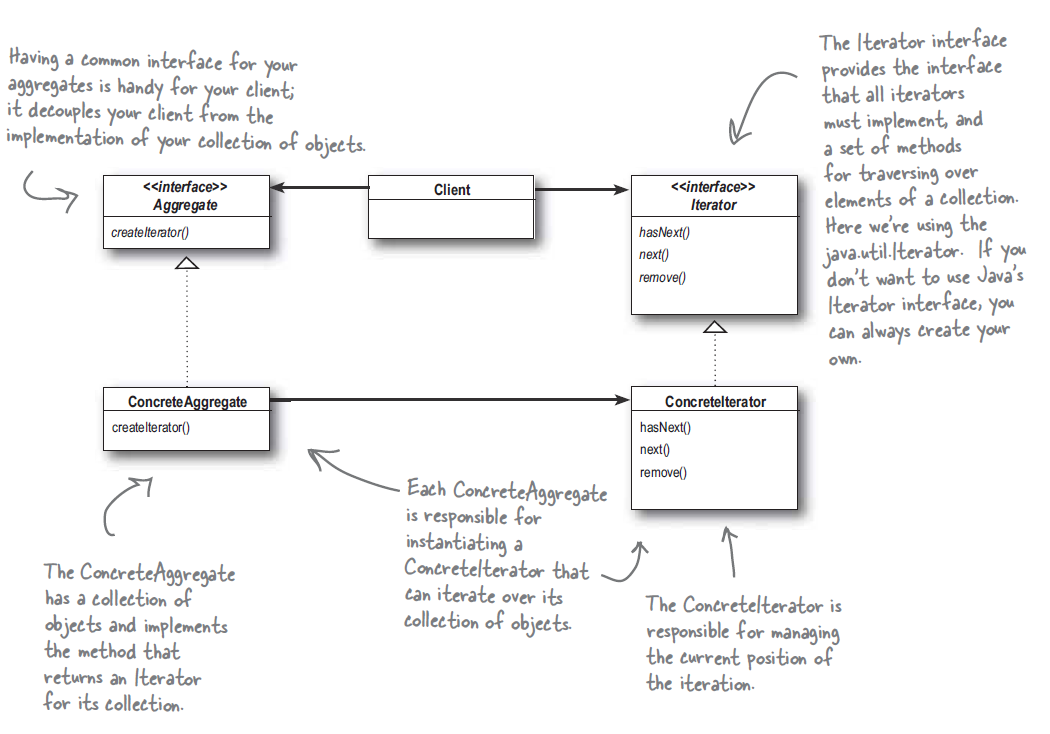
**Iterator Pattern:**

1. **Definition**: The Iterator Pattern let us access the elements of a collection sequentially without exposing the underlying collection representation.
   1. It hides the collection internal data structure from the client who want to traverse it. Collection object knows how to create its own iterator.
   2. It also places the task of traversal of collection object on an iterator which significantly simplifies the Collection interface. Otherwise number of method in the collection interface would rise.
   3. Client uses iterator interface to traverse collection so it’s not bothered about how items are stored. It can call iterate() method on collection object to get hold of its iterator implementation.
   4. **Class diagram**:



1. **Internal vs External iterator**: In external iterator client controls the iteration by calling the next(). In case of internal iterator iteration is controlled by passing an operation to iterator defining operations to be performed on the elements as it iterate through the elements.
2. Java.util.Iterator is the java interface for Iterators having next(), hasNext() and remove() method.
3. ListIterator is an iterator capable of iterating both ways having previous() method.
4. A client should not make any assumption of iteration order for the underlying collection unless specifically documented.
5. **Single responsibility Principle:** A class should have only one reason to change.
   1. Every responsibility of a class is an area of change. More than one responsibility means more than one area of change.
   2. **Cohesion**: A class or module is highly cohesive if its designed around related functions.