            while (Xr.Read() == true)
            {
                if (Xr.NodeType == XmlNodeType.Element)
                {
                    lstName.Items.Add(Xr.Name);
                }
                if (Xr.NodeType == XmlNodeType.Text)
                {
                    lstValue.Items.Add(Xr.Value);
                }
            }
            Xr.Close();
        }
    }
}
```

Design of Form



openFileDialog1

24.5.3 Working with XmlTextWriter Class

- This class will contain necessary properties and functions that help to write the data into any xml file
- Methods with XmlTextWriter Class

Srno	Method	Return Type	Description
01	Close()	Void	Used to close an xml file that has been opened for reading.
02	Flush()	Void	Used to clear the buffer memory occupied by the XmlTextWriter Class object
03	WriteComment(string Text)	Void	Used to write the required comment data into the file
04	WriteEndDocument()	Void	Closes any open elements or attributes and puts the writer back in the Start state.
05	WriteEndElement()	Void	Used to write the end element to the respective opening tag
06	WriteStartDocument()	Void	Used to write the XML declaration with the version "1.0" like <?xml version="1.0" encoding="utf-8" ?>
07	WriteStartElement(string Name)	Void	Used to write the opening tag with the given Name
08	WriteValue(AnyType Data)	Void	Used to write the given value with the given type into the child node of an Xml file

24.5.2 steps to with XmlTextWriter Class

- **Step1:** Include NameSpace
 - **Ex:** using System.Xml;
- **Step2:** Declare XmlTextWriter class Object
 - **Syntax:** ClassName ObjectName;
 - **Ex:** XmlTextWriter XW;
- **Step3:** Define XmlTextWriter class Object
 - **Syntax:** ObjectName=new ClassName("Xml File Path", Encoding Format);
 - **Ex:** XW=new XmlTextWriter("D:\\Sample.Xml", System.Text.Encoding.UTF8);
- **Step4:** Write the data into the Xml File using required Methods
- **Step5:** Close the Xml File after completion of Writing.
 - **Syntax:** ObjectName.Close();
 - **Ex:** XW.Close();

Example 24.2

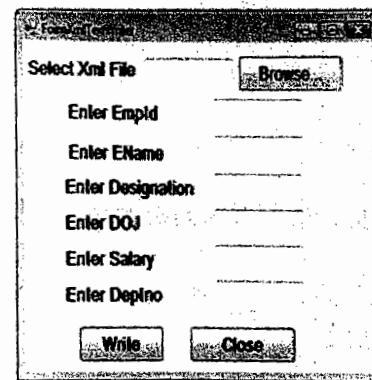
Example with XmlTextWriter Class: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Xml;

namespace WAXML
{
    public partial class FormXmlTextWriter : Form
    {
        XmlTextWriter Xw;
        public FormXmlTextWriter()
        {
            InitializeComponent();
        }
        private void cmdBrowse_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "XML Files (*.xml)|All Files (*.*)";
            openFileDialog1.ShowDialog();
            txtFName.Text = openFileDialog1.FileName;
            Xw = new XmlTextWriter(txtFName.Text, System.Text.Encoding.UTF8);
            Xw.WriteStartDocument();
            Xw.WriteStartElement("Employee");
        }
        private void cmdWrite_Click(object sender, EventArgs e)
        {
            Xw.WriteStartElement("EmpDetails");
            Xw.WriteStartElement("EmpId");
            Xw.WriteValue(Convert.ToInt32(txtEmpId.Text));
            Xw.WriteEndElement();
            Xw.WriteStartElement("EName");
            Xw.WriteValue(txtEName.Text);
            Xw.WriteEndElement();
            Xw.WriteStartElement("DEsignation");
            Xw.WriteValue(txtDesignation.Text);
            Xw.WriteEndElement();
            Xw.WriteStartElement("DOJ");
            Xw.WriteValue(Convert.ToDateTime(txtDOJ.Text));
            Xw.WriteEndElement();
            Xw.WriteStartElement("Salary");
            Xw.WriteValue(Convert.ToDouble(txtSalary.Text));
            Xw.WriteEndElement();
            Xw.WriteStartElement("Deptno");
            Xw.WriteValue(Convert.ToInt32(txtDeptno.Text));
            Xw.WriteEndElement();
        }
    }
}
```

Design of Form



```
Xw.WriteEndElement();
MessageBox.Show("Record is Written into XML File");
}
private void cmdClose_Click(object sender, EventArgs e)
{
    Xw.WriteEndElement();
    Xw.Close();
    MessageBox.Show("File Has Closed");
}
}
```

24.5.4 Working with Dataset Class

This class supports some functions used to write the data into xml file and to read the data from xml file

Methods with Dataset Class:

Srno	Method	Return Type	Description
01	Read() WriteXml("FileName")	bool	Used to write the complete data of dataset into the given Xml File.
02	WriteXmlSchema("FileName")	void	Used to write the structure of tables present in dataset into the given Xsd File..
03	ReadXml("FileName")	enum	Used to read the data of Xml file into the dataset
04	ReadXmlSchema("FileName")	void	Used to read the structure of tables of Xsd file into the dataset

Example 24.3

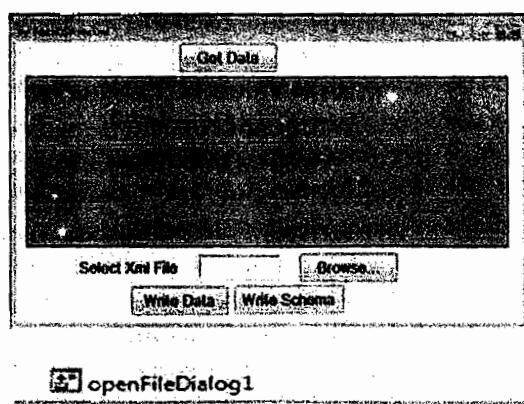
Example with WriteXml and WriteXmlSchema Methods: -

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WAXML
{
    public partial class FormDSWriteXml : Form
```

Design of Form



```

SqlConnection Con = new SqlConnection("Server=.;User Id=sa;Password=abc;
DataBases=Employee");
SqlDataAdapter Da; DataSet Ds;
public FormDSWriteXml()
{
    InitializeComponent();
}
private void cmdGetData_Click(object sender, EventArgs e)
{
    Da = new SqlDataAdapter("Select * from EmpDetails", Con);
    Ds = new DataSet();
    Da.Fill(Ds, "EmpDetails");
    GVSample.DataSource = Ds.Tables[0];
}
private void cmdBrowse_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "Xml Files|*.xml|XSD Files|*.xsd|All Files|*.*";
    openFileDialog1.ShowDialog();
    txtFName.Text = openFileDialog1.FileName;
}
private void cmdWriteData_Click(object sender, EventArgs e)
{
    Ds.WriteXml(txtFName.Text);
    MessageBox.Show("Data Has Written into XML File");
}
private void cmdWriteSchema_Click(object sender, EventArgs e)
{
    Ds.WriteXmlSchema(txtFName.Text);
    MessageBox.Show("Structure Has Written into XSD File");
}
}

```

Example 24.4

Example with ReadXml Method: -

Code:

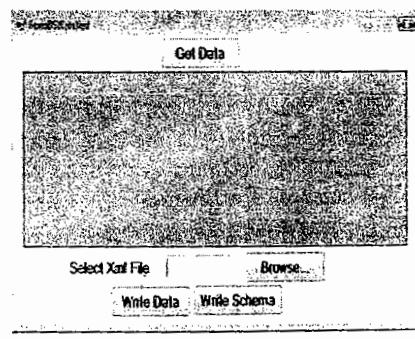
```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WAXML
{
    public partial class FormDSReadXml : Form
    {
        public FormDSReadXml()

```

Design of Form



```

    {
        InitializeComponent();
    }

    private void cmdBrowse_Click(object sender, EventArgs e)
    {
        openFileDialog1.Filter = "XML Files (*.Xml|All Files (*.*)";
        openFileDialog1.ShowDialog();
        txtFName.Text = openFileDialog1.FileName;
    }

    private void cmdReadData_Click(object sender, EventArgs e)
    {
        DataSet Ds = new DataSet();
        Ds.ReadXml(txtFName.Text);
        GVSample.DataSource = Ds.Tables[0];
    }
}

```

25

In
av
to
fil
pe
be
the

25.

>

private void cmdReadData_Click(object sender, EventArgs e)
 {
 DataSet Ds = new DataSet();
 Ds.ReadXml(txtFName.Text);
 GVSample.DataSource = Ds.Tables[0];
 }
}

> 1

> 1

> t

> T

> I

> N

25.3

> In

te:

> Th

Fo

> Th

Bi

Chapter 25

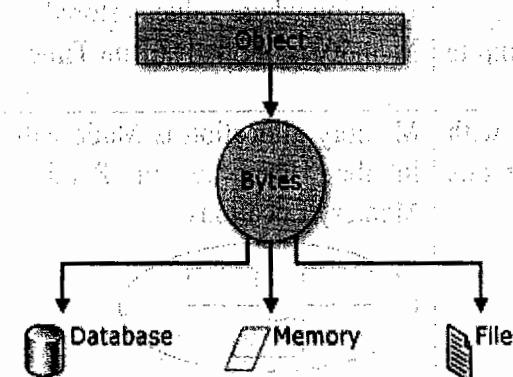
Serialization and Deserialization

25.1 Introduction

In general when an object is created its data is stored in to the object, data of the object is available until the object is destroyed, once object is destroyed its data will be lost. If we want to use the same data next time it is not possible. Making object data permanent by storing into a file or database or to any other memory device is known as **Object Persistence** once object is persisted data can be retrieved and reused again when ever required, for this purpose data should be converted from the object form to other streams, this process is called as **Serialization**, and the reverse process is called as **Deserialization**.

25.2 Serialization and Deserialization

- **Serialization** is the process of converting an object into a stream of bytes in order to persist it to memory, a database, or a file.



- Main purpose of serialization is to save the state of an object in order to be able to recreate it when needed.
- When any object is serialized to a stream, it does not just the data, but also information about the object's type, such as its version, culture, and assembly name.
- To serialize any class the class should be preceded with [Serializable] attribute.
- Deserialization is the reverse process of serialization.

25.3 Types of Serialization

- .NET supports 3 types of serialization
 1. Binary Serialization
 2. SOAP Serialization
 3. XML Serialization

25.3.1 Binary Serialization

- In this object data is converted into binary format by using binary encoding and stored into a text file or passed in the network or for some other purpose.
- The namespace used for binary serialization is System.Runtime.Serialization.Formatters.Binary
- This namespace will contain a class known as BinaryFormatter contains functions to perform Binary Serialization and Deserialization.

Uses of Binary Serialization

Binary serialization is used

- In storage of any data on to the disk.
- In socket-based network streams.

Advantages of Binary Serialization

- Binary Serialization works at faster rate.
- Provides better performance when storing data.

Disadvantages of Binary Serialization

- Binary Serialized data is not suitable for passing data through a firewalls in the network

Methods in BinaryFormatter Class

Srno	Method	Return Type	Description
01	Serialize(System.IO.Stream, SerializationStream, object graph)	void	Serializes an object, or graph of connected objects, to the given stream.
02	DeSerialize(System.IO.Stream Stream)	object	Deserializes a stream into an object graph.

25.3.2 SOAP Serialization

- In this object data is converted into SOAP format by and stored into an XML file or passed in the network or for some other purpose.
- SOAP is Simple Object Access Protocol based on XML.
- SOAP transfers the data in the form SOAP envelops.
- SOAP envelop contains 3 parts
 - SOAP Header
 - SOAP Body
 - SOAP Error
- The namespace used for SOAP serialization is System.Runtime.Serialization.Formatters.SOAP
- This namespace will contain a class known as SOAPFormatter contains functons to perform SOAP Serialization and Desirializatlon.

Uses of SOAP Serialization

- SOAP serialization is widely used in XML web services.

Advantages of SOAP Serialization

- SOAP is open standard format based on XML.
- SOAP formatted data is not stopped by the firewalls in the network.

- **Disadvantages of SOAP Serialization**
- Execution od SOAP serilization is slow as compared to Binary serialization.
- **Methods in BinaryFormatter Class**

Srno	Method	Return Type	Description
01	Serialize(System.Io.Stream.SerializationStream, object graph)	void	Serializes an object, or graph of connected objects, to the given stream.
02	DeSerialize(System.Io.Stream Stream)	object	Deserializes a stream into an object graph.

25.3.3 XML Serialization

- In this object data is converted into XML format by and stored into an XML file or passed in the network or for some other purpose.
- XML is an open standard format used to tranfer the data in the network.
- The namespace used for XML serailization is System.XML.Serialization.
- This namespace will contain a class known asXMLSerializer contains functons to perform XML Serialization and Desirializaton.
- **Uses of XML Serialization**
- SOAP serialization is widely used to communicate among the applicatons designed in various technologies.
- **Advantages of XML Serialization**
- XML is an open standard format can be understood by the all techmologies.
- **Disadvantages of SOAP Serialization**
- Execution od XML serilization is slow as compared to Binary serialization.
- **Methods in XMLSerializer Class**

Srno	Method	Return Type	Description
01	Serialize(System.Io.Stream.SerializationStream, object graph)	void	Serializes the specified Object and writes the XML document to a file using the specified Stream.
02	DeSerialize(System.Io.Stream Stream)	object	Deserializes the XML document contained by the specified Stream.

25.4 Methods of Serialization

- .NET supports 3 types of Serialization.
 - Basic Serialization
 - Custom Serialization
 - Designer Serialization

Note: Following points are as it is from MSDN

25.4.1 Basic Serialization

- The only requirement in basic serialization is that the object have the `SerializableAttribute` attribute applied.
- The `NonSerializedAttribute` can be used to keep specific fields from being serialized.
- When you use basic serialization, the versioning of objects may create problems, in which case custom serialization may be preferable.
- Basic serialization is the easiest way to perform serialization, but it does not provide much control over the process.

25.4.2 Custom Serialization

- In custom serialization, you can specify exactly which objects will be serialized and how it will be done.
- The class must be marked `SerializableAttribute` and implement the `ISerializable` interface.
- If you want your object to be deserialized in a custom manner as well, you must use a custom constructor.

25.4.2 Designer Serialization

- With designer serialization, you can persist the state of your components at design time or run time.
- Designer serialization is a special form of serialization that involves the kind of object persistence usually associated with development tools.
- Designer serialization is the process of converting an object graph into a source file that can later be used to recover the object graph.
- A source file can contain code, markup, or even SQL table information.
- Designer serialization works for all common language runtime objects.
- Designer serialization differs from typical object serialization in several ways:
 - The object performing the serialization is separate from the run-time object, so that design-time logic can be removed from a component.
 - The serialization scheme was devised under the assumption that the object will be created in a fully initialized state and then modified through property and method invocations during deserialization.
 - Properties of an object that have values that were never set on the object are not serialized. Conversely, the deserialization stream may not initialize all property values. For a more detailed description of serialization rules, see the "General Serialization Rules" section later in this topic.

- Emphasis is placed on the quality of the content within the serialization stream, rather than the full serialization of an object. If there is no defined way to serialize an object, that object will be passed over rather than raising an exception. Designer serialization has ways to serialize an object in a simple, human-readable form, rather than as an opaque blob.
- The serialization stream may have more data than is needed for deserialization. For example, source code serialization has user code mixed in with the code needed to deserialize an object graph. This user code must be preserved on serialization and passed over on deserialization.

Example 25.1

Example with Binary Serialization: -

Create a new WindowsFormsApplicaton wih the name WASerialize → Create a new class in that with he name clsEmployee → write the following code.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace WASerialize
{
    [Serializable]
    class ClsEmployee
    {
        public int EmpId, EAge;
        public string EName, EAddress;
    }
}
```

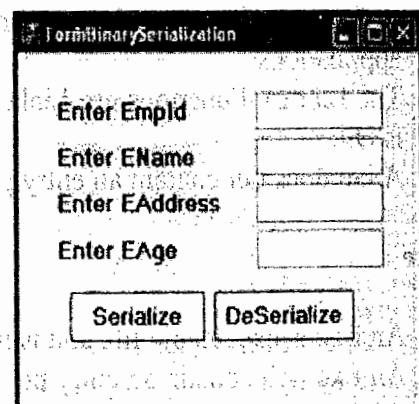
Design the form and write the following code

Design of Form

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace WASerialize
{
    public partial class FormBinarySerialization : Form
    {
```



```

FileStream FSRW; BinaryFormatter Obj1 = new BinaryFormatter();
ClsEmployee Obj2 = new ClsEmployee();
public FormBinarySerialization()
{
    InitializeComponent();
}
private void btnSerialize_Click(object sender, EventArgs e)
{
    Obj2.EmpId = Convert.ToInt32(txtEmpId.Text);
    Obj2.EName = txtEName.Text;
    Obj2.EAddress = txtEAddress.Text;
    Obj2.EAge = Convert.ToInt32(txtEAge.Text);
    FSRW = new FileStream("D:\\Sample.txt", FileMode.Create, FileAccess.Write);
    Obj1.Serialize(FSRW, Obj2);
    FSRW.Flush();
    FSRW.Close();
    MessageBox.Show("Serialization Success");
}
private void btnDeSerialize_Click(object sender, EventArgs e)
{
    FSRW = new FileStream("D:\\Sample.txt", FileMode.Open, FileAccess.Read);
    Obj2 = (ClsEmployee)Obj1.Deserialize(FSRW);
    FSRW.Flush();
    FSRW.Close();
    txtEmpId.Text = Obj2.EmpId.ToString();
    txtEName.Text = Obj2.EName;
    txtEAddress.Text = Obj2.EAddress;
    txtEAge.Text = Obj2.EAge.ToString();
}
}
}
}

```

Go to D:\ and observe the Sample.txt file you find the serialized data.

Example 25.2

Example with SOAP Serialization: -

Create a new WindowsForm in the same application Design the form and write the following code

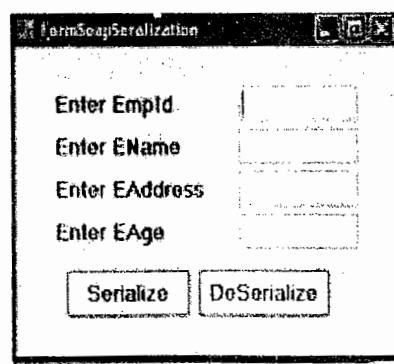
Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.Serialization.Formatters.Soap;
using System.IO;

```

Design of Form



Go to

```

nam espace W ASerialize
{
    public partial class Form SoapSerialization : Form
    {
        FileStream FSRW ; SoapForm atterObj1 = new SoapForm atter();
        ClsEmployee Obj2 = new ClsEmployee();
        public Form SoapSerialization()
        {
            InitializeCom ponent();
        }
        private void btnSerialize_Click (object sender, EventArgs e)
        {
            Obj2 Em pId = Convert.ToInt32(txtEm pId.Text);
            Obj2 ENam e = txtENam e.Text;
            Obj2 EAddress = txtEAddress.Text;
            Obj2 EAge = Convert.ToInt32(txtEAge.Text);
            FSRW = new FileStream ("D :\Sample1.txt", FileMode.Create, FileAccess.Write);
            Obj1 Serialize(FSRW , Obj2);
            FSRW Flush();
            FSRW Close();
            MessageBox.Show ("Serialization Success");
        }
        private void btnDeSerialize_Click (object sender, EventArgs e)
        {
            FSRW = new FileStream ("D :\Sample1.txt", FileMode.Open, FileAccess.Read);
            Obj2 = (ClsEmployee)Obj1 Deserialize(FSRW );
            FSRW Close();
            txtEm pId.Text = Obj2 Em pId.ToString ();
            txtENam e.Text = Obj2 ENam e;
            txtEAddress.Text = Obj2 EAddress;
            txtEAge.Text = Obj2 EAge.ToString ();
        }
    }
}

```

Go to D:\ and observe the Sample1.txt file you find the serialized data

Chapter 26

Setup and Deployment

26.1 Introduction

Setup:

- Setup is the process of preparing the designed software to install in client computer(s) or in other computer(s).
- We gather all the required resources Like
 - .Exe File
 - Supportive dlls if any
 - Image Files / Other resources used with in the Application

Deployment:

- Deployment is the process by which you distribute a finished application or component to be installed on other computers.
- For distributed applications, you usually create a separate deployment project for each tier in your application. For example, an application with three tiers would require three deployment projects, one each for the data, business logic, and client tiers.
- Deployment projects in Visual Studio provide a great deal of flexibility in determining where and how a solution will be deployed.
- In the process of deploying a solution, you might want to specify where files should be installed on a target computer, specify what registry keys should be added, or set special conditions or requirements for installation. You might also want to customize the user interface that is shown during installation, or run code to perform custom actions on a target computer.
- The deployment editors are used to specify and customize settings and properties for all aspects of deployment.

Methods of Setup and Deployment:

- XCopy Method
- Windows Setup Method

Windows Setup Method:

- In this method first we prepare Setup.exe by using
 - Setup Wizard
 - Setup Project
 - Cab Project

Creating Setup.exe using Setup Wizard Method

Here we crate Setup.exe for the application developed in Chapter 18 and for the Example 18.02

Steps

Click on File → Click on New → Click on Project → Double click on “Other Project Types”
→ Select “Setup and Deployment option → Select “Setup Wizard” template → Type the



Application Name (WA3TierSetup) → Select the Location to save → Click on OK → Setup Wizard will start → Click on Next → Select the option "Create a Setup for Windows Application" → Click on Next → Click on Add... (Ellipsis Button) → go to the Location where Exe is saved (i.e. Exe created in Example18.02) → Select the file WA3Tier.exe → Click on Open (Do not select any Dlls Exe file will get its referenced Dlls) → Click on Next → Click on Finish → you find a File System Editor Window with 3 Folders.

Application Folder

Users Desktop

Users Programs Menu

→ Select the Application Folder, here you find 3 Files 1).Exe File 2) 2 Dll Files.

Select the .Exe File → Click with Right Mouse button → Create a Shortcut for this exe → Drag and Drop this shortcut into Users Desktop folder → Similarly create another shortcut for exe and place this in Users Programs Menu Folder → Build the Application → Setup.exe is created and is ready to use.

For installing this software use the following steps

Go to the location where Setup.exe is created → Double Click on it → Follow the wizard → after completion of the installation observe on Desktop one shortcut, in Programs menu you find one more short cut → Run it and Check.

