# List

## ArrayList

### 类的实现：

public class ArrayList<E> extends AbstractList<E>  
 implements List<E>, RandomAccess, Cloneable, java.io.Serializable

//RandomAccess for循环更快，反之迭代器更快（ArrayList和LinkedList）

### 基本结构：

transient Object[] elementData;//数据(自定义了序列化方式)

private int size;//大小

protected transient int modCount = 0;//改动次数用于避免多线程

### 构造方法：

public ArrayList() {//无参构造空的Object[]  
 this.elementData = *DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA*;  
}

public ArrayList(int initialCapacity) {//定义初始化大小  
 this.elementData = new Object[initialCapacity];  
}

public ArrayList(Collection<? extends E> c) {//转为ArrayList  
 elementData = c.toArray();  
 if ((size = elementData.length) != 0) {  
 if (elementData.getClass() != Object[].class)  
 elementData = Arrays.*copyOf*(elementData, size, Object[].class);  
 } else {  
 this.elementData = *EMPTY\_ELEMENTDATA*;  
 }  
}

### 其他：

public void ensureCapacity(int minCapacity) {//扩充容量  
 int minExpand = (elementData != *DEFAULTCAPACITY\_EMPTY\_ELEMENTDATA*)  
 ? 0: *DEFAULT\_CAPACITY*;   
 if (minCapacity > minExpand) {  
 ensureExplicitCapacity(minCapacity);  
 }  
}

public boolean contains(Object o) {//判断包含某元素=查找第一次出现的位置  
 return indexOf(o) >= 0;  
}

public int indexOf(Object o) {//循环找到元素的位置  
 if (o == null) {  
 for (int i = 0; i < size; i++)  
 if (elementData[i]==null)  
 return i;  
 } else {  
 for (int i = 0; i < size; i++)  
 if (o.equals(elementData[i]))  
 return i;  
 }  
 return -1;  
}

public E get(int index) {//得到索引的第几个  
 rangeCheck(index);   
 return elementData(index);  
}

private void rangeCheck(int index) {//检查当前索引是否正确  
 if (index >= size)  
 throw new IndexOutOfBoundsException(outOfBoundsMsg(index));  
}

public boolean retainAll(Collection<?> c) {//清除包含集合c元素的  
 Objects.*requireNonNull*(c);  
 return batchRemove(c, true);//相等的是否清除  
}

## Vector

### 类的实现：

public class Vector<E> extends AbstractList<E>  
 implements List<E>, RandomAccess, Cloneable, java.io.Serializable

### 基本结构：

protected Object[] elementData;//数据

protected int elementCount;//集合大小

protected int capacityIncrement;//集合增量大小

### 构造方法：

public Vector() {  
 this(10); //初始化大小  
}

public Vector(int initialCapacity) {  
 this(initialCapacity, 0);  
}

public Vector(int initialCapacity, int capacityIncrement) { //增量大小  
 super();  
 if (initialCapacity < 0)  
 throw new IllegalArgumentException("Illegal Capacity: "+  
 initialCapacity);  
 this.elementData = new Object[initialCapacity];  
 this.capacityIncrement = capacityIncrement;  
}

public Vector(Collection<? extends E> c) {  
 elementData = c.toArray();  
 elementCount = elementData.length;  
 // c.toArray might (incorrectly) not return Object[] (see 6260652)  
 if (elementData.getClass() != Object[].class)  
 elementData = Arrays.*copyOf*(elementData, elementCount, Object[].class);  
}

### 其他

public synchronized void trimToSize() {}//变成真实容量

public synchronized void ensureCapacity(int minCapacity) {}//扩充容量到

## LinkedList

### 类的实现：

public class LinkedList<E> extends AbstractSequentialList<E>  
 implements List<E>, Deque<E>, Cloneable, java.io.Serializable

### 基本结构：

transient int size = 0;

transient Node<E> first;

transient Node<E> last;

### 构造方法：

public LinkedList() {  
}

public LinkedList(Collection<? extends E> c) {  
 this();  
 addAll(c);  
}

### 其他：

public E peek() {//得到第一个元素  
 final Node<E> f = first;  
 return (f == null) ? null : f.item;  
}

public E poll() {//删除第一个元素  
 final Node<E> f = first;  
 return (f == null) ? null : unlinkFirst(f);  
}

public boolean offer(E e) {//添加元素  
 return add(e);  
}

public void push(E e) {//队列添加元素  
 addFirst(e);  
}

public E pop() {//队列移除元素  
 return removeFirst();  
}

# Set

## TreeSet

### 特点：

能够对元素按照某种规则进行排序。同时也保证元素的唯一。

两种排序方式：自然排序、比较器排序。使用元素的自然排序对元素进行排序，还是根据创建集合时提供的Comparator比较器进行排序，这取决于使用的构造方法。

### 类的实现：

public class TreeSet<E> extends AbstractSet<E>  
 implements NavigableSet<E>, Cloneable, java.io.Serializable

### 基本结构：

private transient NavigableMap<E,Object> m;

### 构造方法：

TreeSet(NavigableMap<E,Object> m) {  
 this.m = m;  
}

public TreeSet() {  
 this(new TreeMap<E,Object>());  
}

public TreeSet(Comparator<? super E> comparator) {  
 this(new TreeMap<>(comparator));  
}

public TreeSet(Collection<? extends E> c) {  
 this();  
 addAll(c);  
}

public TreeSet(SortedSet<E> s) {  
 this(s.comparator());  
 addAll(s);  
}

### 其他：

# Queue

# Map

### 类的实现：

### 基本结构：

### 构造方法：

### 其他：