**شی گرایی**

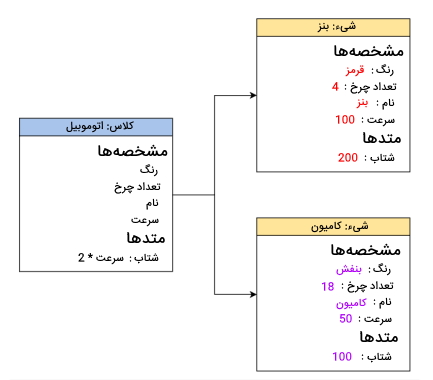
**تعریف:** برنامه نویسی شی گرا رویکردی از برنامه نویسی است که برای طراحی یک پروژه نرم‌افزاری، بر روی داده‌ها یا اشیا (Objects) تمرکز دارد.

**کلاس(Class):**  مشخصه‌ها و متدهای اشیاء، درون کلاس تعریف می‌شوند. کلاس، طرح اولیه‌ ای است که از آن برای ساخت اشیاء در برنامه نویسی شی گرا استفاده می‌شود.

**شی(Object):** نمونه ای از کلاس است که دارای مشخصه های بخصوصی است و از متد های کلاس میتواند استفاده کند.

**مشخصه(Attribute):** زمانی که شی جدیدی ساخته می‌شود، می‌توان اطلاعات آن شی را با استفاده از مشخصه‌ها ذخیره کرد.

**متد (Method):** متدها برای تعریف رفتار یا عملی در قالب تابع تعریف می‌شوند. متدها می‌توانند اطلاعاتی را در خروجی بازگردانند یا اطلاعات مربوط به شی را تغییر دهند.



**اصول شی گرایی:**

1. **اصل کپسوله‌سازی (Encapsulation) :** با ساخت یک شی به عنوان نمونه‌ای از کلاس، مشخصه‌ها و متدهای کپسوله‌سازی شده‌ از بیرون کلاس قابل دسترسی و ویرایش نیستند .
   1. Public می‌توان از بیرون کلاس نیز به مشخصه ها و متدهای دسترسی داشت.
   2. Private فقط می‌توان از طریق متدهای درون کلاس دسترسی داشت.
   3. Protected فقط می توان از خود کلاس و زیرکلاس های آن دسترسی داشت.
2. **اصل انتزاعی (Abstraction) :** پنهان کردن جزئیات غیرضروری از کاربران است تا آن‌ها با پیچیدگی کدهای نوشته شده مواجه نشوند.اطلاعات فقط از طریق متد های خاص قابل دسترس است.
3. **اصل ارث بری (Inheritance):** کلاس فرزند، میتواند از داده ها و متد های کلاس پدر استفاده کند.
4. اصل چندریختی (Polymorphism): یک نام چند شکل
   1. Override **زمان کامپایل/ استاتیک**تصمیم گیری در مورد اینکه کدام متد باید فراخوانی شود در زمان کامپایل. تفاوت در تعداد پارامترهای ارسال شده و نوع پارامتر
   2. Overload**زمان اجرا / پویا** تعداد پارامترها و نوع پارامتر باید یکسان باشد و ممکن است پیاده سازی متفاوتی داشته باشد. در کلاس پایه از کلمه کلیدی virtual با متد استفاده کردیم تا با استفاده از کلمه کلیدی override در کلاس مشتق شده override شود

# **معرفی معماری:** طراحی دامنه محور  **Domain Driven Design**

**دامنه** : "حوزه دانش یا فعالیت". دامنه در حوزه مهندسی نرم افزار معمولاً به موضوع و هدفی که برنامه ی نرم افزاری در آن اعمال می شود، اشاره دارد. به عبارت دیگر، در طول توسعه ی برنامه، دامنه، حوزه ی دانش و فعالیتی است که منطق برنامه در اطراف آن قرار گرفته است. دامنه، ایده ها، دانش و داده های مسئله ای است که می خواهید آن را حل کنید.

**مدل**: وقتی ما میتوانیم مسئله ای را مدل کنیم یعنی توانستیم در مسیر حل آن قدم برداریم. زیرا با مدل کردن توانستیم یه نمای کلی از تمام زوایای آن داشته باشیم و ساختارمند با مسئله رفتار کنیم.

دامنه را حوزه ی فعالیت و کسب و کار خود بدانیم و مدل را راه حل آن

الگوی طراحی (Design Pattern) چیست ؟

الگوهای طراحی راه حل های کلی هستند که در قالب الگو های ارائه می شوند و می توانند برای حل مشکلات در طراحی نرم افزار استفاده می شوند .

### تزریق وابستگی (Dependency Injection)چیست؟

Dependency Injection یک الگوی طراحی است که با هدف حذف وابستگی های موجود بین دو کلاس با استفاده از یک Interface یا همان رابط ایجاد شده است. به عبارت دیگر تزریق وابستگی به معنی تزریق وابستگی های یک کلاس به جای استفاده مستقیم از آن ها درون کلاس است.( یعنی به جای نمونه گیری از کلاس ، از اینترفیس استفاده کنیم )

Inversion of Control (IoC) چیست ؟

IoC یک الگوی طراحی بهبود یافته از Dependency Injection است . در واقع IoC Container به عنوان یک کلاس وظیفه انجام هرچه ساده تر Dependency Injection را به عهده دارد .

Singleton :

Singleton تضمین می کند که تنها یک نمونه از یک کلاس ساخته شود و در سراسر برنامه قابل دسترس باشد.

برای مطالعه بیشتر به لینک زیر مراجعه کنید :

[الگوی طراحی Singleton در سی شارپ (barnamenevisan.org)](https://barnamenevisan.org/Articles/Article6099.html)

کلاس های انتزاعی (Abstract) :

کلاس های که توسط واژه abstract در تعریف کلاس مشخص شده اند و نمی توان از آن نمونه ای ایجاد کرد .

Interface :

یک Interface تنها شامل تعاریفی از متدها ، صفت ها (attributes) ، رویدادها (events) هاست اما فاقد پیاده سازی آنهاست. پیاده سازی این متد ها و رویداد ها به عهده کلاسی است که از اینترفیس ارث بری می کند .

Generic :

Genericها به ما کمک می‌کنند تا بتوانیم کلاس‌ها و یا methodهایی را ایجاد کنیم که هنگام تعریف آن‌ها، تایپشان برایمان مهم نباشد و بتوانیم یک تایپ عمومی برایشان تعریف کنیم و کسی که می‌خواهد از این کلاس یا متدها استفاده کند  تایپ آن را خودش مشخص می‌کند.

مثال :

public void Show(int number)

{

Console.WriteLine(number);

}

public void Show(double number)

{

Console.WriteLine(number);

}

public void Show(string message)

{

Console.WriteLine(message);

}

ما در کد بالا سه تابع یک سان ولی با تایپ متفاوت داریم ، ولی با جنریک می توان هر سه این توابع را در یک تابع پیاده سازی کرد :

public void Show<E>(E item)

{

Console.WriteLine(item);

}