GOF 设计模式小白教程之迭代器模式

作者：valarchie

原文链接：https://ld246.com/article/1567337731270

来源网站：链滴

迭代器模式（Iterator）

定义：

提供一个对象来顺序访问聚合对象中的一系列数据，而不暴露聚合对象的内部表示。迭代器模式是一对象行为型模式

通俗解释：

在开发当中，我们经常需要访问一个聚合对象中的各个元素，如遍历List，通常的做法是将链表的创和遍历都放在同一个类中，但这种方式不利于程序的扩展，如果要更换遍历方法就必须修改程序源代，这违背了 “开闭原则” 。

如果聚合类中不提供遍历方法，而让用户自己实现呢？这也会暴露了聚合类的内部表示，使其数据不全；增加了客户的负担。迭代器模式能较好地解决以上问题，它在客户访问类与聚合类之间插入一个代器，这分离了聚合对象与其遍历行为，对客户也隐藏了其内部细节，且满足“单一职责原则”和开原则，如 Java 中的 Collection、List、Set、Map 等都包含了迭代器。

代码：

抽象聚合类：拥有添加元素、移除元素还有获取迭代器的方法

```
public interface Aggregate {

    void add(Object o);
    void remove(Object o);
    Iterator iterator();

}
```

抽象迭代器：拥有获取第一个元素，下一个元素，以及判断是否有下个元素的方法

```
public interface Iterator {

    Object first();
    Object next();
    boolean hasNext();

}
```

姓名聚合类，本例只是简单说明迭代器模式所以类中方法不严谨。

```
public class NameAggregate implements Aggregate{

    private String[] names = new String[16];

    @Override
    public void add(Object o) {

        for (int i = 0; i < names.length; i++) {
            if (names[i] == null) {
                names[i] = (String) o;
```

```
                break;
            }
        }

    }
    @Override
    public void remove(Object o) {

        for (int i = 0; i < names.length; i++) {
            if (names[i] == o) {
                names[i] = null;
            }
        }

    }
    @Override
    public Iterator iterator() {
        return new NameIterator(names);
    }

}
```

姓名迭代器，本例只是简单说明迭代器模式所以类中方法不严谨。

```
public class NameIterator implements Iterator {

    private String[] names = null;
    // 游标
    private int cursor = -1;

    public NameIterator(String[] strings) {
        this.names = strings;
    }


    @Override
    public Object first() {
        cursor = -1;
        return names[cursor + 1];
    }

    @Override
    public Object next() {
        return names[++cursor];
    }

    @Override
    public boolean hasNext() {
        if (cursor + 1 == names.length) {
            return false;
        }
        return names[cursor + 1] != null;
    }
```

```
}
```

测试迭代器模式

```
public class TestIterator {

    public static void main(String[] args) {

        Aggregate names = new NameAggregate();

        names.add("张三");
        names.add("李四");
        names.add("王五");
        names.add("赵六");

        Iterator iterator = names.iterator();

        while (iterator.hasNext()) {
            System.out.println("姓名：" + iterator.next());
        }

    }

}
```

运行结果：

```
姓名：张三
姓名：李四
姓名：王五
姓名：赵六
```

解析：

1. 使得访问一个聚合对象的内容而无须暴露它的内部表示。
2. 遍历任务交由迭代器完成，这简化了聚合类。
3. 它支持以不同方式遍历一个聚合，甚至可以自定义迭代器的子类以支持新的遍历。
4. 增加新的聚合类和迭代器类都很方便，无须修改原有代码。
5. 封装性良好，为遍历不同的聚合结构提供一个统一的接口。