双循环链表的java实现——《算法》读书笔记

####这是一个双链表环，从头插入元素,可以实现从任意地方删除元素,测试是约瑟夫问题。

添加元素的时间复杂度为O(1)，删除元素的时间复杂度为O(1).由于约瑟夫问题是在不停的删除元素现在假设有n个元素，每hop个元素自杀一次。总共删除n-1次,每删除一次走hop步。所以约瑟夫问题时间复杂度为 O(hop*(n-1))=O(n*
op)。

```
import java.util.Iterator;

/**
 * Created by dog on 3/25/16.
 * 这是一个双链表环，从头插入元素,可以实现从任意地方删除元素
 */
public class CircularDoubleLinked<Item> implements Iterable<Item>{

    private Node head;
    private Node tail;
    private int N;

    public CircularDoubleLinked(){
        head = new Node();
        tail = new Node();

    }

    private class Node{
        Item item;
        Node left;
        Node right;
    }

    public int size(){
        return N;
    }

    public boolean isEmpty(){
        return size()==0;
    }

    public void push(Item item){
        if(isEmpty()){
            Node newFirst = new Node();
            newFirst.item=item;

            head.right = newFirst;
            tail.left=newFirst;

        }else {

            Node newFirst = new Node();
            newFirst.item=item;
            newFirst.right = head.right;
            head.right.left=newFirst;
        newFirst.left = tail.left;
        tail.left.right=newFirst;
        head.right=newFirst;

    }
    N++;
}

public Item remove(Node node){

    if(node!=null) {
        Item item = node.item;
        if (node == head.right) {
            Node last = node.left;
            Node next = node.right;
            last.right = next;
            next.left = last;

            head.right = node.right;

            node.right = null;
            node.left = null;
            node = null;
        } else if (node == tail.left) {
            Node last = node.left;
            Node next = node.right;
            last.right = next;
            next.left = last;

            tail.left = node.left;

            node.right = null;
            node.left = null;
            node = null;
        } else {
            Node last = node.left;
            Node next = node.right;
            last.right = next;
            next.left = last;
            node.right = null;
            node.left = null;
            node = null;
        }
        N--;
        return item;
    }else {
        return null;
    }

}
@Override
public ListIterator iterator() {
    return new ListIterator();
}
```

```
private class ListIterator implements Iterator<Item>{

    Node current = head.right;
    Node follow = head;
    @Override
    public boolean hasNext() {
        if(current!=null && size()>0){
            return true;
        }else {
            return false;
        }

    }

    @Override
    public Item next() {
        Item item = current.item;
        //System.out.println("current:地址"+current);
        follow=current;
        current=current.right;
        return item;
    }

    @Override
    public void remove() {
        Node temp = follow.right ;
        System.out.println(CircularDoubleLinked.this.remove(follow));
        follow = temp;
    }
}

public static void main(String[]args){


    //丢手绢问题的实现
    //test
    //每hop个人报数一次
    int N = 41;
    int hop = 3;

    CircularDoubleLinked d = new CircularDoubleLinked<Integer>();

    //添加元素
    for(int i=N;i>0;i--) d.push(i);

    //开始游戏
    int k=1;
    for(Iterator i = d.iterator() ;d.size()>1;) {
        i.next();
        if(k%(hop+1)==hop){
            i.remove();
            k=0;
            }
            k++;
        }

        System.out.println("剩下："+d.head.right.item);

    }
}
```

原文链接：双循环链表的java实现——《算法》读书笔记