**Iterator Design Pattern**

Iterator design pattern is used to access and traverse an aggregate object without exposing its internal structure.

**Intent**

* Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

The intent is to provide the access of aggregate object without exposing its internal structure.

**Problem**

* Need to access the elements of an aggregate object without exposing the internal details of object.

Accessing elements of object is required with traversal, and at the same time it should not expose the internal structure of aggregate object. Also this may be applicable to different data structures, which provide different algorithms for accessing and traversing.

**Solution**

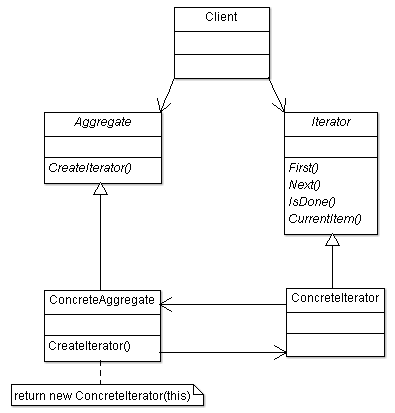
* Provide a way to access and traverse an aggregate object by giving responsibility of access and traversal to another object.

So there can be common interface, which can be applicable for different data structures to provide interface for accessing and traversing. There will be separate objects having this interface and will have responsibility to access the elements and traversal of aggregate objects.

**Where it is applicable?**

* Access and traversal of aggregate object is required without exposing its internal structure.
* Common interface is required to access and traverse different data structures.

**Structure**



**Participant classes**

* **Iterator** class provides the interface for accessing the elements and traversal of the aggregate object.
* **ConcreteIterator** class implements the interface of Iterator class and keeps track of current item of aggregate object.
* **Aggregate** class provides the interface for creating Iterator object.
* **ConcreteAggregate** class implements the method CreateIterator().

**How they work together?**

* The client creates the ConcreteAggregate object. It uses the CreateIterator interface to create the ConcreteIterator object. This will have the current item information to track the aggregate object. Now it uses the Iterator interface to access the elements and traversal of aggregate object.
* So we can have different data structures, and their concrete iterators can implement interface according to their traversal algorithm. This makes the access and traversal simple for different data structures.