1.应用场景

Josephu(约瑟夫问题）

Josephu　问题为：设编号为1，2，… n的n个人围坐一圈，约定编号为k（1<=k<=n）的人从1开始报数，数到m 的那个人出列，它的下一位又从1开始报数，数到m的那个人又出列，依次类推，直到所有人出列为止，由此产生一个出队编号的序列。

可以使用环形链表来解决

2.单向环形链表

1.创建

尾节点指向第一个节点（非头结点）

当只有一个节点的时候，自己指向自己

2.功能

1.添加节点

```
public void add(Node node) {
    if (headNode.next == null) {
        headNode.next = node;
        node.next = node;
        footNode = node;
    } else {
        footNode.next = node;
        node.next = headNode.next;
        footNode = node;
    }
}
```

2.删除节点

删除节点是要考虑特殊情况:

1. 单节点的链表，删除最后一个节点
2. 删除尾结点时，前移尾结点标记
3. 删除的是第一个节点是，需要将尾节点重新指向新的第一个节点

```
public void deleteNode(Node node) {
    Node temp = headNode;
    if (headNode.next == null) {
        System.out.println("空链表");
        return;
    }
    while (!Objects.equals(temp, footNode)) {
        if (temp.next.getNo() == node.getNo()) {
            //只有一个节点时,将链表置空
```

```
            if (temp.next.next.equals(temp.next)) {
                headNode.next = null;
                node.next = null;
                footNode = null;
                return;
            }
            //删除尾节点时，尾结点的标记前移
            if (footNode.equals(temp.next)) {
                temp.next = headNode.next;
                footNode = temp;
                return;
            }
            //删除节点是第一个节点时，需要将最后一个节点指向新的第一个节点
            if(temp.next.equals(headNode.next))
            {
                temp.next = temp.next.next;
                footNode.next=temp.next;
                return;
            }
            temp.next = temp.next.next;
            return;
        }
        temp = temp.next;
    }
}
```

3.模拟约瑟夫问题

实现代码

```
package 线性结构.链表LinkedList.单向环形链表;

import java.util.Scanner;

/**
 * @author jndeng
 * @create 2019-11-13 21:01
 */
public class JosephuGame {
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        System.out.println("游戏开始");
        System.out.println("输入人数 n=");
        int n = Integer.parseInt(scanner.nextLine());
        System.out.println("输入开始数数的人编号：k=");
        int k = Integer.parseInt(scanner.nextLine());
        System.out.println("输入数到m出列： m=");
        int m=Integer.parseInt(scanner.nextLine());
```

```
        SingleCircleLinkedList singleCircleLinkedList=new SingleCircleLinkedList();


        for (int i=0;i<n;i++)
        {
            Node node = new Node(i + 1);
            singleCircleLinkedList.add(node);
        }

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

        Node headNode = singleCircleLinkedList.getHeadNode();
        Node node = headNode;
        for (int i = 0; i < k; i++) {
            node = node.next;
        }
        while (node != null) {
            for (int i = 0; i < m; i++) {
                node = node.next;
            }
            System.out.println(node);
            singleCircleLinkedList.deleteNode(node);
            node=node.next;
        }
    }
}
```

结果演示

```
游戏开始
输入人数 n=
5
输入开始数数的人编号：k=
1
输入数到m出列： m=
2
Node{No=1}
Node{No=2}
Node{No=3}
Node{No=4}
Node{No=5}
------------------
Node{No=3}
Node{No=1}
Node{No=5}
```

Node{No=4}
Node{No=2}