**一、迭代器模式介绍**

迭代器模式：

**提供一种方法顺序的访问一个聚合对象中各个元素，而又不暴露该对象的内部表示。**

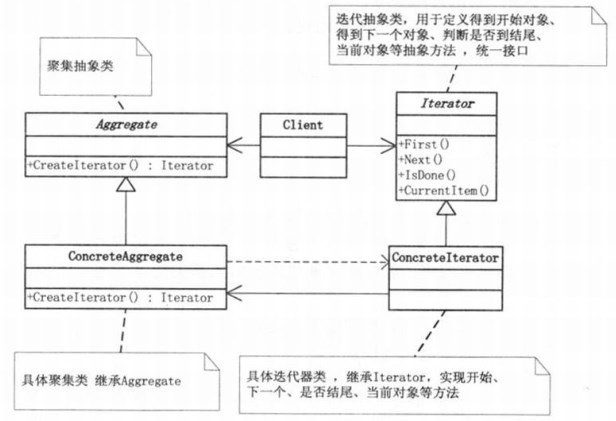
一般情况，我们自己开发时很少自定义迭代器，因为java本身已经把迭代器做到内部中了（比如：常用的list和set中都内置了迭代器）。

当然，如果真有这种需求需要我们自定义迭代器的话，可以参考jdk的迭代器实现方式来实现自己的迭代器。

迭代器是可以**从前往后**，或者**从后往前**遍历的。

为遍历不同聚集结构提供如：开始，下一个，是否有下一个，是否结束，当前哪一个等等的一个统一接口。

迭代器模式UML图：



聚集类：Aggregate(抽象类)和ConcreteAggregate(具体聚集类)表示聚集类，是用来存储迭代器的数据。

在Aggregate(抽象类)中有一个CreateIterator方法，用来获取迭代器

迭代器：迭代器用来为聚集类提供服务，提供了一系列访问聚集类对象元素的方法。

实现（把实现的CreateIterator作为内部类放在ConcreteAggregate里面，可以获取list）：

**package** iterator;  
  
**public interface** MyIterator<T> {  
 **void** first();  
  
 **void** next();  
  
 **boolean** hasNext();  
  
 **boolean** isFirst();  
  
 **boolean** isLast();  
  
 Object getCurrentObj();  
}

**package** iterator;  
  
  
**public interface** Aggregate<T> {  
 **void** addObject(T object);  
  
 **void** removeObject(T object);  
  
 MyIterator<T> iterator();  
  
  
}

**package** iterator;  
  
**import** java.util.ArrayList;  
**import** java.util.Iterator;  
**import** java.util.List;  
**import** java.util.Map;  
  
**public class** ConcreteAggregate<T> **implements** Aggregate<T>{  
 **private** List<T> **list** = **new** ArrayList<>();  
  
 **public** List<T> getList() {  
 **return list**;  
 }  
  
 **public void** setList(List<T> list) {  
 **this**.**list** = list;  
 }  
  
 @Override  
 **public void** addObject(T object) {  
 **list**.add(object);  
 }  
  
 @Override  
 **public void** removeObject(T object) {  
 **list**.remove(object);  
 }  
  
 @Override  
 **public** MyIterator<T> iterator() {  
 **return new** ConcreteIterator();  
 }  
  
 **private class** ConcreteIterator<T> **implements** MyIterator<T> {  
  
 **private int cursor**;  
  
 @Override  
 **public void** first() {  
 **cursor**=0;  
 }  
  
 @Override  
 **public void** next() {  
 **if** (**cursor** < **list**.size()) {  
 **cursor**++;  
 }  
 }  
  
 @Override  
 **public boolean** hasNext() {  
 **if** (**cursor** < **list**.size()) {  
 **cursor**++;  
 **return true**;  
 }  
 **return false**;  
 }  
  
 @Override  
 **public boolean** isFirst() {  
 **if** (**cursor** ==0) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
 @Override  
 **public boolean** isLast() {  
 **if** (**cursor** ==(**list**.size()-1)) {  
 **return true**;  
 }  
 **return false**;  
 }  
  
 @Override  
 **public** Object getCurrentObj() {  
 **return list**.get(**cursor**-1);  
 }  
 }  
}

**package** iterator;  
  
**import** java.util.ArrayList;  
**import** java.util.Iterator;  
**import** java.util.List;  
  
**public class** test {  
 **public static void** main(String[] args) {  
 ConcreteAggregate concreteAggregate = **new** ConcreteAggregate();  
 concreteAggregate.addObject(**"123"**);  
 concreteAggregate.addObject(**"23"**);  
 concreteAggregate.addObject(**"3"**);  
 concreteAggregate.addObject(1);  
 concreteAggregate.addObject(2);  
 concreteAggregate.addObject(3);  
  
 MyIterator myIterator = concreteAggregate.iterator();  
 **while** (myIterator.hasNext()) {  
 System.***out***.println(myIterator.getCurrentObj());  
 *//myIterator.next();* }  
 }  
  
}