**Design Patterns - Iterator Pattern**

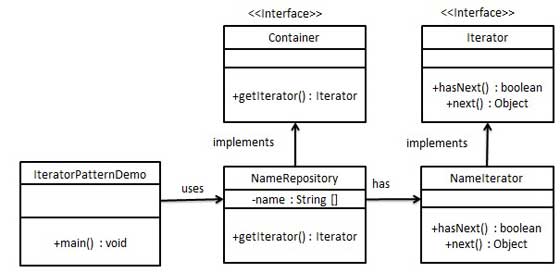
Iterator pattern is very commonly used design pattern in Java and .Net programming environment. This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation.

Iterator pattern falls under behavioral pattern category.

Implementation

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*.



Step 1

Create interfaces.

*Iterator.java*

public interface Iterator {

public boolean hasNext();

public Object next();

}

*Container.java*

public interface Container {

public Iterator getIterator();

}

Step 2

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

*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.

*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.

Name : Robert

Name : John

Name : Julie

Name : Lora