迭代器模式实验

**【实验目的及要求】**

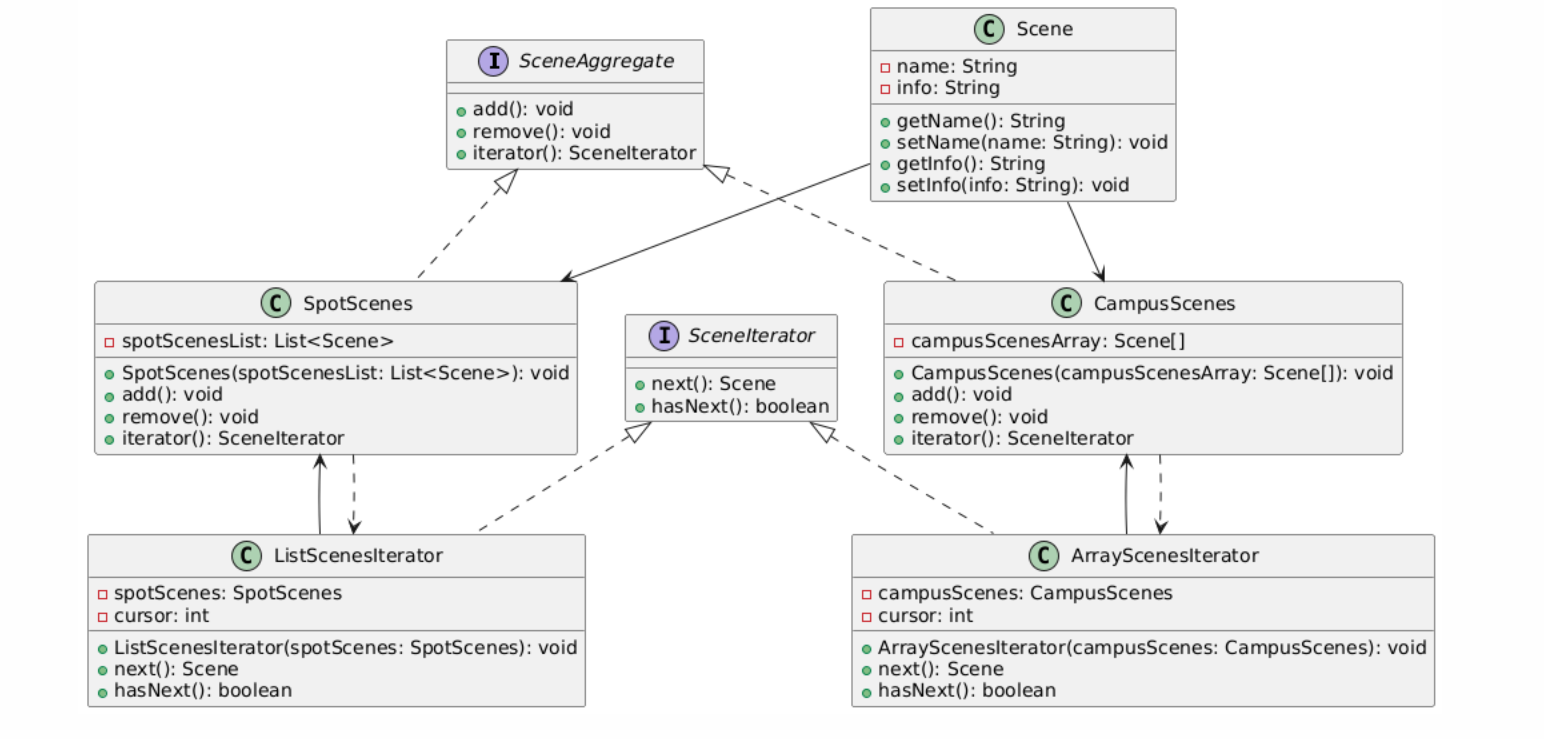
1. 熟悉迭代器模式的定义；
2. 掌握迭代器模式角色及作用，熟悉迭代器模式结构图；
3. 掌握迭代器模式编程。

**【实验内容】**

**景点介绍：在程序中考虑两种景点：一种是旅游景点，用List存放；一种是校区景点，用数组存放。例如：旅游景点有广州、深圳、肇庆，校区有龙洞校区、肇庆校区、清远校区。请使用迭代器模式实现显示所有景点信息。**

**【实验步骤】**

1. **类图**

****

1. **实现代码**
2. **景点类（Scene）**

**package** design.iterator;

**public** **class** Scene {

**private** String name; //景点名称

**private** String info; //景点介绍

**public** Scene(String name, String info) {

**this**.name = name;

**this**.info = info;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getInfo() {

**return** info;

}

**public** **void** setInfo(String info) {

**this**.info = info;

}

}

1. **抽象聚合类（SceneAggregate）**

**package** design.iterator;

**public** **interface** SceneAggregate{

**public** **void** add();

**public** **void** remove();

**public** SceneIterator iterator();

}

1. **抽象迭代器类（SceneIterator）**

**package** design.iterator;

**public** **interface** SceneIterator {

**public** Scene next();

**public** **boolean** hasNext();

}

1. **具体聚合类**

**CampusScenes类，以数组形式存放校区信息**

**package** design.iterator;

//以数组形式存放校区信息

**public** **class** CampusScenes **implements** SceneAggregate {

protected Scene[] campusScenesArray;

public CampusScenes(Scene[] campusScenesArray) {

this.campusScenesArray = campusScenesArray;

}

public Scene[] getCampusScenesArray() {

return campusScenesArray;

}

@Override

public void add() {

System.out.println("添加校区");

}

@Override

public void remove() {

System.out.println("删除校区");

}

@Override

public SceneIterator iterator() {

return new ArrayScenesIterator(this);

}

}

**SpotScenes类，以List形式存放旅游景点信息**

**package** design.iterator;

//以List存放旅游景点

**public** **class** SpotScenes **implements** SceneAggregate {

protected List<Scene> spotScenesList;

public SpotScenes(List<Scene> spotScenesList) {

this.spotScenesList = spotScenesList;

}

public List<Scene> getSpotScenesList() {

return spotScenesList;

}

@Override

public void add() {

System.out.println("添加景点");

}

@Override

public void remove() {

System.out.println("删除景点");

}

@Override

public SceneIterator iterator() {

return new ListScenesIterator(this);

}

}

1. **具体迭代器类**

**ArrayScenesIterator类，实现对数组聚合数据遍历**

**package** design.iterator;

**public** **class** ArrayScenesIterator**implements** SceneIterator {

private CampusScenes campusScenes; //维持对具体聚合类的引用

private int cursor;

public ArrayScenesIterator(CampusScenes campusScenes) {

this.campusScenes = campusScenes;

cursor = 0;

}

@Override

public Scene next() {

return campusScenes.getCampusScenesArray()[cursor++];

}

@Override

public boolean hasNext() {

return cursor < campusScenes.getCampusScenesArray().length

&& campusScenes.getCampusScenesArray()[cursor] != null;

}

}

**ListScenesIterator类，实现对List聚合数据遍历**

**package** design.iterator;

**public** **class** ListScenesIterator **implements** SceneIterator {

private SpotScenes spotScenes;

private int cursor;

public ListScenesIterator(SpotScenes spotScenes) {

this.spotScenes = spotScenes;

cursor = 0;

}

@Override

public Scene next() {

return spotScenes.getSpotScenesList().get(cursor++);

}

@Override

public boolean hasNext() {

return cursor < spotScenes.getSpotScenesList().size();

}

}

1. **客户端测试类Client**

**package** design.iterator;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

SceneAggregate sceneAggregate;

SceneIterator sceneIterator;

//创建校区聚合信息

Scene campusScene1 = new Scene("龙洞校区", "广州");

Scene campusScene2 = new Scene("肇庆校区","肇庆");

Scene campusScene3 = new Scene("清远校区","清远");

Scene[] campusScenesArray = {campusScene1, campusScene2, campusScene3};

sceneAggregate = new CampusScenes(campusScenesArray);

//实例化校区迭代对象

sceneIterator = sceneAggregate.iterator();

//遍历校区信息

while (sceneIterator.hasNext()) {

Scene campusScene = sceneIterator.next();

System.out.println(campusScene.getName()+","+campusScene.getInfo());

}

System.out.println("-----------------------------------");

//创建旅游景点聚合信息

List<Scene> spotScenesList = new ArrayList<>();

Scene spotScene1 = new Scene("广州", "这是广州");

Scene spotScene2 = new Scene("深圳", "这是深圳");

Scene spotScene3 = new Scene("肇庆", "这是肇庆");

spotScenesList.add(spotScene1);

spotScenesList.add(spotScene2);

spotScenesList.add(spotScene3);

sceneAggregate = new SpotScenes(spotScenesList);

//实例化旅游景点迭代对象

sceneIterator = sceneAggregate.iterator();

//遍历旅游景点信息

while (sceneIterator.hasNext()) {

Scene spotScene = sceneIterator.next();

System.out.println(spotScene.getName()+","+spotScene.getInfo());

}

}

}

**【思考】**

1. 使用JDK中提供的Iterator、Iterable等接口简化前面的代码。

public class CampusScenes implements Iterable<Scene> {

private Scene[] campusScenesArray;

public CampusScenes(Scene[] campusScenesArray) {

this.campusScenesArray = campusScenesArray;

}

public Scene[] getCampusScenesArray() {

return campusScenesArray;

}

@Override

public Iterator<Scene> iterator() {

return new ArrayScenesIterator(this); // 返回 ArrayScenesIterator 实例

}

}

public class ListScenesIterator implements Iterator {  
 private SpotScenes spotScenes;  
 private int cursor = 0;  
  
 public ListScenesIterator(SpotScenes spotScenes) {  
 this.spotScenes = spotScenes;  
 }  
  
 @Override  
 public boolean hasNext() {  
 return cursor < spotScenes.getSpotScenesList().size();  
 }  
  
 @Override  
 public Object next() {  
 return spotScenes.getSpotScenesList().get(cursor++);  
 }  
}

//实现Iterator接口，不用自己再写自定义的Iterator接口  
public class ArrayScenesIterator implements Iterator {  
  
 private CampusScenes campusScenes; // 持有对具体聚合类的引用  
 private int cursor = 0;  
  
 public ArrayScenesIterator(CampusScenes campusScenes) {  
 this.campusScenes = campusScenes;  
 }  
  
 @Override  
 public boolean hasNext() {  
 return cursor < campusScenes.getCampusScenesArray().length && campusScenes.getCampusScenesArray()[cursor] != null;  
 }  
  
 @Override  
 public Object next() {  
 return campusScenes.getCampusScenesArray()[cursor++];  
 }  
}

public class SpotScenes implements Iterable<Scene> {  
 private List<Scene> spotScenesList;  
  
 public SpotScenes(List<Scene> spotScenesList) {  
 this.spotScenesList = spotScenesList;  
 }  
  
 public List<Scene> getSpotScenesList() {  
 return spotScenesList;  
 }  
  
 @Override  
 public Iterator<Scene> iterator() {  
 return new ListScenesIterator(this); // 返回 ListScenesIterator 实例  
 }  
}

public class Test {  
 public static void main(String[] args) {  
 Iterator<Scene> iterator;  
 // 创建校区聚合信息  
 Scene campusScene1 = new Scene("龙洞校区", "广州");  
 Scene campusScene2 = new Scene("肇庆校区", "肇庆");  
 Scene campusScene3 = new Scene("清远校区", "清远");  
 Scene[] campusScenesArray = {campusScene1, campusScene2, campusScene3};  
  
 CampusScenes campusScenes = new CampusScenes(campusScenesArray);  
 iterator = campusScenes.iterator();  
 while (iterator.hasNext()) {  
 Scene next = iterator.next();  
 System.*out*.println(next.getName() + "," + next.getInfo());  
 }  
 System.*out*.println("-----------------------------------");  
 // 创建旅游景点聚合信息  
 List<Scene> spotScenesList = new ArrayList<>();  
 Scene spotScene1 = new Scene("广州", "这是广州");  
 Scene spotScene2 = new Scene("深圳", "这是深圳");  
 Scene spotScene3 = new Scene("肇庆", "这是肇庆");  
 spotScenesList.add(spotScene1);  
 spotScenesList.add(spotScene2);  
 spotScenesList.add(spotScene3);  
  
 SpotScenes spotScenes = new SpotScenes(spotScenesList);  
  
 // 遍历旅游景点信息  
 iterator = spotScenes.iterator();  
 while (iterator.hasNext()) {  
 Scene next = iterator.next();  
 System.*out*.println(next.getName() + "," + next.getInfo());  
 }  
 }  
}

**【总结】**

|  |  |
| --- | --- |
| 实验总结  （不少于100字） | 在本次实验中，我深入学习了迭代器模式的定义和应用。首先，理解了迭代器模式的角色与作用，主要包括聚合类（SceneAggregate）、具体聚合类（CampusScenes、SpotScenes）、迭代器接口（SceneIterator）以及具体迭代器类（ArrayScenesIterator、ListScenesIterator）。通过这些角色的协作，迭代器模式实现了对不同集合数据的统一遍历操作。接着，我通过编程实现了迭代器模式，编写了具体的聚合类和迭代器类，掌握了如何通过迭代器对数组和列表进行遍历。此外，我也对迭代器模式的结构图进行了分析，明确了各个角色之间的关系。通过本次实验，我不仅加深了对迭代器模式的理解，还提升了设计模式编程的实际应用能力。 |
| 学号 | 22154A206 |
| 姓名 | 冯艳芳 |
| 成绩 |  |