

بسم رب الحسین (علیه السلام)

سوالات مصاحبه ای C#

## CLR

هسته‌ی مرکزی دات نت با عنوان CLR یا Common Language Runtime شناخته می‌شود.

این هسته شامل کدهای Just In Time Compiler و Garbage Collection می‌شود و رابط بین برنامه‌های دات نت و سیستم عامل است.

مهمنترین قسمت پلتفرم دات نت یا همون CLR در واقع ماشین مجازی یا Virtual Machine دات نت فریمورک هست که وظیفه مدیریت اجرای برنامه‌های دات نت رو بر عهده دارد. در این محیط مجازی در زمان اجرا فرایندی با نام JIT یا Just In Time Compilation کار کامپایل کدهای CIL به زبان ماشین رو انجام میده. CLR سرویسهای متعدد دیگه‌ای مثل مدیریت حافظه و garbage collection، مدیریت خطاها، مدیریت ثردها (Thread Management) و ... رو هم فراهم می‌کنه. CLR در واقع پیاده سازی بخشی از CLI به نام VES یا Virtual Execution System تو دات نت فریمورک هست.

بیانگر یک محیط اجرایی برای برنامه‌ها است و به عنوان یک لایه بین سیستم عامل و برنامه‌های نوشته شده با زبان های تحت Net. عمل می‌کند و از (CLS) Common Language Specification نیز پشتیبانی می‌کند. کار اصلی تبدیل کدهای مدیریت شده به کدهای خام و اصلی و سپس اجرای برنامه است. ویژگی کامپایل (JIT) just in time (MSIL) را در زمان اجرا و در صورت نیاز به کدهای خام و اصلی تبدیل می‌کند.

وقتی برنامه .Net به حالت اجرا می‌رود، control به سیستم عامل می‌رود و سپس سیستم عامل یک فرآیند برای بارگذاری CLR ایجاد می‌کند.

Ref : <https://www.c-sharpcorner.com/UploadFile/puranindia/C-Sharp-interview-questions/>

& Other Website and Article

## 1. What is C#?

C# is a computer programming language. C# was developed by Microsoft in 2000 to provide a modern general-purpose programming language that can be used to develop all kinds of software targeting various platforms including Windows, Web, and Mobile using just one programming language. Today, C# is one of the most popular programming languages in the world. Millions of software developers use C# to build all kinds of software.

C# is the primary language for building Microsoft .NET software applications. Developers can build almost every kind of software using C# including Windows UI apps, console apps, backend services, cloud APIs, Web services, controls and libraries, serverless applications, Web applications, native iOS and Android apps, AI and machine learning software, and blockchain applications.

C# with the help of Visual Studio IDE provides a rapid application development. C# is a modern, object-oriented, simple, versatile, and performance-oriented programming language. C# is developed based on the best features and use cases of several programming languages including C++, Java, Pascal, and SmallTalk.

C# syntaxes are like C++. .NET and the C# library is similar to Java. C# supports modern object-oriented programming language features including Abstraction, Encapsulation, Polymorphism, and Inheritance. C# is a strongly typed language and most types are inherited by the Object class.

C# supports concepts of classes and objects. Classes have members such as fields, properties, events, and methods. Here is a detailed article on C# and OOP.

C# is versatile, modern, and supports modern programming needs. Since its inception, C# language has gone through various upgrades

سی شارپ یک زبان برنامه نویسی کامپیوتری است. سی شارپ توسط مایکروسافت در سال ۲۰۰۰ برای ارائه یک زبان برنامه نویسی همه منظوره مدرن توسعه یافت که می تواند برای توسعه انواع نرم افزارهایی که پلتفرم های مختلف از جمله ویندوز، وب و موبایل را هدف قرار می دهند تنها با استفاده از یک زبان برنامه نویسی مورد استفاده قرار گیرد. امروزه سی شارپ یکی از محبوب ترین زبان های برنامه نویسی در جهان است. میلیون ها توسعه دهنده نرم افزار از سی شارپ برای ساخت انواع نرم افزار استفاده می کنند.

سی شارپ زبان اصلی ساخت نرم افزارهای دات نت مایکروسافت است. توسعه‌دهندگان می‌توانند تقریباً هر نوع نرم‌افزاری را با استفاده از سی شارپ بسازند، از جمله برنامه‌های رابط کاربری ویندوز، برنامه‌های کنسول، سرویس‌های پشتیبان، API‌های ابری، سرویس‌های وب، کنترل‌ها و کتابخانه‌ها، برنامه‌های بدون سورور، برنامه‌های کاربردی وب، برنامه‌های iOS و اندروید بومی، هوش مصنوعی و نرم‌افزارهای یادگیری ماشین، و برنامه‌های بلاک چین

سی شارپ با کمک Visual Studio IDE توسعه سریع برنامه را فراهم می‌کند. سی شارپ یک زبان برنامه نویسی مدرن، شی گرا، ساده، همه کاره و عملکرد گرا است. سی شارپ بر اساس بهترین ویژگی‌ها و موارد استفاده از چندین زبان برنامه نویسی از جمله C++, Pascal، Java و SmallTalk توسعه یافته است.

نحوه‌ای سی شارپ مانند سی پلاس پلاس هستند. دات نت و کتابخانه سی شارپ شبیه جاوا است. سی شارپ از ویژگی‌های زبان برنامه نویسی شی گرا مدرن از جمله Inheritance، Abstraction، Encapsulation و Polymorphism پشتیبانی می‌کند. سی شارپ یک زبان با تایپ قوی است و بیشتر انواع آن توسط کلاس Object به ارت می‌رسد.

سی شارپ از مفاهیم کلاس‌ها و اشیاء پشتیبانی می‌کند. کلاس‌ها دارای اعضایی مانند فیلد، ویژگی‌ها، رویدادها و متدها هستند. در اینجا یک مقاله مفصل در مورد C# و OOP وجود دارد.

سی شارپ همه کاره، مدرن است و نیازهای برنامه نویسی مدرن را پشتیبانی می‌کند. زبان سی شارپ از زمان آغاز به کار خود ارتقاهای مختلفی را پشت سر گذاشته است.

## ۲. What is an object in C#?

C# language is an object-oriented programming language. Classes are the foundation of C#. A class is a template that defines what a data structure will look like, and how data will be stored, managed, and transferred. A class has fields, properties, methods, and other members.

While classes are concepts, objects are real. Objects are created using class instances. A class defines the type of an object. Objects store real values in computer memory.

Any real-world entity which has some certain characteristics or that can perform some work is called an Object. This object is also called an *instance*, i.e. a copy of an entity in a programming language. Objects are instances of classes.

For example, we need to create a program that deals with cars. We need to create an entity for the car. Let's call it a class, Car. A car has four properties, i.e., model, type, color, and size.

To represent a car in programming, we can create a class, Car, with four properties, Model, Type, Color, and Size. These are called members of a class. A class has several types of members, constructors, fields, properties, methods, delegates, and events. A class member can be private, protected, and public. Since these properties may be accessed outside the class, these can be public.

An object is an instance of a class. A class can have as many instances as needed. For example, Honda Civic is an instance of Car. In real programming, Honda Civic is an object. Honda Civic is an instance of the class Car. The Model, Type, Color, and Size properties of Honda Civic are Civic, Honda, Red, 4 respectively. BMW 330, Toyota Carolla, Ford 350, Honda CR4, Honda Accord, and Honda Pilot are some more examples of objects of Car.

زبان سی شارپ یک زبان برنامه نویسی شی گرا است. کلاس ها پایه و اساس سی شارپ هستند. کلاس قالبی است که تعریف می کند ساختار داده چگونه خواهد بود و چگونه داده ها ذخیره، مدیریت و منتقل می شوند. یک کلاس دارای فیلدها، خصوصیات، متدها و سایر اعضاء است.

در حالی که کلاس ها مفاهیم هستند، اشیاء واقعی هستند. اشیاء با استفاده از نمونه های کلاس ایجاد می شوند. یک کلاس نوع یک شی را تعریف می کند. اشیا مقادیر واقعی را در حافظه کامپیوتر ذخیره می کنند.

هر موجودی در دنیای واقعی که دارای ویژگی های خاصی باشد یا بتواند کاری را انجام دهد، یک شی نامیده می شود. به این شی یک نمونه نیز می گویند، یعنی کپی یک موجودیت در یک زبان برنامه نویسی. اشیا نمونه هایی از کلاس ها هستند.

به عنوان مثال، ما باید برنامه ای ایجاد کنیم که با خودروها سروکار داشته باشد. ما باید یک موجودیت برای ماشین ایجاد کنیم. بیایید آن را یک کلاس، ماشین بنامیم. یک ماشین دارای چهار ویژگی است، یعنی مدل، نوع، رنگ و اندازه.

برای نمایش یک ماشین در برنامه نویسی، می توانیم یک کلاس Car با چهار ویژگی Model، Type، Color و Size ایجاد کنیم. به اینها اعضای یک کلاس گفته می شود. یک کلاس دارای چندین نوع عضو، سازنده، فیلد، ویژگی، متدهای نماینده و رویداد است. یک عضو کلاس می تواند خصوصی، محافظت شده و عمومی باشد. از آنجایی که این ویژگی ها ممکن است خارج از کلاس قابل دسترسی باشند، می توانند عمومی باشند.

یک شی نمونه ای از یک کلاس است. یک کلاس می تواند به تعداد مورد نیاز نمونه داشته باشد. به عنوان مثال، هوندا سیویک یک نمونه از Car است. در برنامه نویسی واقعی، هوندا سیویک یک شی است. هوندا سیویک نمونه ای از خودروهای کلاس است. ویژگی های مدل، نوع، رنگ و اندازه هوندا سیویک به ترتیب سیویک، هوندا، قرمز، ۴ است. بی ام و ۳۰، تویوتا کارولا، فورد ۳۵۰، هوندا آکورد و هوندا پایلوت چند نمونه دیگر از اشیاء خودرو هستند.

### ۳. What is Managed or Unmanaged Code?

The code that is developed outside of the .NET framework is known as unmanaged code.

"Applications that do not run under the control of the CLR are said to be unmanaged. Languages such as C or C++ or Visual Basic are unmanaged. The object creation, execution, and disposal of unmanaged code is directly managed by the programmers. If programmers write bad code, it may lead to memory leaks and unwanted resource allocations."

The .NET Framework provides a mechanism for unmanaged code to be used in managed code and vice versa. The process is done with the help of wrapper classes.

#### کد مدیریت شده Managed

"کد مدیریت شده کدی است که با استفاده از چارچوب دات نت و زبان های برنامه نویسی پشتیبانی شده آن مانند C# یا VB.NET ایجاد می شود. کد مدیریت شده مستقیماً توسط Common Language Runtime (CLR) یا (Runtime) اجرا می شود و چرخه حیات آن از جمله ایجاد شی، تخصیص حافظه و دفع شی توسط Runtime مدیریت می شود. هر زبانی که در دات نت فریم ورک نوشته شود، کد مدیریت شده است.«.

#### کد مدیریت نشده Unmanaged Code

کدهایی که خارج از چارچوب دات نت توسعه می یابند به عنوان کد مدیریت نشده شناخته می شوند. گفته می شود برنامه هایی که تحت کنترل CLR اجرا نمی شوند، مدیریت نشده اند. زبان هایی مانند C++ یا Visual Basic مدیریت نمی شوند.

ایجاد، اجرا و دفع کدهای مدیریت نشده مستقیماً توسط برنامه نویسان مدیریت می شود. اگر برنامه نویسان کد بد بنویستند، ممکن است منجر به نشت حافظه و تخصیص منابع ناخواسته شود.

چارچوب دات نت مکانیزمی برای استفاده از کدهای مدیریت نشده در کدهای مدیریت شده و بالعکس فراهم می کند. این فرآیند با کمک کلاس های wrapper انجام می شود.

## ε. What is Boxing and Unboxing in C#?

Unboxing و Boxing هر دو برای تبدیل نوع استفاده می‌شوند.

فرآیند تبدیل از نوع مقدار به نوع مرجع را Boxing می‌گویند. Boxing یک تبدیل ضمنی implicit است.

```
01. // Boxing
02. int anum = 123;
03. Object obj = anum;
04. Console.WriteLine(anum);
05. Console.WriteLine(obj);
```

The process of converting from a reference type to a value type is called unboxing.

```
01. // Unboxing
02. Object obj2 = 123;
03. int anum2 = (int)obj;
04. Console.WriteLine(anum2);
05. Console.WriteLine(obj);
```

## δ. What is the difference between a struct and a class in C#?

### Struct

The struct is a value type in C# and it inherits from System.Value Type.

Struct is usually used for smaller amounts of data.

Struct can't be inherited from other types.

A structure can't be abstract.

No need to create an object with a new keyword.

Do not have permission to create any default constructor.

### Class

The class is a reference type in C# and it inherits from the System.Object Type.

Classes are usually used for large amounts of data.

Classes can be inherited from other classes.

A class can be an abstract type.

We can create a default constructor.

## 7. What is the difference between Interface and Abstract Class in C#?

- A class can implement any number of interfaces but a subclass can at most use only one abstract class.
- An abstract class can have non-abstract methods (concrete methods) while in case of interface, all the methods have to be abstract.
- An abstract class can declare or use any variables while an interface is not allowed to do so.
- In an abstract class, all data members or functions are private by default while in an interface all are public, we can't change them manually.
- In an abstract class, we need to use abstract keywords to declare abstract methods, while in an interface we don't need to use that.
- An abstract class can't be used for multiple inheritance while the interface can be used as multiple inheritance.
- An abstract class use constructor while in an interface we don't have any type of constructor.

### OOPs interface vs abstract class

Interface	Abstract class
Interface support multiple implementations.	Abstract class does not support multiple inheritance.
Interface does not contain Data Member	Abstract class contains Data Member
Interface does not contain Constructors	Abstract class contains Constructors
An interface Contains only incomplete member (signature of member)	An abstract class Contains both incomplete (abstract) and complete member
An interface cannot have access modifiers by default everything is assumed as public	An abstract class can contain access modifiers for the subs, functions, properties
Member of interface can not be Static	Only Complete Member of abstract class can be Static

- Abstract classes are classes that cannot be instantiated ie. that cannot create an object. The interface is like an abstract class because all the methods inside the interface are abstract methods.
- Surprisingly, abstract classes can have both abstract and non-abstract methods but all the methods of an interface are abstract methods.
- Since abstract classes can have both abstract and non-abstract methods, we need to use the Abstract keyword to declare abstract methods. But in the interface, there is no such need.
- An abstract class has constructors while an interface encompasses none.

## V. What is enum in C#?

An enum is a value type with a set of related named constants often referred to as an enumerator list. The enum keyword is used to declare an enumeration. It is a primitive data type that is user-defined.

An enum type can be an integer (float, int, byte, double, etc.). But if you use it beside int it has to be cast.

An enum is used to create numeric constants in the .NET framework. All the members of enum are enum type. There must be a numeric value for each enum type.

The default underlying type of the enumeration element is int. By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1.

یک نوع مقدار با مجموعه‌ای از ثابت‌های نام‌گذاری شده مرتبط است که اغلب به عنوان فهرست شمارشگر شناخته می‌شوند. کلمه کلیدی enum برای اعلام شمارش استفاده می‌شود. این یک نوع داده اولیه است که توسط کاربر تعریف شده است.

نوع enum می‌تواند یک عدد صحیح (double, byte, int, float وغیره) باشد. اما اگر از آن در کنار int استفاده می‌کنید باید ریخته شود.

از enum برای ایجاد ثابت‌های عددی در چارچوب دات نت استفاده می‌شود. همه اعضای enum از نوع enum هستند. برای هر نوع enum باید یک مقدار عددی وجود داشته باشد.

نوع پیش‌فرض زیربنایی عنصر شمارش int است. به طور پیش‌فرض، اولین شمارشگر دارای مقدار 0 است و مقدار هر شمارشگر متوالی 1 افزایش می‌یابد.

| 1. `enum Dow {Sat, Sun, Mon, Tue, Wed, Thu, Fri};`

Some points about enum,

- Enums are enumerated data types in c#.
- Enums are not for the end-user, they are meant for developers.
- Enums are strongly typed constant. They are strongly typed, i.e. an enum of one type may not be implicitly assigned to an enum of another type even though the underlying value of their members is the same.
- Enumerations (enums) make your code much more readable and understandable.
- Enum values are fixed. Enum can be displayed as a string and processed as an integer.
- The default type is int, and the approved types are byte, sbyte, short, ushort, uint, long, and ulong.
- Every enum type automatically derives from System.Enum and thus we can use System.Enum methods on enums.
- Enums are value types and are created on the stack and not on the heap.

## **۸. What is the difference between "continue" and "break" statements in C#?**

Using break statement, you can 'jump out of a loop' whereas by using a continue statement, you can 'jump over one iteration' and then resume your loop execution.

با استفاده از دستور break، می‌توانید از یک حلقه خارج شوید، در حالی که با استفاده از عبارت continue، می‌توانید از یک تکرار پرش کنید و سپس اجرای حلقه خود را از سر بگیرید.

## **۹. What is the difference between constant and readonly in C#?**

Const is nothing but "constant", a variable of which the value is constant but at compile time. It's mandatory to assign a value to it. By default, a const is static and we cannot change the value of a const variable throughout the entire program.

Readonly is the keyword whose value we can change during runtime or we can assign it at run time but only through the non-static constructor.

چیزی نیست جز "constant"، متغیری که مقدار آن ثابت است اما در زمان کامپایل. تخصیص یک مقدار به آن الزامی است. به طور پیش فرض، یک const ثابت است و ما نمی توانیم مقدار متغیر const را در کل برنامه تغییر دهیم.

readonly کلمه کلیدی است که می توانیم مقدار آن را در زمان اجرا تغییر دهیم یا می توانیم آن را در زمان اجرا اختصاص دهیم اما فقط از طریق سازنده غیراستاتیک.

## **۱۰. What is the difference between ref and out keywords?**

The ref keyword passes arguments by reference. It means any changes made to this argument in the method will be reflected in that variable when control returns to the calling method.

The out keyword passes arguments by reference. This is very similar to the ref keyword.

Ref	Out
The parameter or argument must be initialized first before it is passed to ref.	It is not compulsory to initialize a parameter or argument before it is passed to an out.
It is not required to assign or initialize the value of a parameter (which is passed by ref) before returning to the calling method.	A called method is required to assign or initialize a value of a parameter (which is passed to an out) before returning to the calling method.
Passing a parameter value by Ref is useful when the called method is also needed to modify the pass parameter.	Declaring a parameter to an out method is useful when multiple values need to be returned from a function or method.
It is not compulsory to initialize a parameter value before using it in a calling method.	A parameter value must be initialized within the calling method before its use.
When we use REF, data can be passed bi-directionally.	When we use OUT data is passed only in a unidirectional way (from the called method to the caller method).
Both ref and out are treated differently at run time and they are treated the same at compile time.	
Properties are not variables, therefore it cannot be passed as an out or ref parameter.	

هر گاه در اعلان پارامترها از کلمات کلیدی **ref** و **out** استفاده نشود، به صورت **value** فرستاده می شود.

#### ارسال پارامتر به صورت : **ref**

وقتی متغیری را به صورت **ref** به یک تابع ارسال می کنیم ، مقدار متغیر ارسال نمی شود بلکه آدرس متغیر (خود اصلی آن) به بدنہ متده فرستاده می شود و هر تغییری درمتغیر محلی روی متغیر اصلی نیز اعمال می شود. به این نوع پارامترها ارجاعی می گویند.

#### پارامترهای **out**

پارامترهای **out**، پارامترهایی هستند که متغیرهایی که مقدار دهن اولیه نشده اند، را قبول می کنند. کلمه کلیدی **out** زمانی مورد استفاده قرار میگیرد که بخواهیم یک متغیر بدون مقدار را به متده ارسال کنیم. متغیر بدون مقدار اولیه، متغیری است که مقداری به آن اختصاص داده نشده است. در این حالت متده یک مقدار به متغیر می دهد. ارسال متغیر مقداردهی نشده به متده زمانی مفید است که شما بخواهید از طریق متده، متغیر را مقداردهی کنید. استفاده از کلمه کلیدی **out** باعث ارسال آرگومان به روش ارجاع می شود نه مقدار.

از کلمه کلیدی **out** برای پارامترهای متده استفاده شده است، بنابراین می توانند متغیرهای مقداردهی نشده را قبول کنند.

## 11. Can "this" be used within a static method?

We can't use 'this' in a static method because the 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 are called with the name of a class, not by instance, so we can't use this keyword in the body of static Methods. However, in the case of Extension Methods, we can use the parameters of the function.

The "this" keyword in C# 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.

ما نمی‌توانیم از «this» در یک متده استاتیک استفاده کنیم، زیرا کلمه کلیدی «this» یک مرجع به نمونه فعلی کلاس حاوی آن برمی‌گرداند. روش‌های استاتیک (یا هر عضو ایستا) به یک نمونه خاص تعلق ندارند. آنها بدون ایجاد نمونه ای از کلاس وجود دارند و با نام یک کلاس فراخوانی می‌شوند، نه به عنوان نمونه، بنابراین ما نمی‌توانیم از این کلمه کلیدی در بدنه Extension Methods ثابت استفاده کنیم. اما در مورد می‌توانیم از پارامترهای تابع استفاده کنیم.

کلمه کلیدی "this" در سی شارپ نوع خاصی از متغیر مرجع است که به طور ضمنی در هر سازنده و متده غیراستاتیک به عنوان اولین پارامتر از کلاس نوع تعریف شده در آن تعریف می‌شود.

## 12. What are Properties in C#?

C# properties are members of a C# class that provide a flexible mechanism to read, write or compute the values of private fields, in other words, by using properties, we can access private fields and set their values. Properties in C# are always public data members. C# properties use get and set methods, also known as accessors, to access and assign values to private fields.

### What are accessors?

The get and set portions or blocks of a property are called accessors. These are useful to restrict the accessibility of a property. The set accessor specifies that we can assign a value to a private field in a property. Without the set accessor property, it is like a readonly field. With the 'get' accessor we can access the value of the private field. In other words, it returns a single value. A Get accessor specifies that we can access the value of a field publically.

We have three types of properties: Read/Write, ReadOnly, and WriteOnly. Let's see each one by one.

خصوصیات سی شارپ اعضای یک کلاس سی شارپ هستند که مکانیزم انعطاف پذیری را برای خواندن، نوشتن یا محاسبه مقادیر فیلدهای خصوصی ارائه می دهند، به عبارت دیگر با استفاده از خواص می توانیم به فیلدهای خصوصی دسترسی پیدا کرده و مقادیر آنها را تنظیم کنیم. ویژگی ها در سی شارپ همیشه اعضای داده عمومی هستند. خصوصیات سی شارپ از متدهای `get` و `set` که به عنوان `Accessor` نیز شناخته می شوند، برای دسترسی و تخصیص مقادیر به فیلدهای خصوصی استفاده می کنند.

لوازم جانبی چیست؟

بخش ها یا بلوک های `get` and `set` یک ویژگی را `Accessor` می گویند. اینها برای محدود کردن دسترسی به یک ملک مفید هستند. مجموعه دسترسی مشخص می کند که می توانیم یک مقدار به یک فیلد خصوصی در یک ویژگی اختصاص دهیم. بدون ویژگی `set accessor`، مانند یک فیلد فقط خواندنی است. با دسترسی «`get`» می توانیم به مقدار فیلد خصوصی دسترسی پیدا کنیم. به عبارت دیگر، یک مقدار واحد را برمی گرداند. دسترسی دسترسی مشخص می کند که ما می توانیم به مقدار یک فیلد به صورت عمومی دسترسی داشته باشیم.

ما سه نوع ویژگی داریم: `WriteOnly` و `ReadOnly` و `Read/Write`. بیایید هر کدام را یکی یکی ببینیم.

## ۱۳. What are extension methods in C#?

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

An extension method is a special kind of static method, but they are called as if they were instance methods on the extended type.

### How to use extension methods?

An extension method is a static method of a static class, where the "this" modifier is applied to the first parameter. The type of the first parameter will be the type that is extended.

Extension methods are only in scope when you explicitly import the namespace into your source code with a using directive.

متدهایی هستند که متعلق به هیچگونه کلاسی نیستند و از طریق کلاس‌های دیگر اضافه می‌شوند. در واقع متدهای Extension همانطور که از نامشان پیداست متدهای اضافی هستند. میتوان متدهای Extension را به اینترفیس‌ها، ساختارها و کلاس‌ها در سی‌شارپ، اضافه کرد. این کار بدون تغییر و ویرایش ساختارها، کلاس‌ها و اینترفیس‌ها انجام می‌شود. متدهای Extension می‌توانند به کلاس‌های تعریف شده توسط برنامه نویس و یا کلاس‌های تعریف شده در دات نت فرمورک اضافه شوند.

متدهای توسعه امکان اضافی کردن کارایی‌های جدید به کلاسها، ساختارها یا اینترفیس‌هایی که کد آنها در دسترس نیست و یا امکان ارث بری از آنها وجود نداره رو میده!  
تو تعریف متدهای توسعه چند تا محدودیت داریم!  
اول اینکه متدها رو توی یک کلاس استاتیک بنویسیم.  
دوم برای اینکه به کامپایلر بگیم که این تابع، یک تابع توسعه هست باید تو اولین پارامتر ورودی از کلمه‌ی کلیدی this استفاده کنیم.  
یه نکته دیگه اینکه اگر یک تابع توسعه تعریف کردین ولی یک توسعه داخلی با الگوی مشابه وجود داشته باشه، اولویت با توسعه داخلی هستش.  
و نکته دیگه Event‌ها و عملگرها قابل توسعه نیستند!

Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type. Extension methods are static methods, but they're called as if they were instance methods on the extended type.

## ۱۴. What is the difference between the dispose and finalize methods in C#?

In finalize and dispose, both methods are used to free unmanaged resources.

### Finalize

- Finalize is used to free unmanaged resources that are not in use, like files, database connections in the application domain and more. These are resources held by an object before that object is destroyed.
- In the Internal process, it is called by Garbage Collector and can't be 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 that are not in use like files, database connections in the Application domain at any time.
- Dispose is explicitly called by manual user code.
- If we need to use the dispose method, we must implement that class via IDisposable interface.
- It belongs to IDisposable interface.
- Implement this when you are writing a custom class that will be used by other users.

در نهایی سازی و دفع، از هر دو روش برای آزادسازی منابع مدیریت نشده استفاده می شود.

## Finalize

برای آزاد کردن منابع مدیریت نشده ای که استفاده نمی شوند، مانند فایل ها، اتصالات پایگاه داده در دامنه برنامه و موارد دیگر استفاده می شود. اینها منابعی هستند که قبل از این رفتن آن شی، توسط یک شی نگهداری می شوند.

در فرآیند داخلی، توسط Garbage Collector فرآخوانی می شود و با کد کاربر یا هیچ سرویسی نمی توان آن را دستی نامید.

Mتعلق به کلاس System.Object Finalize است.

زمانی که منابع مدیریت نشده ای در کد خود دارید، آن را پیاده سازی کنید و مطمئن شوید که این منابع هنگام جمع آوری زباله آزاد می شوند.

## Dipose

همچنین برای آزاد کردن منابع مدیریت نشده ای استفاده می شود که مانند فایل ها، اتصالات پایگاه داده در دامنه برنامه در هر زمان استفاده نمی شوند.

به صراحت توسط کد کاربر دستی نامیده می شود.

اگر نیاز به استفاده از متده dispose داشته باشیم، باید آن کلاس را از طریق رابط IDisposable پیاده سازی کنیم.

این به رابط IDisposable تعلق دارد.

هنگامی که در حال نوشتمن یک کلاس سفارشی هستید که توسط سایر کاربران استفاده می شود، این را اجرا کنید.

## ۱۰. What is the difference between String and StringBuilder in C#?

StringBuilder and string are both used to string values, but both have many differences on the bases of instance creation and also in performance.

### String

A string is an immutable object. Immutable is when we create string objects in code so we cannot modify or change that object in any operations like insert new value, replace or append any value with the existing value in a string object. When we have to do some operations to change string simply it will dispose of the old value of string object and it will create a new instance in memory for hold the new value in a string object.

### Note

- It's an immutable object that holds a string value.
- Performance-wise, string is slow because it creates a new instance to override or change the previous value.
- String belongs to the System namespace.

string و StringBuilder هر دو برای مقادیر رشته استفاده می شوند، اما هر دو بر اساس ایجاد نمونه و همچنین در عملکرد تفاوت های زیادی دارند.

رشته یک شیء تغییرناپذیر است. Immutable زمانی است که ما اشیاء رشته ای را در کد ایجاد می کنیم، بنابراین نمی توانیم آن شی را در هیچ عملیاتی مانند درج مقدار جدید، جایگزینی یا اضافه کردن هر مقدار با مقدار موجود در یک شی رشته، تغییر یا تغییر دهیم. هنگامی که ما مجبور به انجام برخی عملیات برای تغییر رشته به سادگی هستیم، مقدار قدیمی شی رشته را از بین می برد و یک نمونه جدید در حافظه برای نگهداری مقدار جدید در یک شی رشته ایجاد می کند.

توجه داشته باشید

این یک شیء تغییرناپذیر است که دارای یک مقدار رشته است.  
از نظر عملکرد، رشته کند است زیرا یک نمونه جدید برای لغو یا تغییر مقدار قبلی ایجاد می کند.  
رشته متعلق به فضای نام سیستم است.

### StringBuilder

System.Text.StringBuilder is a mutable object which also holds the string value, mutable means once we create a System.Text.StringBuilder object. We can use this object for any operation like insert value in an existing string with insert

functions also replace or append without creating a new instance of System.Text.Stringbuilder for every time so it's using the previous object. That way, it works fast compared to the System.String. Let's see an example to understand System.Text.Stringbuilder.

### Note

- StringBuilder is a mutable object.
- Performance-wise StringBuilder is very fast because it will use the same instance of StringBuilder object to perform any operation like inserting a value in the existing string.
- StringBuilder belongs to System.Text.Stringbuilder namespace.

### StringBuilder

یک شیء قابل تغییر است که مقدار رشته را نیز در خود نگه می دارد، قابل تغییر یعنی زمانی که یک شی System.Text.Stringbuilder ایجاد می کنیم. ما می توانیم از این شی برای هر عملیاتی مانند مقدار درج در یک رشته موجود استفاده کنیم، با توابع درج، همچنین بدون ایجاد یک نمونه جدید از System.Text.Stringbuilder برای هر بار جایگزین یا اضافه می شود، بنابراین از شی قبلی استفاده می کند. به این ترتیب، در مقایسه با System.String سریع کار می کند. بیایید یک مثال برای درک System.Text.Stringbuilder بینیم.

توجه داشته باشید

StringBuilder یک شیء قابل تغییر است. StringBuilder از نظر عملکرد بسیار سریع است زیرا از همان نمونه شی StringBuilder برای انجام هر عملیاتی مانند درج یک مقدار در رشته موجود استفاده می کند. متعلق به فضای نام System.Text.Stringbuilder است.

## 17. What are delegates in C# and the uses of delegates?

A Delegate is an abstraction of one or more function pointers (as existed in C++; the explanation about this is out of the scope of this article). The .NET has implemented the concept of function pointers in the form of delegates. With delegates, you can treat a function as data. Delegates allow functions to be passed as parameters, returned from a function as a value and stored in an array. Delegates have the following characteristics:

- Delegates are derived from the System.MulticastDelegate class.

- They have a signature and a return type. A function that is added to delegates must be compatible with this signature.
- Delegates can point to either static or instance methods.
- Once a delegate object has been created, it may dynamically invoke the methods it points to at runtime.
- Delegates can call methods synchronously and asynchronously.

The delegate contains a couple of useful fields. The first one holds a reference to an object, and the second holds a method pointer. When you invoke the delegate, the instance method is called on the contained reference. However, if the object reference is null then the runtime understands this to mean that the method is a static method. Moreover, invoking a delegate syntactically is the exact same as calling a regular function. Therefore, delegates are perfect for implementing callbacks.

## Why Do We Need Delegates?

Historically, the Windows API made frequent use of C-style function pointers to create callback functions. Using a callback, programmers were able to configure one function to report back to another function in the application. So the objective of using a callback is to handle button-clicking, menu-selection, and mouse-moving activities. But the problem with this traditional approach is that the callback functions were not type-safe. In the .NET framework, callbacks are still possible using delegates with a more efficient approach. Delegates maintain three important pieces of information:

- The parameters of the method.
- The address of the method it calls.
- The return type of the method.

A delegate is a solution for situations in which you want to pass methods around to other methods. You are so accustomed to passing data to methods as parameters that the idea of passing methods as an argument instead of data might sound a little strange. However, there are cases in which you have a method that does something, for instance, invoking some other method. You do not know at compile time what this second method is. That information is available only at runtime, hence Delegates are the device to overcome such complications.

Delegate انتزاعی از یک یا چند نشانگر تابع است (همانطور که در C++ وجود دارد؛ توضیح در مورد این موضوع خارج از محدوده این مقاله است). دات نت مفهوم نشانگرهای تابع را در قالب نماینده پیاده سازی کرده است. با نماینده‌گان، می‌توانید یک تابع را به عنوان داده در نظر بگیرید. Delegates اجازه می‌دهد تا تابع به عنوان پارامتر ارسال شوند، از یک تابع به عنوان یک مقدار بازگردانده شوند و در یک آرایه ذخیره شوند. نماینده‌گان دارای ویژگی‌های زیر هستند:

- نماینده‌گان از کلاس System.MulticastDelegate مشتق شده‌اند.
- دارای امضا و نوع برگشت هستند. تابعی که به نماینده‌گان اضافه می‌شود باید با این امضا سازگار باشد.
- نماینده‌گان می‌توانند به هر دو روش ثابت یا نمونه اشاره کنند.
- هنگامی که یک شیء نماینده‌گی ایجاد شد، ممکن است به صورت پویا متدهایی را که در زمان اجرا به آنها اشاره می‌کند فراخوانی کند.
- نماینده‌گان می‌توانند متدها را به صورت همزمان و ناهمزمان فراخوانی کنند.

نماینده شامل چند فیلد مفید است. اولی ارجاع به یک شی دارد و دومی نشانگر متده را نگه می‌دارد. هنگامی که نماینده را فراخوانی می‌کنید، متده نمونه بر روی مرجع موجود فراخوانی می‌شود. با این حال، اگر مرجع شی تهی باشد، زمان اجرا این را به این معنی می‌داند که متده که متده یک روش ثابت است. علاوه بر این، فراخوانی یک نماینده به صورت نحوی دقیقاً مشابه فراخوانی یک تابع منظم است. بنابراین، نماینده‌گان برای اجرای callbacks عالی هستند.

چرا به نماینده‌گان نیاز داریم؟

از لحاظ تاریخی، API ویندوز مکرر از نشانگرهای تابع سبک C برای ایجاد تابع برگشت به تماس استفاده می‌کرد. با استفاده از پاسخ به تماس، برنامه نویسان توانستند یک تابع را پیکربندی کنند تا به عملکرد دیگری در برنامه گزارش دهد. بنابراین هدف از استفاده از پاسخ به تماس، مدیریت فعالیتهای کلیک دکمه، انتخاب منو و حرکت ماوس است. اما مشکل این رویکرد سنتی این است که تابع پاسخ به تماس از نظر نوع ایمن نبودند. در چارچوب دات نت، بازخوانی تماس با استفاده از نماینده‌گان با رویکرد کارآمدتر همچنان امکان پذیر است. نماینده‌گان سه اطلاعات مهم را حفظ می‌کنند:

- پارامترهای روش
- آدرس روشی که فراخوانی می‌کند.
- نوع برگشتی روش.

نماینده راه حلی برای موقعیت‌هایی است که می‌خواهید روش‌ها را به روش‌های دیگر منتقل کنید. شما آنقدر به انتقال داده‌ها به متدها به عنوان پارامتر عادت دارید که ممکن است ایده ارسال متدها به عنوان آرگومان به جای داده کمی عجیب به نظر برسد. با این حال، مواردی وجود دارد که در آن شما روشی دارید که کاری را انجام می‌دهد، به عنوان مثال، روش دیگری را فراخوانی می‌کند. شما در زمان کامپایل نمی‌دانید این روش دوم چیست. این اطلاعات فقط در زمان اجرا در دسترس دستگاهی برای غلبه بر چنین عوارضی هستند. از این رو Delegates هستند.

Delegate ها می‌توانند به یک متدهای اضافه کنند. این Delegate اشاره کنند.

Delegate ها انواعی هستند که مرجع یک متدهای خود ذخیره می‌کنند. همچنین می‌توانند رفتار هر متدهای کپی برداری کنند. برای تعریف یک delegate از کلمه کلیدی delegate استفاده می‌شود. تعریف یک delegate بسیار شبیه به تعریف Delegate است، با این تفاوت که متدهای دارد ولی delegate ندارد. دقیقاً مانند متدهای دارای نوع برگشتی و مجموعه‌ای از پارامترها هستند.

نحوه تعریف delegate نشان داده شده است:

```
delegate return Type DelegateName (dt param1, dt param2, ... dt paramN);
```

نحوه فراخوانی دلیگیت در مثال زیر آمده است.

```
delegate void ArithmeticDelegate(int num1, int num2);

static void Add(int x, int y)
{
    Console.WriteLine("Sum is {0}.", x + y);
}

static void Subtract(int x, int y)
{
    Console.WriteLine("Difference is {0}.", x - y);
}

static void Main()
{
    ArithmeticDelegate Operation;
    int num1, num2;

    Console.Write("Enter first number: ");
    num1 = Convert.ToInt32(Console.ReadLine());

    Console.Write("Enter second number: ");
    num2 = Convert.ToInt32(Console.ReadLine());

    if (num1 < num2)
    {
        Operation = new ArithmeticDelegate(Add);
    }
    else
    {
        Operation = new ArithmeticDelegate(Subtract);
    }

    Operation(num1, num2);
}
```

ضمیر اگر بخواهیم بیش از یک متدهای اضافه کنیم از += استفاده می‌کنیم.

## ۱۷. What are sealed classes in C#?

Sealed classes are used to restrict the inheritance feature of object-oriented programming. Once a class is defined as a sealed class, the class cannot be inherited.

In C#, the sealed modifier is used to define a class as sealed. In Visual Basic .NET the Not Inheritable keyword serves the purpose of the sealed class. If a class is derived from a sealed class then the compiler throws an error.

If you have ever noticed, structs are sealed. You cannot derive a class from a struct.

کلاس های مهر و موم شده برای محدود کردن ویژگی وراثت برنامه نویسی شی گرا استفاده می شوند. هنگامی که یک کلاس به عنوان یک کلاس مهر و موم شده تعریف می شود، کلاس نمی تواند ارث بری شود. در سی شارپ از اصلاح کننده sealed برای تعریف کلاس به صورت sealed استفاده می شود. اگر یک کلاس از یک کلاس مهر و موم شده مشتق شده باشد، کامپایلر یک خطا ایجاد می کند. اگر تا به حال متوجه شده اید، سازه ها مهر و موم شده اند. شما نمی توانید یک کلاس را از یک ساختار استخراج کنید.

کلاسی است که دیگر کلاس ها نمیتوانند از آن ارث بری کنند و به همین دلیل abstract نیز نمیشود.

## ۱۸. What are partial classes?

A partial class is only used to split the definition of a class in two or more classes in the same source code file or more than one source file. You can create a class definition in multiple files, but it will be compiled as one class at run time. Also, when you create an instance of this class, you can access all the methods from all source files with the same object.

Partial Classes can be created in the same namespace. It isn't possible to create a partial class in a different namespace. So use the "partial" keyword with all the class names that you want to bind together with the same name of a class in the same namespace.

استفاده از کلمه کلیدی `partial` به شما اجازه می دهد که یک کلاس را در چندین فایل جداگانه تعریف کنید. به عنوان مثال می توانید فیلد ها، خصیت ها و سازنده ها را در یک فایل و متد ها را در فایل دیگر قرار دهید. برای تعریف این نوع کلاس ها از کلمه کلیدی `partial` استفاده می شود. در مثال زیر نحوه تعریف یک کلاس `partial` در دو فایل جدا نشان داده شده است:

## 19. What is the difference between boxing and unboxing in C#?

Boxing and Unboxing are both used for type converting, but have some differences:

### Boxing

Boxing is the process of converting **a value type data type to the object** or to any interface data type which is implemented by this value type. When the CLR boxes a value means when CLR converting a value type to Object Type, it wraps the value inside a System.Object and stores it on the heap area in the application domain.

### Example

```
public void Function1()
{
    int i = 111;
    object o = i;//implicit Boxing
    Console.WriteLine(o);
}
```

### Unboxing

Unboxing is also a process that is used to extract the value type from the object or any implemented interface type. Boxing may be done implicitly, but unboxing has to be explicit by code.

### Example:

```
public void Function1()
{
    object o = 111;
    int i = (int)o;//explicit Unboxing
    Console.WriteLine(i);
}
```

The concept of boxing and unboxing underlies the C# unified view of the type system in which a value of any type can be treated as an object.

## ۲۰. What is `IEnumerable<T>` in C#?

`IEnumerable` is the parent interface for all non-generic collections in `System.Collections` namespace like `ArrayList`, `HastTable` etc. that can be enumerated. For the generic version of this interface as `IEnumerable<T>` which a parent interface of all generic collections class in `System.Collections.Generic` namespace like `List<T>` and more.

In `System.Collections.Generic.IEnumerable<T>` have only a single method which is `GetEnumerator()` that returns an `IEnumerator`. `IEnumerator` provides the power to iterate through the collection by exposing a `Current` property and `Move Next` and `Reset` methods if we don't have this interface as a parent so we can't use iteration by `foreach` loop or can't use that class object in our LINQ query.

رابطه والد برای همه مجموعه‌های غیر عمومی در فضای نام `System.Collections` مانند `HastTable`، `ArrayList` و غیره است که می‌توان آنها را بر شمرد. برای نسخه عمومی این رابطه به عنوان `IEnumerable<T>` که یک رابطه والد از همه مجموعه‌های عمومی در `System.Collections` نام عمومی مانند `List<T>` و بیشتر کلاس می‌کند.

در `T` در `System.Collections.Generic.IEnumerable<T>` فقط یک متده دارد که `Get Enumerator()` است که یک `IEnumerator` را بر می‌گرداند. قدرت تکرار را از طریق مجموعه با نمایش ویژگی `Current` و متدهای `Move Next` و `Reset` در صورتی که این رابطه را به عنوان والد نداریم، فراهم می‌کند، بنابراین نمی‌توانیم از تکرار توسط حلقه `foreach` استفاده کنیم یا نمی‌توانیم از آن شی کلاس در استفاده کنیم. پرسش LINQ ما.

## ۲۱. What is the difference between late binding and early binding in C#?

Early Binding and Late Binding concepts belong to polymorphism in C#. Polymorphism is the feature of object-oriented programming that allows a language to use the same name in different forms. For example, a method named `Add` can add integers, doubles, and decimals.

Polymorphism we have 2 different types to achieve that:

- Compile Time also known as Early Binding or Overloading.
- Run Time is also known as Late Binding or Overriding.

### Compile Time Polymorphism or Early Binding

In Compile time polymorphism or Early Binding, we will use multiple methods with the same name but different types of parameters, or maybe the number of parameters. Because of this, we can perform different-different tasks with the same method name in the same class which is also known as Method overloading.

See how we can do that in the following example:

```
class MyMath
{
    public int Sum(int val1, int val2)
    {
        return val1 + val2;
    }
    public string Sum(string val1, string val2)
    {
        return val1 + " " + val2;
    }
}
```

## Run Time Polymorphism or Late Binding

Run time polymorphism is also known as late binding. In Run Time Polymorphism or Late Binding, we can use the same method names with the same signatures, which means the same type or the same number of parameters, but not in the same class because the compiler doesn't allow for that at compile time. Therefore, we can use that bind at run time in the derived class when a child class or derived class object will be instantiated. That's why we call it Late Binding. We have to create my parent class functions as partial and in driver or child class as override functions with the override keyword.

### Example

```

class Class1
{
    public virtual string TestFunction()
    {
        return "Hello";
    }
}
class Class2 : Class1
{
    public override string TestFunction()
    {
        return "Bye Bye";
    }
}
class Program
{
    static void Main(string[] args)
    {
        Class2 obj = new Class2();
        Console.WriteLine(obj.TestFunction());
        Console.ReadLine();
    }
}

```

### چیست؟ Data Binding

مفهوم data binding مربوط به عناصری مانند Dataset, array, string است که می توانند مجموعه ای از داده ها را در خود نگهداری کنند. بعد از تعریف هر کدام از اشیا مذکور، آنها را به کنترل ها نسبت می دهیم. data binding می تواند از منابعی مانند دیتابیس، فایل XML، یا Script برای خواندن داده ها استفاده کند.

فرآیند بازیابی داده ها از منابع و اختصاص پویای آنها به یک کنترل است. بسته به عنصری که باید نمایش داده شود می توانید هر عنصر را به تگ زبان HTML و یا کنترل .NET نسبت دهید. این عناصر در ASP.NET کنترل های وب هستند.

### Compile Time Polymorphism or Early Binding

مفهوم آن برابر Overloading است.

### Run Time Polymorphism or Late Binding

مفهوم این یکی نیز برابر Override است.

## 11. What are the differences between IEnumerable and IQueryable?

Before we go into the differences, let's learn what the IEnumerable and IQueryable are.

### IEnumerable

Is the parent interface for all non-generic collections in System.Collections namespace like ArrayList, HastTable, etc. that can be enumerated. The generic version of this interface is IEnumerable<T>, which a parent interface of all generic collections class in System.Collections.Generic namespace, like List<> and more.

### IQueryable

As per MSDN, the IQueryable interface is intended for implementation by query providers. It is only supposed to be implemented by providers that also implement IQueryable<T>. If the provider does not also implement IQueryable<T>, the standard query operators cannot be used on the provider's data source.

The IQueryable interface inherits the IEnumerable interface so that if it represents a query, the results of that query can be enumerated. Enumeration causes the expression tree associated with an IQueryable object to be executed. The definition of "executing an expression tree" is specific to a query provider. For example, it may involve translating the expression tree to an appropriate query language for the underlying data source. Queries that do not return enumerable results are executed when the Execute method is called.

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.

## ۲۳. What happens if the inherited interfaces have conflicting method names?

If we implement multiple interfaces in the same class with conflict method names, we don't need to define all. In other words, we can say if we have conflict methods in the same class, we can't implement their body independently in the same class because of the same name and same signature. Therefore, we have to use the interface name before the method name to remove this method confiscation.

اگر چندین اینترفیس را در یک کلاس با نام متدهای تضاد پیاده سازی کنیم، نیازی نیست همه را تعریف کنیم. به عبارت دیگر، می‌توان گفت اگر متدهای تضاد در یک کلاس داشته باشیم، به دلیل نام و امضای یکسان، نمی‌توانیم بدنه آن‌ها را به طور مستقل در یک کلاس پیاده‌سازی کنیم. بنابراین، ما باید از نام رابط قبل از نام متد برای حذف این روش مصادره استفاده کنیم.

## ۲۴. What are the Arrays in C#?

In C#, an array index starts at zero. That means the first item of an array starts at the 0th position. The position of the last item on an array will total the number of items - 1. So if an array has 10 items, the last 10th item is in the 9th position.

In C#, arrays can be declared as fixed-length or dynamic.

A *fixed-length* array can store a predefined number of items.

A *dynamic array* does not have a predefined size. The size of a *dynamic array* increases as you add new items to the array. You can declare an array of fixed length or dynamic. You can even change a dynamic array to static after it is defined.

Let's take a look at simple declarations of arrays in C#. The following code snippet defines the simplest dynamic array of integer types that do not have a fixed size.

```
int[] intArray;
```

*As you can see from the above code snippet, the declaration of an array starts with a type of array followed by a square bracket ([]) and the name of the array.*

The following code snippet declares an array that can store 5 items only starting from index 0 to 4.

```
1. int[] intArray;  
2. intArray = new int[5];
```

The following code snippet declares an array that can store 100 items starting from index 0 to 99.

```
1. int[] intArray;  
2. intArray = new int[100];
```

در سی شارپ، یک شاخص آرایه از صفر شروع می شود. این بدان معناست که اولین مورد از یک آرایه از موقعیت ۰ شروع می شود. موقعیت آخرین آیتم در یک آرایه، مجموع تعداد آیتم ها را خواهد داشت - ۱. بنابراین اگر یک آرایه دارای ۱۰ آیتم باشد، آخرین مورد دهم در جایگاه نهم قرار دارد.

در سی شارپ، آرایه ها را می توان به صورت ثابت یا پویا اعلام کرد.

یک آرایه با طول ثابت می تواند تعداد از پیش تعریف شده ای از آیتم ها را ذخیره کند.

یک آرایه پویا اندازه از پیش تعریف شده ندارد. با اضافه کردن آیتم های جدید به آرایه، اندازه یک آرایه پویا افزایش می یابد. شما می توانید یک آرایه با طول ثابت یا پویا را اعلام کنید. حتی می توانید یک آرایه پویا را پس از تعریف به استاتیک تغییر دهید.

بیایید نگاهی به اعلان های ساده آرایه ها در سی شارپ بیندازیم. قطعه کد زیر ساده ترین آرایه پویا از انواع عدد صحیح را که اندازه ثابتی ندارند را تعریف می کند.

;int[] intArray

همانطور که از قطعه کد بالا می بینید، اعلان یک آرایه با یک نوع آرایه و سپس یک برآکت مربع ([]) و نام آرایه شروع می شود.

## ۲۵. What is the Constructor Chaining in C#?

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 a parent class Constructor implicitly by base keyword, so when you create an instance of the child class, it will call the parent's class Constructor. Without it, inheritance is not possible.

زنجیره سازی سازنده راهی برای اتصال دو یا چند کلاس در یک رابطه به عنوان وراثت است. در Constructor Chaining، هر سازنده کلاس فرزند به طور ضمنی با کلمه کلیدی پایه به سازنده کلاس والد نگاشت می شود، بنابراین وقتی نمونه ای از کلاس فرزند ایجاد می کنید، کلاس سازنده کلاس والد را فراخوانی می کند. بدون آن، ارث ممکن نیست.

## ۲۶. What's the difference between the Array.CopyTo() and Array.Clone()?

The Array.Clone() method creates a shallow copy of an array. A shallow copy of an Array copies only the elements of the Array, whether they are reference types or value types, but it does not copy the objects that the references refer to. The references in the new Array point to the same objects that the references in the original Array point to.

The CopyTo() static method of the Array class copies a section of an array to another array. The CopyTo method copies all the elements of an array to another one-dimension array. The code listed in Listing 9 copies contents of an integer array to an array of object types.

متدهایی که کپی کم عمق از یک آرایه ایجاد می کنند. یک کپی کم عمق از یک آرایه فقط عناصر آرایه را کپی می کند، خواه انواع مرجع باشند یا انواع مقادیر، اما اشیایی را که مراجع به آن ارجاع می دهند کپی نمی کند. ارجاعات موجود در آرایه جدید به همان اشیایی اشاره می کنند که مراجع در آرایه اصلی به آنها اشاره می کنند.

متدهای استاتیک CopyTo() از کلاس Array بخشی از یک آرایه را در آرایه دیگری کپی می کند. متدهای CopyTo تمام عناصر یک آرایه را در یک آرایه تک بعدی دیگر کپی می کند. کد فهرست شده در فهرست ۹ محتویات یک آرایه عدد صحیح را در آرایه ای از انواع شی کپی می کند.

## ۲۷. Can Multiple Catch Blocks be executed in C#?

We can use multiple catch blocks with a try statement. Each catch block can catch a different exception. The following code example shows how to implement multiple catch statements with a single try statement.

ما می توانیم از چندین بلوک catch با دستور try استفاده کنیم. هر بلوک catch می تواند استثناهای مختلفی را بگیرد.

```
1. using System;
2. class MyClient {
3.     public static void Main() {
4.         int x = 0;
5.         int div = 0;
6.         try {
7.             div = 100 / x;
8.             Console.WriteLine("Not executed line");
9.         } catch (DivideByZeroException de) {
10.             Console.WriteLine("DivideByZeroException");
11.         } catch (Exception ee) {
12.             Console.WriteLine("Exception");
13.         } finally {
14.             Console.WriteLine("Finally Block");
15.         }
16.         Console.WriteLine("Result is {0}", div);
17.     }
18. }
```

## ۲۸. What are Singleton Design Patterns and how to implement them in C#?

### What is a 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 the 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.

الگوی طراحی Singleton چیست؟

۱. اطمینان حاصل می کند که یک کلاس فقط یک نمونه دارد و یک نقطه دسترسی جهانی به آن فراهم می کند.

۲. Singleton کلاسی است که فقط اجازه می دهد یک نمونه از خودش ایجاد شود و معمولاً به آن نمونه دسترسی ساده می دهد.

۳. معمولاً Singleton اجازه نمی دهد هیچ پارامتری هنگام ایجاد نمونه مشخص شود، زیرا درخواست دوم یک نمونه با پارامتر متفاوت می تواند مشکل ساز باشد! (اگر برای همه درخواست ها با پارامتر یکسان باید به یک نمونه دسترسی داشت، الگوی کارخانه مناسبتر است.)

۴. راه های مختلفی برای پیاده سازی الگوی Singleton در سی شارپ وجود دارد. در زیر ویژگی های مشترک یک الگوی Singleton آورده شده است.

۰ یک سازنده واحد که خصوصی و بدون پارامتر است.

۰ کلاس مهر و موم شده است.

۰ یک متغیر ایستا که در صورت وجود ارجاع به نمونه ایجاد شده را دارد.

۰ یک وسیله ثابت عمومی برای دریافت ارجاع به نمونه ایجاد شده واحد، ایجاد یکی در صورت لزوم.

**Example of how to write code with Singleton:**

```
1. namespace Singleton {  
2.     class Program {  
3.         static void Main(string[] args) {  
4.             Calculate.Instance.ValueOne = 10.5;  
5.             Calculate.Instance.ValueTwo = 5.5;  
6.             Console.WriteLine("Addition : " + Calculate.I  
nstance.Addition());  
7.             Console.WriteLine("Subtraction : " + Calculat  
e.Instance.Subtraction());  
8.             Console.WriteLine("Multiplication : " + Calcu  
late.Instance.Multiplication());  
}
```

```
9.             Console.WriteLine("Division : " + Calculate.I
  nstance.Division());
10.            Console.WriteLine("\n-----
 \n");
11.            Calculate.Instance.ValueTwo = 10.5;
12.            Console.WriteLine("Addition : " + Calculate.I
  nstance.Addition());
13.            Console.WriteLine("Subtraction : " + Calculat
 e.Instance.Subtraction());
14.            Console.WriteLine("Multiplication : " + Calcu
 late.Instance.Multiplication());
15.            Console.WriteLine("Division : " + Calculate.I
  nstance.Division());
16.            Console.ReadLine();
17.        }
18.    }
19.    public sealed class Calculate {
20.        private Calculate() {}
21.        private static Calculate instance = null;
22.        public static Calculate Instance {
23.            get {
24.                if (instance == null) {
25.                    instance = new Calculate();
26.                }
27.                return instance;
28.            }
29.        }
30.        public double ValueOne {
31.            get;
32.            set;
33.        }
34.        public double ValueTwo {
35.            get;
36.            set;
37.        }
38.        public double Addition() {
39.            return ValueOne + ValueTwo;
40.        }
41.        public double Subtraction() {
42.            return ValueOne - ValueTwo;
43.        }
44.        public double Multiplication() {
45.            return ValueOne * ValueTwo;
46.        }
47.        public double Division() {
48.            return ValueOne / ValueTwo;
49.        }
50.    }
51.}
```

## ۲۹. Difference between Throw Exception and Throw Clause

The basic difference is that the Throw exception overwrites the stack trace. This makes it hard to find the original code line number that has thrown the exception.

Throw basically retains the stack information and adds to the stack information in the exception that it is thrown.

Let's see what it means to better understand the differences. I am using a console application to easily test and see how the usage of the two differ in their functionality.

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. namespace TestingThrowExceptions {
6.     class Program {
7.         public void ExceptionMethod() {
8.             throw new Exception("Original Exception occurred in ExceptionMethod");
9.         }
10.        static void Main(string[] args) {
11.            Program p = new Program();
12.            try {
13.                p.ExceptionMethod();
14.            } catch (Exception ex) {
15.                throw ex;
16.            }
17.        }
18.    }
19.}
```

Now run the code by pressing the F5 key and see what happens. It returns an exception and look at the stack trace.

تفاوت اساسی این است که استثنای Throw رد پشته را بازنویسی می کند. این امر یافتن شماره خط کد اصلی را که استثناء را ایجاد کرده است دشوار می کند.

اساساً اطلاعات پشته را حفظ می کند و در استثنایی که پرتاب می شود به اطلاعات پشته اضافه می کند. بیایید ببینیم که درک بهتر تفاوت ها به چه معناست. من از یک برنامه کنسول استفاده می کنم تا به راحتی آزمایش کنم و ببینم که استفاده از این دو در عملکرد آنها چقدر متفاوت است.

## ۳۰. What are Indexers in C#?

C# introduces a new concept known as Indexers which are used for treating an object as an array. The indexers are usually known as smart arrays in C#. They are not an essential part of object-oriented programming.

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

سی شارپ مفهوم جدیدی به نام **Indexers** را معرفی می کند که برای رفتار یک شی به عنوان یک آرایه استفاده می شود. ایندکس کننده ها معمولاً به عنوان آرایه های هوشمند در سی شارپ شناخته می شوند. آنها بخش اساسی برنامه نویسی شی گرا نیستند.

تعريف یک نمایه ساز به شما امکان می دهد کلاس هایی ایجاد کنید که به عنوان آرایه های مجازی عمل می کنند. نمونه هایی از آن کلاس با استفاده از عملگر دسترسی آرایه [] قابل دسترسی هستند.

اندیس‌گذارها (Indexer) با نام آرایه‌های هوشمند شناخته می‌شوند و به واسطه آن‌ها، امکان دسترسی به یک متغیر عضو فراهم خواهد شد. اندیس‌گذارها با استفاده از کلمه کلیدی «ایجاد می‌شوند و اعضای ایستا به حساب نمی‌آیند.

```
<return type> this[<parameter type> index]  
{  
    get{  
        // return the value from the specified index of an internal  
        collection  
    }  
    set{  
        // set values at the specified index in an internal  
        collection  
    }  
}
```

## ۳۱. What is a multicast delegate in C#?

Delegate is one of the base types in .NET. Delegate is a class that is used to create and invoke delegates at runtime.

A delegate in C# allows developers to treat methods as objects and invoke them from their code.

یک از Type های پایه در دات نت است Delegate کلاسی است که برای ایجاد و فراخوانی نماینده‌گان در زمان اجرا استفاده می‌شود.

یک نماینده در سی شارپ به توسعه دهنده‌گان این امکان را می‌دهد که متدها را به عنوان اشیا در نظر بگیرند و آنها را از کد خود فراخوانی کنند.

## ۳۲. Difference between the Equality Operator (==) and Equals() Method in C#

Both the == Operator and the Equals() method are used to compare two value type data items or reference type data items.

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. Let's see with some examples.

هر دو روش == و Equals() برای مقایسه دو آیتم داده نوع مقدار یا آیتم داده نوع مرجع استفاده می‌شوند. عملگر برابری (==) عملگر مقایسه است و متدهای Equals() محتويات یک رشته را مقایسه می‌کند. عملگر == هويت مرجع را مقایسه می‌کند در حالی که متدهای Equals() فقط محتويات را مقایسه می‌کند.

### ۳۴. What's the Difference between the Is and As operator in C#

#### "is" operator

In C# language, we use the "is" operator to check the object type. If two objects are of the same type, it returns true, else it returns false.

در زبان سی شارپ از عملگر "is" برای بررسی نوع شی استفاده می کنیم. اگر دو شی از یک نوع باشند، true را برمی گرداند، در غیر این صورت false را برمی گرداند.

#### "as" operator

The "as" operator behaves in a similar way as the "is" operator. The only difference is it returns the object if both are compatible with that type. Else it returns a null.

عملگر "as" به روشه مشابه عملگر "is" رفتار می کند. تنها تفاوت این است که اگر هر دو با آن نوع سازگار باشند، شی را برمی گرداند. در غیر این صورت یک عدد تهی برمی گرداند.

در حالت `Is a` کلاس ما به صورت مستقیم از یک کلاس دیگر مشتق می شود، اما در حالت `Has a`، کلاس ما یک کلاس دیگر را در خود جای داده است.

Teacher is a Person

Car Has a Player

### ۳۵. How to use Nullable<> Types in C#?

A nullable type is a data type that contains the defined data type or the null value.

This nullable type concept is not compatible with "var".

Any data type can be declared nullable type with the help of operator "?".

نوع nullable یک نوع داده است که حاوی نوع داده تعریف شده یا مقدار null باشد.  
این مفهوم نوع nullable با "var" سازگار نیست.

هر نوع داده ای را می توان با کمک عملگر "?", نوع nullable اعلام کرد.

```
1. int? i = null;
```

## ۳۵. What are Different Ways a Method can be Overloaded?

Method overloading is a way to achieve compile-time polymorphism where we can use a method with the same name but different signatures. For example, the following code example has a method volume with three different signatures based on the number and type of parameters and return values.

### Note

If we have a method that has two parameter object type and has the same name method with two integer parameters, when we call that method with int value, it will call that method with integer parameters instead of the object type parameters method.

راهی برای دستیابی به چند شکلی زمان کامپایل است که در آن می‌توانیم از روشی با نام مشابه اما امضاهای متفاوت استفاده کنیم.

مفهوم سربارگذاری (Overloading) متده در سی شارپ یعنی شرایطی که در آن متده با یک نام یکسان استفاده شود، اما بر اساس زمینه (Context) مورد استفاده، مقادیر مختلفی را جا به جا کند. تنها متده که در آن قابلیت سربارگذاری وجود ندارد، متده main() است. به منظور سربارگذاری متدها در سی شارپ، یکی از موارد زیر ضروری هستند:

تغییر تعداد پارامترهای یک متده

تغییر ترتیب پارامترهای یک متده

استفاده از نوعهای داده مختلف برای پارامترها

## ۳۶. What is an Object Pool in .Net?

Object Pooling in .NET allows objects to keep in the memory pool so the objects can be reused without recreating them. This article explains what object pooling is in .NET and how to implement object pooling in C#.

### What does it mean?

Object Pool is a container of objects that are ready for use. Whenever there is a request for a new object, the pool manager will take the request and it will be served by allocating an object from the pool.

## How does it work?

We are going to use the Factory pattern for this purpose. We will have a factory method, which will take care of the creation of objects. Whenever there is a request for a new object, the factory method will look into the object pool (we use Queue object). If there is any object available within the allowed limit, it will return the object (value object), otherwise, a new object will be created and give you back.

یک نگهدارنده است که درون خود اشیایی آماده استفاده دارد. در داخل این نگهدارنده، اشیایی که در زمان حال مورد استفاده قرار می‌گیرند و همچنین، مجموع تعداد اشیا موجود در Pool دنبال می‌شوند.

Object Pooling در دات نت به اشیاء اجازه می‌دهد تا در حوضه حافظه نگهداری شوند تا بتوان از آنها بدون ایجاد مجدد آنها استفاده مجدد کرد. این مقاله توضیح می‌دهد که ادغام آبجکت در دات نت چیست و چگونه می‌توان ادغام شی در سی شارپ را پیاده سازی کرد.

چه مفهومی داره؟

ظرفی از اشیا است که آماده استفاده هستند. هر زمان که درخواستی برای یک شی جدید وجود داشته باشد، مدیر استخراج درخواست را می‌گیرد و با تخصیص یک شی از استخراج، به آن سرویس داده می‌شود.

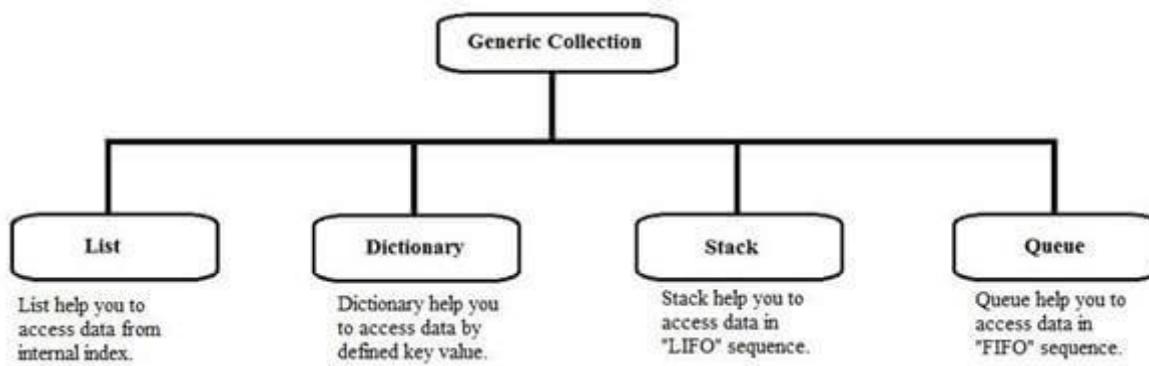
چگونه کار می‌کند؟

برای این منظور از الگوی Factory استفاده می‌کنیم. ما یک روش کارخانه ای خواهیم داشت که از ایجاد اشیاء مراقبت می‌کند. هر زمان که درخواستی برای یک شی جدید وجود داشته باشد، متند کارخانه به مخزن آبجکت نگاه می‌کند (ما از شی Queue استفاده می‌کنیم). اگر در محدوده مجاز، شیء موجود باشد، شیء را بر می‌گرداند (مقدار شی)، در غیر این صورت، یک شی جدید ایجاد می‌شود و به شما پس می‌دهد.

## ۴. What are Generics in C#?

Generics allow you to delay the specification of the data type of programming elements in a class or a method until it is actually used in the program. In other words, generics allow you to write a class or method that can work with any data type.

You write the specifications for the class or the method, with substitute parameters for data types. When the compiler encounters a constructor for the class or a function call for the method, it generates code to handle the specific data type.



Generic classes and methods combine reusability, type safety and efficiency in a way that their non-generic counterparts cannot. Generics are most frequently used with collections and the methods that operate on them. Version 2.0 of the .NET Framework class library provides a new namespace, `System.Collections.Generic`, that contains several new generic-based collection classes. It is recommended that all applications that target the .NET Framework 2.0 and later use the new generic collection classes instead of the older non-generic counterparts such as `ArrayList`.

در مجموعه‌های C# یا همان `Collection`‌های سی شارپ، تعریف هر نوعی از شی امکان‌پذیر است که با قانون اساسی اینمی نوع (Type-Safety) در تضاد است. به همین دلیل، از `Generic`‌ها استفاده می‌شود تا اینمی نوع در کدها رعایت شود. این کار با فراهم کردن امکان استفاده مجدد از الگوریتم‌های پردازش داده میسر می‌شود. `Generic`‌ها در سی شارپ یعنی به هیچ نوع داده خاصی پیوند وجود نداشته باشد.

های سی شارپ، `Boxing` و `Unboxing` و همچنین `Typecasting` (تغییر نوع) را کاهش می‌دهند. `Generic`‌ها همیشه بین دو علامت بزرگتر و کوچکتر یعنی در داخل `<>` قرار می‌گیرند.

## **#A. Describe Accessibility Modifiers in C#**

Access modifiers are keywords used to specify the declared accessibility of a member or a type.

Access modifiers are keywords used to specify the scope of accessibility of a member of a type or the type itself. For example, a public class is accessible to the entire world, while an internal class may be accessible to the assembly only.

### **Why use access modifiers?**

Access modifiers are an integral part of object-oriented programming. Access modifiers are used to implement the encapsulation of OOP. Access modifiers allow you to define who does or who doesn't have access to certain features.

In C# there are 6 different types of Access Modifiers:

Modifier	Description
public	There are no restrictions on accessing public members.
private	Access is limited to within the class definition. This is the default access modifier type if none is formally specified
protected	Access is limited to within the class definition and any class that inherits from the class
internal	Access is limited exclusively to classes defined within the current project assembly
protected internal	Access is limited to the current assembly and types derived from the containing class. All members in the current project and all members in derived class can access the variables.
private protected	Access is limited to the containing class or types derived from the containing class within the current assembly.

## ٣٩. What is a Virtual Method in C#?

A virtual method is a method that can be redefined in derived classes. A virtual method has an implementation in a base class as well as derived the class. It is used when a method's basic functionality is the same but sometimes more functionality is needed in the derived class. A virtual method is created in the base class that can be overridden in the derived class. We create a virtual method in the base class using the `virtual` keyword and that method is overridden in the derived class using the `override` keyword.

When a method is declared as a virtual method in a base class then that method can be defined in a base class and it is optional for the derived class to override that method. The overriding method also provides more than one form for a method. Hence, it is also an example of polymorphism.

When a method is declared as a virtual method in a base class and that method has the same definition in a derived class then there is no need to override it in the derived class. But when a virtual method has a different definition in the base class and the derived class then there is a need to override it in the derived class.

When a virtual method is invoked, the run-time type of the object is checked for an overriding member. The overriding member in the most derived class is called, which might be the original member if no derived class has overridden the member.

## **Virtual Method**

1. By default, methods are non-virtual. We can't override a non-virtual method.
  2. We can't use the virtual modifier with static, abstract, private or override modifiers.

متد مجازی متدی است که می‌تواند در کلاس‌های مشتق شده مجدداً تعریف شود. یک متد مجازی دارای یک پیاده‌سازی در کلاس پایه و همچنین مشتق شده از کلاس است. زمانی استفاده می‌شود که عملکرد پایه یک متد یکسان

باشد اما گاهی اوقات به عملکرد بیشتری در کلاس مشتق شده نیاز است. یک متده مجازی در کلاس پایه ایجاد می شود که می تواند در کلاس مشتق شده بازنویسی شود. یک متده مجازی در کلاس پایه با استفاده از کلمه کلیدی مجازی ایجاد می کنیم و آن متده در کلاس مشتق شده با استفاده از کلمه کلیدی `override` لغو می شود.

هنگامی که یک متده به عنوان یک متده مجازی در یک کلاس پایه اعلام می شود، آن متده را می توان در یک کلاس پایه تعریف کرد و برای کلاس مشتق شده اختیاری است که آن متده را لغو کند. روش `overriding` نیز بیش از یک فرم برای یک متده ارائه می دهد. از این رو، آن نیز نمونه ای از چندشکلی است.

وقتی یک متده به عنوان یک متده مجازی در یک کلاس پایه اعلان می شود و آن متده در یک کلاس مشتق شده همان تعریف را دارد، دیگر نیازی به لغو آن در کلاس مشتق شده نیست. اما زمانی که یک متده مجازی در کلاس پایه و کلاس مشتق شده تعریف متقابلی دارد، نیاز به لغو آن در کلاس مشتق شده وجود دارد.

هنگامی که یک متده مجازی فراخوانی می شود، نوع زمان اجرا شی برای یک عضو اصلی بررسی می شود. عضو اصلی در مشتق شده ترین کلاس فراخوانی می شود که اگر هیچ کلاس مشتق شده ای عضو را لغو نکرده باشد، ممکن است عضو اصلی باشد.

### روش مجازی

به طور پیش فرض، روش ها غیر مجازی هستند. ما نمی توانیم یک روش غیر مجازی را لغو کنیم. ما نمی توانیم از اصلاح کننده مجازی با اصلاح کننده های ایستا، انتزاعی، خصوصی یا لغو استفاده کنیم.

## ۴۰. What is the Difference between an Array and ArrayList in C#?

Here is a list of differences between the two:

Array	ArrayList
Array uses the Vector array to store the elements	ArrayList uses the LinkedList to store the elements.
Size of the Array must be defined until redim used( vb)	No need to specify the storage size.
Array is a specific data type storage	ArrayList can be stored everything as object.
No need to do the type casting	Every time type casting has to do.
It will not lead to Runtime exception	It leads to the Run time error exception.
Element cannot be inserted or deleted in between.	Elements can be inserted and deleted.
There is no built in members to do ascending or descending.	ArrayList has many methods to do operation like Sort, Insert, Remove, BinarySearch,etc..

آرایه (Array) مجموعه‌ای از متغیرهای مشابه به حساب می‌آید که تحت یک عنوان مشترک در کنار یکدیگر قرار داده می‌شوند. در صورتی که، `ArrayList` مجموعه‌ای از اشیا است که امکان شاخص‌گذاری یا همان ایندکس کردن آن‌ها به صورت تک به تک وجود دارد. با استفاده از `ArrayList` می‌توان به ویژگی‌های مختلفی، از جمله تخصیص حافظه پویا (Dynamic Memory Allocation)، افزودن، جستجو و مرتب‌سازی آیتم‌ها دسترسی داشت. به منظور درک بهتر تفاوت `ArrayList` با `Array`، برخی از نقاط تفاوت آن‌ها در ادامه فهرست شده است:

- زمان تعریف یک آرایه، اندازه آیتم‌ها ثابت است، بنابراین تخصیص حافظه نیز ثابت خواهد بود. اما در `ArrayList`، این اندازه به را می‌توان به صورت پویا اضافه کرد یا کاهش داد.
- آرایه به فضای نام `ArrayList` تعلق دارد، اما `array` متعلق به فضای نام `System` است.
- تمام آیتم‌های موجود در آرایه از یک نوع داده یکسان هستند؛ در حالی که مقادیر یک `ArrayList` می‌توانند از یک نوع داده یکسان یا از نوع داده‌های متفاوتی باشند.
- آرایه‌ها در سی شارپ امکان پذیرش مقادیر `Null` را ندارند، ولی `ArrayList` می‌تواند مقدار `Null` را بپذیرد.

## ۴۱. What are Value types and Reference types in C#?

In C#, data types can be of two types, value types, and reference types. Value type variables contain their object (or data) directly. If we copy one value type variable to another then we are actually making a copy of the object for the second variable. Both of them will independently operate on their values, Value type data types are stored on a stack and reference data types are stored on a heap.

In C#, basic data types include `int`, `char`, `bool`, and `long`, which are value types. Classes and collections are reference types.

در سی شارپ، انواع داده‌ها می‌توانند دو نوع باشند، نوع ارزش و نوع مرجع. متغیرهای نوع مقدار مستقیماً شی (یا داده) خود را در بر می‌گیرند. اگر یک متغیر نوع مقدار را به متغیر دیگری کپی کنیم، در واقع یک کپی از شی برای متغیر دوم ایجاد می‌کنیم. هر دوی آنها به طور مستقل بر روی مقادیر خود عمل می‌کنند، انواع داده نوع ارزش در یک پشته و انواع داده‌های مرجع در یک پشته ذخیره می‌شوند.

در سی شارپ، انواع داده های پایه شامل int، char، long و bool هستند که از انواع مقدار هستند. کلاس ها و مجموعه ها انواع مرجع هستند.

### Value type

Value types are generally (not always) stored on the stack and are passed by copying.

The way in which a variable assignment works differs between reference and value types.

انواع مقادیر معمولاً (نه همیشه) در پشته ذخیره می شوند و با کپی کردن منتقل می شوند.  
روشی که یک انتساب متغیر کار می کند بین انواع مرجع و مقدار متفاوت است.

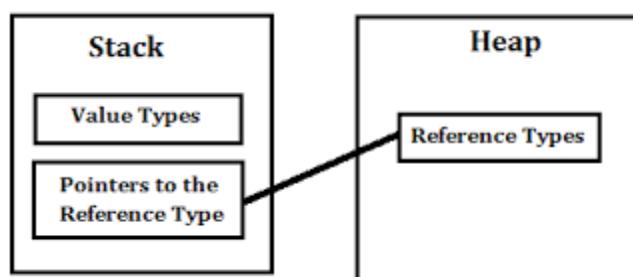
### Reference Type

A value type is basically stored on the heap and passed by creating a reference.

یک نوع مقدار اساساً روی پشته ذخیره می شود و با ایجاد یک مرجع ارسال می شود.

## تفاوت Reference type و Value type

در یک برنامه دو نوع حافظه Stack و Managed Heap وجود دارد. Stack حافظه کمتری دارد اما سرعت کار کردن با آن بالاست و managed heap حافظه بیشتری دارد و سرعت آن یک مقدار پایین تر است.



وقتی یک متغیر از نوع value type معرفی می کنیم، قسمتی از حافظه stack گرفته می شود و مقدار آن متغیر نیز در stack ذخیره می شود. برای مثال وقتی یک متغیر از نوع int معرفی می کنیم، چهار بایت از حافظه stack گرفته می شود و مقدار آن نیز در stack نگهداری می شود. اما وقتی یک reference type معرفی می کنیم، یک قسمت از حافظه stack گرفته می شود که داخل آن یک آدرس نگهداری می شود و این آدرس به قسمتی از managed heap اشاره می کند. مقدار آن نیز در managed heap نگهداری می شود.

در نتیجه stack reference type می شوند و برای آنها، آدرس در stack مقدار در managed heap نگهداری می شود.

یکی از امکاناتی که دات نت در اختیار ما قرار می دهد مدیریت حافظه از طریق garbage collector به این صورت که در managed heap متغیرها را بررسی می کند و اگر به ازای آنها در stack آدرسی وجود نداشته باشد حافظه را آزاد می کند.

انواعی که در سی شارپ به عنوان value type می شوند عبارتند از:

انواع عددی:

sbyte, short, int, long

انواع اعشاری:

float, double, decimal

انواع داده های دو وضعیتی که true و false می باشد و بعد از آن کاراکترها هستند.

در مقابل reference type پیش فرض سی شارپ که اولین آن ها object می باشد. تمام کلاس هایی که در سی شارپ تعریف می کنیم از یک کلاس به نام object ارث بری می کنند.

یکی دیگر از reference type های پرکاربرد، رشته ها می باشند که به کمک کلمه کلیدی string تعریف می شوند. اصلاحاً به آن ها immutable گفته می شود. به متغیرهایی که مقدار آن ها می تواند تغییر کنند mutable متغیرهای گفته می شود. اما در مقابل متغیرهایی هستند که اگر بخواهیم مقدار آن ها را عوض کنیم باید ماهیت آن ها را عوض کنیم. به این نوع متغیرها immutable گفته می شود. متغیرها از نوع رشته نیز یکی از آنها می باشد.

برای مثال وقتی یک متغیر string تعریف می کنیم، آدرس آن در stack و مقدار آن در heap نگهداری می شود. اگر من بخواهم مقدار متغیر تعریف شده را تغییر دهم، یک خانه جدید در heap ایجاد می شود و مقدار جدید را نگهداری می کند و آدرس قبل در stack به مقدار جدید اشاره می کند. بنابراین یک متغیر جدید ایجاد می شود و مقدار جدید در آن قرار می گیرد.

## ۴۲. What is Serialization in C#?

Serialization in C# is the process of converting an object into a stream of bytes to store the object to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed. The reverse process is called deserialization.

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).

سربیال سازی در سی شارپ فرآیند تبدیل یک شی به جریانی از بایت ها برای ذخیره شی در حافظه، پایگاه داده یا فایل است. هدف اصلی آن این است که وضعیت یک شی را ذخیره کند تا بتوان در صورت نیاز آن را بازسازی کرد. فرآیند معکوس، deserialization نامیده می شود.

سه نوع سربیال سازی وجود دارد،

سربیال سازی باینری (داده های شی خود را در قالب باینری ذخیره کنید).

سربیال سازی صابون (داده های شی خود را در قالب باینری ذخیره کنید؛ عمدتاً در ارتباطات مرتبط با شبکه استفاده می شود).

XmlSerialization (داده های شی خود را در یک فایل XML ذخیره کنید).

زمانی که لازم باشد یک شی از طریق یک شبکه منتقل شود، لازم است شی به جریانی از بایت ها تبدیل شود. به فرآیندی که در آن یک شی به جریانی از بایت ها تبدیل می شود، پیاپی سازی (Derialize) یا ISerialize (Serialization) می گویند. به منظور تسلیسل سازی (Serialize) کردن یک شی، باید رابط De-Serialize (Deserialization) را پیاده سازی کند. معکوس این فرآیند، یعنی ساخت یک شی با کمک جریانی از بایت ها، نامگذاری شده است.

<https://www.c-sharpcorner.com/article/serializing-objects-in-C-Sharp/>

## ۴۴. What is a Jagged Array in C#?

A jagged array is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes. A jagged array is sometimes called an "array of arrays."

A special type of array is introduced in C#. A Jagged Array is an array of an array in which the length of each array index can differ.

### Example

```
1. int[][] jagArray = new int[5][];
```

In the above declaration, the rows are fixed in size. But columns are not specified as they can vary.

Declaring and initializing a jagged array.

```
1. int[][] jaggedArray = new int[5][];
2. jaggedArray[0] = new int[3];
3. jaggedArray[1] = new int[5];
4. jaggedArray[2] = new int[2];
5. jaggedArray[3] = new int[8];
6. jaggedArray[4] = new int[10];
7. jaggedArray[0] = new int[] { 3, 5, 7, };
8. jaggedArray[1] = new int[] { 1, 0, 2, 4, 6 };
9. jaggedArray[2] = new int[] { 1, 6 };
10. jaggedArray[3] = new int[] { 1, 0, 2, 4, 6, 45, 67, 78 };

11. jaggedArray[4] = new int[] { 1, 0, 2, 4, 6, 34, 54, 67, 8
    7, 78 };
```

آرایه دندانه دار آرایه ای است که عناصر آن آرایه هستند. عناصر یک آرایه دندانه دار می توانند ابعاد و اندازه های مختلف داشته باشند. یک آرایه ناهموار گاهی اوقات "آرایه آرایه ها" نامیده می شود.

نوع خاصی از آرایه در سی شارپ معرفی شده است. آرایه دندانه دار آرایه ای از یک آرایه است که در آن طول هر شاخص آرایه می تواند متفاوت باشد.

در سی شارپ، آن دسته از آرایه هایی که عناصر آن هم خود از نوع آرایه هستند، با نام «آرایه های دندانه دار» (Jagged Arrays) شناخته می شوند. عناصر آرایه دندانه دار می توانند در ابعاد و اندازه های مختلفی باشند. همچنین می توان این نوع از آرایه ها را «آرایه ای از آرایه ها» نیز خطاب کرد.

آرایه های دندانه ای دقیقا همان آرایه های دو بعدی است، با این تفاوت که تعداد ستون ها می تواند در هر سطر متغیر بوده و با توجه به نیاز ما در حافظه خانه اشغال می شود.

نکته : برخلاف آرایه ی دوبعدی که در بین دو براکت تنها یک ویرگول گذاشته میشد که نمایانگر آرایه ی دو بعدی بود، در آرایه های دندانه ای از دو براکت جداگانه استفاده می شود.

## ۴۰. What is Multithreading with .NET?

Multithreading allows a program to run multiple threads concurrently. This article explains how multithreading works in .NET. This article covers the entire range of threading areas from thread creation, race conditions, deadlocks, monitors, mutexes, synchronization and semaphores and so on.

The real usage of a thread is not about a single sequential thread, but rather using multiple threads in a single program. Multiple threads running at the same time and performing various tasks are referred to as Multithreading. A thread is considered to be a lightweight process because it runs within the context of a program and takes advantage of the resources allocated for that program.

A single-threaded process contains only one thread while a multithreaded process contains more than one thread for execution.

## ۴۱. What are Anonymous Types in C#?

Anonymous types allow us to create new types without defining them. This is a way of defining read-only properties in a single object without having to define each type explicitly. Here, Type is generated by the compiler and is accessible only for the current block of code. The type of properties is also inferred by the compiler.

We can create anonymous types by using "new" keyword together with the object initializer.

انواع ناشناس به ما اجازه می دهد تا بدون تعریف انواع جدید ایجاد کنیم. این روشی برای تعریف ویژگی های فقط خواندنی در یک شی واحد بدون نیاز به تعریف صریح هر نوع است. در اینجا Type توسط کامپایلر تولید می شود و فقط برای بلوک فعلی کد قابل دسترسی است. نوع خواص نیز توسط کامپایلر استنباط می شود.

ما می توانیم انواع ناشناس را با استفاده از کلمه کلیدی "جدید" به همراه مقداردهی اولیه شی ایجاد کنیم.

## ۵۷. What is a Hashtable in C#?

A Hashtable is a collection that stores (Keys, Values) pairs. Here, the Keys are used to find the storage location and is immutable and cannot have duplicate entries in a Hashtable. The .Net Framework has provided a Hash Table class that contains all the functionality required to implement a hash table without any additional development. The hash table is a general-purpose dictionary collection. Each item within the collection is a DictionaryEntry object with two properties: a key object and a value object. These are known as Key/Value. When items are added to a hash table, a hash code is generated automatically. This code is hidden from the developer. Access to the table's values is achieved using the key object for identification. As the items in the collection are sorted according to the hidden hash code, the items should be considered to be randomly ordered.

### The Hashtable Collection

The Base Class libraries offer a Hashtable Class that is defined in the System.Collections namespace, so you don't have to code your own hash tables. It processes each key of the hash that you add every time and then uses the hash code to look up the element very quickly. The capacity of a hash table is the number of elements the hash table can hold. As elements are added to a hash table, the capacity is automatically increased as required through reallocation. It is an older .Net Framework type.

### Declaring a Hashtable

The Hashtable class is generally found in the namespace called System.Collections. So to execute any of the examples, we have to add using System.Collections; to the source code. The declaration for the Hashtable is:

```
1. Hashtable HT = new Hashtable ();
```

Hashtable

در فضای نام System.Collections کالکشن Hashtable وجود دارد. این کالکشن مقادیر را به صورت- key ذخیره می کند و با محاسبه کد هش مربوط به هر key ، عملیات جستجو در خود را بهینه سازی می کند.

Hashtable

از متده Add() برای افزودن یک آیتم با مقادیر key و value به Hashtable استفاده می شود. مقادیر Key و value می توانند از هر نوعی باشند. Key نمی تواند null باشد ، در حالی که value می تواند مقدار null را بپذیرد.

اما باید توجه داشت که در اصل داده شما در Value ذخیره نمی شود. بلکه یک مقدار هش شده هست! که براساس یک الگوریتم هش انجام می گیرد. که هش شده مقدار شماست.

## ۴۸. What is LINQ in C#?

LINQ stands for Language Integrated Query. LINQ is a data querying methodology that provides querying capabilities to .NET languages with a syntax similar to a SQL query.

LINQ has a great power of querying on any source of data. The data source could be collections of objects, database or XML files. We can easily retrieve data from any object that implements the `IEnumerable<T>` interface.

### Advantages of LINQ

1. LINQ offers an object-based, language-integrated way to query over data no matter where that data came from. So through LINQ, we can query a database and XML as well as collections.
2. Compile-time syntax checking.

It allows you to query collections like arrays, enumerable classes, etc... in the native language of your application, like in VB or C# in much the same way you would query a database using SQL.

LINQ مخفف عبارت Language Integrated Query است. LINQ یک متدولوژی پرس و جوی داده است که قابلیت های پرس و جو را برای زبان های دات نت با نحوی شبیه به پرس و جوی SQL فراهم می کند.

LINQ قدرت زیادی در پرس و جو در هر منبع داده ای دارد. منبع داده می تواند مجموعه ای از اشیا، پایگاه داده یا فایل های XML باشد. ما به راحتی می توانیم داده ها را از هر شی که رابط `IEnumerable<T>` را پیاده سازی می کند، بازیابی کنیم.

### مزایای LINQ

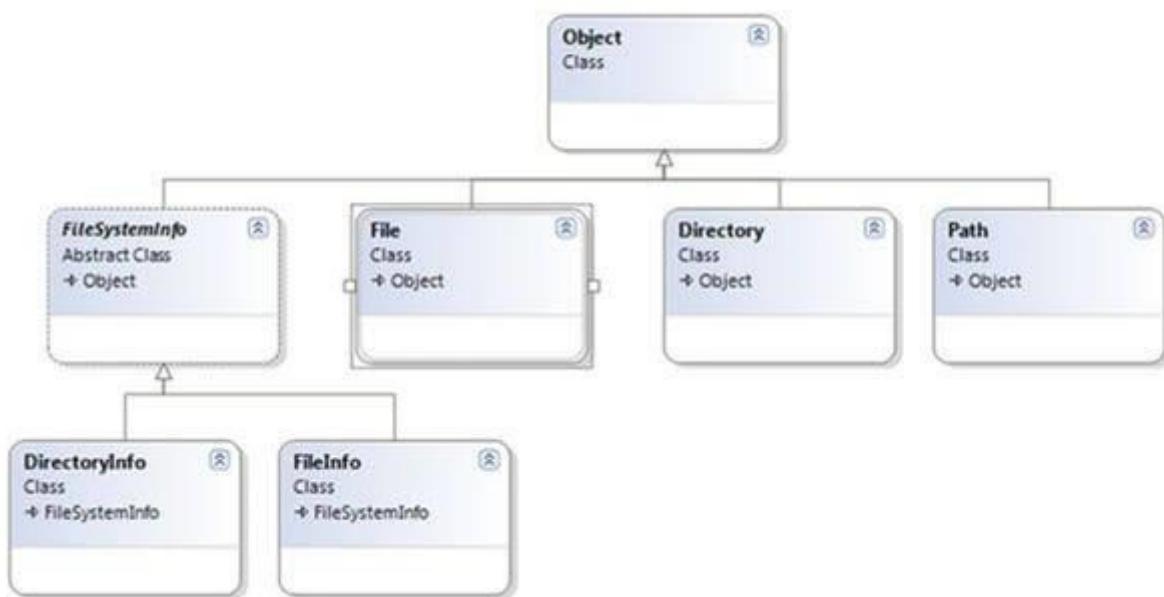
LINQ روشی مبتنی بر شی و زبان یکپارچه برای پرس و جو روی داده ها بدون توجه به اینکه آن داده ها از کجا آمده اند، ارائه می دهد. بنابراین از طریق LINQ می توانیم از پایگاه داده و XML و همچنین مجموعه ها پرس و جو کنیم.

### چک کردن نحو زمان کامپایل

این به شما امکان می دهد مجموعه هایی مانند آرایه ها، کلاس های شمارش پذیر، و غیره را به زبان مادری برنامه خود، مانند VB یا C#، به همان روشی که از پایگاه داده با استفاده از SQL پرس و جو می کنید، جستجو کنید.

## ۴۹. What is File Handling in C#.Net?

The System.IO namespace provides four classes that allow you to manipulate individual files, as well as interact with a machine directory structure. The Directory and File directly extend System.Object and supports the creation, copying, moving and deletion of files using various static methods. They only contain static methods and are never instantiated. The FileInfo and DirectoryInfo types are derived from the abstract class FileSystemInfo type and they are typically employed for obtaining the full details of a file or directory because their members tend to return strongly typed objects. They implement roughly the same public methods as a Directory and a File but they are stateful and members of these classes are not static.



فضای نام System.IO چهار کلاس را فراهم می کند که به شما امکان می دهد فایل های فردی را دستکاری کنید و همچنین با ساختار دایرکتوری ماشین تعامل داشته باشید. **File** و **Directory** به طور مستقیم **System.Object** را گسترش می دهند و از ایجاد، کپی، انتقال و حذف فایل ها با استفاده از روش های مختلف ثابت پشتیبانی می کند. آنها فقط شامل روش های ایستاده هستند و هرگز نمونه سازی نمی شوند. انواع **FileInfo** و **DirectoryInfo** از نوع کلاس انتزاعی **FileSystemInfo** مشتق شده اند و معمولاً برای به دست آوردن جزئیات کامل یک فایل یا دایرکتوری استفاده می شوند، زیرا اعضای آنها تمایل دارند اشیاء تایپ شده قوی را برگردانند. آنها تقریباً همان متدهای عمومی را مانند **Directory** و **File** پیاده سازی می کنند، اما حالتی هستند و اعضای این کلاس ها ثابت نیستند.

## ۰۰. What is Reflection in C#?

Reflection is the process of runtime type discovery to inspect metadata, CIL code, late binding, and self-generating code. At the 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.

Reflection typically is used to dump out the loaded assemblies list, their reference to inspect methods, properties etcetera. Reflection is also used in the external disassembling tools such as Reflector, Fxcop, and NUnit because .NET tools don't need to parse the source code similar to C++.

## Metadata Investigation

The following program depicts the process of reflection by creating a console-based application. This program will display the details of the fields, methods, properties, and interfaces for any type within the mscorelib.dll assembly. Before proceeding, it is mandatory to import "System.Reflection".

Here, we are defining a number of static methods in the program class to enumerate fields, methods, and interfaces in the specified type. The static method takes a single "System.Type" parameter and returns void.

در سی شارپ به منظور استخراج «فراداده» (Metadata) از نوعهای داده، در حین زمان اجرا به کار می‌رود. برای اضافه کردن Reflection در فریم ورک .NET، می‌توان به سادگی از فضای نام System.Reflection نام داده استفاده کرد. بدین طریق، می‌توان هر نوع خاصی از داده‌ها را بازیابی کرد. در ادامه انواع این داده‌های قابل بازیابی فهرست شده‌اند:

اسمبلی (Assembly)  
ماژول (Module)  
نوع شمارشی (Enum)  
MethodInfo  
ConstructorInfo  
MemberInfo  
ParameterInfo  
نوع (Type)  
FieldInfo  
EventInfo  
 PropertyInfo

## Encapsulation

Wrapping up a data member and a method together into a single unit (in other words class) is called Encapsulation.

Encapsulation is like enclosing in a capsule. That is enclosing the related operations and data related to an object into that object.

Encapsulation is like your bag in which you can keep your pen, book etcetera. It means this is the property of encapsulating members and functions.

Encapsulation means hiding the internal details of an object, in other words how an object does something.

Encapsulation prevents clients from seeing its inside view, where the behaviour of the abstraction is implemented.

Encapsulation is a technique used to protect the information in an object from another object.

Hide the data for security such as making the variables private, and expose the property to access the private data that will be public.

So, when you access the property you can validate the data and set it.

کیپوله کردن (تلفیق داده ها با یکدیگر) یا مخفی کردن اطلاعات فرایندی است که طی آن اطلاعات حساس یک موضوع از دید کاربر مخفی می شود و فقط اطلاعاتی که لازم باشد برای او نشان داده میشود. وقتی که یک کلاس تعریف می کنیم، معمولاً تعدادی اعضای داده ای (فیلید) برای ذخیره مقادیر مربوط به شیء نیز تعریف می کنیم. برخی از این اعضای داده ای توسط خود کلاس برای عملکرد متدها و برخی دیگر از آنها به عنوان یک متغیر موقت به کار می روند. به این اعضای داده ای، اعضای مفید نیز می گویند چون فقط در عملکرد متدها تأثیر دارند و مانند یک داده قابل رویت کلاس نیستند. لازم نیست که کاربر به تمام اعضای داده ای یا متدهای کلاس دسترسی داشته باشد. اینکه فیلیدها را طوری تعریف کنیم که در خارج از کلاس قابل دسترسی باشند بسیار خطروناک است، چون ممکن است کاربر رفتار و نتیجه یک متند را تغییر دهد.

کپسولاسیون به معنای پنهان کردن جزئیات داخلی یک شی است، به عبارت دیگر چگونه یک شی چیزی را انجام می‌دهد.

کپسوله‌سازی مانع از دیدن نمای داخلی آن توسط مشتری می‌شود، جایی که رفتار انتزاعی در آن پیاده‌سازی می‌شود.

کپسوله سازی تکنیکی است که برای محافظت از اطلاعات موجود در یک شی از یک شی دیگر استفاده می‌شود.

داده‌ها را برای امنیت پنهان کنید، مانند خصوصی کردن متغیرها، و ویژگی را برای دسترسی به داده‌های خصوصی که عمومی خواهند بود، در معرض دید قرار دهید.

بنابراین، هنگامی که به ویژگی دسترسی پیدا می‌کنید، می‌توانید داده‌ها را تأیید کرده و آن را تنظیم کنید.

## Inheritance

When a class includes a property of another class it is known as inheritance.

Inheritance is a process of object reusability.

For example, a child includes the properties of its parents.

وراثت

هنگامی که یک کلاس شامل یک ویژگی از یک کلاس دیگر می‌شود، به عنوان وراثت شناخته می‌شود.

وراثت یک فرآیند استفاده مجدد از شی است.

مثلًاً فرزند شامل اموال والدینش می‌شود.

در ساده‌ترین شکل ممکن، Inheritance را می‌توان به «استفاده از کدهای از قبل نوشته شده» ترجمه کرد.

## Polymorphism

Polymorphism means one name, many forms.

One function behaves in different forms.

In other words, "Many forms of a single object is called Polymorphism."

به معنای یک نام، اشکال متعدد است.

یک تابع به اشکال مختلف رفتار می کند.

به عبارت دیگر، به بسیاری از اشکال یک شیء واحد، چند شکلی می گویند.

"Encapsulation is accomplished using classes. Keeping data and methods that access that data into a single unit."

"Abstraction is accomplished using an Interface. Just giving the abstract information about what it can do without specifying the details."

"Information/Data hiding is accomplished using modifiers by keeping the instance variables private or protected."

کپسول سازی با استفاده از کلاس ها انجام می شود. نگهداری داده ها و روش هایی که به آن داده ها دسترسی دارند در یک واحد واحد.

"انتزاع با استفاده از یک رابط انجام می شود. فقط ارائه اطلاعات انتزاعی در مورد آنچه که می تواند انجام دهد بدون مشخص کردن جزئیات".

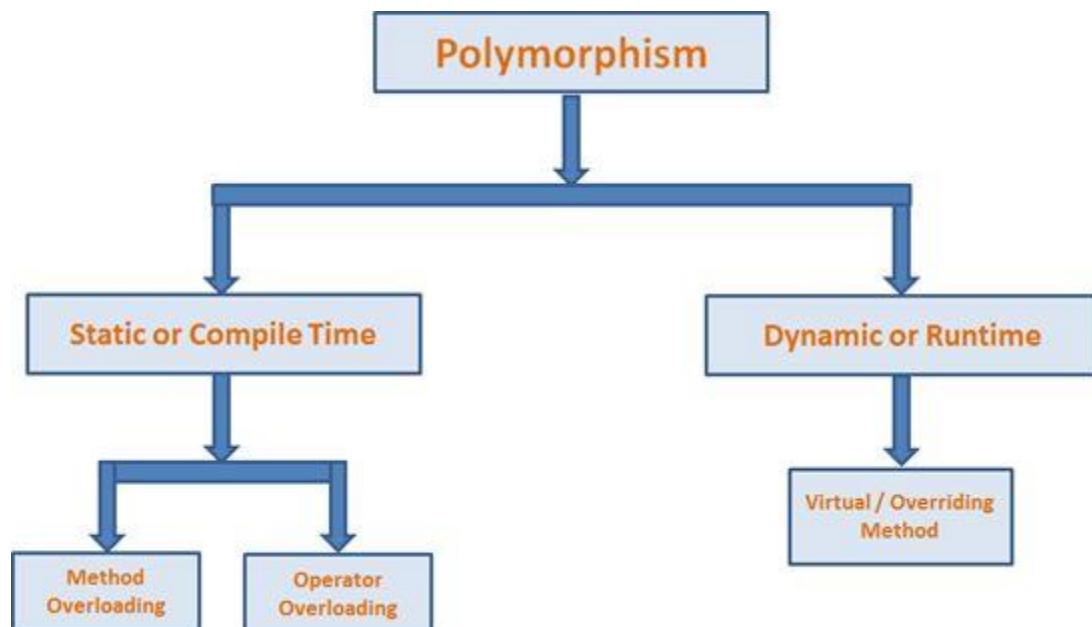
"مخفی کردن اطلاعات/داده ها با استفاده از اصلاح کننده ها با حفظ خصوصی یا محافظت شده متغیرهای نمونه انجام می شود".

Polymorphism is a Greek word, meaning "one name many forms". In other words, one object has many forms or has one name with multiple functionalities. "Poly" means many and "morph" means forms. Polymorphism provides the ability to a class to have multiple implementations with the same name. It is one of the core principles of Object Oriented Programming after encapsulation and inheritance. In this article, you'll learn what polymorphism is, how it works, and how to implement polymorphism in C#.

## Types of Polymorphism

There are two types of polymorphism in C#:

- Static / Compile Time Polymorphism.
- Dynamic / Runtime Polymorphism.



- The meaning of Polymorphism is one name having multiple forms.
- The following are the two types of Polymorphism:
  - Static or compile-time polymorphism (for example, method overloading and operator overloading).
  - Dynamic or runtime polymorphism (for example, overriding).
- Method Overriding differs from shadowing.
- Using the "new" keyword, we can hide the base class member.
- We can prevent a derived class from overriding virtual members.
- We can access a base class virtual member from the derived class.

## سایر سوالات مصاحبه ای:

سوال ۱: در برنامه نویسی سی شارپ Namespace ها چیستند و چگونه به کار گرفته می شوند؟

جواب: در برنامه نویسی سی شارپ Namespace ها برای سازماندهی کلاس ها در درون فریم ورک .NET استفاده می شوند. تنظیم ساختار منطقی کد بر عهده این Namespace ها می باشد. Java package ها مانند Namespace ها می باشند، با این فرق اساسی که Java package ها ، قالب فیزیکی source file ها را شکل می دهند، اما های دات نت این عمل را انجام نمی دهند. با این وجود، بیشتر توسعه دهندگان این شیوه را پی می گیرند و C# source file هایشان را در دایرکتوری هایی سازماندهی می کنند که با این فضای نام ها در ارتباط باشند.

سوال ۲: از داده عددی نوع double در زبان برنامه نویسی سی شارپ چه استفاده ای می شود؟

جواب: برای ذخیره اعداد اعشاری بسیار بزرگ یا بسیار کوچک با دقت ۱۵ رقم از داده نوع double استفاده میشود. همچنین این نوع داده برای رند کردن مقدار اعشاری به صورت اتوماتیک بکار می رود.

سوال ۳: نوع داده float و decimal و double در C# چه تفاوتی با هم دارند؟

جواب: در زبان برنامه نویسی C# داده های عددی float و double عموماً در هنگام اندازه گیری مقادیری که دقت در آنها خیلی مهم نیست مانند فاصله، مسافت و ... کاربرد دارند ولی از داده عددی decimal جهت زمانی که برای ما دقت عددی معیار است نظری واحد پول، محاسبات حسابداری و ... استفاده می گردد.

سوال ۴: عملگرهای رابطه ای را نام ببرید و به جر آن چه عملگرها در سی شارپ وجود دارد؟

جواب: عملگرهای رابطه ای به شرح زیر هستند.

== به معنی تساوی دو مقدار

!= به معنی عدم تساوی دو مقدار

< به معنی بزرگتر

> به معنی کوچکتر

=> به معنی بزرگتر مساوی

=> به معنی کوچکتر مساوی

عملگرهای ریاضی، منطقی، بیتی، انتسابی و متغرقه به همراه عملگرهای رابطه ای انواع عملگرها در زبان برنامه نویسی C# می باشند.

سوال ۵: در چه صورتی نتیجه عملگر true && خواهد بود؟

جواب: عملگر && تنها در صورتی که هر دو عملوند true باشد نتیجه آن true می گردد.

سوال ۶: کدام string ها در immutable.C# هستند؟

جواب: Immutable یعنی اینکه که مقادیر رشته نمی توانند بعد از ایجاد شدن، تغییر کنند. هر گونه تغییر باعث ایجاد یک نمونه کاملاً جدید از رشته می شود که این امر موجب به هدر رفتن حافظه و افزایش اطلاعات مازاد می گردد. هنگامی که مقادیر رشته ها می خواهند تغییر کنند، باید کلاس System.Text.StringBuilder بکار گرفته شود.

سوال ۷: متدها در برنامه نویسی C# چگونه overload می شوند؟

جواب: در برنامه نویسی C# متدها با قواعد گوناگونی overload می شوند. در نتیجه، شما می توانید یک متدها را با دادن نوع داده های مختلف، تعداد پارامترهای غیریکسان و یا تغییر ترتیب پارامترها overload نمایید.

سوال ۸: دستورات سی شارپی زیر چه عملی را انجام می دهند؟

A

```
;Console.ForegroundColor= ConsoleColor.Red  
;Console.WriteLine("IRAN")
```

:B

```
;Console.Beep(200,1000)
```

جواب: الف) این دستور کلمه ایران به رنگ قرمز را چاپ می کند.

ب) صدایی با فرکانس ۲۰۰ هرتز به مدت ۱۰۰۰ میلی ثانیه نواخته می شود.

سوال ۹: خروجی قطعه کد زیر کدام است؟

```
;Int x = 3  
;int y = 4  
; x*= y  
;Console.WriteLine (x)
```

جواب: این قطعه کد زوج بودن عدد را بررسی می کند.

سوال ۱۰: برنامه ای به زبان C# بنویسید که یک عدد صحیح را دریافت نماید و سپس اعداد فرد از ۱ تا آن عدد را چاپ نماید؟

جواب:

```
;int n  
;Console.WriteLine("Enter n:")  
;n = int.Parse(Console.ReadLine())  
for (int i = 1; i <= n; i = i + 2)  
;Console.WriteLine(i)  
();Console.ReadKey
```

## نکات مهم بین جلسات استاد:

Session · ۸, ۹

- Linq
- lambda expression
- List
- enum
- Anonymous Type
- IEnumerable
- Generic
- Set , Get

```
private int _ID;
public int ID {
    get { return _ID; }
    set { _ID = value; }
```

- Read only Properties

```
public string FullName
{
    get { return FirstName + " " + LastName; }
}
```

```
public int ID
{
    get
    {
        return SadrTools.CommonMethods.NumberMethods.GetRandom();
    }
}
```

- override string ToString ()
- Recursive Method

مثال متدهای بازگشتهای فاکتوریل است که مدام یک عدد کوچکتر از خودش را صدا میزنند و این عمل تا رسیدن به عدد یک مداوم تکرار میشود و عدد یک تعریف شده است که یک مقدار پیش فرض را برگرداند(خاتمه آن).

```
public static ulong GetFactorial_Recursive(byte number)
{
    // n! = n * (n-1)!
    // *** شرط پایان - خروج
    // stackoverflow

    if (number < 2)
        return 1;
```

```
        return number * GetFactorial_Recursive((byte)(number - 1));  
    }  
• Has-a , Is-a
```

در حالت Is a کلاس ما به صورت مستقیم از یک کلاس دیگر مشتق می شود، اما در حالت Has a، کلاس ما یک کلاس دیگر را در خود جای داده است.

Teacher is a Person

Car Has a Player

- Sealed

Session ۱ .

- Polymorphism
- Abstract
  - In Abstract : cannot declare a body(when declare abstract method) because it is marked abstract.
- Virtual
  - In virtual Method we can don't declare abstract to class and we can declare virtual to any class. In Virtual we Can Override virtual Method (.base can be used).

Session ۱۱

- Interface
  - Low Coupling - High Cohesion
- Sort in List, Creat `operator ==` , `i=` , `+`
- Compile And Run Time Error (Try catch)

Session ۱۲

- `StringBuilder`
  - Initializes a new instance of the `System.Text.StringBuilder` class.
- `Array.BinarySearch (array, value)`
- `IEnumerable , IEnumerator`

```
// I Enumerable<T> = You can iterate my elements !
```

// `ICollection<T> = I know how many elements I have ! You can modify my contents !`

تمامی کلاس‌هایی که به نحوی شامل یک `Collection` هستند، این دو رابط رو پیاده سازی می‌کنند. وجود `IEnumerable` که توسط کلاسها پیاده سازی می‌شود به کلاس این امکان را می‌دهد که بصورت ضمنی و توکار بشود شیء را پیمایش کرد. دقیقاً به همین دلیل می‌توان با استفاده از حلقه `foreach` یک آرایه را پیمایش کرد، چون که رابط `IEnumerable` توسط کلاس `System.Array` پیاده سازی می‌شود. رابط `IEnumerator` در سطح پایین‌تری از یک `IEnumerable` قرار دارد. با استفاده از این رابط می‌توان در هرجای بدنی متدهایی را برگشت دهیم بدون اینکه مجبور باشیم ابتدا نتایج را مثلاً در یک آرایه بریزیم و بعد آن آرایه را برگشت دهیم.

برای حلقه تکرار زدن روی یک مجموعه فقط خواندنی است (`read only collection`), که تنها یک متدهای `GetEnumerator()` دارد که این امکان را فراهم می‌کند با استفاده از حلقه `foreach` مجموعه‌های `IEnumerable` را تکرار کند. تکرار فقط در جهت جلو امکان پذیر است. `<T>` به صورت زیر تعریف می‌شود:

- یک مجموعه فقط خواندنی است
- تکرار فقط در جهت جلو (مستقیم) صورت می‌گیرد.
- اعمال جمع یا حذف روی `objects` امکان پذیر نیست.
- برای تکرار در مجموعه (در جهت مستقیم)، `enumerator` فراهم شده است.

برای انجام عمل تکرار از یک `foreach` استفاده می‌کنیم و یک مجموعه عمومی باید `<T>` را پیاده سازی کرده و متدهای `GetEnumerator()` را تعریف کند.

اینترفیس `IEnumerator` در واقع سه عضو دارد :

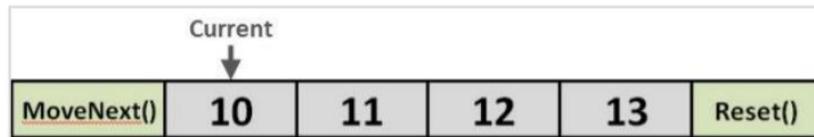
عضو اول یک پرایپریتی هست به نام `Current` که آیتم در پوزیشن فعلی را پیمایش می‌کنه.

عضو دوم یه متدهاست که اسمش `MoveNext` هست که پیمایش به آیتم بعدی می‌کنه

عضو سوم هم یک متدهاست با نام `Reset`

## رابط IEnumarator

یک شمارنده (enumerator) را پیاده سازی می کند که دارای دو متده است `MoveNext()` و `Reset()` و یک خاصیت به نام `Current` می باشد. خاصیت `Current` عنصر جاری یک مجموعه را بر می گرداند. این خاصیت یک خاصیت فقط خواندن (-read-only) است و چیزی که برمی گرداند از نوع `object` می باشد. متده است که، مکان شمارنده را از یک آیتم در مجموعه به آیتم بعدی منتقل می کند. این متده یک مقدار بولی را بر می گرداند، که نشان می دهد که آیا مکان دیگری برای خواندن در دسترس است یا به انتهای مجموعه رسیده است. اگر مکان جدیدی وجود داشته باشد مقدار `true` و در غیر اینصورت مقدار `false` را بر می گرداند. مکان اولیه شمارنده، قبل از اولین آیتم مجموعه است، بنابراین `MoveNext()` باید قبل از اولین دسترسی خاصیت `Current` فراخوانی شود. متده `Reset()`، متده برای برگرداندن شمارنده به مکان اولیه خود قبل از جابجایی است. به تعبیری دیگر، مکان اولیه مجموعه را برمی گرداند. با در اختیار داشتن یک شمارنده (enumerator) شما قادر خواهید بود که حلقه `foreach` را شبیه سازی کرده و عناصر یک مجموعه را با استفاده از متده `MoveNext()` و خاصیت `Current` پیمایش کنید. برای درک بهتر عملکرد دو متده و خاصیت مذکور به شکل و کد زیر توجه کنید :



- Queue , Stack (استک) (پشته)

```
// Queue : First In First Out  
قانون صف  
// First Come , First Service  
// Last in Last Out
```

```
//-----
```

```
// Stack : Last In First Out  
قانون پشته  
// First In Last Out  
// menu delete photo !!
```

## Session ۱۳

- آمار میده که انفاق مورد نظر افتاد و این پیغام به تمام آبجکت های شنونده فرستاده میشود

Delegate : The pipeline between an event and an event handler - پل ارتباطی

Event handler : is responsible for receiving and processing data from a delegate

EventArgs : responsible for encapsulating event data

باید مشخص کنید که به چه مدل فانکشن هایی می تواند اشاره کند :

Func ,Action

## Session ۱۴

- سرعت در بین لیست ها به ترتیب ( آرایه های کلاسیک > آرایه های جزئیک > آرایه معمولی "array list")
- تغییر هدر گردید در ویندوز فرم

## Session ۱۵

- Shallow copy vs Deep copy

- در کپی سطحی Shallow copy ، یک متغیر ساخته می شود و به مکانی در حافظه، که مقدار متغیر قبلی در آن قرار گرفته است، اشاره می کند. پس اگر شما مقدار متغیر اول را تغییر دهید، متغیر دوم هم تغییر می کند. و همین طور اگر مقدار متغیر دوم را تغییر دهید، مقدار متغیر اول هم تغییر می کند.

این تغییر صرفا برای reference type های موجود مورد اعمال قرار میگیرد و value type ها یکسان مانده و تغییر نمیکند.

- در کپی عمیق Deep copy ، یک متغیر ساخته می شود و مقدار متغیر قبلی در آن کپی می شود. توجه شود که در اینجا پس از کپی کردن، اگر مقدار هر کدام از متغیرها را تغییر دهیم، تغییری در مقدار متغیر دیگر مشاهده نمی کنیم.

- Thread

برای عدم صبر و تکمیل پروسه ترد اول در بکگراند بعد از استارت آن از دستور join . نیز استفاده میکنیم.

- lock (\_sync)

برای مثال حساب بانکی، بطور مثال چندین تراکنش پشت سر هم بخواهد از یک حساب کسر شود. از دستور lock که یک شی (که از نوع آبجکت ساخته شده باشد) را قبل از عملیات برداشت (مثل try ) درج میکنیم.

## Extension Method

متدهایی هستند که متعلق به هیچگونه کلاسی نیستند و از طریق کلاس‌های دیگر اضافه می‌شوند. در واقع متدهای Extension همانطور که از نامشان پیداست متدهای اضافی هستند. میتوان متدهای Extension را به اینترفیس‌ها، ساختارها و کلاس‌ها در سی‌شارپ، اضافه کرد. این کار بدون تغییر و ویرایش ساختارها، کلاس‌ها و اینترفیس‌ها انجام می‌شود. متدهای Extension می‌توانند به کلاس‌های تعریف شده توسط برنامه نویس و یا کلاس‌های تعریف شده در دات‌نوت فریمورک اضافه شوند.

متدهای توسعه امکان اضافی کردن کارایی‌های جدید به کلاسها، ساختارها یا اینترفیس‌هایی که کد آنها در دسترس نیست و یا امکان ارث بری از اونها وجود نداره رو میده! تو تعریف متدهای توسعه چند تا محدودیت داریم!  
اول اینکه متدها را توی یک کلاس استاتیک بنویسیم.  
دوم برای اینکه به کامپایلر بگیم که این تابع، یک تابع توسعه هست باید تو اولین پارامتر ورودی از کلمه‌ی کلیدی this استفاده کنیم.  
یه نکته دیگه اینکه اگر یک تابع توسعه تعریف کردین ولی یک توسعه داخلی با الگوی مشابه وجود داشته باشه، اولویت با توسعه داخلی هستش.  
و نکته دیگه رویدادها و عملگرها قابل توسعه نیستند!

Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type. Extension methods are static methods, but they're called as if they were instance methods on the extended type.

## Call By Reference (Ref and Out)

ارسال پارامتر به صورت **:value**

ارسال به روش value parameter به صورت پیش فرض می‌باشد. هنگامی که ما متغیری (variable) به ورودی یک متدهار ارسال می‌کنیم، در صورتی که نوع آن پارامتر را ذکر نکنیم به صورت مقداری ارسال می‌شود. به عبارت دیگر مقدار آن متغیر به تابع فرستاده می‌شود (یک کپی از آن، نه خود اصلی آن). در صورت تغییر مقدار متغیردر بدنه تابع، تغییری در پارامتر ارسال شده، ایجاد نمی‌شود.

تذکر : هر گاه در اعلان پارامترها از کلمات کلیدی `ref` و `out` استفاده نشود، به صورت `value` فرستاده می شود.

#### ارسال پارامتر به صورت `ref` :

وقتی متغیری را به صورت `ref` به یک تابع ارسال می کنیم ، مقدار متغیر ارسال نمی شود بلکه آدرس متغیر (خود اصلی آن) به بدن متد فرستاده می شود و هر تغییری درمتغیر محلی روی متغیر اصلی نیز اعمال می شود. به این نوع پارامترها ارجاعی می گویند.

#### پارامترهای `out` :

پارامترهای `out`، پارامترهایی هستند که مقدار دهنده اولیه نشده اند، را قبول می کنند. کلمه کلیدی `out` زمانی مورد استفاده قرار میگیرد که بخواهیم یک متغیر بدون مقدار را به متد ارسال کنیم. متغیر بدون مقدار اولیه، متغیری است که مقداری به آن اختصاص داده نشده است. در این حالت متد یک مقدار به متغیر می دهد. ارسال متغیر مقداردهی نشده به متد زمانی مفید است که شما بخواهید از طریق متد، متغیر را مقداردهی کنید. استفاده از کلمه کلیدی `out` باعث ارسال آرگومان به روش ارجاع می شود نه مقدار.

از کلمه کلیدی `out` برای پارامترهای متد استفاده شده است، بنابراین می توانند متغیرهای مقداردهی نشده را قبول کنند. استفاده از پارامترهای `out` بدین معنا نیست که شما همیشه نیاز دارید که آرگومانهای مقداردهی نشده را به متد ارسال کنید، بلکه آرگومانهایی که شامل مقدار هستند را هم می توان به متد ارسال کرد. این کار در حکم استفاده از کلمه کلیدی `ref` است.

تفاوت `ref` با `out` این است که کلمه کلیدی `ref` به کامپایلر میگوید که متغیر مقدار دهنده اولیه و بعد به متد ارسال شده است ولی `out` به کامپایلر می گوید که متغیر مقدار دهنده اولیه نشده و باید در داخل متد مقدار دهنده اولیه شود.

نکته: زمانی که لازم باشد یک متد دارای چندین خروجی باشد از `out` استفاده می کنیم.

## Inheritance

مفهوم ارث بری یا Inheritance می باشد. ارث بری یعنی تولید کلاس هایی جدید که برخی از ویژگی های خود را از کلاس مادر Parent Class به ارث بردہ اند.

## Access Modifier

- **عمومی Public** : در این حالت عنصر به صورت عمومی تعریف شده و از هر جای برنامه توسط هر عنصر دیگر مثل سایر کلاس ها و توابع قابل دسترسی است. این حالت دارای حداقل محدودیت برای عنصر بوده و Enums و Interface ها به صورت پیش فرض public هستند.
- محافظت شده یا **Protected** : در این حالت عنصر فقط توسط عوامل کلاس خود یا کلاس هایی که از کلاس آن به ارث رفته اند، قابل دسترس است.
- درونی یا **internal** : در این حالت عنصر فقط درون پروژه جاری قابل دسترسی است.
- درونی محافظت شده یا **Protected internal** : این حالت، همانند حالت internal است با این تفاوت که عناصر موجود در کلاس هایی که از کلاس عنصر به ارث رفته اند، حتی اگر در پروژه های دیگر باشند قابلیت دسترسی به آن را دارند.
- خصوصی یا **Private** : در این حالت فقط اعضای همان کلاس امکان دسترسی به آیتم مورد نظر را دارند. این حالت دارای بیشترین میزان محدودیت بوده و Class ها و Struct ها به صورت پیش فرض خصوصی private هستند.

## Abstract

کلاس های مجرد (Abstract) کلاس هایی هستند که کلاس پایه سایر کلاس ها هستند. این نوع کلاس ها نمی توانند مانند کلاس های عادی دارای سازنده باشند. به همین دلیل نمیتوان از کلاس های انتزاعی نمونه ایجاد کنید؛ چون که هدف اصلی از به کار بردن کلاس های انتزاعی استفاده از آنها به عنوان کلاس پایه برای کلاس های مشتق است. برای تعریف یک کلاس انتزاعی از کلمه کلیدی abstract استفاده می شود.

میتوان پراپرتی های Abstract نیز داشته باشیم که برای تعریف این خاصیت کلمه کلیدی abstract را به هنگام تعریف فیلد کار بردہ ایم. این property باید به

وسیله کلاس هایی که از این کلاس ارث می بردند override شود ولی از آن جایی که به صورت abstract تعریف شده است قسمتهای set و get قادر بدنی هستند.

کلاس های Abstract می توانند شامل Property های معمولی مانند خاصیت Name باشند.

کلاس های Abstract حداقل باید یک عضو Abstract داشته باشند.

## Abstract Method

در کلاس های Abstract میتوانیم متدهایی پیاده سازی کنیم که حتماً در کلاس فرزند Inheritance شوند. به دلیل این که متدهای (Abstract Method) یک تعمید یا اجبار است تا آن متدها در تمامی کلاس های فرزند کلاس جاری اجرا شود. در واقع، به وسیله این کار در زمان اجرای برنامه چک می کنیم، آیا تمامی کلاس های فرزند، اینتابع را در خود تعریف کرده اند یا خیر.

## Abstract (Override – Virtual)

باید توجه داشت در کلاس های Abstract میتوانیم متدهایی که دارای بدنی باشند را تعریف نماییم و در کلاس هایی که مشتق هستند از آنها نیز بهره ببریم اما با توجه به آنکه در این متدها باید شخصی سازی صورت گیرد میتوان متدها را override کرد بدین منظور در هنگام تعریف متدها در کلاس Abstract باید از کلمه virtual استفاده کنیم سپس با استفاده از کلمه کلیدی override در متدهای مشتق شده میتوانیم متدهای موجود را (بازنویسی) کنیم. همچنین میتوانیم با استفاده از کلمه کلیدی base مقادیر پارامترها را در سازنده مشتق شده به parent ارسال کنیم.

تذکر: نمیتوان از یک کلاس abstract نمونه ایجاد کرد، ولی از کلاس هایی که از این نوع کلاسها مشتق می شوند، می توان نمونه ایجاد کرد.

## Interface

- اینترفیس ها جهت پیاده سازی والد parent بودن در ارث بری ساخته شده اند.
- برخلاف کلاس ها که فقط یک کلاس میتواند به عنوان parent قرار گیرد، در اینجا میتوان چندین interface را بعنوان والد خود تعیین کرد.
- از `C#` به بعد میتوان interface های ایجاد کرد که ۱۰۰ درصد انتزاعی نباشند یعنی میتوان بعضی موارد آن را طوری معین کرد تا در کلاس فرزند پیاده سازی نشود. از ورژن قبل از آن باید تمامی موارد موجود در اینترفیس ها در فرزند پیاده سازی میشدند.
- تفاوت آن با abstract در این است که داخل کلاس abstract باید معین کرد آنها بی که abstract هستند مجددا در کلاس فرزند پیاده سازی شوند.
- در اینترفیس امکان مشخص کردن سطح دسترسی و امکان ایجاد فیلد وجود ندارد و باید پرپرتبی (بدون سطح دسترسی) تعیین کرد و همه به صورت پیشفرض public هستند..
- داخل interface کانسٹراکتور نداریم بنابر این نمیتوان از اینترفیس نمونه جدید ایجاد کرد.
- بعد از Inheritance (ارث بری) کردن از یک اینترفیس باید بیاییم و تمام اعضاء داخل آن اینترفیس را implement (پیاده سازی) نماییم.
- خود interface ها نیز میتوانند فرزند چند interface دیگر باشند. بدین منظور کلاس باید به هنگام ارث بری از interface نخست ، اعضاء هم خود آن اینترفیس و هم آنکه ارث رفته است را پیاده سازی نماید.
- متدهای ایجاد شده در interface ها قادر بدنی هستند و فقط تعریف میشوند تا در کلاس مورد نظر الزاما پیاده سازی شوند.

## Boxing & Unboxing

برای انتقال یک نوع داده مقداری (Value Type) به یک نوع داده ارجاعی (Reference Type) استفاده می‌شود.

برای انتقال یک نوع داده ارجاعی به یک نوع داده مقداری می‌باشد.

## Implicit & Explicit

در C#, داده‌ها را می‌توان از یک نوع به نوع دیگر با استفاده از روش‌های Implicit و یا Explicit تبدیل کرد. فرض کنید دو متغیر به نام A و B داریم، می‌خواهیم محتوای متغیر B را در متغیر A بروزیم در صورتی می‌توانیم این کار انجام دهیم که از یک نوع داده باشند و متغیر مقصد محدودش مساوی یا بزرگتر از متغیر مبدا باشد. عمل Implicit زمانی اتفاق می‌افتد که این ویژگی‌ها را داشته باشد، یعنی نیازی به تبدیل نوع داده محتوای متغیر نیست و به راحتی می‌توانیم محتوای یک متغیر را در متغیر دیگری بروزیم. ولی اگر مجبور باشیم نوع داده محتوای متغیر را تغییر دهیم، عمل Explicit رخ می‌دهد.

## Delegate

Delegate‌ها می‌توانند به یک متدهاست که امضای آن مانند امضای این Delegate است اشاره کنند.

Delegate‌ها انواعی هستند که مرجع یک متدهاست را در خود ذخیره می‌کنند. همچنین می‌توانند رفتار هر متدهاست را کپی برداری کنند. برای تعریف یک delegate از کلمه کلیدی delegate استفاده می‌شود. تعریف یک delegate بسیار شبیه به تعریف Delegate است، با این تفاوت که متدهاست بدنی دارد ولی delegate ندارد. دقیقاً مانند متدهاست دارای نوع برگشتی و مجموعه‌ای از پارامترها هستند.

Delegate‌ها، می‌گویند که چه نوع متدهاست را می‌توانند در خود ذخیره کنند. در زیر نحوه تعریف delegate نشان داده شده است:

```
delegate return Type DelegateName (dt param1, dt param2, ... dt paramN);
```

نحوه فراخوانی دلیگیت در مثال زیر آمده است.

```
delegate void ArithmeticDelegate(int num1, int num2);

static void Add(int x, int y)
{
    Console.WriteLine("Sum is {0}.", x + y);
}

static void Subtract(int x, int y)
{
    Console.WriteLine("Difference is {0}.", x - y);
}

static void Main()
{
    ArithmeticDelegate Operation;

    int num1, num2;

    Console.Write("Enter first number: ");
    num1 = Convert.ToInt32(Console.ReadLine());

    Console.Write("Enter second number: ");
    num2 = Convert.ToInt32(Console.ReadLine());

    if (num1 < num2)
    {
        Operation = new ArithmeticDelegate(Add);
    }
    else
    {
        Operation = new ArithmeticDelegate(Subtract);
    }

    Operation(num1, num2);
}
```

ضمنا اگر بخواهیم بیش از یک متده را به دلیگیت اضافه کنیم از `=` استفاده میکنیم.