**Constructors in c#**

**یک Special Method در کلاس که initialized Variables آن کلاس را به عهده دارد.**

**Constructor هیچ ReturnType Value ندارد.**

**هر کلاسی به یک سازنده حتما نیاز دارد در صورتی که سازنده ای را برای کلاسی تعریف نکرده باشیم**

**به صورت default برای آن کلاس implicit constructor توسط کامپایلر برای آن کلاس تعریف می شود و متغیر های کلاس را مقدار دهی اولیه (پیش فرض متغیر ها) می کند.**

**بدین ترتیب مقدار دهی اولیه متغیرها در کلاس به عهده سازنده می باشد : سازندهدر کلاس بسیار مهم است.**

**ما می توانیم Explicit Constructor در کلاس تعریف کنیم.**

**Types of Constructors in C#**

1. **Default or Parameter Less Constructor**
2. **Parameterized Constructor**
3. **Copy Constructor**
4. **Static Constructor**
5. **Private Constructor**

##### **Default or Parameterless Constructors in C#**

اگر ctor فاقد پارامتر باشد default Constructor است. 2 نوع است

System-Defined Default Constructor : سیستم تعریف می کند

User-Defined Default Constructor کاربر تعریف میکند

Default Constructor مقادیر پیش فرض را به data Member non-staticاختصاص می دهد به

implicit Constructor توسط کامپایلر ساخته می شودو همیشه public و Parameter-Less می باشد. و چون متغیر ها را مقدار دهی اولیه می کند System-Defined Default Constructor می نامند

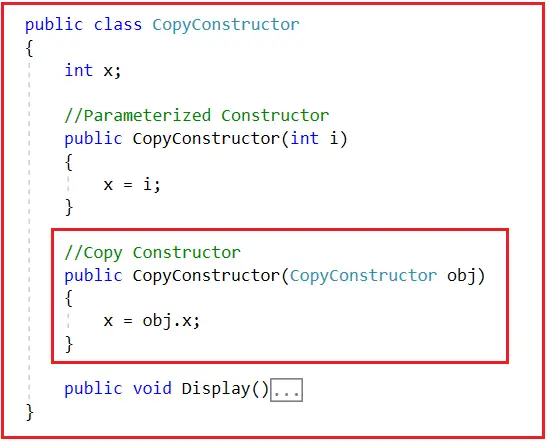
اگر بخواهیم یک custom logic را در زمان ایجاد object اجرا کنیم، آن منطق ممکن است

object initialization logic یا useful logic (منطق مفید) دیگری باشد، به عنوان یک Developer، باید Explicit Constructor در سی شارپ ارائه کنیم.

اگر بخواهیم object را به صورت dynamically )پویا (با مقادیر داده شده توسط کاربر initialize کنیم یا اگر بخواهیم each instance از یک class را با مجموعه ای متفاوت از مقادیر initialize کنیم، باید از سازنده Parameterized در سی شارپ استفاده کنیم

اگر یک constructor با پارامترها تعریف شود، آن را parameterize constructor در سی شارپ می نامیم

Copy Constructor : اگر بخواهیم Multiple instance با Same-Value داشته باشیم آنگاه باید از copy constructor استفاده کینم



در کلاس می توان چندین سازنده تعریف کرد اما باید signature متفاوت داشته باشند.

در سی شارپ امکان ایجاد یک سازنده به صورت Static وجود دارد Static Constructor :

Static Constructor مسول اولیه سازی متغیر های Static هستند

Static Constructor هرگز به صورت explicitly فراخوانی نمی شوند و Implicitly فراخوانی می شوند

Non-Static Constructor هرگز به صورت Implicitly فراخوانی نمی شود و به صورت explicitly فراخوانی می شود.

نکته : Static Constructor اولین بلوک اجرای کد در اجرای Class است.

Static Constructor را نمی توان پارامتر کرد، زیرا به صورت Implicitly اجرا میشود و فرصتی برای ارسال پارمتر نیست، پس Overloading Static Constructor امکان پذیر نیست.

Static Constructor برای Initialize the Static Members به صورت اتومات Call می شود.

در تعریف Static Constructor هیچ Access Specifiers وجود ندارد.

Static Constructor یکبار یعنی در زمان Class Loading فراخوانی می شود.

Initialize Static members در Non – Static Constructor امکان دارد اما

Initialize Non Static members در Static Constructor امکان ندارد

Private Constructor : با استفاده از Access Specifiers Private ساخته می شود در اینصورت نمی توان Object از Class در خارج از Class ایجاد کرد

Private Constructor برای ایجاد Object برای Class در همان Class استفاده می شود.

با استفاده از Private Constructor در سی شارپ می توانیم الگوی طراحی singleton را پیاده سازی کنیم.

یادآوری : **singleton چیست؟**

سینگلتون یک الگوی طراحی سازنده (Creational) است که به شما اطمینان می‌دهد از یک کلاس، تنها یک نمونه یا شی (Object) ساخته شود. بنابراین در هر زمان که استفاده از این کلاس نیاز شود، سینگلتون یک دسترسی سراسری به تنها شی‌ای که از این کلاس ساخته شده است ایجاد می‌کند و اجازه ساخت شی جدیدی را از آن کلاس نمی‌دهد.

زمانی که کلاس فقط شامل Static members باشد، باید از Private Constructor در سی شارپ استفاده کنیم.

چرا به Constructors در C# نیاز داریم؟ اگر بخواهیم instance آن Class را ایجاد کنیم، هر class نیاز به یک Constructorدارد که در آن وجود داشته باشد.

نیاز به تعریف مجدد Constructors به Explicitly چیست؟

اگر Constructors را به طور Explicitly با Parameters تعریف کنیم، هر بار که می خواهیم Instance آن Class را ایجاد کنیم، شانس initialize فیلدها یا متغیرهای class با یک مقدار جدید را خواهیم داشت.

هر زمان که class را ایجاد می کنیم یک چیز را به خاطر بسپاریم به این معنی که می توانیم از class reuse کنیم و هر زمان و هر کجا که بخواهیم می توانیم instance را ایجاد کنیم. این code reusability است.

وقتی class را Define می‌کنیم، ابتدا مشخص کنیم که آیا class Variables (منظور filed های کلاس می باشد که سطح دسترسی آنها Private است) برای اجرا به مقادیری نیاز دارند یا خیر و در صورت نیاز، constructor را به explicitly تعریف کرده و مقادیر را از constructor عبور می‌دهیم تا هر بار که instance class ایجاد می‌شود، شانس ارسال مقادیر جدید را داشته باشیم.

با کمک یک Parameterized Constructor ، می توانیم هر instance از class را با different set از مقادیر initialize کنیم

Constructor ای که parameter از نوع class را می گیرد، Copy Constructor نامیده می شود.

Static Constructor در سی شارپ فقط یک بار Call می شود

**Static vs Non-Static Constructors in C#**

1. اگر constructor ای به Explictly با استفاده از Static Modifire اعلام شود، static Constructor و بقیه Non Static Constructor است.

Static Fields / Variables -2 توسط static Constructor و

Non static Fields / Variables توسط Non-Static Constructor یا Instance

در سی شارپ Initialize می شوند

Static Constructor -3 به طور implicitly Call می شوند در حالی

که Non-static Constructor به طور Explicitly Call می شوند.

Static Constructor - 4 ، immediately از شروع اجرای یک Class اجرا می شوند و علاوه بر این، اولین بلوک کدی است که تحت یک Class اجرا می شود در حالی که

Non-Static Constructor فقط پس از ایجاد Class instance و همچنین هر بار که Class Instance ایجاد می شود، اجرا می شوند.

Static Constructor - 5 فقط یک بار اجرا می شوند در حالی که

Non-Static Constructor 0 یا n تعداد بار بسته به تعداد Class Instance ایجاد می کنیم اجرا می شوند.

1. - Non-Static Constructor را می توان Parameter کرد در حالی

Static Constructor نمی توانند هیچ Parameter داشته باشند چون به صورت call Implicitly می شود

Non-Static Constructor – 7 را می توان Overloading کرد در حالی که

Static Constructor را نمی توان Overloading کرد

Static constructor - 8 به طور Implictly تنها در صورتی تعریف می شوند که آن کلاس دارای Const Filedباشد یا در غیر این صورت آن سازنده وجود نخواهد داشت مشروط بر اینکه کلاس سازنده ایستا صریح نداشته باشد.