学号 E21514033 专业 软件工程 姓名

实验日期 **2018.5.31**  教师签字 成绩

实验报告

【实验名称】 迭代器模式实验

【实验目的】

1. 熟悉迭代器模式的实验
2. 能够自行建立链表并实现操作

【实验原理】

1. 节点类中实现前后节点指针
2. 容器中实现迭代器和节点操作

【实验内容】

Node节点：

class Node {

int a;

double b;

Node next = null;

Node pre = null;

Node() {

}

Node(int a, double b) {

this.a = a;

this.b = b;

}

public void setNext(Node next) {

12 this.next = next;

}

public void setPre(Node pre) {

this.pre = pre;

@Override

public Iterator getIterator() {

return iterator;

}

}

}

}

迭代器接口及其实现类：

interface Iterator {

Node getNext();

Node getPre();

Node getFirst();

Node getLast();

}

class ConcreteIterator implements Iterator {

Node first = new Node();

Node last = new Node();

Node current = first;

ConcreteIterator() {

}

@Override

public Node getNext() {

current = current.next;

if (current == last) {

return null;

}

return current;

}

@Override

public Node getPre() {

current = current.pre;

if (current == first) {

return null;

}

return current;

}

@Override

public Node getFirst() {

return first;

}

@Override

public Node getLast() {

return last;

}

}

容器接口及其实现类：

interface Container {

void addNode(Node n);

Iterator getIterator();

}

class ConcreteContainer implements Container {

Iterator iterator;

ConcreteContainer() {

iterator = new ConcreteIterator();

}

@Override

public void addNode(Node n) {

if (iterator.getFirst().next == null) {

iterator.getFirst().setNext(n);

iterator.getLast().setPre(n);

n.setPre(iterator.getFirst());

n.setNext(iterator.getLast());

} else {

iterator.getLast().pre.setNext(n);

n.setPre(iterator.getLast().pre);

n.setNext(iterator.getLast());

iterator.getLast().setPre(n);

}

}

@Override

public Iterator getIterator() {

return iterator;

}

}

【小结或讨论】

1. 使用迭代器模式，不需要在列表类中实现访问操作，降低了耦合性。
2. 通过迭代器能够轻松快捷的实现访问操作。