Design Patterns - Iterator Pattern

Iterator pattern is very commonly used design pattern in Java and .Net programming environment. This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

Iterator pattern falls under behavioral pattern category.

迭代器是Java和.Net程序环境下经常使用的一种设计模式。这种设计模式通常用来获取能顺序访问集合对元素象的方式，并且不需要了解底层是如何实现的。

迭代器模式属于行为型模式下的一种。

Implementation

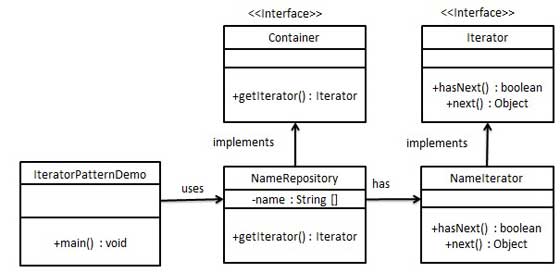
实现

We're going to create a *Iterator* interface which narrates navigation method and a *Container* interface which retruns the iterator . Concrete classes implementing the *Container* interface will be responsible to implement *Iterator* interface and use it

*IteratorPatternDemo*, our demo class will use *NamesRepository*, a concrete class implementation to print a *Names* stored as a collection in*NamesRepository*.

我们将创建一个Iterator接口，该接口描述迭代所需要的方法；紧接着声明了一个Container接口，该接口返回一个iterator对象。我们会创建具体的类实现Container接口和Iterator接口，并去使用该它们。

IteratorPatternDemo，我们的demo类将使用NamesRepository类，该类有一个集合存储要被打印的名字。



Step 1

第一步

Create interfaces.

创建接口

*Iterator.java*

public interface Iterator {

public boolean hasNext();

public Object next();

}

*Container.java*

public interface Container {

public Iterator getIterator();

}

Step 2

第二步

Create concrete class implementing the *Container* interface. This class has inner class *NameIterator* implementing the *Iterator* interface.

创建具体类实现Container接口，该类还有一个内部类NameIterator实现了Iterator接口。

*NameRepository.java*

public class NameRepository implements Container {

public String names[] = {"Robert" , "John" ,"Julie" , "Lora"};

@Override

public Iterator getIterator() {

return new NameIterator();

}

private class NameIterator implements Iterator {

int index;

@Override

public boolean hasNext() {

if(index < names.length){

return true;

}

return false;

}

@Override

public Object next() {

if(this.hasNext()){

return names[index++];

}

return null;

}

}

}

Step 3

Use the *NameRepository* to get iterator and print names.

使用NameRepository获得迭代器并且打印name。

*IteratorPatternDemo.java*

public class IteratorPatternDemo {

public static void main(String[] args) {

NameRepository namesRepository = new NameRepository();

for(Iterator iter = namesRepository.getIterator(); iter.hasNext();){

String name = (String)iter.next();

System.out.println("Name : " + name);

}

}

}

Step 4

Verify the output.

校验输出

Name : Robert

Name : John

Name : Julie

Name : Lora