4. 迭代器模式

看一个具体的需求

编写程序展示一个学校院系结构：需求是这样，要在一个页面中展示出学校的院系组成，一个学校有多个学院， 一个学院有多个系。如图：

```
--------计算机学院有以下专业----------
Java工程师
大数据工程师
前端工程师
信息安全
--------信息工程学院有以下专业----------
网络信息安全
电子技术
```

传统的设计方案(类图)

传统的方式的问题分析

1） 将学院看做是学校的子类，系是学院的子类，这样实际上是站在组织大小来进行分层次的

2） 实际上我们的要求是 ：在一个页面中展示出学校的院系组成，一个学校有多个学院，一个学院有多个系， 因此这种方案，不能很好实现的遍历的操作

3） 解决方案：=> 迭代器模式

迭代器模式基本介绍

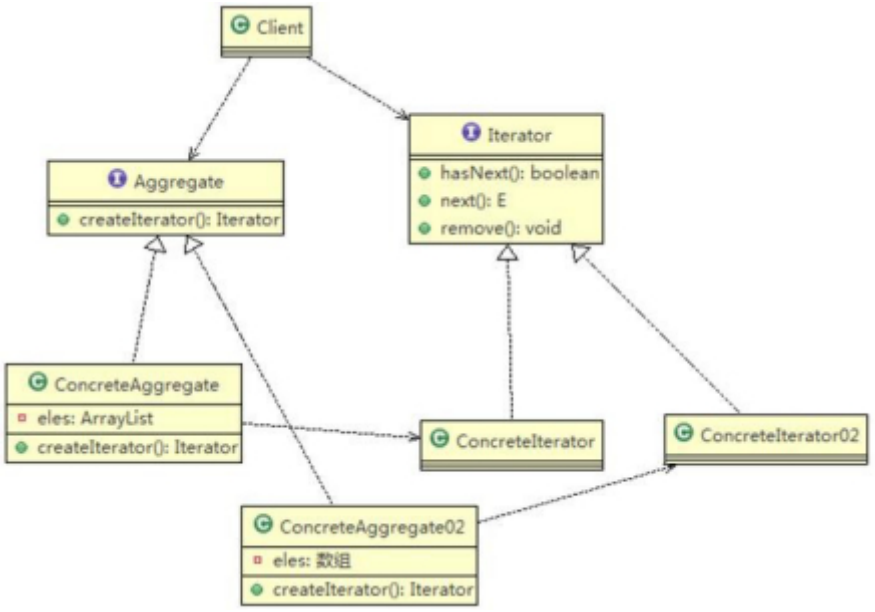
基本介绍

1） 迭代器模式（Iterator Pattern）是常用的设计模式，属于行为型模式

2） 如果我们的集合元素是用不同的方式实现的，有数组，还有 java 的集合类，或者还有其他方式，当客户端要遍历这些集合元素的时候就要使用多种遍历方式，而且还会暴露元素的内部结构，可以考虑使用迭代器模式解决。

3） 迭代器模式，提供一种遍历集合元素的统一接口，用一致的方法遍历集合元素，不需要知道集合对象的底层表示，即：不暴露其内部的结构。

迭代器模式的原理类图



对原理类图的说明-即(迭代器模式的角色及职责)

1） Iterator ： 迭代器接口，是系统提供，含义 hasNext， next，remove

2） ConcreteIterator ： 具体的迭代器类，管理迭代

3） Aggregate ：一个统一的聚合接口， 将客户端和具体聚合解耦

4） ConcreteAggreage ：具体的聚合持有对象集合， 并提供一个方法，返回一个迭代器， 该迭代器可以正确遍历集合

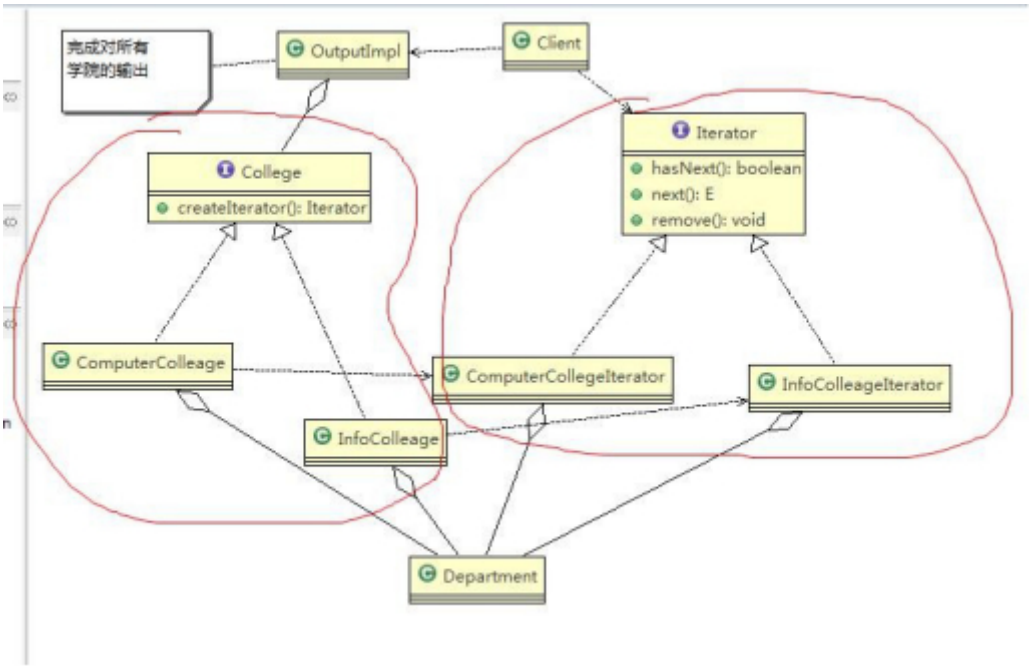
5） Client ：客户端， 通过 Iterator 和 Aggregate 依赖子类

迭代器模式应用实例

1） 应用实例要求

编写程序展示一个学校院系结构：需求是这样，要在一个页面中展示出学校的院系组成，一个学校有多个学院， 一个学院有多个系。

2） 设计思路分析



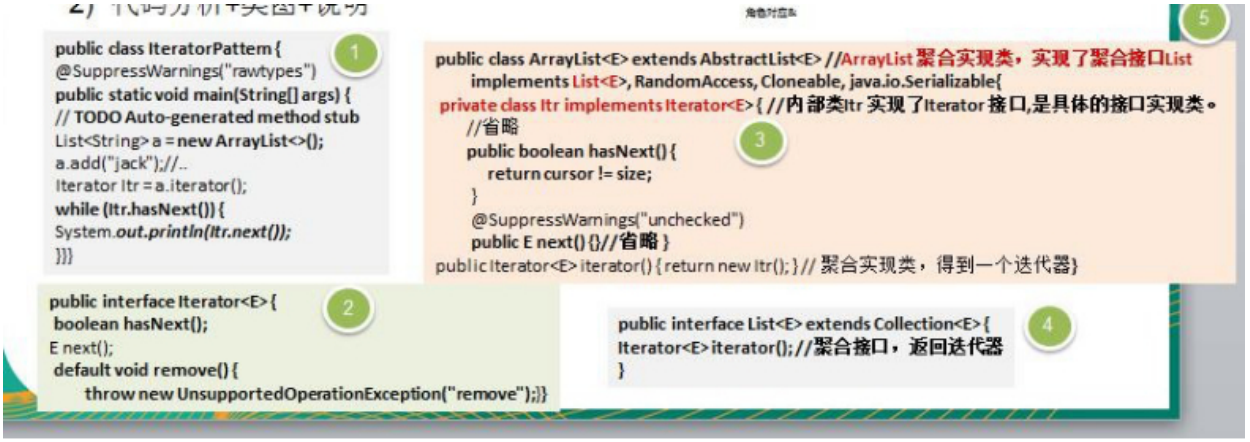

```
public class ComputerCollegeIterator implements Iterator {
  //这里我们需要Department 是以怎样的方式存放=>数组
  Department[] departments;
  int position = 0; //遍历的位置

  public ComputerCollegeIterator(Department[] departments) {
    this.departments = departments;
  }

  //判断是否还有下一个元素
  @Override
  public boolean hasNext() {
    if(position >= departments.length || departments[position] == null) {
      return false;
    }else {
      return true;
    }
  }

  @Override
  public Object next() {
    // TODO Auto-generated method stub
    Department department = departments[position];
    position += 1;
    return department;
  }

  //删除的方法，默认空实现
  public void remove() {
  }
}
```

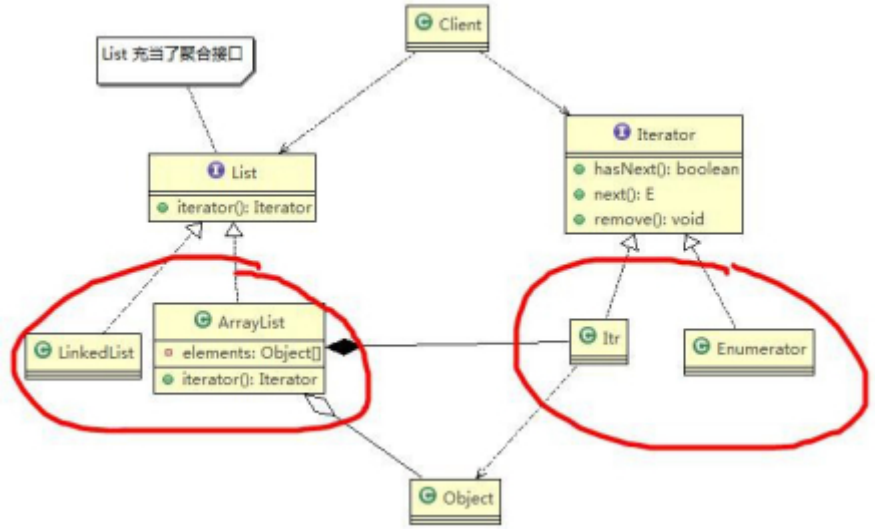
```
public class ComputerCollege implements College {
//部门数组
Department[] departments=null;
@Override
public Iterator createIterator() {
  return new ComputerCollegeIterator(departments);
}
}
```

迭代器模式在 JDK-ArrayList 集合应用的源码分析

1） JDK 的 ArrayList 集合中就使用了迭代器模式

2） 代码分析+类图+说明





1） 对类图的角色分析和说明

内部类 Itr 充当具体实现迭代器 Iterator 的类， 作为 ArrayList 内部类

List 就是充当了聚合接口，含有一个 iterator() 方法，返回一个迭代器对象

ArrayList 是实现聚合接口 List 的子类，实现了 iterator()

Iterator 接口系统提供

迭代器模式解决了 不同集合(ArrayList ,LinkedList) 统一遍历问题

迭代器模式的注意事项和细节

优点

1) 提供一个统一的方法遍历对象，客户不用再考虑聚合的类型，使用一种方法就可以遍历对象了。

2) 隐藏了聚合的内部结构，客户端要遍历聚合的时候只能取到迭代器，而不会知道聚合的具体组成。

3) 提供了一种设计思想，就是一个类应该只有一个引起变化的原因（叫做单一责任原则）。在聚合类中，我们把迭代器分开，就是要把管理对象集合和遍历对象集合的责任分开，这样一来集合改变的话，只影响到聚合对象。而如果遍历方式改变的话，只影响到了迭代器。

4) 当要展示一组相似对象，或者遍历一组相同对象时使用，适合使用迭代器模式

缺点

每个聚合对象都要一个迭代器，会生成多个迭代器不好管理类