概念：

顺序表：顺序存储的线性表。顺序存储使用的是一组地址连续的存储单元（数组）依次存放各个元素数据。例如：ArrayList

特点：

1.在线性表中逻辑相邻的数据元素，在物理存储位置上也是相邻的。

2.存储密度高，但需要预先分配足够的应用的存储空间，会造成空间浪费。

存储密度=数据本身值所需的空间/数据实际所占用空间

3.便于随机存取。

随机取（访问）的时候，输入要访问的位置I, array[i]就是访问数值；随机存就直接存入到数组的末尾。

4.不便于插入和删除。

插入和删除的时候需要大量移动数据。

功能：

1.clear() 清空

2.isEmpty() 判断是否位空，返回true/false

3.length() 当前顺序表长度，返回int

4.display() 打印顺序表

5.add(Object x) 添加数据x

6.insert(int I,Object x) 按位置i插入数据元素x

7.remove(int i) 按位置i删除元素

8.get(int i) 按位置i取元素

9.indexOf(Object x) 查找数据元素x的位置索引

实现：

private Object[] listElem; //用于内部存储的数组

private int curLen; //当前顺序表的长度，注意并不是数组的最大长度。

public Sqlist(int maxSize){ //顺序表的初始化，这里给出带参数的构造方法

this.curLen=0; //实际arraylist是有无参数构造的，初始的存储数组大小为10

this.listElem=new Object[maxSize];

}

public void clear(){ //清空，这里清空并不能真的清空，因为计算机中数据就是直接覆盖使用的

curLen=0; //所以只需要把当前的顺序表长度初始为0就好了

listElem = new Object[1];

}

public boolean isEmpty(){ //判断是否为空，判断当前长度为0返回true。

if(curLen==0)

return true;

else

return false;

}

public int length(){ //返回当前顺序表的长度，注意这里并不是内在数组的大小，因为数组有的位置可能为空

return curLen;

}

public void add(Object x){ //注意修改当前长度curLen

listElem[curLen]=x;

curLen++;

}

public Object get(int i)throws Exception{

if(i<1||i>curLen)

throw new Exception("第"+i+"元素不存在");

return listElem[i-1];

}

public void insert(int i,Object x)throws Exception{

if(curLen==listElem.length){

throw new Exception("顺序表已满！");

}

for(int j=curLen;j>i-1;j--){ //在插入数据的时候，预插入位置后的数据要向后移动一位

listElem[j]=listElem[j-1]; //在移动的过程中，注意一定要从最后的那个元素开始向后移动1位要是从前边移动的话，后边的数据都会被覆盖

}

listElem[i-1]=x; //移动完毕后，直接覆盖原来那个位置的元素就行

curLen+=1; //修改当前长度，插入长度+1

}

public void remove(int i){

if(i<1||i>curLen)

System.out.println("位置不合法！");

for(int j=i;j<curLen;j++){ //删除，删除后边的数据都要向前移动一位

listElem[j-1]=listElem[j]; //注意，移动的时候一定要从前边开始移动，从后边的话数据都会被覆盖。

}

curLen--; //注意修改长度，删除 -1

}

public int indexOf(Object x){

for(int i=0;i<curLen;i++){

if(x.equals(listElem[i])) //注意，判断相等的时候要用equals(),Object是引用类型 用==的话会出错

return i+1; //要是基础数据类型可以用 ==

}

return -1; //找不到的时候返回-1

}

public void display(){

System.out.print("[");

for(int i=0;i<curLen;i++){

if(i==curLen-1)

System.out.print(listElem[i]);

else

System.out.print(listElem[i]+",");

}

System.out.println("]");

}

}

**ArrayList**

ArrayList就是典型的顺序表。ArrayList实现了ICollection和IList接口。刚才列出的基本功能除了add是ICollection的方法，其他全是IList接口中的方法。ArrayList内部封装了一个Object类型的数组。优点是便于对集合进行快速的随机访问，如果经常需要根据索引位置访问集合中的对象，使用由ArrayList类实现的List集合的效率较好。

ArrayList的方法：

1.构造器

（1）用一个ICollection对象来构造，并将该集合的元素添加到ArrayList

（2）构造器参数为int，定义内部存储数组的大小

（3）默认无参数，大小为10

2. 添加、删除、插入元素

Add方法用于添加一个元素到当前列表的末尾

AddRange方法用于添加一批元素到当前列表的末尾

Remove方法用于删除一个元素，通过元素本身的引用来删除

RemoveAt方法用于删除一个元素，通过索引值来删除

RemoveRange用于删除一批元素，通过指定开始的索引和删除的数量来删除

Insert用于添加一个元素到指定位置，列表后面的元素依次往后移动

InsertRange用于从指定位置开始添加一批元素，列表后面的元素依次往后移动

注意：添加只能在当前列表尾添加，要是向在中间添加需要使用Insert

3.清空和是否存在

Clear方法用于清除现有所有的元素

Contains方法用来查找某个对象在不在列表之中

4.固定长度到实际需要大小

TrimSize方法，这个方法用于将ArrayList固定到实际元素的大小，当动态数组元素确定不在添加的时候，可以调用这个方法来释放空余的内存。

5.转数组

ToArray()方法

6.动态扩容

每当执行添加元素的方法，都会检查内部数组的容量是否不够了，如果是，它就会以当前容量的两倍来重新构建一个数组，将旧元素Copy到新数组中，然后丢弃旧数组，在这个临界点的扩容操作，应该来说是比较影响效率的。

注意：因为ArrayList中是Object类型的数组，所以对一般引用类型的存储影响不大，要是对于基本数据类型就需要频繁的装箱拆箱，十分影响效率。