**الگوی Iterator**

**پیمایش بر روی مجموعه ها**

**زمانی که مجموعه از اشیا را در کنار هم قرار می دهیم باید نحوه پیمایش آن را نیز مشخص کنیم که client بتواند به راحتی آن را پیمایش کند و پردازش مورد نیاز را انجام دهد. برای مثال برای List نحوه پیمایش را در کنار لیست پیاده سازی کنیم.**

**نوع های مختلفی برای list داریم :**

* **Collections**
* **List**
* **Queue**
* **Stack**
* **HashSet**

**هر کدام از موارد بالا یک api برای پیمایش خود ارائه میکنند. اما الگوی Iterator وظیفه پیمایش هر نوع list را بر عهده میگیرد.**

**Dotnet از این الگو استفاده میکند.**

**نکات:**

1. **Iterator باید CurrentItem را نگه داری کند.**
2. **برای هر Aggregate می توانیم چند concreteAggregate بنویسیم.**
3. **یک شی ConcreteAggregate را می توانیم با چند ConcreteIterator شمارش کنیم.**
4. **در حال پیشفرض زمانی که از Iterator دات نت استفاده می کنیم فقط دو متد MoveNext() و Reset() را داریم که جز حداقل ترین متدهایی است که میتواند داشته باشد که می توانیم متدهای بیشتری هم اضافه کنیم. برای مثال MovePrevious() ، Skip() و ....**

**مزایا:**

1. **رعایت اصل SRP: الگوریتم های شمارش هر یک ،در یک کلاس مجزا توسعه داده می شوند.**
2. **اصل Open/Close : الگوریتم و یا ساختار شمارش جدید نیاز باشد می توانیم مجزا بسازیم.**
3. **میتوانیم چندین تکرارگر مختلف برای هر مجموعه ای بنویسیم.**

**معایب:**

1. **اگر در پروژه از مجموعه های ساده ای کار میکنید استفاده از این الگو به صرفه نیست.(در هر زبان برنامه نویسی این الگو اضافه شده است)**

**چه زمانی از این الگو استفاده میکنیم:**

1. **زمانی که ساختار داده پیچیده داریم و میخواهیم این پیچیدگی را از دید کاربر استفاده کننده پنهان کنیم.**
2. **زمانی که میخواهیم یک ساختار استاندارد برای پیمایش مجموعه های مختلف ایجاد کنیم.**
3. **زمانی که میخواهیم از تکرار کد پیمایش در پروژه پرهیز کنیم.**