# 单链表抽象数据类型

## 抽象数据类型

**Node节点类：**

```
public class Node<T> {
    public T data;
    public Node<T> next;
    public Node(T data,Node<T> next){
        this.data = data;
        this.next = next;
    }
    public Node(){
        this.data = null;
        this.next = null;
    }
    public String toString(){
        return this.data.toString();
    }
}
```

**链表类：**

```
public class SinglyList<T> {
     public Node<T> head;

     public SinglyList(){
         this.head = new Node<T>();
     }
     public SinglyList(T[] values){
         this();
         Node<T> rear = this.head;
         for(int j = 0;j<values.length;j++){
             rear.next = new Node<T>(values[j],null);
             rear = rear.next;
         }
     }
     public String toString() {
         String str = this.getClass().getName() + "(";
         Node<T> p = this.head.next;
         while(p!=null){
             str += p.data;
             if(p.next!=null){str += ",";}
             p = p.next;
         }
         return str+")";
     }
     public boolean isEmpty(){
         return this.head.next == null;
     }
     public T get(int i){
         Node<T> p = this.head.next;
```

```
30          for(int j=0;p!=null&&j<i;j++){
31              p = p.next;
32          }
33          return ( i>=0 && p!=null)?p.data:null;
34      }
35      public void set(int i,T x){
36          Node<T> p = this.head.next;
37          for(int j=0;j<i;j++){
38              p = p.next;
39          }
40          p.data = x;
41      }
42      public Node<T> insert(int i,T x){
43          if(x == null){
44              throw new NullPointerException();
45          }
46          Node<T> p = this.head;
47          for(int j = 0;p.next!=null&&j<i;j++){
48              p = p.next;
49          }
50          p.next = new Node<T>(x,p.next);
51          return p.next;
52      }
53
54      public Node<T> remove(int i) {
55          Node<T> p = this.head;
56          for (int j = 0;p!=null&&j<i;j++ ){
57                p = p.next;
58          }
59          if(i<=0 && p.next!=null) {
60              Node<T> old = p.next;
61              p.next = p.next.next;
62              return old;
63          }
64          return null;
65      }
66  }
```