1. Describe state management in ASP.NET

State management is a technique to manage a state of an object on different request.

Maintaining state is important in any web application. There are two types of state management system in ASP.NET.

- Client-side state management
- Server-side state management

Client side state management

View state: ASP.NET uses view state to track values in controls between page requests. It works within the page only. You cannot use view state value in next page. The value is stored in view state as a hidden field. In encoded form. Base 64 encoded form

Control state: You can persist information about a control that is not part of the view state. If view state is disabled for a control or the page, the control state will still work. Heavy usage of Control State can impact the performance of application because it involves serialization and deserialization for its functioning.

Hidden fields: It stores data without displaying that control and data to the user's browser. This data is presented back to the server and is available when the form is processed. Hidden fields data is available within the page only (page-scoped data).

Cookies: Cookies are small piece of information that server creates on the browser. Cookies store a value in the user's browser that the browser sends with every page request to the web server.

```
//Storing value in cookie
HttpCookie cookie = new HttpCookie("NickName");
cookie.Value = "David";
Request.Cookies.Add(cookie);

// Creating a cookie
myCookie.Values.Add("muffin", "chocolate");
myCookie.Values.Add("babka", "cinnamon");

// Adding Cookie to Collection
Response.Cookies.Add(myCookie);
```

Query strings: In query strings, values are stored at the end of the URL. These values are visible to the user through his or her browser's address bar. Query strings are not secure. You should not send secret information through the query string.

Server side state management

Application State:
This object stores the data that is accessible to all pages in a given Web application. The Application object contains global variables for your ASP.NET application.

Use of application state

When you run this you will get new id and then again when you open new browser and again you will see the current id even though even if you deletes the session id from the browser means it is we server is maintaining the state for the other user as well.

Cache Object:
Caching is the process of storing data that is used frequently by the user. Caching increases your application's performance, scalability, and availability. You can catch the data on the server or client.

Session State:
Session object stores user-specific data between individual requests. This object is same as application object but it stores the data about particular user.

2. Cookies

A cookie is a small amount of data that server creates on the client. There are two types of cookies

Session cookies - A session cookie exists only in memory. If a user closes the web browser, the session cookie delete permanently.

Persistent cookies - A persistent cookie, on the other hand, can available for months or even years. When you create a persistent cookie, the cookie is stored permanently by the user's browser on the user's computer.

If you want to create a persistent cookie, then you need to specify an expiration date for the cookie.

```
Response.Cookies["message"].Expires = DateTime.Now.AddYears(1);
```

Disadvantage of cookies.

- Cookie can store only string value.
- Cookies are browser dependent.
- Cookies are not secure.
- Cookies can store small amount of data.

3. What is Session object?

HTTP is a stateless protocol; it can't hold the user information on web page. If user inserts some information, and move to the next page, that data will be lost and user would not able to retrieve the information. For accessing that information we have to store information. Session provides that facility to store information on server memory. It can support any type of object to store. For every user Session data store separately means session is user specific.

4. Authentication And authorization

Authentication is the process of verifying the identity of a user using some credentials like username and password while authorization determines the parts of the system to which a particular identity has access.
- Authentication is required before authorization.
Example: If an employee authenticates himself with his credentials on a system, authorization will determine if he has the control over just publishing the content or also editing it.

Types of Authentication **-**

1. Windows Authentication: This authentication method uses built-in windows security features to authenticate user.
2. Forms Authentication: authenticate against a customized list of users or users in a database.
3. Passport Authentication: validates against Microsoft Passport service which is basically a centralized authentication service.

5. ASP.Net

It is a framework developed by Microsoft on which we can develop new generation web sites using web forms (aspx), MVC, HTML, JavaScript, CSS etc. Its successor of Microsoft Active Server Pages (ASP). Currently there is ASP.NET 4.0, which is used to develop web sites. There are various page extensions provided by Microsoft that are being used for web site development. Eg: aspx, asmx, ascx, ashx, cs, vb, html, XML etc.

6. Validation in Asp.Net

1. Required field Validator
2. Range Validator

3. Compare Validator
4. Custom Validator
5. Regular expression Validator
6. Summary Validator

7. Can we have a web application running without web.Config file?

Yes.

8. Is it possible to create web application with both webforms and mvc?

Yes. We have to include below mvc assembly references in the web forms application to create hybrid application.

9. What is Cross Page Posting

When we click submit button on a web page, the page post the data to the same page. The technique in which we post the data to different pages is called Cross Page posting. This can be achieved by setting POSTBACKURL property of the button that causes the post back. Find control method of Previous Page can be used to get the posted values on the page to which the page has been posted.

10. Boxing and Unboxing

Boxing is assigning a value type to reference type variable.

```
int myvar = 10;
object myObj = myvar;
```

Unboxing is reverse of boxing ie. Assigning reference type variable to value type variable.

```
int myvar2 = (int)myObj;
```

11. Strong typing and Weak typing

In strong typing, the data types of variable are checked at compile time. On the other hand, in case of weak typing the variable data types are checked at runtime. In case of strong typing, there is no chance of compilation error. Scripts use weak typing and hence issues arises at runtime.

12. Difference between a Hash Table and a Dictionary

The main differences are listed below.

Dictionary

Returns an error if the key does not exist

No boxing and unboxing

Faster than a Hash table

Hashtable:

Returns NULL even if the key does not exist

Requires boxing and unboxing

Slower than a Dictionary

13. differences between Array list and Hash table

1) Hash table store data as name, value pair. While in array only value is store.

2) To access value from hash table, you need to pass name. While in array, to access value, you need to pass index number.

3) you can store different type of data in hash table, say int, string etc. while in array you can store only similar type of data.

14. system.stringbuilder and system.string

The main difference is system.string is immutable and system.stringbuilder is a mutable. Append keyword is used in string builder but not in system.string.

Immutable means once we created we cannot modified. Suppose if we want give new value to old value simply it will discarded the old value and it will create new instance in memory to hold the new value.

Example – String Us = “A”;

US += “B”;

US + “C”;

You are basically changing the same string 3 times , but how it works it keeps last value in memory like this - A, AB, ABC… that is string is immutable means they can not be changed once created but in stringbuilder it is mutable. , No differernce in output but string buiider is faster.

String builder sb = new strin gul(“A”)

Sb.append(“B”)

Sb.append(“C);

15. Types of memories are there in .net

Two types of memories are there in .net stack memory and heap memory

16. Value type and reference type

Value type contain variable and reference type are not containing value directly in its memory.

Memory is allocated in managed heap in reference type and in value type memory allocated in stack. Reference type ex-class value type-struct, enumeration

17. Difference between GET and POST Methods

GET Method ():

1) Data is appended to the URL.

2) Data is not secret.
3) It is a single call system
4) Maximum data that can be sent is 256.
5) Data transmission is faster
6) this is the default method for many browsers

POST Method ():

1) Data is not appended to the URL.
2) Data is Secret
3) it is a two call system.
4) There is no Limit on the amount of data. That is characters any amount of data can be sent.
5) Data transmission is comparatively slow.
6) No default and should be explicitly specified.

18. String objects are immutable?

String objects are immutable as its state cannot be modified once created. Every time when we perform any operation like add, copy, replace, and case conversion or when we pass a string object as a parameter to a method a new object will be created.

Example:

```
String str = "ABC";


str.Replace("A","X");
```

Here Replace() method will not change data that "str" contains, instead a new string object is created to hold data "XBC" and the reference to this object is returned by Replace() method.

So in order to point *str* to this object we need to write below line.

```
str = str.Replace("A","X");
```

Now the new object is assigned to the variable str. earlier object that was assigned to str will take care by garbage collector as this one is no longer in used.

19. Dll hell problem in .NET

Dll hell, is kind of conflict that occurred previously, due to the lack of version supportability of dll for (within) an application

.NET Framework provides operating system with a global assembly cache. This cache is a repository for all the .net components that are shared globally on a particular machine. When a .net component installed onto the machine, the global assembly cache looks at its version, its public key and its language information and creates a strong name for the component. The component is then registered in the repository and indexed by its strong name, so there is no confusion between the different versions of same component, or DLL

20. constants, read-only and, static

Constants: The value can't be changed
Read-only: The value will be initialized only once from the constructor of the class.
Static: Value can be initialized once

21. Usage of Global.asax

Global.asax is basically ASP.NET Application file. It's a place to write code for Application-level events such as Application start, Application end, Session start and end, Application error etc. raised by ASP.NET or by HTTP Modules.

There is a good list of events that are fired but following are few of the important events in Global.asax:

- Application_Init occurs in case of application initialization for the very first time.
- Application_Start fires on application start.
- Session_Start fires when a new user session starts
- Application_Error occurs in case of an unhandled exception generated from application.
- Session_End fires when user session ends.
- Application_End fires when application ends or time out.

22. HttpHandlers and HttpModules in ASP.NET

- **HttpHandler**: ASP.NET Engine uses HttpHandlers to handle specific requests on the basis of it's extensions. ASP.NET Page Handler handles all requests coming for (.aspx) pages. We can define our own custom HttpHandler to handle a specific request with a specific extension, say .jpeg, .gif, or .ahmad. But there will always be only one handler for a specific request.

  HttpModule: ASP.NET Engine uses HttpModules to inject some specific functionality along with ASP.NET default functionality for all incoming requests regardless of its extensions. There are a number of built-in modules already available in ASP.NET HTTP Pipeline. But we can write our own

custom HTTP module to perform some additional functionality (for example, URL rewriting or implementing some security mechanism) for all incoming requests.

23. Session.Clear() and Session.Abandon()

As we understand that Session is a Collection and it stores data as Key/Value pair. So,
Session.Clear() clears all the session values but doesn't destroy the Session. however,
Session.Abandon() destroys the session object.
In other words, Session.Clear() is like deleting all files inside a folder (say "Root") but Session.Abandon() means deleting the "Root" folder.

24. Label Control and Literal Control

A Label control in ASP.NET renders text inside tags while a Literal Control renders just the text without any tags.
With Label controls we can easily apply styles using it's CssClass property, however, if we don't want to apply style/formatting, it's better to go for a Literal control

25. What is CLR

The CLR stands for Common Language Runtime and it is an Execution Environment. The main function of Common Language Runtime (CLR) is to convert the Managed Code into native code and then execute the program. The Managed Code compiled only when it is needed, that is it converts the appropriate instructions when each function is called. The Common Language Runtime (CLR)'s just in time (JIT) compilation converts Intermediate Language (MSIL) to native code on demand at application run time.

When a .NET application is executed at that time the control will go to Operating System, then Operating System create a process to load CLR.

Feature of CLR

IL to Native translation

CTS- CTS ensure that data types defined in two different languages get compiled to a common data type. This is useful because there may be situations when we want code in one language to be called in other language.

We can see practical demonstration of CTS by creating same application in C# and VB.Net and then compare the IL code of both application. Here the datatype of both IL code is same.

CLS- CLS is a subset of CTS. CLS is a set of rules or guidelines. When any programming language adheres to these set of rules it can be consumed by any .Net language.

JIT-JIT compiles the IL code to Machine code just before execution and then saves this transaction in memory. The JIT compiler is an important element of CLR, which loads MSIL on target machines for execution. The MSIL is stored in .NET assemblies after the developer has compiled the code written in any .NET-compliant programming language, such as Visual Basic and C#.

The responsibilities of CLR are listed as follows:

- Automatic memory management
- Garbage Collection
- Code Access Security
- Code verification
- JIT compilation of .NET code

26. Managed Vs Unmanaged

.NET supports two kind of coding

Managed Code – Code which is executed with the help of CLR instead of operating system, Now since code is executed with clr so it will provide services like garbage collector, type checking, exception handling. Code is compiled by the language complier in the Intermediate language.
Unmanaged Code – code which is executed by operating system directly, so it does not provide such services, it should be taken care by the programmer itself. Like in C , C++, like in those language for de-allocating the memory programmer needs to write the code explicitly. Code is compiled into native code.

Managed Code - The code, which is developed in .NET framework, is known as managed code. This code is directly executed by CLR with help of managed code execution. Any language that is written in .NET Framework is managed code.

A piece of managed code is executed as follows:

- Choosing a language compiler
- Compiling the code to MSIL
- Compiling MSIL to native code
- Executing the code.

Unmanaged Code -The code, which is developed outside .NET, Framework is known as unmanaged code. Applications that do not run under the control of the CLR are said to be unmanaged, and

certain languages such as C++ can be used to write such applications, which, for example, access low - level functions of the operating system.

Unmanaged code is executed with help of wrapper classes. Wrapper classes are of two types: CCW (COM Callable Wrapper) and RCW (Runtime Callable Wrapper).

27. Assembly

Assembly can be in form of exe or dll, whenever we compile the class it going to generate dll, or whenever we going to compile application it going to generate exe. Dll dynamic link library, for reuse purpose. We cannot execute the dll files, we can only compile the files.

When we build the solution the code is complied in assembly. If you complie a project an assembly is generated . 2 Types of assembly .exe or dll , console application generates .exe, windows app generates exe but web application generates .dll,

Assembly contains the intermediate language of you project source code, so all your project classes , methods etc will be complied in intermidate lang. is generated and packaged in the assembly

Private Assembly – which can be used only single application, location of private assembly bin folder.

Public Assembly – which can be used for multiple application, location of public assembly is - c drive windows assembly

28. Difference between int and int32

There is no difference between int and int32. System.Int32 is a .NET Class and int is an alias name forSystem.Int32.

29. Differences between the Dispose() and Finalize().

CLR uses the Dispose and Finalize methods to perform garbage collection of run-time objects of .NET applications.

The Finalize method is called automatically by the runtime. CLR has a garbage collector (GC), which periodically checks for objects in heap that are no longer referenced by any object or program. It calls the Finalize method to free the memory used by such objects. The Dispose method is called by the and programmer. Dispose is another method to release the memory used by an object. The Dispose method

needs to be explicitly called in code to dereference an object from the heap. The Dispose method can be invoked only by the classes that implement the IDisposableinterface.

- **Dispose** - This method uses interface – "IDisposable" interface and it will free up both managed and unmanaged codes like – database connection, files etc.
- **Finalize** - This method is called internally unlike Dispose method which is called explicitly. It is called by garbage collector and can't be called from the code. This method is used for garbage collection. So before destroying an object this method is called as part of clean up activity.
- **Finally -** This method is used for executing the code irrespective of exception occurred or not.

Finalize

- Finalize used to free unmanaged resources those are not in use like files, database connections in application domain and more, held by an object before that object is destroyed.
- In the Internal process it is called by Garbage Collector and can't called manual by user code or any service.
- Finalize belongs to System.Object class.
- Implement it when you have unmanaged resources in your code, and make sure that these resources are freed when the Garbage collection happens.

Dispose

- Dispose is also used to free unmanaged resources those are not in use like files, database connections in Application domain at any time.
- Dispose explicitly it is called by manual user code.
- If we need to dispose method so must implement that class by IDisposable interface.
- It belongs to IDisposable interface.
- Implement this when you are writing a custom class that will be used by other users.

30. What are tuples

Tuple is a fixed-size collection that can have elements of either same or different data types. Similar to arrays, a user must have to specify the size of a tuple at the time of declaration. Tuples are allowed to

hold up from 1 to 8 elements and if there are more than 8 elements, then the 8th element can be defined as another tuple. Tuples can be specified as parameter or return type of a method.

```
Tuple<String, int> t = new Tuple<String, int> ("Hellow", 2);
```

31. Garbage collector

Garbage Collector - AMM – Automatic memory Management – At this time, programmer needs not to worry about the clearing the memory, and Garbage collector allocates the memory to the heap very effectively. So in the heap we will be having live objects and dead objects, now if new object is coming and you have the space for this, than its fine, that object will be given the space, but what happens when then GC will clear the dead objects and then provide the memory for that newly object. Now question is how GC identifies that object is dead or not. The Process is first it checks for the Application root, if the application root is the any of the object then it doesn't remove that object , and even any reference provided for that object, it will not remove that object. It works like in 3 generation –

Generation 2 – Very Long lived objects resides here

Generation 1 – Long lived objects resides here

Generation 0 – Newly created objects resides here

When no space in generation 0 then some object will be shifted to upper generation and so on

This memory is separated in to three parts.

- Generation Zero.
- Generation One and
- Generation Two

Ideally Generation zero will be in smaller size, Generation one will be in medium size and Generation two will be larger.
When Generation Zero is full and it does not have enough space to occupy other objects but still the program wants to allocate some more memory for some other objects, then the garbage collector will be given the REALTIME priority and will come in to picture.

Now the garbage collector will come and check all the objects in the Generation Zero level. If an object's scope and lifetime goes off then the system will automatically mark it for garbage collection.

You may have a doubt, all the generations are filled with the referred objects and still system or our program wants to allocate some objects, then what will happen? If so, then the MemoryOutofRangeException will be thrown.

Generation 0

This is the youngest generation and contains the newly created objects. Generation 0 has short-lived objects and collected frequently. The objects that survive the Generation 0 are promoted to Generation 1.

Example : A temporary object.

Generation 1

This generation contains the longer lived objects that are promoted from generation 0. The objects that survive the Generation 1 are promoted to Generation 2. Basically this generation serves as a buffer between short-lived objects and longest-lived objects.

Generation 2

This generation contains the longest lived objects that are promoted from generation 1 and collected infrequently.

Example : An object at application level that contains static data which is available for the duration of the process.

```
Console.WriteLine("GC Maximum Generations:" + GC.MaxGeneration);

Console.WriteLine("Total Memory:" + GC.GetTotalMemory(false));
```

We can force garbage collector to run using System.GC.Collect().

32. Asp.Net Versions Differences

2. .Net Framework 2.0 Features

- ADO.NET 2.0
- SQL Server data provider (SqlClient)
- XML
- .NET Remoting
- ASP.NET 2.0

3. .Net Framework 3.0/3.5 Features

- Windows Presentation Foundation (WPF)
- Windows Communication Foundation (WCF)
- Windows Workflow Foundation (WWF)
- Windows Card Space (WCS)

- Core New Features and Improvements:
    - Auto Implemented
    - Implicit Typed local variable
    - Implicitly Typed Arrays
    - Anonymous Types
    - Extension Methods (3.5 new feature)
    - Object and Collection Initializers
    - Lambda Expressions

4. .Net Framework 4.0 Features

- Application Compatibility and Deployment
- Core New Features and Improvements
    - BigInteger and Complex Numbers
    - Tuples
    - Covariance and Contravariance
    - Dynamic Language Runtime
- Managed Extensibility Framework
- Parallel Computing
- Networking
- Web
- Client
- Data
- Windows Communication Foundation
- Windows Workflow Foundation

5. .Net Framework 4.5 Features

- .NET for Windows Store Apps
- Portable Class Libraries
- Core New Features and Improvements
- Tools
- Parallel Computing
- Web
  Windows Presentation Foundation (WPF)
- Windows Communication Foundation (WCF)

- Windows Workflow Foundation (WF)

33. Difference between NameSpace and Assembly

Namespace:

- A namespace is collection of classes like
    - Using system
    - Console.writeline("cp")

  Means console class is present in the system namespace.

- A namespace can contain another namespace.
- A namespace is used to organize the code and it is the collection of the classes , structures, delegates, enum, interface, another namespace etc.
- You can create the alias for your namespace.
- {} is the symbol of Namespaces
- Forms the logical boundary for a Group of classes.
- It is a Collection of names where each name is Unique.
- The namespace must be specified in Project Properties.

Assembly:

- Assemblies are Self-Describing
- It is an Output Unit. It is a unit of deployment and is used for versioning. Assemblies contain MSIL code.

34. Reflection

Reflection is used to dynamically load a class, create object and invoke methods at runtime. It can also be used to read its own meta data to find assemblies, modules and type information at runtime.

Reflection typically is the process of runtime type discovery to inspect metadata, CIL code, late binding and self-generating code. At run time by using reflection, we can access the same "type" information as displayed by the ildasm utility at design time. The reflection is analogous to reverse engineering in which we can break an existing *.exe or *.dll assembly to explore defined significant contents information, including methods, fields, events and properties.

You can dynamically discover the set of interfaces supported by a given type using the System.Reflection namespace. This namespace contains numerous related types as follows:

Types	Description
Assembly	This static class allows you to load, investigate and manipulate an assembly.
AssemblyName	Allows to exploration of abundant details behind an assembly.
EventInfo	Information about a given event.
PropertyInfo	Holds information of a specified property.
MethodInfo	Contains information about a specified method.

35. Value types and Reference types

There are two types of data types in .Net, Value types and Reference types. Value types are stored in stack part of the memory. Reference type are stored in managed heap. Let have a look at the example for better understanding.

Int iCount = 0; \\ Value Type

int NewiCount = iCount; \\ Reference Type

36. satellite assembly

A satellite assembly are used when multilingual (UI) application are created. Satellite assembly is a compiled library that contains localized resources which provides us with the capability of designing and deploying solutions to multiple cultures, rather than hard coding texts, bitmaps etc

Difference between static class and singleton pattern?

- The big difference between a singleton and a bunch of static methods is that singletons can implement interfaces.
- A singleton allows access to a single created instance - that instance (or rather, a reference to that instance) can be passed as a parameter to other methods, and treated as a normal object. A static class allows only static methods.

- Singleton object stores in **Heap** but, static object stores in **stack**
- We can **clone** the object of Singleton but, we can not clone the static class object
- Singleton class follow the **OOP**(object oriented principles) but not static class
- we can implement **interface** with Singleton class but not with Static class.
- Singleton Can be lazy loaded when need (static classes are always loaded).,

What is the difference between "throw ex" and "throw" methods in C#?

- "throw ex" will replace the stack trace of the exception with stack trace info of re throw point.
- "throw" will preserve the original stack trace info.

Can we use "this" inside a static method in C#?

No. We can't use "this" in static method.

We can't use this in static method because keyword 'this' returns a reference to the current instance of the class containing it. Static methods (or any static member) do not belong to a particular instance. They exist without creating an instance of the class and call with the name of a class not by instance so we can't use this keyword in the body of static Methods, but in case of Extension Methods we can use it the functions parameters. Let's have a look on "this" keyword.

The "this" keyword is a special type of reference variable that is implicitly defined within each constructor and non-static method as a first parameter of the type class in which it is defined. For example, consider the following class written in C#.

Can we override private virtual method in C#?

No. We can't override private virtual methods as it is not accessible outside the class.

Explain Indexers in C#?

Indexers are used for allowing the classes to be indexed like arrays. Indexers will resemble the property structure but only difference is indexer's accessors will take parameters. For example,

```
class MyCollection<T>

{

 private T[] myArr = new T[100];

 public T this[int t]

 {

 get

 {

 return myArr[t];

 }

 set

 {

 myArr[t] = value;

 }

 }

}
```

What is Operator Overloading in C# .net ?

We had seen function overloading in the previous example. For operator Overloading, we will have a look at the example given below. We had defined a class rectangle with two operator overloading methods.

```
class Rectangle


{
```

```
 private int Height;

private int Width;



 public Rectangle(int w,int h)

{

  Width=w;

  Height=h;

}

 public static bool operator >(Rectangle a,Rectangle b)

{

  return a.Height > b.Height ;

}

 public static bool operator <(Rectangle a,Rectangle b)

{

  return a.Height < b.Height ;

}

}
```

Let us call the operator overloaded functions from the method given below. When first if condition is triggered, the first overloaded function in the rectangle class will be triggered. When second if condition is triggered, the second overloaded function in the rectangle class will be triggered.

```
public static void Main()

{

Rectangle obj1 =new Rectangle();

Rectangle obj2 =new Rectangle();



 if(obj1 > obj2)

 {

  Console.WriteLine("Rectangle1 is greater than Rectangle2");

 }



 if(obj1 < obj2)

 {

  Console.WriteLine("Rectangle1 is less than Rectangle2");

 }

}
```

---

1. C-Sharp (C#)

C# is a type-safe, managed and object oriented language, which is compiled by .Net framework for generating intermediate language (IL)

2. Types of comments in C#

Below are the types of comments in C# -

- Single Line Comment Eg : //
- Multiline Comments Eg: /* */
- XML Comments Eg : ///

3. Sealed class and Sealed method in C#

Sealed Methods
Sealed method is used to define the overriding level of a virtual method.

Sealed keyword is always used with override keyword.

So “sealed” modifier also can be used with methods to avoid the methods to override in the child classes.

```
 class Program
    {
        public class BaseClass
        {

            public virtual void Display()
            {
                Console.WriteLine("Virtual method");
            }
        }

       public class DerivedClass : BaseClass
        {
            // Now the display method have been sealed and can;t be
overridden
            public override sealed void Display()
            {
                Console.WriteLine("Sealed method");
            }
        }

       //public class ThirdClass : DerivedClass
       //{

       //    public override void Display()
       //    {
       //        Console.WriteLine("Here we try again to override display
method which is not possible and will give error");
       //    }
       //}
```

Sealed class is used to prevent the class from being inherited from other classes. So “sealed” modifier also can be used with methods to avoid the methods to override in the child classes.

1. A class, which restricts inheritance for security reason is declared, sealed class.
2. Sealed class is the last class in the hierarchy.
3. Sealed class can be a derived class but can't be a base class.
4. A sealed class cannot also be an abstract class. Because abstract class has to provide functionality and here we are restricting it to inherit.

In some situation we will get requirement like we don't want to give permission for the users to derive the classes from it or don't allow users to inherit the properties from particular class in that situations what we can do?

For that purpose we have keyword called "Sealed" in OOPS. When we defined class with keyword "Sealed" then we don't have a chance to derive that particular class and we don't have permission to inherit the properties from that particular class.

Another Important information is if we create a static class, it becomes automatically sealed. This means that you cannot derive a class from a static class. So, the sealed and the static class have in common that both are sealed. The difference is that you can declared a variable of a sealed class to access its members but you use the name of a static class to access its members.

Here we can declare a class that is derived from another class can also be sealed. Check below

```
Public sealed class Test
{
public int Number;
public string Name;
}

Public sealed class child1:Test
{
public string Name;
}
```

4. Array and ArrayList in C#

- Array stores the values or elements of same data type but arraylist stores values of different datatypes.
- Arrays will use the fixed length but arraylist does not uses fixed length like array.

5. use of "using" in C#

“Using” statement calls – “dispose” method internally, whenever any exception occurred in any method call and in “Using” statement objects are read only and cannot be reassignable or modifiable.

6. Difference between “constant” and “readonly”

- “Const” keyword is used for making an entity constant. We cannot modify the value later in the code. Value assigning is mandatory to constant variables.
- “readonly” variable value can be changed during runtime and value to readonly variables can be assigned in the constructor or at the time of declaration.
- Constant can not set to static, where as readonly can be set to readonly
- Constanct value at the time of declaration is mandatory where is for readonly it is optional but you can define in constructer.
- If you have set static to readonly then you can not define its value in constructer even.

7. “static” keyword in C#

“Static” keyword can be used for declaring a static member. If the class is made static then all the members of the class are also made static. If the variable is made static then it will have a single instance and the value change is updated in this instance.

8. Can we have only “try” block without “catch” block in C#?

Yes we can have only try block without catch block, but you have to specify finally in that case. So either catch or finally has to be there with try.

9. Can we execute multiple catch blocks in C#?

Yes, but first catch block should not catch all the exception like first can have filenotfoundexception and then second one can have Exception , but vice versa is not possible. Because all other exception inherits from exception class.

10. “out” and “ref” parameters in C#

“out” parameter can be passed to a method and it need not be initialized where as “ref” parameter has to be initialized before it is used.

11. Jagged Arrays

If the elements of an array is an array then it’s called as jagged array. The elements can be of different sizes and dimensions.

```
int[][] scores = new int[2][]{new int[]{92,93,94},new int[]{85,66,87,88}};
```

Where, scores is an array of two arrays of integers - scores[0] is an array of 3 integers and scores[1] is an array of 4 integers.

12. Can we use “this” inside a static method in C#

No. We can’t use “this” in static method.

13. String Builder class in C#

This will represent the mutable string of characters and this class cannot be inherited. It allows us to Insert, Remove, Append and Replace the characters. “ToString()” method can be used for the final string obtained from StringBuilder. For example,

```
StringBuilder TestBuilder = new StringBuilder("Hello");

TestBuilder.Remove(2, 3); // result - "He"

TestBuilder.Insert(2, "lp"); // result - "Help"

TestBuilder.Replace('l', 'a'); // result - "Heap"
```

14 “System.Array.Clone()” and “System.Array.CopyTo()”

- “CopyTo()” method can be used to copy the elements of one array to other.
- “Clone()” method is used to create a new array to contain all the elements which are in the original array.

15 List out some of the exceptions in C#

Below are some of the exceptions in C# -

- NullReferenceException
- ArgumentNullException
- DivideByZeroException
- IndexOutOfRangeException
- InvalidOperationException
- StackOverflowException etc.

16. Delegate in C#

Delegates are type safe pointers unlike function pointers as in C++. Delegate is used to represent the reference of the methods of some return type and parameters.

Below are the types of delegates in C# -

- Single Delegate
- Multicast Delegate
- Generic Delegate

Below are the list of uses of delegates in C# -

- Callback Mechanism
- Asynchronous Processing
- Abstract and Encapsulate method
- Multicasting

17 “as” and “is” operators in C#

- “as” operator is used for casting object to type or class.
  - is operator determines whether an object is of a certain type.
  - If(Ford is Car) // checks if Ford is an object of the Car class.
    -
- “is” operator is used for checking the object with type and this will return a Boolean value.
  - as operator casts without raising an exception if the cast fails.

```
Object obj = new StringReader("Hello");
StringReader r = obj as StringReader;
```

-

18 Enum in C#

enum keyword is used for declaring an enumeration, which consists of named constants and it is called as enumerator lists. Enums are value types in C# and these can’t be inherited. Below is the sample code of using Enums

> Eg: enum Fruits { Apple, Orange, Banana, WaterMelon};

19 “Continue” and “break” statements

- “continue” statement is used to pass the control to next iteration. This statement can be used with – “while”, “for”, “foreach” loops.
- “break” statement is used to exit the loop.

20 Partial Class in C#

Partial classes concept added in .Net Framework 2.0 and it allows us to split the business logic in multiple files with the same class name along with “partial” keyword.

21 Anonymous type in C#

This is being added in C# 3.0 version. This feature enables us to create an object at compile time. Below is the sample code for the same –

```
Var myTestCategory = new { CategoryId = 1, CategoryName = “Category1”};
```

22 Name the compiler of C#

C# Compiler is – CSC.

23 types of unit test cases

Below are the list of unit test case types –

- Positive Test cases
- Negative Test cases
- Exception Test cases

24 Copy constructor in C#

If the constructor contains the same class in the constructor parameter then it is called as copy constructor.

```
public MyClass(MyClass myobj) // Copy Constructor
 {
 prop1 = myobj.prop1;
 prop2 = myobj.prop2;
 }
}
```

25 Collection types in C#

Below are the collection types in C# -

- ArrayList

- Stack
- Queue
- SortedList
- HashTable
- Bit Array

26 Attributes in C#

- Attributes are used to convey the info for runtime about the behavior of elements like – “methods”, “classes”, “enums” etc.
- Attributes can be used to add metadata like – comments, classes, compiler instruction etc

Below are the predefined attributes in C# -

- Conditional
- Obsolete
- Attribute Usage

27 Thread in C#

Thread is an execution path of a program. Thread is used to define the different or unique flow of control. If our application involves some time consuming processes then it’s better to use Multithreading., which involves multiple threads.

Below are the states of thread –

- Unstarted State

- Ready State
- Not Runnable State
- Dead State

Below are the methods and properties of thread class –

- CurrentCulture
- CurrentThread
- CurrentContext
- IsAlive
- IsThreadPoolThread
- IsBackground
- Priority

28 Overloading in C#

When methods are created with the same name, but with different signature its called overloading. For example, WriteLine method in console class is an example of overloading. In the first instance, it takes one variable. In the second instance, “WriteLine” method takes two variable.

Different types of overloading in C# are

- Constructor overloading

- Function overloading
- Operator overloading

In Constructor overloading, n number of constructors can be created for the same class. But the signatures of each constructor should vary

In Function overloading, n number of functions can be created for the same class. But the signatures of each function should vary.

- Method overloading allows a class to have multiple methods with the same name, but with a differenct signatrure.
- Can be overloaded based on Number, type (like int,string) kind (out, ref)

29 Inheritance in C#

In object-oriented programming (OOP), inheritance is a way to reuse code of existing objects. In inheritance, there will be two classes - base class and derived classes. A class can inherit attributes and methods from existing class called base class or parent class. The class which inherits from a base class is called derived classes or child class.

- Inheritance allow code reuse , which can reduce the time and errors
- C# supports only single inheritance, means derived class can only have one base class
- Parent class constructer called before child class constructer
- Multi level inheritance is possible.
- Multiple inheritance is possible with interfaces
- When we create the child class instance , the base class constructer automatically executed. And before child class, parent constructer will be called.
- If parent class has 2 const, with or without parms then by default without parms const will be called but if you want to call with parms const then
    - Public Class () : base (“params”)
      {
      }

30 Polymorphism in C#

The ability of a programming language to process objects in different ways depending on their data type or class is known as Polymorphism. There are two types of polymorphism

- Compile time polymorphism. Best example is Overloading
- Runtime polymorphism. Best example is Overriding
- **Compile Time Polymorphism** (Called as Early Binding or Overloading or static binding)
- **Run Time Polymorphism** (Called as Late Binding or Overriding or dynamic binding)

31 DataTypes in C#

Boolean

Integeral

- Sbyte -> Signed -> -128 to +127
- Byte -> 0 to 255
- Short -> -32768 to 32767
- Ushort – 0 to 65535
- Int
- Uint
- Long
- Ulong
- Char
- dynamic type- You can store any type of value in the dynamic data type variable. Type checking for these types of variables takes place at run-time.
- Object type -Dynamic types are similar to object types except that type checking for object type variables takes place at compile time,

whereas that for the dynamic type variables takes place at run time.

Integers are smaller than float datatype

To check min or max value

Int.minValue

Int.maxValue

Floating Types

- Float -> 7 digits 32 bit
- Double -> 15 – 16 digits 64 bit
- Decimal -> 28-29 digits 64 bit

String Type

String name =”cp”

Now suppose I want to print name with the double code -> Escape sequence because double quote has different meaning in c# but if want double quote to be treated as any other printable character we use the concept of escape seql

String name = “\”cp\”

We can replace blackslash with @ symbol as well if there are 20-30 directives then we can instead of writing \ so many times we can simply add @ symbol at the beginning.

32 Operators in C# -

Assignment Operators ----------→ =

Arthmetic Operators ----------→ +, - , * , / , %

Comprasion ----------→ == , != , > , <=

Conditional ----------→ && , ||

Ternery ----------→ ?

Null colassing Operator ---------→ ??

Nullable Operator

In C# types are divided in 2 category

1 Value Type – int, float, double, struct, enum etc

2 Refernce Type – Interface, class, delegates, array string etc.

By default value type are non nullable

Use ? to make tham nullable.

Int I =0 – I is non nullable, so I cant set to null I = null complie time error

Int? j = 0 (j is nullable , so j = null is ok)

Null Collasing operatior

Int? ticketsonsale = null

Int availableticic;

If(ticketOnsale == nul)

availableticic = 0

else

availableticic = (int)TicketOnsale

33 DataTypeConversion

Implicit

Expilicit

Implicit conversion is done by the complier

When there is no loss of information

If there is no possibility of throwing exception dureing the conversion

Int I = 100

Float f = i

Console.writline(f)

Float f = 123.45f

Int I = f (error) compile tiem (as float is large then int) but if know that our number is ok for this then we can use convewrsion. But we will lose fraction part

Int I = (int) f;

Output = 123 –

Here we could also use convert class.

Diffiernce - if you have large number you will have output in minumn value of float.. it will not give you exception , but if you convert class you will get an exeption.

Parse Vs TryParse

TryParse

String n = “100cp”

Int result = 0

Int.tryParse(N, out result)

It will convert the value and will store in result variable.

Return type of parse is int but of trypasr is bool , if the conversion is success it will return trye otherwise false.

34 Arrays in C#

If you want to store a group of similler datatypes in a single variable use arrays.

Int []

evenNo = new Int [3]

eveNo[0] = 3

they are strongly type means if declared int then only int type can be assigned.

They can not grow in size once initialized.

Array Properties

- The length cannot be changed once created.

- Elements are initialized to default values.
- Arrays are reference types and are instances of System.Array.
- Their number of dimensions or ranks can be determined by the Rank property.
- An array length can be determined by the GetLength() method or Length property.

35 Comments in C#

Single line ------→ //

Multi line ---------→ /* */

Xml documentiation ------→ ///

36 Difference Between Static and Instance Method

Methods are also called functions , they are very useful as they allow you to define your logic once and use it at many places

Maintance of your application is easi

Method Body – AccessModifier, ReturnType, MethodName, Parameters

Method can return only one value

Method with static keyword known as static method

If we want to call a public (non staic method or instance fun) in a static method than first we have to create the object of that classs in which the (instance non static fun) resides.

If both are static then you don't need to create instance of program class.. call it direct. With class.methodname

37 Types of parameters method can have

Value – seprate copy will be created

Refernce – both will point to same location.. changes in one reflect in another

Out – when you want function to return more than 1 value

Parameters array – to make parameters optional , params should be ur last paramets, and more than 1 is not allowed

When you paremeters in mehod – they are called parmeters , while calling that method and you pass values for those parmets are known as argumets

38 Class

A class is simply chunk of code that does a particular job

You might have a class that handles all your database work, The idea is that you can resue this code whenever you need it or when you need it another projects

39 Constructers

- To initialize its field class can have the constructers
- Same name as class
- Constr. Does not return a value, so don't have return type
- Const. will be called automatically when you create instance of your class
- Constructers are mandatory for the class if you don't create c# will provide it for you , default one, which has no parameters

Types of Constructors - Basically constructors are 5 types those are

- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Static Constructor
- Private Constructor

Destructer –

- Class can also have destructor to clean up the all resource that your class was holding at its life time
- Same name as the class , att ~ sign before name
- They don't accepts parameters

- You don’t need to call this, GC will call it automaticlay

40 Static Vs Instance Class

Suppose

Class circle

{

    Float _Pi 3.14f

Int _radious ;

Public Circle(int radious)

{ this._Radious = radious}

Pubic CalcualteArea()

{

    Return this.Pi * this._radios * this.readious

}

}

Now suppose if you create 10 times object of your class to create the area, then 10 times radious and Pi value value will be initilized

Radious value is fine as it may chagen each time when you carete the object , but PI is same all the time, but still with this approace unnessory meemor is being occupied, so we can make PI staic.

Differences

- Static methods are invoked by the name of the class, And it is pretty obvious for static method that they are called directly by the class name because it does not change on object base only 1 memory of the member , no matters how many object are created.
- Instance method are invoked by creating the object of that class

- Access modifiers are not allowed in static constructer , you don't need public, simple function without access modifier by default private, static does not allow modifiers, and again it is pretty obvious because you know static members will be same throught the whole program and you can not explicitly call the static constructor and you know by default is private
- To initialize static fields we need static constructor.
- Static const. called before access construter, and will be called only once
- Static const are called directly by class name.

-

41 Method Hiding

When you use the concept of inheritance and base class and parent class has the same function then the parent class function is hide . function in the base class give you msg or green line that this method is hiding the function of parent class. But his is just warning. So we have to use new keyword to tell dotnet that we are aware of this thing.

But still you want to call base class function even after hiding that

You can do like this

```
Base.PrintName();
```

Or

```
ChildClass CC = new ChildClass();

((ParentClass)CC).PrintName();
```

Or

```
ParentClass PE = new childclass();

PE.PrintName();
```

42 Polymorophism

It is the ability to create a variable or a function or an object that has more than one form.

Suppose

ParentClass Pe = new ChildClass()

Pe.PrintName()

Now this will call the function which is in parent class, because you have created the base class reference.

But you want to call the method which is in your child class , how to do this you have to use 'Override' but before this you have to set your method in parent class ' Virtual' basically virtual allows the derived class can override the method if they widh to do so

So polymorphism enables you to invoke derived class methods through base class reference variable at runtime.

Method Overriding Vs Method Hiding

Overriding is cleer with Virtual and override keyword, even though you point the parent class variable still the child method will be called,

Method overloadin is clear with new keyword.

43 Properties

- Making the class field public and exposing to the external workd is bad ad you will not have control over what gets assigned and returned
- Like id should not accept negative no but since you have declared it publick so use can provide any value in it. To prevent this we use the concept of properties
    - Private int _id
      Public void SsetId (int Id)
      {
      If(I < 0)
      {
      {
      Throw ex
      }
      }
      Else
      {
      _Id = Id
      }
      }

      Public void GetId(){
      Return this.Id;

}

Now the concept of createing private fileds and accessing them using setter and getter method is known as encapsulation the concept of hiding.

When you don't have any logic code in properties than instead of writing the bigger code you can write the auto implemented properties.

When you use the auto implemented properties the compiler creates a private, anonyoumes filed that can only be accessed throu the property get and set acessor.

Compiler will automatically creates privates fileds fo us.

44 Struct

Just like class struct can have priate fields, public properties, constructor , method

Using struct keyword

Public strint customer

{

Private int _id;

}

Structure can also have the constructor to initiize these fileds.

Difference between class and struct

- Struct Value type and Class – Reference type
- Struc are stored on stack where as class on heap
- Struc can not have parmeters less construstor
- Struct can not inheritance another class
- When dotnet runs a program the variables and the objects you created in that programe are strored in the memory, till your programe executes so the physical memory of a computes is logically devided by the dotnet , something called stack and heap.
- We cannot declare structure as "protected",because "protected" is related to inheritance and A struct cannot inherit from another struct or class, and also it cannot be the base of a class.

  Customer C = new customer()
  C -> is reference variable

2 things for this , int I and int j are stored in the stack as they are value type and C is also stored in stack but the C1.Id stored in the heap
Int I = 10;
Int j = I ;
They are value type . in the memory 2 varialbe I and j
J operation will not affect on I , because I is sotred seprataly now let see what happesns when we create refence thype

Custmoer C1 = new Customer();
C1.Id = 101
C1.Name = “ha”

Customer C2 = C1;

It will not create the copy of the objects it only creates the copy of the refence variable so we have 2 ref var c2 and c2 and signle object

- When you copy a struct into another struc , a new copy of that struc is created, and modification on one stuch will not effect the value contained by the other struct
- When you copy one class into another class, we only get a copy of reference variable, both the ref variable point to the same object in the heap. Changes in one will effect to other

45 Interface

- Using interface keyword, like – interface ICustomer
- Only declaration , you can not have implementation
- Interface members are by default public, but you can not explicitly set public keyword like ---→ public void Print() --→error
- Interface can not contain fields like ---→ int id (error)
- Class which inerhits interface , that class has to provide the implementation for all the methods defined in that interface, if one interface another interface as well then derived class has to provide implemention for methods defined in both the interfaces
- Classes allows multiple interface inheritance like --- > Class customer : IC1, IC2
- We can not create the instance of Interface

46 Explicit Interface Implementation

Suppose I have 2 Interfaces and both have the method with the same name and same signature

Now suppose some class inherits both the Interfaces, and provide the implementation of that method, when you build the application it will successfully build, because dotnet thinks that you have provided the implememtaion for both method, but since we are inheriting with 2 interface we are not clear with this.

Program P = new Program();

P.Add()

So to make this thing clear we use explicit interface like this

```
Class Programe () {
        Void I1.Add() { // your content}
        Void I2.Add() { // you content }
    }
```

Prog P = new Prog();

((I1)P).Inter();

((I2)P).Inter();

Second way

I1 i1 = new Programe();

I1.Add();

And notice we have not used access modifier while providing the implementation in of Add method as

Acess modifiers are not allowed on explicitly implemented interface members.

47 Abstract Class

- Using abstact keyword
- Abstract class are incomplete , that is they have abstact membbers (properties, methods, events)

- When you use abstract keyword then method can not have impomentation because abstract class can have incomplete methods and this method is here for any class that gonna derived from this abstract class to provide implementation for this method
- You can not create instance of the abstract class, as it no sence because they have incomplete methods that is abstract class can not be instatiaed, it can only be used as base class for other class.
- When a non abstract class inherit the abstract class then that class has to provide the implementation for all the abstract methods which is in the base class, if it doesn't then yhou will get complier error
- Note - A parent class reference variable can point to a derived class object. As we know that abstract class can not be instainted that is we can not create the instance of an abstract class, but we can do this
    - Public abstract Class Customer () { public abstract void print(); }
    - Public class Programe : Customer {public void print { //impmentaton}}
        - Customer c = new Programe()
        - C.Print();
- Abstract class ref variable can point to derived class object
- Abstract class can note be sealed, sealed means that this class can not be inherited, so abstract and sealed at the same time can not possible.
- Abstract class can have abstract members (methods, properties, indexes and events) but not mendatory
- If a class is a abstract class then here is no gurantee that it is gonna have abstract method.

48 Abstract Vs Interface

- Interface can not have implementation for its any method, where as abstract class can have
- Interface memebers are public by default , and they can not have access modifier where as abstract class members can have access modifiers and abstract class memebers are private by default
- Fields in abstract, no fields in interface
- An interface can inherit from another interface only and can not inherit from an abstract class or non abstract class, whereas abstract class can inherit from another abstract class or another interface
- A class can inherit from multiple interfaces at the same time where as a class cannot inherit from multiple classes at the same time as we know classes doest not support multiple inheritance but they do support multiple interface inheritance
- Abstract class memebers can have access modifiers, interface can not have

49 Problem with Multiple Classes Inheritance

```
Public Class A
{
        Public virtual Print() { “A”}
}
Public Class B : A
{
        Public override Print() {“B”}
}
Public Class C:A
{
        Public override Print() {“C”}
}
Now Suppose
Class D : B,C
{
        // this is not possible in dotnet .. but if we do so then dotnet will get confuse when you create the instance of D, then B says exues my method c says my…
}
```

This is called dymaon problem because it has dymand shape

50 Multiple Class Inheritance using interface

```
Interface IA
```

```
{
        Void AMethhod();
}
Class A : IA
{
        Public void AMthods { //}
}

Interface IB
{
        Void AMethhod();
}
Class A : IB
{
        Public void BMthods { //}
}
```

Now create a new class

```
Class AB : IA,IB { //
}
```

Now this class is inheritinf the both interface than it has to provide implemtation for the methods, but as you know that class A and class B already implemented then you don't need to do it again

```
Class AB : IA, IB
{
        A a = new A();
        B b = new B();
        Public void AMethod()
                {
                        a.Amethod();
```

```
        }
    Public void BMethod()
        {
            b.Bmethod();
        }

}
```

Now how to use

Ab ab = new Ab();

Ab.AMethod()

Ab,BMethod();

51 Delegates

A delegate is a type safe function pointer , meaning it points to your function.

Pubic static void Hello (str message) { // }

Syntax of delegate is similler to a function apart from delegate keyword

Public delegate void Hello (str message);

When you put delegate meaning that this delegate now can be used to point function that has the same signature like this above delegate will point to a function which has void data type and string parameters, now how to point this delegate to a function you have to create instance of delegate

Hello del = new Hello(“cp”)

Now to call

Del(“Hi frind”)

A delegate is a type safe function pointer meaning change your hello function from void to string datatype you will get an error., that's why type safe

Suppose you have a function in what you have writtern some logic in the class, but I want to make this class resulabe . anyone can use this class and provide its own logic but this class is not resulable bdcuase yhou have hard coded the logic of employee promotion, if anyone whants to resue it then it has to recode

Suppose someone has to promote on the basis of salalry so I wan to make this class flexible enough to plugin the logic they want , so will create delegate and remove hard code and make class resulab.e

52 Exception Handling

We have to go for exception handling for the –

- For the end user message should be useful that's why we use exception handling
- Proflem if we don't sue exp handling then hacker can use this information for chacking
- Like in this example error occur in first line will be skipped and your stream reader will never closed
- Try catch and finally blocks…
- Finnaly block is optional
- Inner Exception – suppose some error occurred in try block and comes in catch block and again in catch some error occur like file not found while writing the code, then unhandled expection may occur so you always put this try catch in main try catch so if inner catch any expetion occurred then reaches to main cartch block. So first we have diveide by 0 ex and now another exp occure file not found , now I am throwing the exception fo file not found but I don't want to loose the original 1st exp , so to retain that expction pass ex as a parmeter. Expan it you will see inner expetion divde by zero there.

You can not have 2 catch blocks like this 1st is main catch and 2nd is filenotfound , because 1st block already catches all the exception but revers is possible.

Custom Exception – As we know exception is nothing but just a class and we know all the exception inherit from the exception class, so you will also have to inherit from the esception class to create the custom exception

Serization – it is nothing but breaking down the object in packets that can be transmitted over the network.

53 Enum

Suppose we have 8 .. east west ..north .. ect and we have hsed 0 1 2 for that .. but then programe becomes unmangable , after some time even you don't remember which no belongs to which string .. that's why we use enums.

```
Public enum Gender
{
        Unknown,
        Male,
        Female
}
```

If you don't specify anything then it uses the default , meaning the underline datatype of this enum is interger and the values of enum is started with 0

Enum Class has 2 methods

GetName (Type enumtype)

GetValue (Type enumType)

To reterive value

Enum.getValue(typeof (Gender))

To get names

String [] names = Enum.GetNames (typeof (Gender)

It is possible to change the enum default value like by default it starts with 0

```
Public enum Gender :short
{
        Unknown  =1
Male,
female
}
```

You don’t need to provide for other male and female , automatically 2 and 3 will be assigned. Even you can assigne explicit as well .. female 23 , male 44

Capital Enum vs enum

Enum is class which has got static method enum keyword we use to create enumrations

Enums aer strongly types constants. As by default the are integer type but you can not assign int value dierctlylike this

Gender gender = 3

You have to do like this

Gender gender (Gender)3;

54 Type Vs Type Members

Private fields, properties, methods – Type Members

Classes, Struct, Enum, Delegate, Interface – Types

Access Modifers for Type Members

- Private – only in the container where they declared
- Public – are accessible everywhere
- Protacted – with in the containing types and the types derived from the containing type
- Internal – avilabe to all the classes in the assembly
- Protacted internal – with derived class and in the assembly

If you don’t specify acess modifier then by default is private

Access Modifiers for Types

- Public – Avilable to everywhere even in the all assemblies in solution
- Internal – available only in the assembly

If you don’t specify any access modifier by default is internal.

55 Attributes

Use – suppose your uses are already using some method of your class, now you some of customer wants to modify this, and after modification it is perfect but now you want to disclose this method to user but you can not remove first methods because currently users are using that , now how you will tell the user to use new method this can be done using attributes

Now when I have 2 method 2 is old and second is new and I complie there is no eror, now we want to show warning message in the complie window when a user uses a old mehod, that the methods are using is obsulte

[Obsolete (“your are suing old one”) , true]

Public int (a.b)

{

}

If we set the property true, than it will be mandatory for all others to use this new method

56 Reflection

Your data is in form of assembly… there is some concept

Early and Late binding – suppose you have a class and you are createing the instance of your class means you knows your class at compli time, this is early binding but therir might be situations that at complit time you might not have the knowledge of you that class for which you have to create instance , this only can be done at run time and when you does that is late binding

And late binding sis possible using reflection so late bindin is nothing but creating the instance of class at run time

To use reflection

Type t = Type.GetType(“namespace.classname”);

ProerpttyInfo [] prop = T.getProperties()

Foreach(pro p in prop)

{

p.propertytype.name

}

First you have to load assembly

Assebmly a = Assebly.GetEx.

a.GetType(_"namespace.classname")

now to initliae the class activator.CreateInstance();

57 Generics

Equal programe , you can see the method is tightly coupled of int now suppose you want to compare 2 strings it is not possible because int I int no , now how can I make this proggrame to oerate in any datataype

One way is using object datatype but the problem is when you pass 10 and 10 .. boxing -> at runtime these to value types 10 10 will be converted into reference type so here boxing is happing unnecessary iproblem with boxing is degrade the performance second problem of using object is it allows (10, 'A')

So the best way to make the programe to be use with diff datataypes using generics

```
Public static bool AreEqual <t> (T val1, T val2)
{
        Return value.equals(values2)
}
```

Classes can also be generics instead of spefies T in the method you can do this

Public Class Cal<t>

Delegates interface can be generics

Gen allows you to write a class or method that can work with any data type . it helps you to minimize code resue, tpe safty and performance.

58 Partial Classes

- Partial classes allow us to split a class in 2 or more classes, All these parts are then combined into a single class , when the application is complied,
- The partial keyword can also be used to split struct or an interface into 2 or more files
- When you are working for large project speratding a class over seprate files allows multiple programeers to work on it simultenseouly.

Rules for partial class

- All those classes must use partial keyword
- All those must have the same access modifiers
- If any of the part is declared abstract then all has to be abstract
- If any of the part is declared sealed then all has will be considerd as sealed
- If any of the part class inherits then all should inherit
- And all should inherit same class as dotnet does not support multiple inheritance
- Any member that are declared in a partial definition are avialbe to all of the other parts of the parital class

Rules for partial method

- A partial class or struct can contain partial methods
- Partial method created using partial keyword
- Partial method declareaion in 2 parts one is definition (only the method signatiure) second is implementation
- The implementation is optional, if we don't provide then complier removes the signature
- Partial methods are private by default, and it is complie time error to declare any access modifier even including private
- It is complie time error, to include declaration and implemntaion at the same time for partial method.
- A partial method return time must be void, incliding any other type is comple time error

59 Method Parameters Options

- Using parameters array
- Method overloading
- Spcify parameers default
- Use optional attribute that is present in system.runtime.interoptservice namespace

Parameters array

```
Public int  add (int a, int b, params object[] restofnumbers)

{
```

}

Add(2,3, new object[34,34.32]);

Method overloading

Public int Add (int a , int b) {

Add(a,b,null)

}

Public int Add (int a, int b , int [] restno) {}

Calling

If user pass – Add(3,4)// ok

If use pass – Add (3,4, new int[] 2,3) // ok

Specifing parmas default with = operatior

Public int add (int a , int b =10 , int c= 19)

{}

Add (1,2)

Now 1 will be pass to a and 2 will be pass to b

But suppose I want to pass for c instead of B ..

Add(1 ,c: 2)

Now b will get its default value

Optional keyword

Public int add (int a , int b, [Optional] int[] restofnu){}

60 Private Constructor

Private constructor is a special instance constructor used in a class that contains static member only. If a class has one or more private constructor and no public constructor then other classes is not allowed to create instance of this class this mean we can neither create the object of the class nor it can be inherit by other class. The main purpose of creating private constructor is used to restrict the class from being instantiated when it contains every member as static.

Important points of private constructor

- One use of private construct is when we have only static member.
- Once we provide a constructor that is either private or public or any, the compiler will not allow us to add public constructor without parameters to the class.
- If we want to create object of class even if we have private constructors then we need to have public constructor along with private constructor

61 Extension Methods

Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.

Feature and Property of Extension Methods

The following list contains basic features and properties of extension methods:

1. It is a `static` method.
2. It must be located in a `static` class.
3. It uses the "`this`" keyword as the first parameter with a type in .NET and this method will be called by a given type instance on the client side.
4. It also shown by VS intellisense. When we press the dot (.) after a type instance, then it comes in VS intellisense.
5. An extension method should be in the same namespace as it is used or you need to import the namespace of the class by a `using` statement.
6. You can give any name for the class that has an extension method but the class should be `static`.
7. If you want to add new methods to a type and you don't have the source code for it, then the solution is to use and implement extension methods of that type.
8. If you create extension methods that have the same signature methods as the type you are extending, then the extension methods will never be called.

```
public static class ExtensionMethods
{
    public static string UppercaseFirstLetter(this string value)
```

```
    {
        //
        // Uppercase the first letter in the string.
        //
        if (value.Length > 0)
        {
            char[] array = value.ToCharArray();
            array[0] = char.ToUpper(array[0]);
            return new string(array);
        }
        return value;
    }
}


static void Main()
    {
        //
        // Use the string extension method on this value.
        //
        string value = "dot net perls";
        value = value.UppercaseFirstLetter();
        Console.WriteLine(value);
    }
```

62 Indexes

63 IEnumerable <> vs Iqueryable in C#

IEnumerable	IQueryable
IEnumerable belongs to System.Collections namespace.	IQueryable belongs to System.Linq namespace.
IEnumerable is the best way to write query on collections data type like List, Array etc.	IQueryable is the best way to write query data like remote database, service collections.
IEnumerable is the return type for LINQ to Object and LINQ to XML queries.	IQueryable is the return type of LINQ to SQL queries.
IEnumerable doesn't support lazy loading. So it's not a recommended approach for paging kind of scenarios.	IQueryable support lazy loading so we can also use in paging kind of scenarios.
Extension methods are supports by IEnumerable takes functional objects for LINQ Query's.	IQueryable implements IEnumerable so indirectly it's also supports Extensions methods.

64 Abstraction and Encapsulation in C#

- Encapsulation and abstraction is the advanced mechanism in C# that lets your program to hide unwanted code within a capsule and shows only essential features of an object. Encapsulation is used to hide its members from outside class or interface, whereas abstraction is used to show only essential features.
- In C# programming, Encapsulation uses five types of modifier to encapsulate data. These modifiers are public, private, internal, protected and protected internal. These all includes different types of characteristics and makes different types of boundary of code

Encapsulation: hiding data using getters and setters etc.

Abstraction: hiding implementation using abstract classes and interfaces etc

32 Serialization in c#

Serialization means saving the state of your object to secondary memory, such as a file.

When we want to transport an object through network then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should implement ISerialize Interface. De-serialization is the reverse process of creating an object from a stream of bytes.

Suppose you have a business layer where you have many classes to perform your business data.

Now suppose you want to test whether your business classes give the correct data out without verifying the result from the UI or from a database. Because it will take some time to process.

SO what you will you do my friend?

Here comes Serialization. You will serialize all your necessary business classes and save them into a text or XML file.

on your hard disk. So you can easily test your desired result by comparing your serialized saved data with.

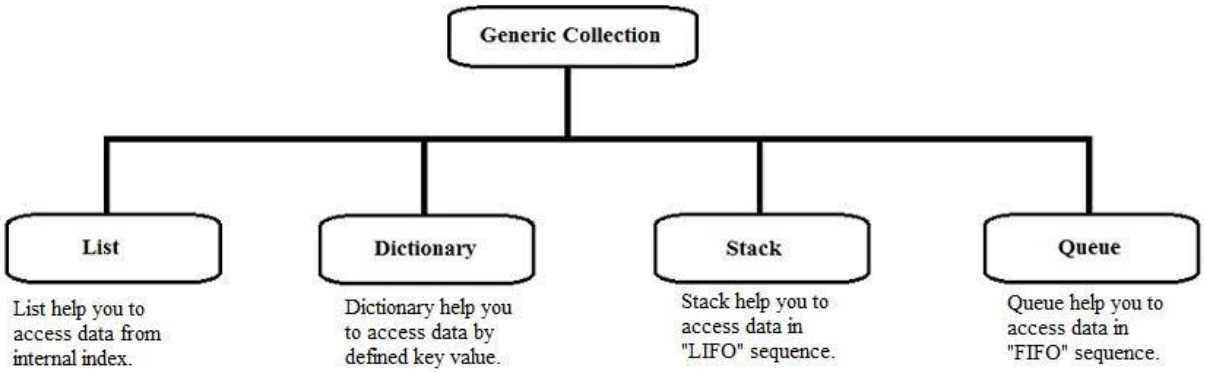
your desired output data. You can say it is a little bit of autonomic unit testing performed by the developer.

There are three types of serialization:

1. Binary serialization (Save your object data into binary format).
2. Soap Serialization (Save your object data into binary format; mainly used in network related communication).
3. XmlSerialization (Save your object data into an XML file).

33 Generics in C#

generics allow you to write a class or method that can work with any data type.



Features of Generics

Generics is a technique that enriches your programs in the following ways:

- It helps you to maximize code reuse, type safety and performance.
- You can create generic collection classes. The .NET Framework class library contains several new generic collection classes in the System.Collections.Generic namespace. You may use these generic collection classes instead of the collection classes in the System.Collections namespace.
- You can create your own generic interfaces, classes, methods, events and delegates.
- You may create generic classes constrained to enable access to methods on specific data types.
- You may get information on the types used in a generic data type at run-time using reflection.

34 Equal Vs == Operator

Both the == Operator and the Equals() method are used to compare two value type data items or reference type data items. This article explains the basic difference between these two. The Equality Operator (==) is the comparison operator and the Equals() method compares the contents of a string. The == Operator compares the reference identity while the Equals() method compares only contents.

```
string name = "sandeep";
        string myName = name;
        Console.WriteLine("== operator result is {0}", name == myName);
         Console.WriteLine("Equals method result is {0}", name.Equals(myName));
```

both will return true

second example

```
object name = "sandeep";
        char[] values = {'s','a','n','d','e','e','p'};
        object myName = new string(values);
        Console.WriteLine("== operator result is {0}", name == myName);
        Console.WriteLine("Equals method result is {0}", myName.Equals(name));
        Console.ReadKey();
```

== operator will return false

Whereas equals will return true

35 Singlton Desging Pattern

What is Singleton Design Pattern?

1. Ensures a class has only one instance and provides a global point of access to it.
2. A singleton is a class that only allows a single instance of itself to be created, and usually gives simple access to that instance.
3. Most commonly, singletons don't allow any parameters to be specified when creating the instance, since a second request of an instance with a different parameter could be problematic! (If the same instance should be accessed for all requests with the same parameter then the factory pattern is more appropriate.)
4. There are various ways to implement the Singleton Pattern in C#. The following are the common characteristics of a Singleton Pattern.

   • A single constructor, that is private and parameterless.
   • The class is sealed.
   • A static variable that holds a reference to the single created instance, if any.
   • A public static means of getting the reference to the single created instance, creating one if necessary.

36 Constructer Chaning

Answer: constructor chaining is a way to connect two or more classes in a relationship as Inheritance, in Constructor Chaining every child class constructor is mapped to parent class Constructor implicitly by base keyword so when you create an instance of child class to it'll call parent's class Constructor without it inheritance is not possible.