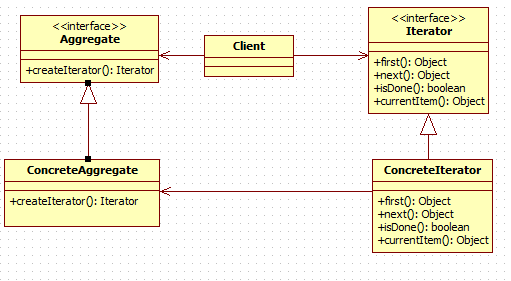
**Iterator pattern**

1. **Description:** Iterator pattern provides an interface called Iterator to access the aggregate and still keeps the encapsulation of the its representation.
2. **Implementation:**



The classes that participate to the Iterator pattern are:

**Aggregate** - defines the Aggregate.

**ConcreteAggregate** – define the implementation of the collection.

**Iterator** - defines the iterator.

**ConcreteIterator** - defines implementation of the iterator.

**Client** - uses Aggregate and Iterator interfaces to traverse the collection in ConcreteIterator .

1. **Advantages and disadvantages**

**Advantages:**

* Access the content but still encapsulate the aggregate object.
  + Client calls the iterator to traverse through the collection
  + Depend on the type of collection, the iterator will use certain methods.
* Take the responsibility of traversing different aggregate structures.
  + Client does not need to know the actual implemented collection.
* Support multiple traversals of aggregate objects.
  + Each aggregate object can have many iterator for different traversals

**Disadvantages:**

* Do not have some useful method (add, set,...)
  + The iterator only provides a way to access, not modify an aggregate object.
* The order of the iteration is fixed in the implementation.
  + Each iterator has its own traversal algorithm.
  + If we need a new traversal, we should have a different iterator

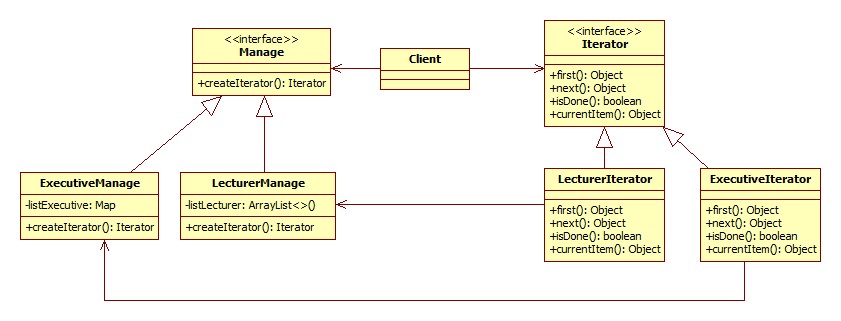
**Usage**

* When the system needs to be independent of how its products are created composed and represented.
* When the system needs to be configured with one of multiple families of products.

1. **Example:**

We have two classes for managing executives and lecturers in the university. We use ArrayList for lecturer management because it is easy to extend. On the other hand, the executive position is usually not duplicated, so we can use Map to store the position and information of executive people.

As a result, we have two different kinds of collection and we can use Iterator pattern to traverse both of them.



1. **Alternative pattern:**

Visitor pattern can be used to traverse the object structure. The Iterator pattern is ussually used on collections with same type objects. On the other hand, the Visitor pattern can be used to traverse complex structures (hierarchical structures or composite structures). The Visitor pattern also helps to perform operations on the objects which have different interfaces.

1. **Reference**

1) Gamma, E, Helm, R, Johnson, R & Vlissides, J 1994, *Design Patterns - Elements of Reusable Object-Oriented Software,* Addison-Wesley, Holland

2) Object Oriented Design, *Iterator pattern*, Object Oriented Design, viewed 20 November 2012, <<http://www.oodesign.com/iterator-pattern.html>>