Basic Questions .Net -

3. **What is static class/method, why used**

Advance Questions . Net-

**Solid principals with example**

Middle ware , Create Custom middle ware.

1. Middle ware is like a pipeline that handles http request and responses. Every request coming to your app passes through multiple middle ware one after the other each middle can do something before passing to next middleware and can do something after the response is received.  
     
   **Built in Middleware** there are several built in middle ware that we can find in program.cs page. List of the built in middle ware is -  
     
   a. useRouting(): find the routes.  
   b. useAuthentication : check who is the user.

c. useAuthorization() : check if your is authorized to access.

d. useEndPoint() : Executes the matched endpoints.

Validation, how to create custom validation

Design patterns with example.

Factory and Singleton design pattern

Repository and CQRS pattern

Inversion of control (IOC) vs DIP

How to make singleton class

SQL profilers, Waterfall Model

Loosely & tightly couple example

Normalization, 1 NF and 2 NF

DB Session, Sql injection, Sentization

Negative USE Case of your last project

Types of caching, Cookie vs cache

Database-

2nd highest salary

How to delete duplicate records

Rank vs dena rank vs Row Number

Tempoery variable vs Temporary Table

Where vs Having vs group by

Primary vs Unique vs Foreign key

Trigger and Types

No lock and CTE

Self and cross join

Indexing

Transaction

Program write -

1. Find duplicate and Non duplicate value

2. Reverse - String or integer value

4. Write Factorial number program

5. Write a program prime and non prime no

6. Write a program for check is palindrome

Unconditional Questions-

How to do unit testing in c#

How to provide High quality code

How can be optimize the c# code

How to optimize store procedure

Another ask-

Authentication vs Authorization

Filter, Type / Routing, Type

View bag vs Temp data vs View data

Linq query with join

Mvc return type method

MVC Life cycle

Reflection

Extension Method

Boxing and unboxing

Pass by value and pass by reference

Private vs protected vs internal

Static vs Seald vs Private

WCF vs Web service vs Web API

IEmurable vs IEmurater

JavaScript/ jQuery-

Var vs Let vs Const

empty vs remove vs detach

Selector

.Bind vs .on

Promise

Props vs attr

Event delegation

Loose and Strict Quality(==,===)

Focus vs blur

filter vs grep

Find vs is(\*)

Asp.net :-

State Management Technique

Session State and ViewState

Caching and Types

Response.Redirect vs Server.Transfer

Gridview vs Repeater

Postback Event Handling

Asp.net Page life cycle

Web Form vs MVC

WPF :-

What is MVVM

One & two way binding, one time

Static resource vs dynamic resource

Localizations vs Globalization

INotifyPropertyChange

How to bind gridview in WPF

Managerial Round -

\* Tell 5 things for secure the application

\* How to fix production issue when is perfectly work in Devlopment and QA environment.

\* How do you verify 1k records have been successfully synced with another system and ensure that duplicate data is not re-sync.

\* How do you re-sync 1k missing records when 9k records have already been processed

\* If you have multiple high-priority tasks, how do you decide which one to work on first?

\* How do you design a scalable and high-performance system?

\* What best practices do you follow for database or code optimization?

\* How do you approach debugging a critical issue in production?

\* What steps do you take before deploying a major release?

\* How do you handle rollback scenarios in case of deployment failure?

Angular -

What is Pipes

How many type form in angular

Angular lifecycle hooks

Diff components and directives

How Angular handle dependency injection

lazy loading in Angular

How to bind one to second html components

Market Requirements(At least one come)-

\* Front End - Angular or React or Vue.js

\* Devops/Cloud - AWS or Azure Cloud

\* Methodologies - Agile or Scrum

\* Unit Testing - NUnit or XUnit

\* Tools - CI/CD pipeline and Sonarqube

\* Messaging Tools- kafka or rabbitmq

\* Containerization & Orchestration- Docker, Kubernetes

Q. What are Partial views?  
  
A. Partial view is a view which we do not use itself. They will be consumed inside the main view.

Q. Dependency Injection with example.

A. Dependency Injection is a design pattern that reduce the dependency of modules on each other. To implement dependency injection, we use dependency injection container and we register all the required services in it and then inject it to the constructor of the class that request for the registered service, rather that creating object of it.  
  
Example

Page 1

Page3

Page2

As shown in the above example suppose there are two classes Emails and DB Connection and they have been instantiated in each of these pages now suppose if there is a change in the project and for some reason these classes have been replaced with new classes named Email\_New and DBConnection\_New. In that scenario we need to go back to each page page1, page2 and page3 and update our code and replace older classes with the new one. which is still fine for 3 pages but what if we had 100 or 150 or even more than that we had to make changes in each one of them. Dependency injection helps us to get rid of this problem by introducing an dependency injection container in between the classes and pages.

Page1

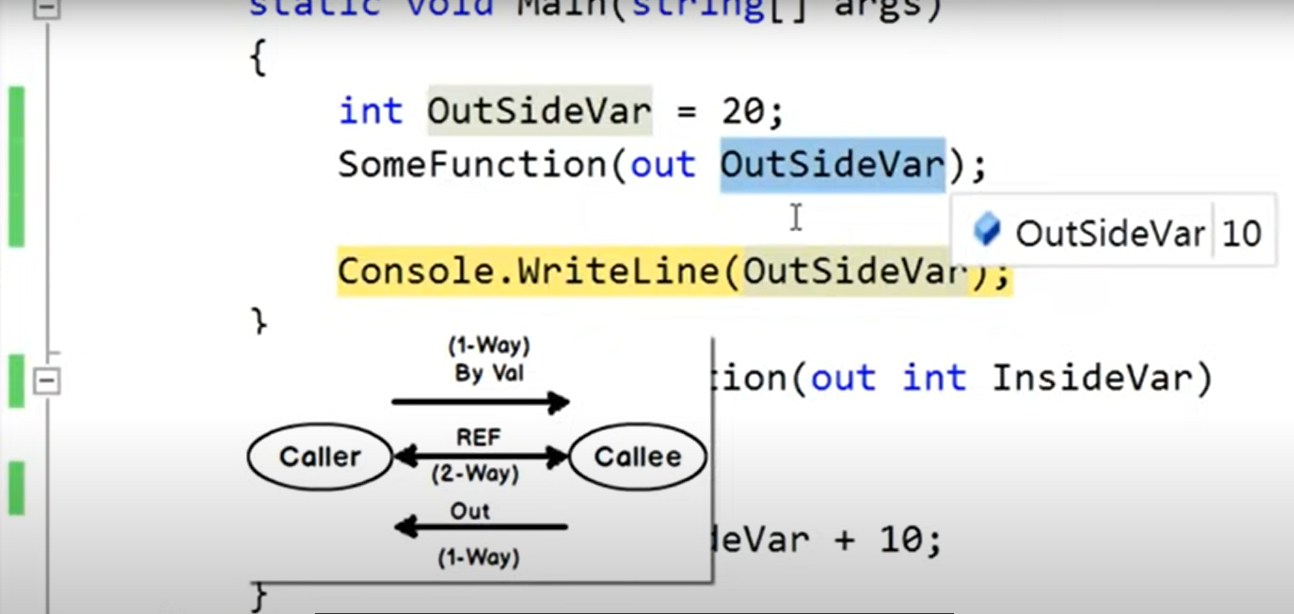
Page3

Page2

DI

Container

Q2. Difference between Ref vs out?

A2. Ref and out allows you to pass values of variable to functions by reference. But the only deference between these two is in ref you can pass the reference of the variable from the caller to the callee and you not need to initialize it in the callee. But in case of out even if you initialize the variable in the caller method that would be discarded and you will be required to initialize it in the callee.  
  


**Q3**. Array vs Array List

**A3.**

|  |  |
| --- | --- |
| Array | ArrayList |
| Strongly typed (it can store only one specific type of data) | Array list is not strongly typed and can store any type of data. |
| Can store fix number of element. | It is not fixed. |
| Performance Efficient | It is not performance efficient in comparison to array as it requires boxing and unboxing. |

**Q4.** Var vs dynamic

**A4.** Var and dynamic both are used for variable declaration but there are some differences between them. Like  
  
1. In case of Var type of variable is based on the assigned value, once the type of variable is inferred it cannot be change. Where as in dynamic variable type is inferred at run time and variable type can change during the execution of program.

2. Var is early binding and dynamics is late bindings which means error related to var will come during the compilation where as error related to dynamic will come during run time.   
  
Note: As error related to dynamic comes during run time, we should not prefer using dynamic over var.

Q5. Read only vs Const

A5. const and read-only, both are used to store constant values. In case of const we store values which are absolute constant. Which means once initialized with a value will not be changed again. Whereas read only variables can also not be changed after initialization but it is not required to initialize them during declaration, we can declare them and initialize them later using non static constructor.

Q6. Hash table vs dictionary

A6. Hash table and Dictionary in C# are used to store data in key-value pairs, but there are some important differences between them.

 **Hashtable**:

* A Hashtable **does not maintain the order** of the elements. It is an **unordered** collection of key-value pairs. When you iterate over a Hashtable, the order in which the elements are returned is not guaranteed.

 **Dictionary**:

* A Dictionary **maintains the order** of elements based on the insertion order.

**Type Safety (Loose vs. Tight Typing)**

* **Hashtable**:
  + Hashtable is **loosely typed**. This means it can store keys and values of any type, as it uses object as the type for both the keys and the values. You can store a string key with an int value, or a DateTime key with a double value. However, you would need to cast the values back to their original types when retrieving them.

Example:  
Hashtable hashtable = new Hashtable();

hashtable.Add("age", 30);

hashtable.Add("name", "John");

var age = (int)hashtable["age"]; // Needs casting

var name = (string)hashtable["name"]; // Needs casting

**Dictionary**:

* Dictionary<TKey, TValue> is **strongly typed**. You define the types for the key and the value when you declare the dictionary. This provides compile-time type checking and avoids the need for casting when retrieving values.

Dictionary<string, int> dictionary = new Dictionary<string, int>();

dictionary.Add("age", 30);

dictionary.Add("year", 2025);

int age = dictionary["age"]; // No casting needed Dictionary<string, int> dictionary = new Dictionary<string, int>();

dictionary.Add("age", 30);

dictionary.Add("year", 2025);

int age = dictionary["age"]; // No casting needed

**3. Performance**

* **Hashtable**:
  + Since Hashtable is loosely typed, it often involves boxing and unboxing operations when storing and retrieving value types. This can lead to additional overhead. In addition, because Hashtable works with object type, it might not be as efficient as a Dictionary in terms of performance, especially when storing value types.
* Dictionary  
  Dictionary<TKey, TValue> is more **efficient** than Hashtable because it uses generics and avoids the overhead of boxing/unboxing. Since the key and value types are known at compile time, it can be optimized by the compiler.

Q6. IEnumerable vs IQueryable

A6. I Enumerable and IQueryable both are interface of .NET collection. IQueryable Implements IEnumerable which means IQueryable can do everything that IEnumerable can do but it has some additional features. The major difference between IEnumerable and IQueryable is that IEnumerable fetches all the data from db and implement filters on user side, where as IQueryable implements filter on db side and fetches on the required part. IEnumerable is best to work with the collection which is not connected to database where as for the collection that is connected to database IQueryable is best to work with.

Q7. Interface vs Abstract Class

A7.

|  |  |
| --- | --- |
| Interface | Abstract Class |
| Interface is like a contract | Abstract class is a partial implementation. |
| It is about what to do. | It is about how to do. |
| I can have abstract members only. | It can have abstract as well as definition of the members. |
| Access modifier of the members is always public | Access modifier of members can be public, protected, private etc.. |
| It can not have constructor | It can have constructor. |
| A class can implement multiple interface. | Can not inherit multiple abstract classes. |

Q8. Method overloading vs overriding

A8.

| **Feature** | **Method Overloading** | **Method Overriding** |
| --- | --- | --- |
| **What is it?** | Same method name, different parameters (in same class) | Same method signature, different implementation (in child class) |
| **Polymorphism Type** | Compile-time (Static) | Run-time (Dynamic) |
| **Class Relationship** | Same class | Involves inheritance (base & derived classes) |
| **Signature** | Must change parameter type, number, or order | Must keep same signature as in base class |
| **Keyword Used** | No special keyword needed | Uses override in derived class, virtual in base |
| **When used** | To perform similar tasks in different ways | To change base class behavior in derived class |

Q8. Abstraction vs Encapsulation diff

A8.

Q9. What is constructor, why used, type?

A9. Constructor is a special method which automatically gets called on the time of object creation. Constructors are majorly used for initialization of object to set the default values at the time of object creation.

There are different types of Constructors:

* 1. Default Constructor: without any parameter.
  2. Parameterized Constructor: which receives parameters.
  3. Copy Constructor: Used to pass an object’s values to another object.
  4. Static Constructor: It is used to execute any peace of code that we want to execute before execution of any static member of the class.
  5. Private Constructor: It stops other classes to instantiate any class or derive from a class. Majorly used in singleton design pattern.

Q10. What is abstract class, why used, type

A10. An Abstract class is an in complete implementation of a class. It is used to provide a common base class that shares the common code between child classes. It can have both abstract as well as non-abstract methods in it. It forces derived classes to implement specific behavior.

Suppose there is a class Cat and a class Dog both the dog and cat have a common functionality like eat, drinkMilk, sleep etc. but they also have some different behavior also like cat has Meow and dog Bhow. So, to keep the code clean and futuristic we can create an abstract class called Animal and move the common functionality there and let the Dog and Cat class to derive the common functionality from there

What is inheritance, why used, type