Design Pattern Lab Manual

Name: Jaivik Jariwala

Roll No.: 21BCP004

Division: 1

Group: 1

# Behavioral Design Pattern

|  |  |
| --- | --- |
| Sr. No | Name |
| 1 | Iterator |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |

## Iterator Behavioral Design Pattern

### Example 1 : Name Repository

Iterator.java

public interface *Iterator* {  
 public boolean hasNext();  
 public *Object* next();  
}

Container.java

public interface *Container* {  
 public *Iterator* getIterator();  
}

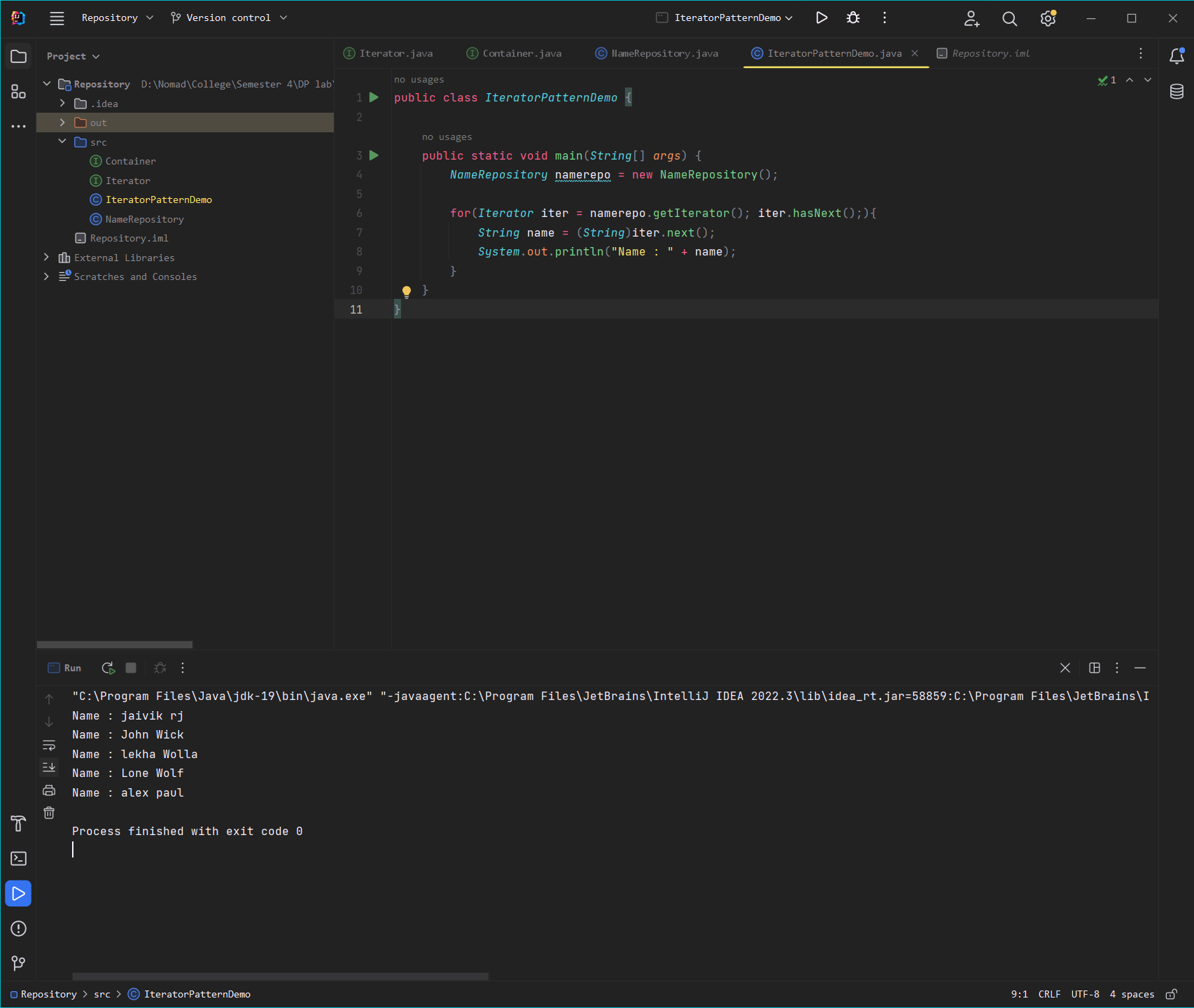
NameRepository.java

public class *NameRepository* implements *Container* {  
 public *String* names[] = {"jaivik rj" , "John Wick" ,"lekha Wolla" , "Lone Wolf" , "alex paul"};  
  
 *@Override* public *Iterator* getIterator() {  
 return new NameIterator();  
 }  
  
 private class *NameIterator* implements *Iterator* {  
  
 int i;  
  
 *@Override* public boolean hasNext() {  
  
 if(i < names.length){  
 return true;  
 }  
 return false;  
 }  
  
 *@Override* public *Object* next() {  
  
 if(this.hasNext()){  
 return names[i++];  
 }  
 return null;  
 }  
 }  
}

IteratorPatternDemo.java

public class *IteratorPatternDemo* {  
  
 public static void main(*String*[] *args*) {  
 *NameRepository* namerepo = new NameRepository();  
  
 for(*Iterator* iter = namerepo.getIterator(); iter.hasNext();){  
 *String* name = (*String*)iter.next();  
 *System*.out.println("Name : " + name);  
 }  
 }  
}

Output :



Example 2 : Shape maker

Shape.java

public class *Shape* {  
 private int id;  
 private *String* name;  
  
 public Shape(int *id* , *String name*){  
 this.id = *id*;  
 this.name = *name*;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int *id*) {  
 this.id = *id*;;  
 }  
  
 public *String* getName() {  
 return name;  
 }  
  
 public void setName(*String name*) {  
 this.name = *name*;  
 }  
  
 *@Override* public *String* toString() {  
 return " ID number : " +id+ " Shape is : " +name;  
 }  
}

ShapeStorage.java

public class *ShapeStorage* {  
  
 private *Shape*[]shapes = new Shape[3];  
 private int index;  
  
 public void addShape(*String name*){  
 int i = index++;  
 shapes[i]= new Shape(i ,*name*);  
 }  
  
 public *Shape*[] getShapes(){  
 return shapes;  
 }  
}

ShapeIterator.java

import *java.util.Iterator*;  
  
public class *ShapeIterator* implements *Iterator*<*Shape*> {  
  
 private *Shape*[] shapes;  
 int pos;  
  
 public ShapeIterator(*Shape* []*shapes*){  
 this.shapes = *shapes*;  
 }  
  
 *@Override* public boolean hasNext(){  
 if(pos >= shapes.length || shapes[pos] == null)  
 return false;  
 return true;  
 }  
  
 *@Override* public *Shape* next(){  
 return shapes[pos++];  
 }  
  
 *@Override* public void remove(){  
 if(pos <= 0)  
 throw new IllegalStateException("wrong place buddy");  
 if(shapes[pos-1] != null){  
 for(int i=pos-1; i <(shapes.length-1);i++){  
 shapes[i] = shapes[i+1];  
 }  
 shapes[shapes.length-1] = null;  
 }  
 }  
}

testIteratirPattern.java

public class *testIteraorPattern* {  
 public static void main(*String*[] *args*){  
 *ShapeStorage* store = new ShapeStorage();  
 store.addShape("triangle");  
 store.addShape("square");  
 store.addShape("n-gon");  
  
 *ShapeIterator* iterator = new ShapeIterator(store.getShapes());  
 while (iterator.hasNext()){  
 *System*.out.println(iterator.next());  
 }  
 *System*.out.println("remove");  
 iterator = new ShapeIterator(store.getShapes());  
 while (iterator.hasNext()){  
 *System*.out.println(iterator.next());  
 iterator.remove();  
 }  
 }  
}

Output

