## 2.3链表

Node类

Link类

#### 定义node节点类

**package** NodeList;  
  
**public class** Node {  
 **public** NodeData **nodeData**;*//保存数据* **public** Node **next**;*//用于保存下一个节点* **public** Node() {  
 **this**.**nodeData**=**null**;  
 **this**.**next** = **null**;  
 }  
  
 **public** Node(NodeData nodeData) {  
 **this**.**nodeData** = nodeData;  
 **this**.**next** = **null**;  
 }  
  
 **public** NodeData getNodeData() {  
 **return nodeData**;  
 }  
  
 **public void** setNodeData(NodeData nodeData) {  
 **this**.**nodeData** = nodeData;  
 }  
  
 **public** Node getNext() {  
 **return next**;  
 }  
  
 **public void** setNext(Node next) {  
 **this**.**next** = next;  
 }  
}

#### 定义Link链表类

**package** NodeList;  
  
**public class** LinkList {  
 Node **head**=**new** Node();*//定义头结点  
  
  
 /\*  
 \* 单链表常见操作  
 \* \*/  
 //头插法添加链表  
 /\*  
 \* 每新插入一个结点放在头结点的后面  
 \* 创建的链表为逆序  
 \* \*/* **public** Node headAddNodeList(String name,**int** number){  
 NodeData nodeData=**new** NodeData(name,number);  
 Node node=**new** Node(nodeData);  
 node.**next**=**head**.**next**;  
 **head**.**next**=node;  
 **return head**;  
 }  
 *//尾插法添加链表  
 /\*  
 \* 每新插入一个结点都放在尾节点的后面  
 \* 创建的链表为正序  
 \* \*/* **public** Node tailAddNodeList(String name,**int** number ){  
 NodeData nodeData=**new** NodeData(name,number);  
 Node node=**new** Node(nodeData);  
 Node temp=**head**;  
 **while** (temp.**next**!=**null**){  
 temp=temp.**next**;  
 }  
 temp.**next**=node;  
 *//node.next=null;* **return head**;  
 }  
  
 */\*  
 \* 链表遍历  
 \* \*/* **public void** selNodeList(Node head){  
 Node temp=head;  
 **int** cnt=1;  
 **while** (temp.**next**!=**null**){  
 temp=temp.**next**;  
 NodeData nodeData=temp.getNodeData();  
 System.***out***.println(**"第"**+cnt+**"个结点\_名字："**+nodeData.getName()+**"\_\_数字:"**+nodeData.getNumber());  
 cnt++;  
 }  
  
 }  
  
 */\*  
 \* 将数据插入到指定位置  
 \*  
 \* \*/* **public void** insertNodeListByindex(String name,**int** number,**int** index){  
 Node temp=**head**;  
 **int** cnt=1;  
 **while**(index!=cnt&&temp.**next**!=**null**){  
 temp=temp.**next**;  
 cnt++;  
 }  
 NodeData nodeData=**new** NodeData(name,number);  
 Node node=**new** Node(nodeData);  
 node.**next**=temp.**next**;  
 temp.**next**=node;  
 }  
 */\*  
 \* 删除某个位置的链表  
 \* \*/* **public void** deleteNodeListByindex(**int** index){  
  
 **int** cnt=1;  
 Node pre=**head**;*//记录删除的前一个结点* Node cur=pre.**next**;*//记录删除的当前节点* **while**(index!=cnt&&cur.**next**!=**null**){*//找到要删除的结点* pre=cur;*//记录要删除结点的前一个结点* cur=pre.**next**;  
 cnt++;  
 }  
 pre.**next**=cur.**next**;  
 }  
  
 */\*  
 \* 返回链表长度  
 \* \*/* **public void** getNodeListLen(){  
 **int** cnt=0;  
 Node temp=**head**;  
 **while**(temp.**next**!=**null**){  
 temp=temp.**next**;  
 cnt++;  
 }  
 System.***out***.println(**"链表长度为："**+cnt);  
 }  
 */\*  
 \* 头插法链表反转  
 \* 就地反转 别不1->2->3 将1，2指针由2指向1  
 \* head 1 2 3  
 \* head 2 1 3  
 \* head 3 2 1  
 \* \*/* **public void** reverseNodeList(){  
 Node cur=**head**.**next**;  
 Node n=cur.**next**;  
 *//头插法反转结点* Node Phead=**new** Node();*//反转头结点* **while**(cur.**next**!=**null**){  
 cur.**next**=n.**next**;  
 n.**next**=**head**.**next**;  
 **head**.**next**=n;  
 n=cur.**next**;  
 }  
 }  
  
  
 */\*  
 \* 递归反转  
 \* \*/* **public** Node reverseList(Node cur) {  
 **if** (cur==**null**)  
 **return** cur;  
 Node next = cur.**next**;  
 Node new\_head = reverseList(cur.**next**);  
 **if**(new\_head==**null**){  
 new\_head=**new** Node();  
 new\_head.**next**=cur;  
 }**else if**(cur==**head**) {  
 **return** new\_head;  
 }**else** {  
 next.**next** = cur;  
 cur.**next** = **null**;  
 }  
 **return** new\_head;  
  
 }  
}

#### 单链表常用方法

**迭代和递归**

##### 3.1单链表反转

###### A:头插法链表反转==迭代

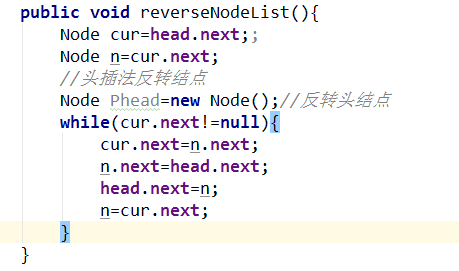
思路：把链表分为两个部分一个为反转部分一个为未反转部分

从前往后开始处理

流程：

*\* head 1 2 3（原链表）  
\* head 2 1 3  
\* head 3 2 1*

具体方法：



java不支持函数参数默认值

###### B：递归

**public** Node reverseList(Node cur) {  
 **if** (cur==**null**)  
 **return** cur;  
 Node next = cur.**next**;  
 Node new\_head = reverseList(cur.**next**);  
 **if**(new\_head==**null**){  
 new\_head=**new** Node();  
 new\_head.**next**=cur;  
 }**else if**(cur==**head**) {  
 **return** new\_head;  
 }**else** {  
 next.**next** = cur;  
 cur.**next** = **null**;  
 }  
 **return** new\_head;  
  
}