Iterator:

**Other name (if any)**

The Iterator pattern is also known as **Cursor.**

**What it does**

This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation. It is a behavioral design pattern. It is used in java collection (arraylist, linked list, priority queue) to iterate over items.

**Where to use**

1. When you want to access a collection of objects without exposing its internal representation.  
2. When there are multiple traversals of objects need to be supported in the collection.

**Steps**

1. Aggregator Interface: Aggregator means the actual collection. For example: linked list, arraylist etc. It has only one function createIterator.  
2. Concrete Aggregator Class: It will take the actual collection as data. In the createIterator function, concreteIterator object will be created.  
3. Iterator Interface: It has two functions. One is hasNext and another one is next function.  
4. Concrete Iterator Class: It will take the actual collection as data. In the hasNext function, it will return true or false based on the condition whether there are next item in the item list. In the next function, next item will be returned.  
5. Client Code: In the client code, concrete aggregator object will be created using concrete aggregator class. Then the  concrete iterator object will be created using iterator interface and createIterator function. Then write this piece of code:  
while([iteratorObject.hasNext](http://iteratorObject.hasNext" \t "_blank)())  
{  
Item itemObject = (Item) [iteratorObject.next](http://iteratorObject.next)();  
}

**Special cases (if any)**

placeholder

**Advantages**

1. It supports variations in the traversal of a collection.  
2. It simplifies the interface to the collection.

**Disadvantages**

placeholder

**Code**

Coding Concept

**Difference with similar pattern**

placeholder

**Diagram**

Coding Concept + Tutorialspoint