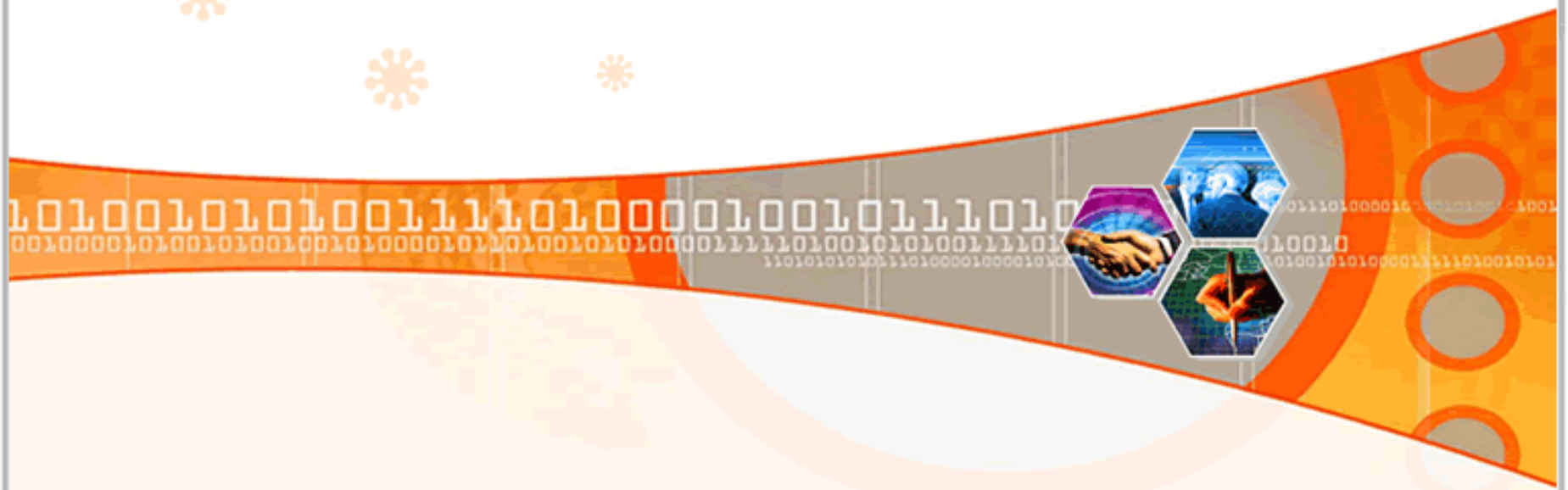
第20章

迭代器模式



主讲教师：程细柱　韶关学院计算机系
本书主编：刘　伟　　清华大学出版社

本章教学内容

- **迭代器模式**
  - ✓ 模式动机与定义
  - ✓ 模式结构与分析
  - ✓ 模式实例与解析
  - ✓ 模式效果与应用
  - ✓ 模式扩展

迭代器模式

- **模式动机**
  - ✓ 一个聚合对象，如一个列表(List)或者一个集合(Set)，应该提供一种方法来让别人可以访问它的元素，而又不需要暴露它的内部结构。
  - ✓ 针对不同的需要，可能还要以不同的方式遍历整个聚合对象，但是我们并不希望在聚合对象的抽象层接口中充斥着各种不同遍历的操作。
  - ✓ 怎样遍历一个聚合对象，又不需要了解聚合对象的内部结构，还能够提供多种不同的遍历方式，这就是迭代器模式所要解决的问题。

迭代器模式

- 模式动机

遍历频道

电视机遥控器

电视机

（电视频道的集合）

迭代器模式

模式动机

- 在迭代器模式中，提供一个外部的迭代器来对聚合对象进行访问和遍历，迭代器定义了一个访问该聚合元素的接口，并且可以跟踪当前遍历的元素，了解哪些元素已经遍历过而哪些没有。
- 有了迭代器模式，我们会发现对一个复杂的聚合对象的操作会变得如此简单。

迭代器模式

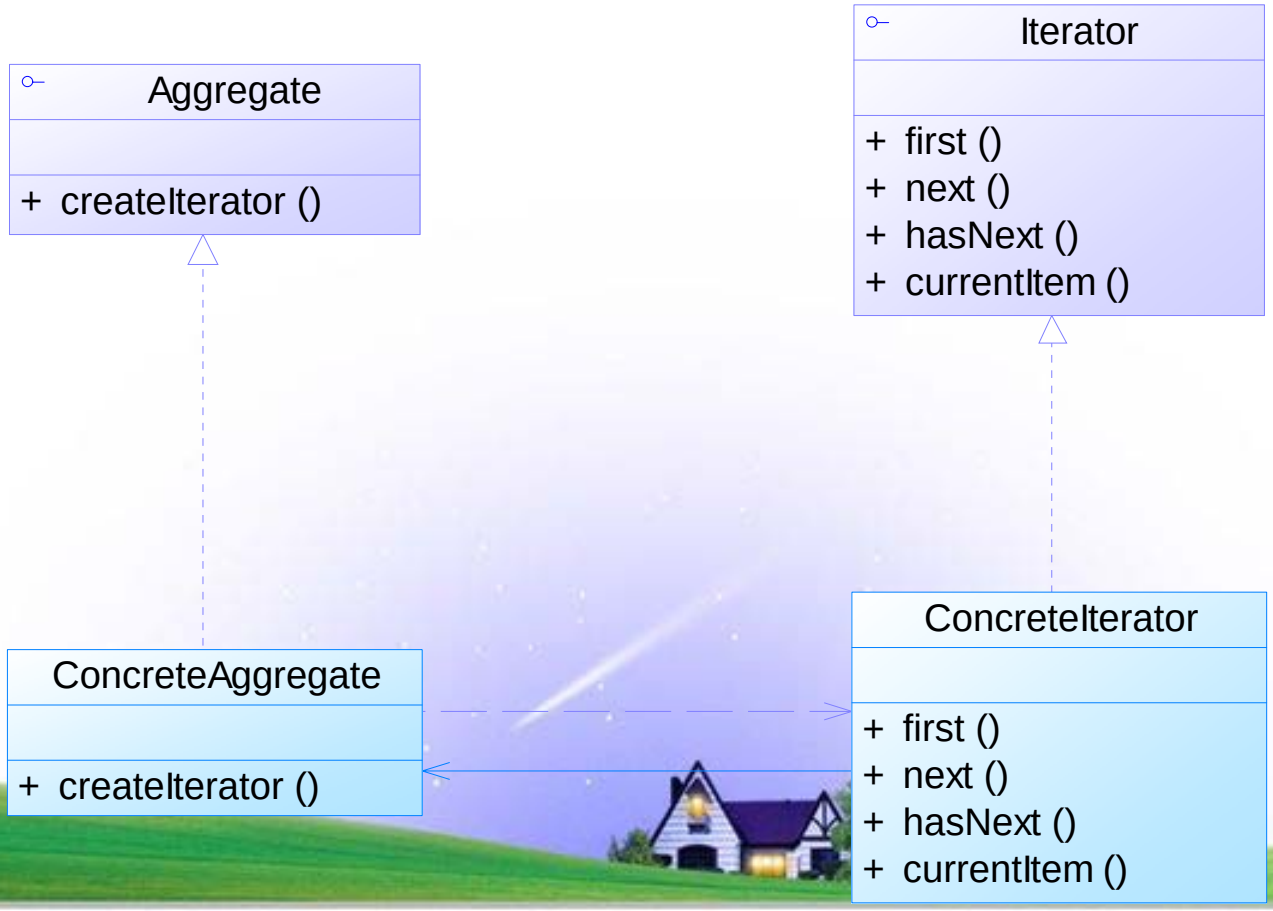
- **模式定义**
  - ✓迭代器模式**(Iterator Pattern)**：提供一种方法来访问聚合对象，而不用暴露这个对象的内部表示，其别名为游标**(Cursor)**。迭代器模式是一种对象行为型模式。

迭代器模式

- 模式定义
  - ✓ **Iterator Pattern:** Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.
  - ✓ Frequency of use: high

迭代器模式

- 模式结构


Aggregate
+ createIterator ()
Iterator
+ first ()
+ next ()
+ hasNext ()
+ currentItem ()
ConcreteAggregate
+ createIterator ()
ConcreteIterator
+ first ()
+ next ()
+ hasNext ()
+ currentItem ()

迭代器模式

- 模式结构
  - ✓迭代器模式包含如下角色:
    - Iterator: 抽象迭代器
    - ConcreteIterator: 具体迭代器
    - Aggregate: 抽象聚合类
    - ConcreteAggregate: 具体聚合类

迭代器模式

- **模式分析**
  - ✓聚合是一个管理和组织数据对象的数据结构。
  - ✓聚合对象主要拥有两个职责：一是存储内部数据；二是遍历内部数据。
  - ✓存储数据是聚合对象最基本的职责。
  - ✓将遍历聚合对象中数据的行为提取出来，封装到一个迭代器中，通过专门的迭代器来遍历聚合对象的内部数据，这就是迭代器模式的本质。迭代器模式是“单一职责原则”的完美体现。

迭代器模式

- 模式分析
  - ✓自定义迭代器
    - MyIterator——抽象迭代器
    - MyCollection——抽象聚合类
    - NewCollection——具体聚合类
    - NewIterator——具体迭代器
    - Client

迭代器模式

- 模式分析
  - ✓自定义迭代器
    - 参考代码：Chapter 20 Iterator\SimpleIterator
    - 下载地址：http://download.csdn.net/user/cflynn

演示……

迭代器模式

- 模式分析
  - ✓注意：以上代码中将具体迭代器类作为具体聚合类的内部类，另外迭代器模式也可以使用常规方式实现，代码如下：

```
public class ConcreteIterator implements Iterator
{
        private ConcreteAggregate objects;
       public  ConcreteIterator(ConcreteAggregate  objects)
        {         this.objects=objects; }
        public void first() {  ......  }
        public void next() {  ......  }
        public boolean hasNext() {  ......  }
        public Object currentItem() {  ......  }
}
public class ConcreteAggregate implements Aggregate
{         ......
        public Iterator createIterator()
        {  return new ConcreteIterator(this);        }
        ......
}
```

迭代器模式

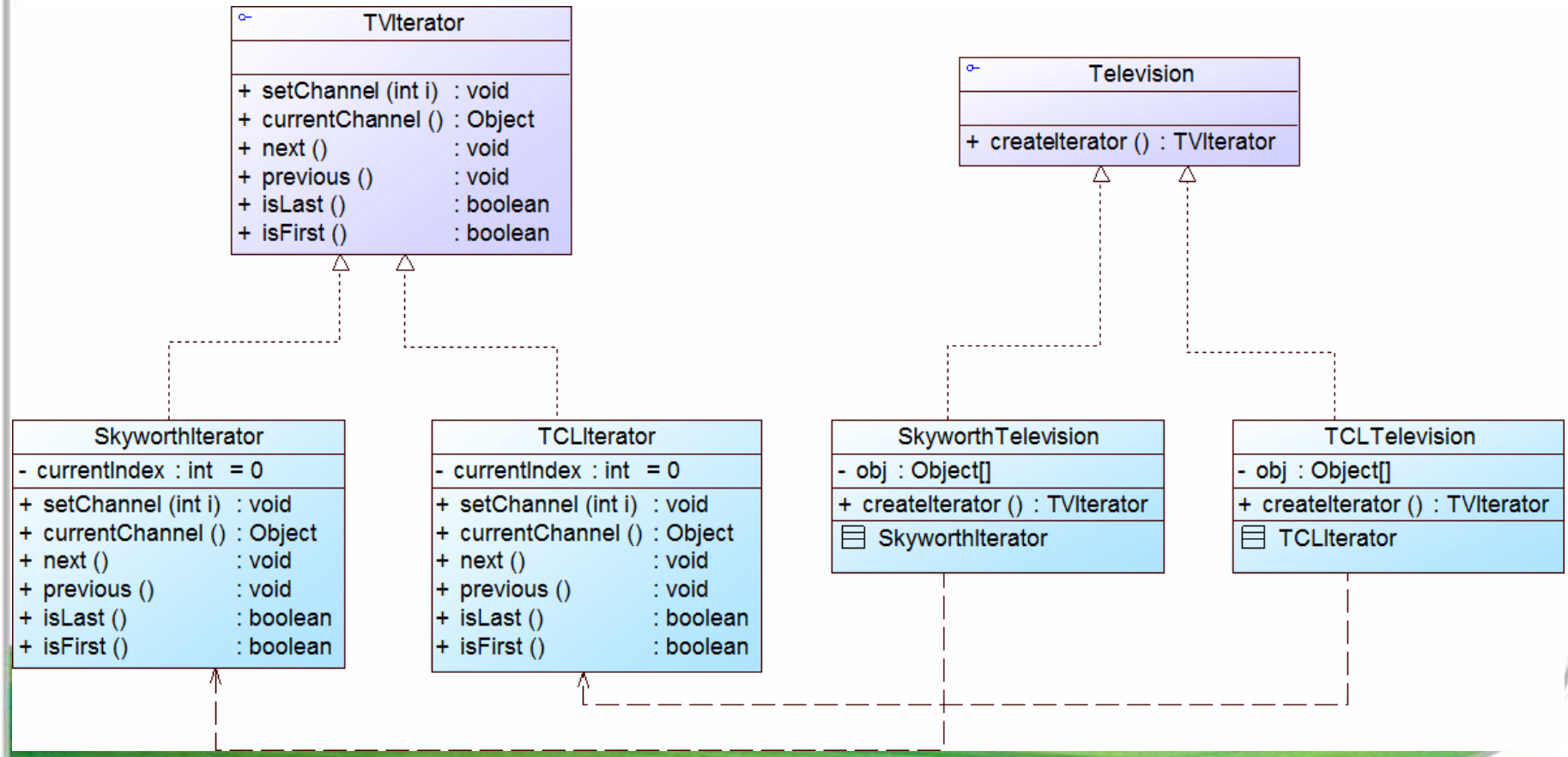
- **模式分析**
  - ✓ 在迭代器模式中应用了工厂方法模式，聚合类充当工厂类，而迭代器充当产品类，由于定义了抽象层，系统的扩展性很好，在客户端可以针对抽象聚合类和抽象迭代器进行编程。
  - ✓ 由于很多编程语言的类库都已经实现了迭代器模式，因此在实际使用中我们很少自定义迭代器，只需要直接使用Java、C#等语言中已定义好的迭代器即可，迭代器已经成为我们操作聚合对象的基本工具之一。

迭代器模式

- 迭代器模式实例与解析
  - ✓实例：电视机遥控器
    - 电视机遥控器就是一个迭代器的实例，通过它可以实现对电视机频道集合的遍历操作，本实例我们将模拟电视机遥控器的实现。

迭代器模式

✓ 实例：电视机遥控器



迭代器模式

- 迭代器模式实例与解析
  - ✓ 实例：电视机遥控器
    - 参考代码：Chapter 20 Iterator\sample01
    - 下载地址：http://download.csdn.net/user/cflynn

演示.....

迭代器模式

- 模式优缺点
  - ✓迭代器模式的优点
    - 它支持以不同的方式遍历一个聚合对象。
    - 迭代器简化了聚合类。
    - 在同一个聚合上可以有多个遍历。
    - 在迭代器模式中，增加新的聚合类和迭代器类都很方便，无须修改原有代码，满足“开闭原则”的要求。

迭代器模式

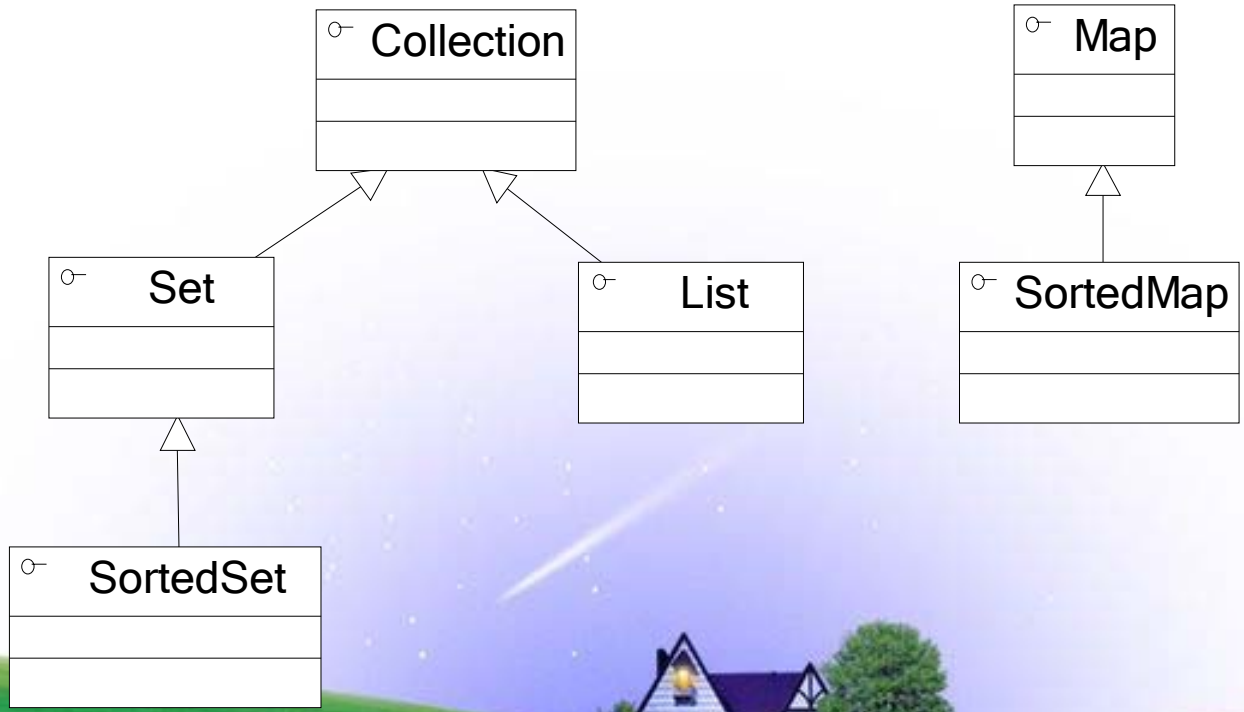
- 模式优缺点
  - ✓迭代器模式的缺点
    - 由于迭代器模式将存储数据和遍历数据的职责分离，增加新的聚合类需要对应增加新的迭代器类，类的个数成对增加，这在一定程度上增加了系统的复杂性。

迭代器模式

- 模式适用环境
  - ✓在以下情况下可以使用迭代器模式:
    - 访问一个聚合对象的内容而无须暴露它的内部表示。
    - 需要为聚合对象提供多种遍历方式。
    - 为遍历不同的聚合结构提供一个统一的接口。

迭代器模式

- 模式应用
  - ✓JDK 1.2 引入了新的Java聚合框架Collections。



迭代器模式

- **模式应用**
  - ✓ Collection是所有Java聚合类的根接口。
  - ✓ 在JDK类库中，Collection的iterator()方法返回一个java.util.Iterator类型的对象，而其子接口java.util.List的listIterator()方法返回一个java.util.ListIterator类型的对象，ListIterator是Iterator的子类。它们构成了Java语言对迭代器模式的支持，Java语言的java.util.Iterator接口就是迭代器模式的应用。

迭代器模式

- **模式应用**
  - ✓ Java内置迭代器
    - 参考代码：**Chapter 20 Iterator\JavaIterator**
    - 下载地址：http://download.csdn.net/user/cflynn

演示.....

迭代器模式

- 模式扩展
  - ✓ Java迭代器
    - 在JDK中，Iterator接口具有如下3个基本方法:

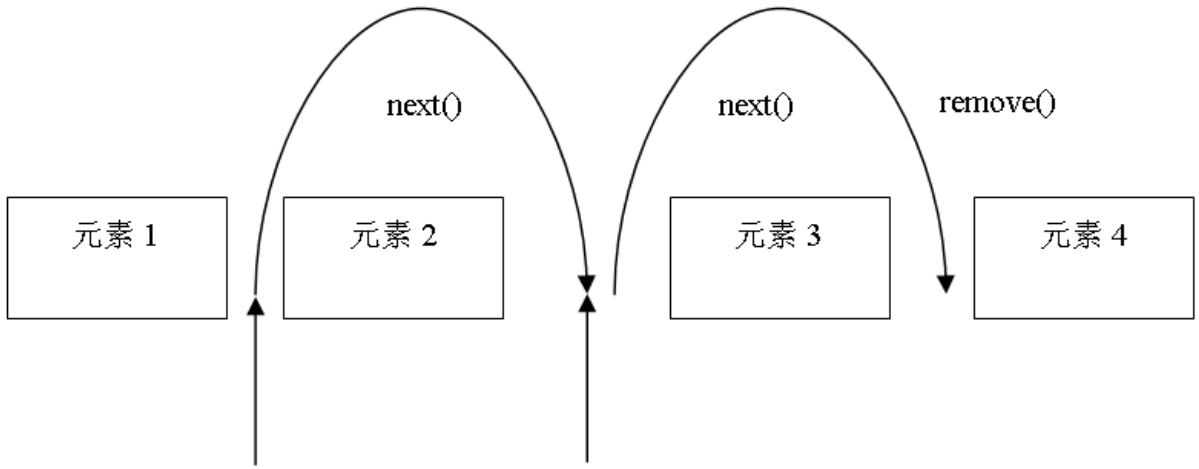
      (1) Object next()：通过反复调用next()方法可以逐个访问聚合中的元素。

      (2) boolean hasNext()：hasNext()方法用于判断聚合对象中是否还存在下一个元素，为了不抛出异常，必须在调用next()之前先调用hasNext()。如果迭代对象仍然拥有可供访问的元素，那么hasNext()返回true。

      (3) void remove()：用于删除上次调用next()时所返回的元素。

迭代器模式

- 模式扩展
  - ✓ Java迭代器
    - Java迭代器可以理解为它工作在聚合对象的各个元素之间，每调用一次next()方法，迭代器便越过下个元素，并且返回它刚越过的那个元素的地址引用。但是，它也有一些限制，如某些迭代器只能单向移动。在使用迭代器时，访问某个元素的唯一方法就是调用next()。



迭代器模式

- **模式扩展**
  - ✓ **Java迭代器**
    - 代码示例：

```
Iterator iterator = collection.iterator();   //collection是已实例
    化的集合对象
iterator.next();                              // 跳过第一个元素
iterator.remove();                            // 删除第一个元素
```

```
iterator.remove();
iterator.next();                              //该语句不能去掉
iterator.remove();
```

本章小结

- 迭代器模式提供一种方法来访问聚合对象，而不用暴露这个对象的内部表示，其别名为游标。迭代器模式是一种对象行为型模式。
- 迭代器模式包含四个角色：抽象迭代器定义了访问和遍历元素的接口；具体迭代器实现了抽象迭代器接口，完成对聚合对象的遍历；抽象聚合类用于存储对象，并定义创建相应迭代器对象的接口；具体聚合类实现了创建相应迭代器的接口。

本章小结

- 将遍历聚合对象中数据的行为提取出来，封装到一个迭代器中，通过专门的迭代器来遍历聚合对象的内部数据，这就是迭代器模式的本质。迭代器模式是“单一职责原则”的完美体现。
- 迭代器模式的主要优点在于它支持以不同的方式遍历一个聚合对象，还简化了聚合类，而且在同一个聚合上可以有多个遍历；其缺点在于增加新的聚合类需要对应增加新的迭代器类，类的个数成对增加，这在一定程度上增加了系统的复杂性。

本章小结

- 迭代器模式适用情况包括：访问一个聚合对象的内容而无须暴露它的内部表示；需要为聚合对象提供多种遍历方式；为遍历不同的聚合结构提供一个统一的接口。
- 在JDK类库中，Collection的iterator()方法返回一个Iterator类型的对象，而其子接口List的listIterator()方法返回一个ListIterator类型的对象，ListIterator是Iterator的子类。它们构成了Java语言对迭代器模式的支持，Java语言的Iterator接口就是迭代器模式的应用。

END

Thanks!