Q1) Explain .net framework in detail :

The .NET Framework is a software development platform developed by Microsoft. The framework was meant to create applications which would run on the window platform.

The first version of .Net framework was released in the year 2002. The version was called .Net framework 1.0

.NET provide :-

1) Integrated Environment.
2) provide portable environment.
3) managed environment.

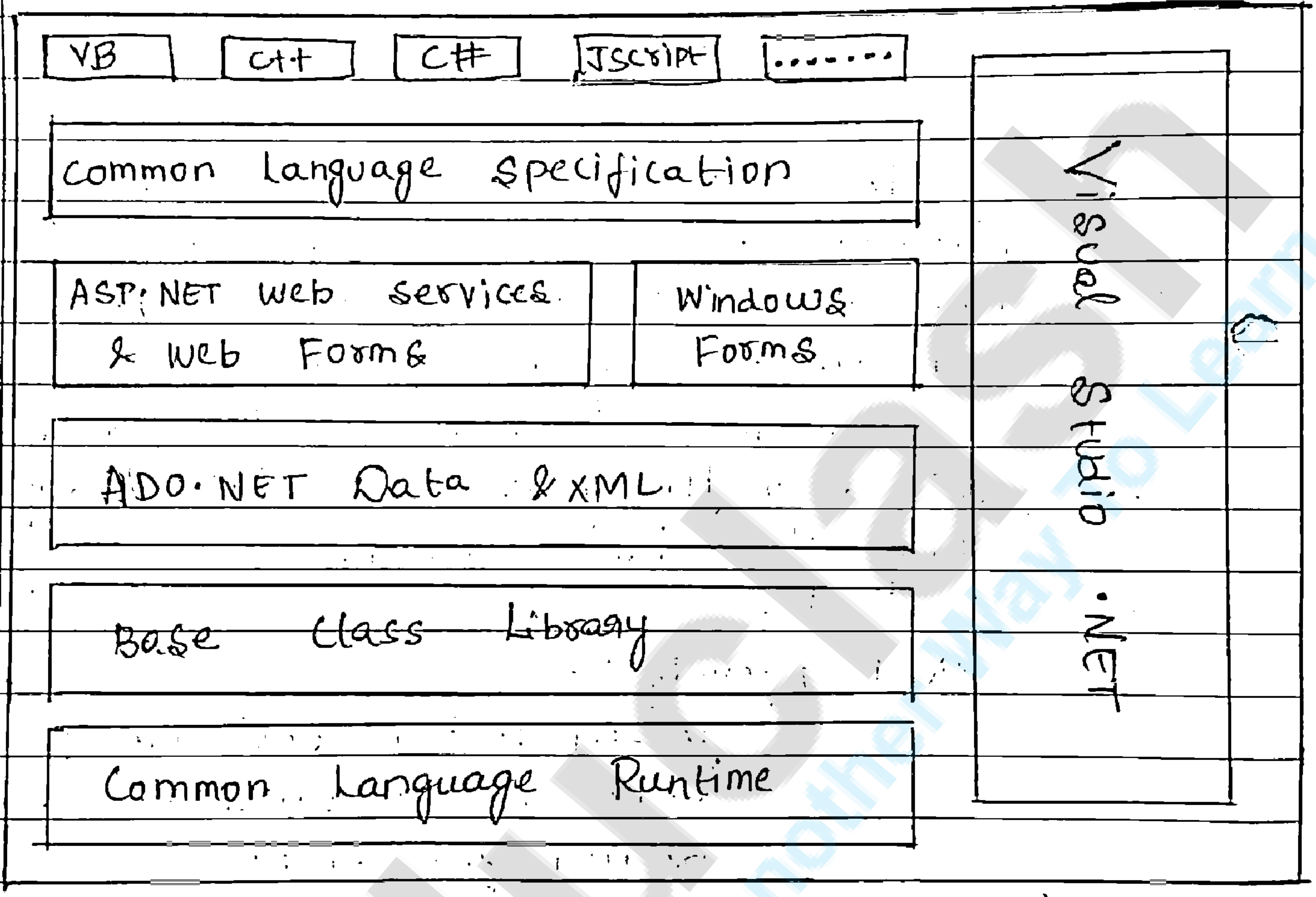
↳ .NET framework can be used to create both Form based & web-based application. Web services can also be developed using the .NET framework

↳ This framework contains large no. of class libraries known as Framework class library (FCL)

↳ The s/w program written in .NET are execute in execu^n environment that is called CLR (Common language Runtime)

↳ These both are based on core & essential parts of the .NET framework.

↳ This Framework provides various services like : memory management , networking , security & memory safety . It also supports numerous programing languages like C#, VB etc.



~~Common Language Runtime (CLR)~~

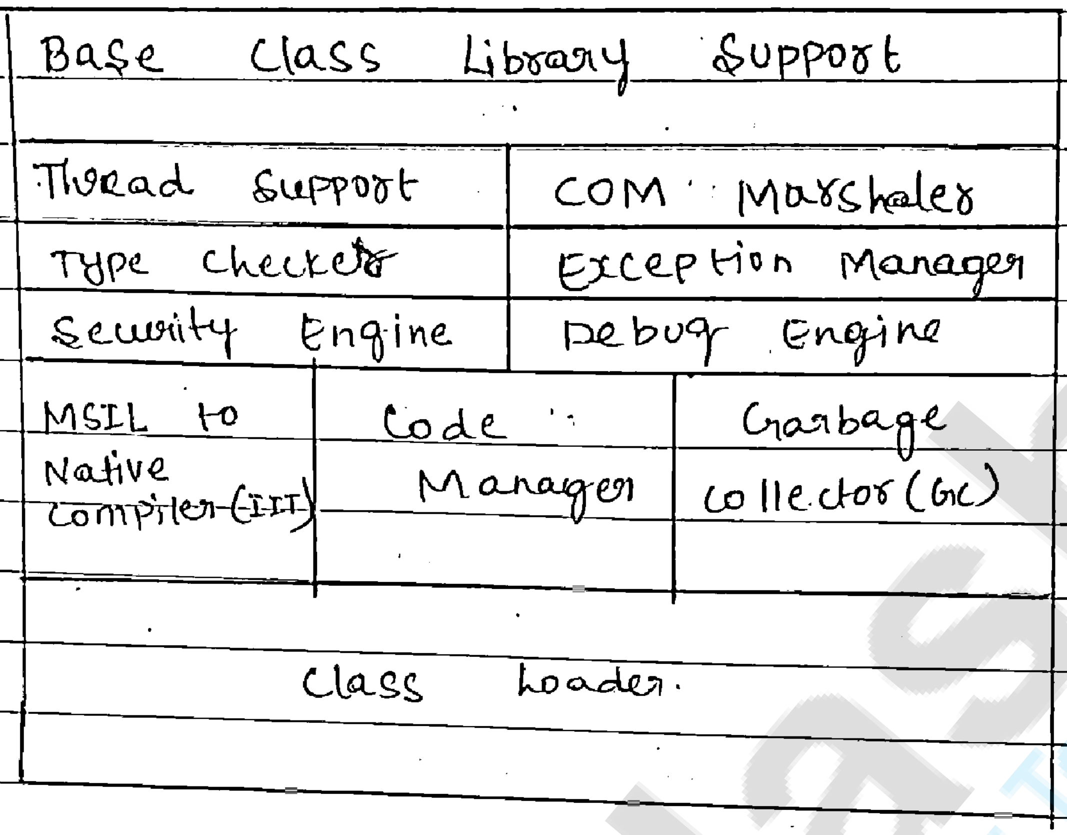
~~- Common Type System (CTS)~~

~~- Common metadata~~

~~- Intermediate Language (IL) to native code Compiler~~

CLR Architecture :-

Responsible for converting source code to respective language.



Exception ~~The~~ Handling :- Exceptions are error which occur when the application is executed

↳ If an application tries to open a file on the local machine, but the file is not open

Garbage Collection - Garbage Collection is the process of removing unwanted resources when they are no longer required.

↳ ~~A File~~ handle which is no ~~longer~~ longer required. If the application has finished all operations on a file, then the file handle may no longer be required.

Working with various programming language.

1) ~~Ho~~ Language - The first level is the programming language itself. The most common ones are VB.Net & C#.

2) Compiler :- There is compiler which will be separate for each programming language. So underlying the VB.Net language, there will be a separate VB.Net compiler. Similarly for C# you will have another compiler.

3) Common language Interpretor :- This is final layer in .NET which would be used to run a .net program developed in any programming language. So the subsequent compiler will send the program to CLI layer to run .Net application.

Base Class Library :-

The .NET Framework includes a set of standard class library. A class library is a collection of methods & functions that can be used for the core purpose.

For eg. there is a class library with methods to handle all file - ~~Vot~~ level operations. So there is a method which can be used to read the text from a file. Similarly there is a method to write text to file.

Unified Classes

| | |
|---|---|
| Web classes (ASP.Net) (controls caching, security, session, config etc) | |
| Data (ADO.Net) (ADO. SQL, type etc) | Windows Forms (Design o/p model etc) |
| XML Classes (XSLT, path, serializatn etc) | Drawing classes (drawing's & text etc) |
| System classes Collections, diagnostics, globalization, I/O. security Security, Threading, serialization, Reflection message etc) | |

## FCL (Framework class library)

Most of methods are split into either the system or microsoft namespaces.

Namespace is a collection of different classes. Also in C# its userdefined

## Common Language Specification (CLS)

CLS is a fundamental set of language features supported by the Common Language Runtime (CLR) of the .NET Framework. CLS was designed to support language constructs commonly used by developers & to produce verifiable code, which allows all CLS - compliant language to ensure the type of safety of code.

CLS includes features common to many object oriented programming language. It forms a functionality of common type system (CTS).

## Languages :-

1) WinForms :- This is used for developing Forms - based application which would run on an user machine. Notepad is an example of a client - based application.

2) ASP. Net :- This is used for developing web - based applications, which are made to run on any browser such as Internet Explorer, Chrome or Firefox.

- The web application would be processed on a server, which would have internet information services installed.
- Internet information services or IIS is a microsoft component which is used to execute on ASP.Net application.
- The result of execution is then sent to the client machine, & the output is shown in browser.

3) ADO.NET :- This technology is used to develop applications to interact with all databases such as oracle or Microsoft SQL Server.

write a Short not on

Working of CLR :-

↳ CLR is the basic & virtual machine component of .NET framework. It's the run time environment in the .NET framework that runs the codes & helps in making the development process easier by providing various services

↳ Basically it is responsible for managing the execution of .Net programming language

↳ Internally, CLR implements the VES (virtual execution system) which is defined in the microsoft's implementation of CLI (common language Infrastructure

↳ The code that runs under the CLR is termed as the managed code. In other words you can say that CLR provides a managed execution environment for the .Net programs by improving the security including the cross language integration & a rich set of class library etc.

↳ CLR is present in every .NET Framework version. Below table illustrate the CLR version .NET Framework.

| CLR Version | .NET Framework Version |
|---|---|
| 1.0 | 1.0 |
| 1.1 | 1.1 |
| 2.0 | 2.0 |
| 2.0 | 3.0 |
| 2.0 | 3.5 |
| 4 | 4 |
| 4 | 4.5 |
| 4 | 4.6 |
| 4 | 4.7 |

## * Role of CLR in the execution of a C# program

↳ Suppose you have written a C# program & save it in a file which is known as source code.
↳ language specific compiler compiles the source code into the MSIL which is also known as the CIL, along with its metadata, metadata includes ~~the~~ all the types, actual implementation of each function of the program

↳ MSIL is machine independent code.
↳ Now CLR comes into existence, CLR provides the services & runtime environment to the MSIL code. Internally CLR includes the JIT (Just in Time) compiler which converts the MSIL code to machine code which further executed by CPU.

↳ CLR also uses the .NET framework class library. Meta data provides the information about the programming language, environment, Version & class libraries to the CLR by which CLR handles the MSIL Code.

↳ As a CLR is a common so it allows an instance of a class that written in a different languages to call a method of the class which written in another language.

2) Q
b) Assembly & It's type.

↳ Assembly is a compiled output of program which are used for easy deployment of an application. They are executables in the form of exe or dll

↳ Assembly is building block of .NET application

↳ Every application after the execution generate the file called assembly

↳ An assembly contains self describing binary data that can be either dll or executable(exe)

↳ It is also contain collection of time such as classes, interfaces, structures the single assembly contains multiple code file or single code file can have more than one assembly.

↳ The assembly contains CLR executable code

↳ Assemblies provide the infrastructure to allow the runtime to fully understand the contents of an application & to enforce the versioning & dependency rules defined by the application

In .Net 3 types of Assemblies are available

1) Private Assemblies:-

↳ private assemblies are accessible by a single application

↳ They are reside within application folder & are unique by name.

↳ They are 1, can be directly used by copying & pasting them to the bin folder.

2) Shared Assemblies :-

↳ Microsoft offers the shared assembly for those components that must be distributed
↳ It centered around two principles
(1) Firstly called side-by-side execution, allows the CLR to house multiple version of the same component on a single machine
(2) Secondly, termed binding, ensures that client obtain the version of the component they expect.

3) Satellite Assemblies :-

Satellite assemblies are the assemblies to provide the support for multiple languages based on different cultures.

These are kept in different modules based on the different categories available.

Q) Metadata & GAC

Metadata :-

Metadata describe a program (i.e) in the form of Binary information stored in CLR Portable executable (PE) file or in a memory, When a compilation of code takes place in a PE file. The metadata is inserted into I part of a file the code is converted into IL & inserted into another part of file.

↳ The metadata describe every type & member.

↳ When the code is in run mode the CLR loads the metadata into the memory & findout into about the code classes from a member from etc.

↳ metadata describe info about the code in the language neutral manner. The metadata contains the following.

1) Assembly Information - Such as its identity that can be name, version, culture, public key the type of Assemblies. Other refrence assembly & security permission.

2) Information about the type such as name, visibility, baseclass, interfaces use & members (methods, fields, properties, events, nested values)

3) Attribute information which modifies type & members of class after providing the listed information ~~the~~ it creates objects, excess data & calls member functions.

GAC [Global Assembly Cache] :-

The Global Assembly Cache (GAC) is a folder in windows directly to store the .NET assembly that are specifically designated to be Shared by all applications executed on a System.

The Concept of GAC is result of the .Net architecture whose design addresses the issue of "DLL hell" that existed in COM (Component Object Model).

Unlike in COM there is no need for assembly in GAC to be registered before its use. Each assembly is identifying its name, version, architecture, culture & Public key.

The GAC is machine-wide code cache used for side by side execution of assemblies. The GAC implements the feature of shared library where different applications reuse the code placed in the files located in a common folder.

The GAC is also 1st in the search path while loading a .NET assembly. The only requirement for assembly to be deployed in GAC is that it should have strong name. The CLR refers an assembly based on specific Version mentioned by ~~calleg~~ calling application. The virtual file sys of GAC helps to fetch the version specific assembly.

GAC provides benefits of code reuse, file security side-by-side execution etc.

Q. 3 What is the use of generics in C# ? Explain with suitable example ?

Generic is a class which allows the user to define classes & methods with the placeholder. The basic idea behind using Generic is to allow type (Integer, String .....etc & user defined types) to be a parameter to methods, classes & interfaces. Generics are commonly used to create type-safe collections for both reference & value types. The .NET framework provides an extensive set of interfaces & classes in the system. collections Generic namespace for implementing generic collection

Generic Class :-

Generic in C# is its most powerful feature. It allows you to define the type-safe data structure. This out turn in a remarkable performance boost & high-grade code, because it help to reuse data processing algorithms without replicating type-specific code.

Generics are similar to templates in C++ but are different in implementation & capabilities. Generics introduces the concept of type parameter, because of which it is possible to create methods & classes that defer the framing of data type until the class or method is declared & is initialed by client code. Generic types because they reduce the need for boxing, unboxing, type casting the variables or objects parameter types are specified in generic class creation.

Syntax :-

BaseType obj = new BaseType()

* A Generic method with various parameters :-

Just as a method can take one argument, generics can take various parameters one argument can be passed as a familiar type & other as a generic type

Example :-

```
using System;
public class GFG
{
 public void Display <TypeOfValue> (string msg,
                       TypeOfValue)
  {
  console.writeline ("{0}:{1}", msg.Value);
  }}
  public class Example
  {
  public static int main()
  { GFG P = new GFG ();
  P.Display <int> ("Integer", 122);
  P.Display <char> ("Character", 'H');
  P.Display <double> ("Decimal", 255.67);
  return 0;
  }
```

## Features of Generics :-

1) It helps you in code reuse, performance & type safety
2) You can create your own generic classes, methods, interface & delegates.
3) You can create generic collection classes. The .NET framework class library contains many new generic collection classes in system collection. Generics namespace
4) You can get information on the types used in generic data type at run-time

## Advantages of Generics :-

1) Reusability -
You can use a single generic type definition for multiple purpose in the same code without any alteration
Example :- You can create a generic method to add two numbers. This method can be used to add two integer as well as two floats without any modification in the code.

2) Type Safety :-
Generics provide type safety without the overhead of multiple implementation

3) Generics eliminates boxing & unboxing

4) Generic Collection types generally perform better for storing & manipulating value types because there is no need to box the value types.

5) Generic delegates enable type-safe callbacks withbest the need to create multiple delegate classes.

• Disadvantages of Generics:-

1) Enumeration cannot have generic type parameters

2) Lightweight dynamic methods cannot be generic.

3) In C# a nested type that is enclosed in a generic type canot be instantiated unless type have been assigned to the type parameters of all enclosing types.

11

4 Q What is the Constructor ? Explain various type of Constructor in C# with an example?

Constructor :-

A special method of the class that will be automatically invoked when an instance of the class is created is called a Constructor. The main use of the constructors is to initialize private fields of the class while creating an instance of the class. when you have not created a constructor in the class. the compiler will automatically create a default constructor in the class. The default constructor initializes all numeric fields in the class to zero & all string & object fields to null.

Key points of the Constructor :-

1) A class can have any number of constructors.

2) A constructor doesn't have any return type, not even void.

3) A static constructor can not be a parameterized Constructor.

4) Within a class you can create only one static constructors.

Constructors can be divided into 5 types :-

1) Default Constructor
2) parameterized Constructor
3) copy Constructor
4) static Constructor
5) private Constructor.

* Default Constructor :-

A Constructor without any parameters is called a default constructor ; in other words this type of constructor does not take parameter. The drawbacks of a default constructor is that every instance of a class will be initialized to the same value & it is not possible to initializes each instance of the class to different values. The default Constructor initializes :

1) All Numeric fields in the class to zero.
2) All String & object fields to Null.

Example :- [ Copy Constructor ]

```
Using system ;
namespace CopyConctructor
{
 class employee
 {
 Private String name ;
 private int age;
 public employee (employee emp)
  {
 name = emp.name;
 age = em.age;
  }
  public employee (String name ,int age)
 { this.name = name;
   this.age = age;
 }
  public String Details
 { get
 { return " The age OF" + name + " is " + age.Tostring
}}}
class empdetail
{ static Void main()
{ employee emp1 = new employee ("Abc", 123);
  employee emp2 = new employee (emp1);
 Console.WriteLine (emp2.Details);
Console.Readline ();
}}
```

Ex :- Default Constructor

```
Using system :
namespace Default Constructor
{
 class addition
 { int a,b ;
public addition () // default constructor
 {
     a=100 ;
     b= 175;
}
 public static void main()
{ addition obj = new addition();
console.WriteLine (obj.a);
console. WriteLine. (obj.b);
console .Read ();
 }
```

* Parameterized Constructor

A constructor with atleast one parameter is called a parameterized constructor. The advantage of a parameterized constructor is that you can initialize each instance of the class to different values.

Eg :- parameterized Constructor :-

```
using system ;
namespace constructor
{
 class ParaConstructor
 {
 public int a,b ;
 public paraConstructor (int x, int y)
 {
  a=x ;
  b=y ;
 }}
class Mainclass
{
static void main()
{
paraConstructor v= new paraConstructor (100 ,176);
console.WriteLine ("Value Of a=" + V.a);
Console . WriteLine ("Value OF b = "+V.b);
Console . ReadLine ();
 }
 }
```

## * Copy Constructor

The Constructor which creates an object by copying variables from another object is called a copy constructor. The purpose of a copy constructor is to initialize a new instance to the values of an existing instance.

Eg:-

Syntax -

```
public employee (employee emp)
{
  name = emp.name;
  age = emp.age;
 }
```

The copy ~~constructor~~ is invoked by instantiating an object of type employee & object to be copied.

```
employee emp1 = new employee (em)
```

Now, emp1 is a copy of emp2

Q.5 Write Short note on

a) Delegates :-

A delegates is a pointer to a method with a same as a concept of function pointer, we had in is your C++ program
↳ we can use delegates for function pointer for invoking of method or function
↳ Advantages execution of method takes place within a single function stack even if called for multiple times.

Step 1 : Delegates Creation

Syntax : [<modifier>] delegates <void / type> Name of method (parameters)

↳ while declaring the delegates the I/O parameters of delegates should exactly the same as I/O parameter of method. we want to call with delegates

↳ eg :- public void add (int x, int y)
{ Console.WriteLine (x+y); }
public delegates ~~used~~ void AddDel (int x, int y)

Step 2 : Creating Object of delegates
As a delegates is also a type (Ref type) to custome it we need to create an object of delegates while creating the object the method name has to be passed as to a parameter to the constructor of this
eg :- AddDel ad = new AddDel (Add);

## Step 3 :- Calling the delegates

↳ Now, we can call the delegates, so that the method which was bound to the delegates which gets executed.
eq ad (100,200)

Delegates are of two types
1) Single cast delegates
2) Multi cast delegates

↳ If delegates is used for calling single method is called single cast delegate. where as delegates can be used for invoking multiple method its called as multicast delegates.

Note :- A delegates can be declare either under a class type or under namespace.

Programs :-

```
namespace OOP Project
{ public delegates void maths (int x, int y)
class multiDel
{
public void Add (int x, int y)
{
Console. writeline ("Add ="+ (x+y)); }
public void sub (int x, int y)
{ console. writeline ("sub=" + (x-y)); }
public void mul (int x, int y)
{
console.writeline ("Mul =" + (x*y));}
public void Div (int x, int y)
{ console.writeline ("DIV =" + (x/y)); }
static void main()
{
  MultiDel md = new multiDel();
  Maths m = new Maths (md.add);
  m+ = obj.sub;
  m+ =obj.mul;
  m+ = obj.div;
  m (100, 50);
  console.writeLine();
  m (678, 234);
  console.writeLine();
  m- = obj.sub;
  m (625, 25);
  c.ReadLine();
}
}
```

Q5

b) Indexer :-

Indexer allows classes to be used in more intuitive manner. C# introduces a new concept known as indexers which are used for treating an Object as an array. The indexers are usually known as Smart arrays in C#. They are not essential part of object oriented programming.

↳ An Indexer also called an indexer property is a class property that allows you to access a member variable of a class using the features of an array.

↳ Defining an indexer allows you to create classes that act like virtual arrays. Instances of that class can be accessed using the array access operator.

↳ Creating an Indexer :-

```
<modifier> <return type> this [arg list]
{
    get {// code block }
    get {// code block}
}
```

<modifier> - private, public, protected

<return type> - can be any valid C# types

this - Special keyword in C# to indicates the object of the current.

## Important points to remember on Indexers

↳ Indexers are always created with this keyword
↳ parameterized property are called Indexer
↳ Indexer are implemented through get & set accessors for the [ ] operator
↳ ref & out parameter modifiers are not permited in Indexer.
↳ The Indexer as an instance no. so can't be static but property can be static
↳ Indexer are used on group of elements.
↳ Indexer is identified by its signature where as a property is identified by its name.
↳ Indexer are accessed using indexers where as properties are accessed by names.
↳ Indexer can be overloaded.
↳ The formal parameter list of an indexer correspondi to that of a method & at least are parameters should be specified.

↳ Indexer are defined in pretty much same way as properties with get & set function, The main difference is that the name of the Indexer is the keyword this.

Q5 Sealed Class :-

c) ↳ The sealed keyword in C# language is used to create a sealed class.

↳ sealed classes are used to restrict the inheritance feature of Object oriented programming. once a class is defined as a sealed class, this class cannot be inherited.

↳ In C# the Sealed modifier is used to declare a class as sealed. In Visual Basic. NET, NotInheritable keyword serves the purpose of sealed. If a class is derived from a sealed class, compiler throws an error.

↳ If structs are sealed, you cannot derive a class from a struct.

Syntax :-

```
        sealed class Sealedclass
        {
        }
```

Eg program :-

```
        using System;
        class class 1
        {
        static void main (string[]args)
        {
        sealed class Sealed class = new sealedclass();
        int total = SealedClass.Add (4,5);
        Console.writeLine("Total ="+ total. ToString
                                                  ());
```

```
}
  }
  Sealed class SealedClass
   {
     public int Add (int x ,int y)
     {
      return x + y ;
    }
   }
```

↳ If you run this code, it will work just fine. But if you try to derive a class from a SealedClass, you will get an error

↳ The main purpose of a sealed class is to take away the inheritance feature from the class users so they cannot derive a class from it.

[illegible]
20/12/19.

Q1) Explain various asp.Net Coding models in details

Ans A web Consists of controls (Label. Button etc) & business logic. we can use ASP.NET coding technique to manage these controls & business logic

Two types of coding technique

1) Single. File page model:-
In single File page model. the HTML markup & functionality of web application are implemented in same file.
In single File coding approach. developer write code directly in .aspx page of the application
A major drawback of sing file code model is that writing code in single file, So it is very difficult to read & manage the web pages

eg:-

```
<html xmlns = "http://www.w3.org/1999/xhtm
<head runat = "server">
    <title> Single File page Model</title>
```

```
</head>
</body>
<form id = "form1" runat = "server">
  <div> <br/>
&nbsp; &nbsp; ;<asp : Label id = Label 1
   runat = "server" > </asp : Label >
 &nbsp; <asp : Button id = "Button 1"
  runat = "server" onclick = "Button 1_Click"
  Text = "show system date & time :" >
  </asp: Button > <br/>
  <asp : Button ID = "Button2" runat = "server"
   OnClick = "Button2_Click" Text = "Ne">
   </div >
   </form > </ body > </html >
```

2) Code - Behind page model :-

↳ In code Behind page model there are two seperate files Default.aspx & Default.aspx.cs

↳ There two files are linked together to run the web application. All the different version of ASP.NET Support this model by default.

↳ Both the files are combined together during compilation to implement the complete application

↳ In this model you need to maintain separate code filer for each web page, are file storer the code to implement the functionality of web page written is some programming language & other file storer the HTML markup of web application

Example :-

Default . aspx

```
<html xmlns = "http:// www. w3. org /1999/xhtml">
<head runat = " server" >
    <title > code Behind page model</title>
  </head>
  < body >
  < form id = "form" runat = "server">
  <div
< asp: Label id = "Label 1" runat = "server>
</asp: Label1 >
& nbsp: <asp: Button id = "Button1"
runat = " server" Onclick = " Button 1-click"
Text = "show sytem date &amp; time">
</asp: Button>
< / div> </ form > </ body > </html>
```

Default2.aspx.cs

```
using System;
using system . collection . Generic;
using system . Linq;
using system . web;
using system. web . UI
using system . web. UI . web controls;
 public partial class-Default ;
 system . web.UIpage
 {
```

```
protected void Button1_click (Object
                sender EventArgs e)
{
   Label1.Text = DateTime.Now.ToString();
 }
  }
```

Q2. Explain In details Various Validation Control with example

Ans. Validation Controls are very useful while submitting data into database table

These Controls prevent users from submitting wrong type of data

Six Validation Controls available in asp.Net.

1) RequiredField Validator
2) RangeValidator
3) CompareValidator
4) Regular ExpressionValidator
5) Custom Validator
6) Validation Summary

By default, Validation Control perform Validation on both client (the browser) & server.

These Validation Controls uses JavaScript to perform Validation on client side. Clientside Validation is very fast because user gets immediate response whether user enters an invalid Value into a form field

If browser does not support JavaScript, then serverside Validation still perform.

You can disable client-side Validation by assigning value False to EnableClientScript property of Validation Controls.

Validation Controls inherit from Base Validator class therefore they inherit its properties & methods. Base Validator Class is an abstract class.

Display property :-
All Validation control support Display property that ensures how Validation error message will be displayed.

Following 3 Possible Values
↳ Static
↳ Dynamic
↳ None
By default, Display property has Value Static

1) RequiredFieldValidator
↳ used to make an input control as a mandatory field or Should not be empty.
↳ other Validation controls do not Validate for empty field, they Validate, when user enter data

For eg:- you can specify user must fill password in Password TextBox before submit login form.
You have to set two imp properties.
1) Control To Validate :- ID of control for which RequiredField Validator control will be associated.
2) Error Message :- used to show error message, if user forget to enter data, then custom error message will appear.

[illegible]

Example :-

```
< asp : TextBox  ID = "txtName"  runat = "server" width
                                                 = 200 px" >
< /asp  : TextBox >
<asp : RequiredFieldValidator  ID = "RequiredFieldValidator
runat = "Server"  ControlToValidate  = "txt Name" 1"
ErrorMessage = "Enter Name"  Fore Color = "Red" >
 < /asp : RequiredFieldValidator >
```

2) Rangevalidator

If confirms that user input data is within a specific range of values. The i/p value should come between a certain minimum & maximum value otherwise it will give error

properties

1) MinimumValue :- The minimum value of validation range

2) MaximumValue :- The maximum value of the validation range

3) Type :- Possible values are String, Integer Double Data & currency.
By default the type property has value String

3) Compare Validator :-
Used to perform 3 different types of validation
1) To perform a datatype check
2) To compare value entered into a form field against a given fixed value.
3) To compare value of one form field against another field.

Properties :-
1) ValueToCompare :- The fixed value against which to compare.
2) ControlToCompare :- The ID of control against which to compare
3) Type :- type of value for comparison. Given values are String, Integer, Double, Date & Currency
4) Operator :- Given values are DataTypeCheck, Equal, GreaterThan, GreaterThanEqual, Less Than, Less ThanEqual & NotEqual

4) RegularExpressionValidator :-
The RegularExpressionValidator control performs it's validation according to regular expression.
You can check predefined pattern, such as phone no... postal code, e-mail address, dates & so on

Eg :-

```
<asp:TextBox ID = "txtEmailID" runat = "Server"
    Width = "200 px" > </asp:TextBox >
<asp:RegularExpressionValidator ID = "Regular-
-Expression1" runat = "server" ForeColor = "Red"
    ControlToValidate = "txtEmailID'
    ErrorMessage = "Enter Correct EmailID"
    ValidationExpression = "\w + ([-+.']\w+)*
@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp : RegularExpressionValidator >
```

5) ValidationSummary

Control Summarizes the error message from all validation controls on webpage in a Single location

↳ useful when working with large forms

↳ If user enters wrong value in field located at the end of page, then the user might never see the error message. ValidationSummary Control can displays a lists of errors at the top of the form.

↳ It does not support ErrorMessage & ControlToValidate property

properties :-

1) DisplayMode :- BulletList, List & SingleParagraph

2) HeaderText - Display header text above the Validation Summary.

3) ShowMessageBox :- used to display a pop up alert box

4) ShowSummary :- It hides Validation Summary in the page

By default ShowMessageBox property value is set False & ShowSummary property value is set True.

Eg :-

```
<asp:ValidationSummary
ID="ValidationSummary1" runat="server">
</asp:ValidationSummary>
```

6) CustomValidation :-

Custom Validator control enables you to create your own Validation control on the page.

performs Validation based upon your code you write. "You can write Validation code that will be executed on client-side using Javascript or with Server-side validation

The CustomValidator control can work client side & server side both.

Eg :-

```
using system
using system.web.UI.webControls;
public partial Class CustomValidation:
system.web.UI.page
{
Protected Void Page-Load (object sender
                          EventArgs e)
{
}
```

```
protected void
CustomValidator1 ServerValidate (object source,
ServerValidateEventArgs args)
  {
     String inputData ; args.Value ;
     args.IsValid = False;
     if (inputData .Length<6 ||inputData.
                                Length>12)
     return;
   bool upperCase = False ;
    foreach (char ch in inputData)
     {
      if (ch> = 'A' && Ch < = 'z')
       {
        upperCase = true;
         break;
       }
     }
      if (;upperCase ) return
      bool LowerCase = false;
      foreach (char Ch in inPutData)
       {
        if (ch> = "a" && Ch = 'z'
         {
         LowerCase = true ; break;
        }
       }
if (!lowerCase) return;
  bool number = false;
  foreach (Char ch in inputData)
  {

    if (ch>='0' && ch<='9')
    {
    number = true;
    break;
    } }
if (!number) return;
args.IsValid = true;
  }
   }
```

Q3) What is ajax ? Explain timer Control & update Progess Control with suitable example.

Ans :- ↳ Ajax Stands for Asynchronous Javascript & XML
↳ Ajax is new technology for creating better, faster, & more interactive web application with help of XML, HTML, CSS & Javascript
↳ Ajax uses XHTML for content, CSS for presentation along with Document Object Model & Javascript for dynamic content display.
↳ Conventional web application transmit info to & from the server using Synchronous request
↳ with Ajax, when you hit submit Javascript will make a request to Server, interpret the result & update the current screen.
↳ XML is commonly used as format for receiving server data. although any format including plain text can be used
↳ AJAX is web browser technology independent of web server Software
↳ Data - driven as opposed to page - driven

• AJAX Based on Open Standards
↳ Browser - based Presentation using HTML & cascading style sheets (CSS)
↳ Data is stored in XML format & fetch from the Server
↳ Behind the scenes data fetches using XMLHtt

Request Objects in the browser
↳ Javascript to make everything happen

- ~~As~~ Ajax use with other technologies to create interactive webpages.
↳ Javascript - Glue for whole AJAX operation
↳ DOM - Represents Structure of XML & HTM doc
↳ CSS
↳ XML HttpRequest.

- Web application that make use of Ajax
↳ Google Maps
↳ Google suggest
↳ Gmail
Yahoo Maps

1) Timer Control :-
↳ Allows you to do postbacks at certain interval
↳ It is used together with Updatepanel which is the most common approach, it allows for timed partial updates of your page but can be used for posting back the entire page as well

Eg :-

```
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="Serve
"/>
<asp:Timer runat="Server" id="UpdateTimer
interval="5000" ontick="UpdateTimer_Tick"/>
```

```
<asp:UpdatePanel runat="server" id="TimedPanel1"
 Updatemode = "Conditional">
  <Triggers>
  <asp: AsyncPostBackTrigger . controlid ="UpdateTime"
 eventname = "Tick"/>
 <|Triggers>
 <ContentTemplate>
  <asp:Label runat ="server" id="DatestampLabel"/>
  <|ContentTemplate>
  <|asp: UpdatePanel|>
  < |form>
```

2) update progress control -
It will use your own template to show that an asynchronous method is working

```
<form id = "form1" runat="server">
 < asp: scriptManager ID= "scriptManager1"
                                 runat="server"/>
<asp :Update Progress runat="server" id =
                "pageUpdateProgress">
 < ProgressTemplate >
  Loading . . . .
 <|ProgressTemplate>
 <| asp : update progress>
 <asp: Update panel runat = "server" id=
                                  "panel">
 <content Template>
   <asp: Button runat="server" id ="
                         Update Button"
```

```
onclick = "UpdateButton_click" text="Update"/>
 <|content Template>
<|asp: updatepanel>
 <|form >
```

You can even have multiple update progress controls on page & using Associated update panel ID property you can make sure that the update progress is only shown when a certain updatepanel is updated.

↳ Dynamic Layout property tells whether or not the page should reserve space for your progress - control

Q.4 Explain State Management in details.

Hyper Text Transfer protocol (HTTP) is Stateless protocol. when the client disconnected from the Server, the ASP.NET engine discards the page objects.

This way each web application can scale up to serve numerics request simultaneously without running out of server memory

ASP.NET manges four types of states:
1) View State
2) Control state
3) Session State
4) Application state

1) View State

The view state is state of the page & all its Controls. It is automatically maintained across posts by ASP.NET framework

when page sent back to client the changes in properties of page & ~~is~~ stored in the value of hidden input field named. VIEWSTATE. when page is again posted back. the ~~View~~ VIEWSTATE field is sent to server with HTTP request

Properties :-
1) Item (name) - Value of view state item with specific name.

2) Count - no. of items in View state collection
3) Keys - collection of keys for all items in collection
4) Values - collection of values for all items

Method :-
1) Add (name, Value) - Add an item & existing item updated
2) Clear - Remove all items
3) Equals (object) - determines whether specific object is equal to current object
4) Finalize - Allows it to free resources

5) GetType - Gets type of current instance
6) Remove (name) - Remove specified item
7) SetItemDirty - Sets Dirty property for specified StateItem object in StateBag object
8) ToString - Return a string representing the state bag object.

2) Session State :-
↳ When a user connects to an ASP.NET website, a new session object is created
↳ When session state is turned on, a new session state object is created for each new request
↳ Used for storing application data such as inventory, supplier list, customer record. It can also keep information about user & his preferences & keep track of pending operations
↳ Sessions are identified & tracked with 120-bit session ID, which is passed from

client to server & back as cookie. The sessionID is globally unique & random
The state of session object is created from HttpSessionState.class.

Properties :-
1) SessionID - Unique session Identifier
2) Item (name) - Value of session state item with specific name
3) Count - no. of items in the session state collection

4) Timeout - Gets & sets the amount of time in minutes allowed betn requests before the session-state provide terminated the session

Methods :-
1) Add (name, value)
2) Clear
3) Remove(name) - removes specific items
4) Remove All
5) Remove At - Deletes an item at a specified index from session-state collection

3) Application State.
i) ASP.NET application is collection of all web pages, code & other files within a single virtual directory on web server. When information is stored in application state, it is available to all the users.
i) To provide for use of application state

ASP.NET creates an application state object each application from HTTPApplicationState class & store this object in server memory. This object is represented by class file global.aspx
↳ used to store hit counters & other statistical data, global application data & keep the track of users visiting the site.

properties :-

1) Item(name) - Value of applicatn state item with specified name.
2) Count :- no. of items in application state collection.

Methods

1) Add (name) - Value of applicatn state items with specified name.

2) clear
3) Remove (name).
4) RemoveAll
5) RemoveAt
6) Lock () - Locks the application state collection so only current user can access it

7) Unlock () - Unlocks the application state Collection so all the users can access it.

Q.5) Explain ASP.NET page lifecycle & compilation model.

ASP.NET Page Life cycle :-
When a page is requested, it is loaded into the server memory, processed & send to browser. Then it is uploaded from the memory. At each steps, methods & events are available which could be overriden according to the end of the application.
The page class creates a hierarchical tree of all controls on the page. All components on the page, except directives, are part of this control tree.

The page life cycle Phases are :-
↳ Initialization
↳ Instantiation of the controls on the page
↳ Restoration & maintenance of the state
↳ Execution of event handler codes
↳ Page rendering.

Different stages of an ASP.NET Pages

1) Page Request :- when ASP.NET gets a page request, it decides whether to parse & compile the page or there would be cached version of page
2) Start :- In this stage request & response properties are set Start stage also

determines request type.
3) Initialization:- In initialization stage each control has set the unique property ID. Themes need to be initialized
4) Page load:- Control Properties are set using the view state & control state values.
5) Postback Event Handling:- If the request is Postback the related event handler is invoked
6) Page rendering:- View state for the page & all controls are saved. The page calls the Reder method for each control & the o/p of rendering is written in outputstream class of the Response property of page
7) Unload:- The rendered page is sent to the client & page properties, such as response & request, are unloaded & all cleanup done.

ASP.NET Page life cycle Events:-

ASP.NET pages support the automatic wire-up event, which means it looks for methods with particular names & automatically runs those methods when particular events are raised.

1) preInit:- PreInit is 1st event in ASP.NET page lifecycle. This event is taking the start stage is completed & before initialization stage. This event is used to set the Theme property dynamically

2) Init :- This event raised once all controls have been initialized. The use of this event is to read or initialne control properties.
3) Init Complete :- Event tracking of view state changes is turned on, until view state tracking is turned on any values added to view state are lost accross postback.
4) preload :- used for performing page processing on the page before control is loaded.
5) Load :- using this method we can set the control properties.
6) Control Events :- use to handle specific control events such as a Button controls click event
7) Load Complete :- handled by overloading the unload Complete method, used for tasks that require that all controls on the page be loaded
8) PreRender Complete :- This event is fired for child controls
9) Savestate Complete :- In this event, any changes to the page on controls effect rendering but changes will not be retrieved ~~on~~ next postback.
10) Unload - occurs at last in ASP.NET page life cycle unload event used to do cleanup task like the closing of open files, closing the database connections or other requested specific tasks.

Q6) Explain various datasource & data bound control

Ans:- Data source :-

↳ Data Source control interact with data-bound controls & hides the complex data binding processes.

↳ provide data to databound controls & support execution of operations like insertion, deletion, sorting & updates.

Each data source control wraps a particular data provider relational database XML documents or custom classes & helps in

↳ managing connection
↳ selecting data
↳ Managing presentation aspects like paging, caching etc
↳ Manipulating data.

Based on type of data, these controls could be divided into two categories

↳ Hirarchical datasource controls
↳ Table-based data source controls

Data Source control used for hierarchical data :-

↳ XML Datasource :- allows binding to XML file & storing with or without Schema Info

↳ Site Map Data Source :- allows binding to a provider that supplies Site Map information

Templates

1) EditItemTemplate :- Specifies layout of item when edit mode is working.

2) FooterTemplate :- Contains all text & controls to be rendered at the end of the List

3) SeperatorTemplate :- Contains all elements to render bet$^{n}$ each item.

3) GridView

↳ Display data in tabular form rendered as HTML table

↳ Easy to configure paging, sorting & editing

↳ Contains collection of GridViewRow & collection of DataControlField.

↳ GridView supports Initialize~~Row~~ Row method for creating new GridViewRow & initialising its cells

property :-

1) AllowPaging :- It is Boolean value Indicating control support paging

2) AllowSorting :- " "

3) SortExpression - accepts current expression determining the order of row

4) DataSourceID - indicates dataSource control

5) SortDirection - It gets sorting direction of the column for the control.

4) ListView :- used to bind the data items returned to datasource & display them. The control displays data in a format defined by using templates & styles.

Templates :-

1) ItemTemplate :- identifies databound content to display for single item.

2) GroupTemplate :- identifies the content of group layout

3) EditItemTemplate :- identifies the content to render when item is lost

4) InsertItemTemplate :- identifies the content to render when an item is being inserted

5) LayoutTemplate :- It identifies the root template that defines the main layout.

5) FormView :-

↳ Display single record from database but not within pre-defined HTML table.

↳ Developers create template that defines how data is displayed

↳ Can define different templates for viewing, editing & updating record.

Properties :-

1) EditItem Template :- used when the record is being edited by the user.

2) InsertItem Template :- used when a record

is being Created by the user.
3) Item Template:- used to render the record to display only in an application

The ObjectDataSource Control
↳ enables user-defined class to associate o/p of their methods to data bound controls.

important aspects of binding business objects
↳ Bindable class should have default constructor it should be stateless & have methods that mapped to select, update, insert & delete semantics.
↳ The Object must update one item at a time batch operations are not supported.

Access DataSource Control:-
↳ It represents connection to an Access db.
↳ The Access DataSource Control opens db in read only mode, also used for performing insert update or delete operations. This is done using ADO.NET commands & parameter collection.

## Data-Bound Controls :-

↳ used to display data to end-user within web applications & using databound controls allows to manipulate data within web application very easily

↳ Databound controls are bound to DataSource property

1) Repeater :-

It is data bound control created by using templates to display items. The control does not support editing paging or sorting of data rendered through the control.

Templates :-

1) Header Template :-

2) Footer Template :-

3) Seperator Template :-

2) DataList :-

↳ works like repeater control

↳ Repeats data for each row in dataset based on provided template.

Data Source controls used for tabular data

1) SqlDataSource :- represents connection to an ADO.NET data provider that returns SQL data
2) ObjectDataSource :- allows binding to a custom .Net business object that returns data
3) LinqdataSource :- allows binding to results of a Linq-to-SQL query
4) AccessDataSource :- Represents connection to microsoft access database.

Data Source Views :-
Data Source views are objects of DataSourceView class. which is serves as the base class for all data source view classes which define capability of data source controls

Properties :-

1) CanDelete :- deletion is allowed on underlying data source
2) CanInsert :- whether insertion is allowed
3) Canpage :- whether paging is allowed
4) CanRetrieveTotalRowCount :- whether total row total info is available
5) CanSort - Whether data could be sorted
6) canUpdate - whether updates are allowed
7) Name - Name of the view

Methods :-

1) Can Execute

2) ExecuteCommon
3) ExecuteSelect
4) Delete
5) Select
6) OnDataSourceViewChanged

SqlDataSource Control :-
Represents connection to relational db such as SQL Server or oracle db or open Database connectivity (ODBC) connection to data is made through two important properties connection String & providerName

Properties
1) Delete Command
2) Deleteparameters
3) Delete CommanType
} Get SQL Statement, Parameter & type for deleting rows
4) Filter Expression
5) Filter parameters
} Gets or sets data filtering
6) Insert command
7) Insert parameter
8) InsertCommandType
} Gets or sets SQL SE parameters & type for inserting rows
9) SortparameterName - Gets or sets name of the parameter that command's stored procedure will use to sort data