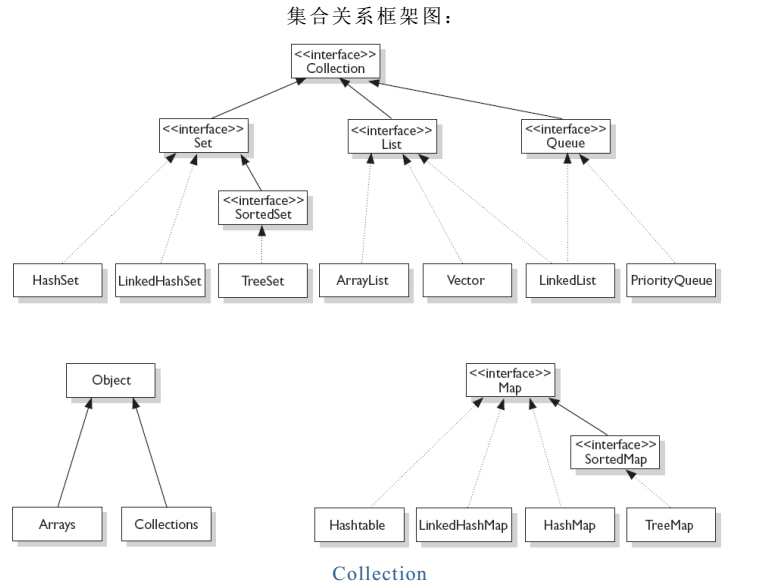
一、List接口实现

**1.Collection继承与接口操作架构**



可以看出，List接口继承Collection接口，Collection接口继承Iterable接口。实现List接口的子类必须实现Iterable和Collection接口中的方法。Iterable可以进行元素的迭代。

**2.Iterable接口、Collection接口及Iterator接口的定义**

实现这个接口允许对象成为 "foreach" 语句的目标。

Iterable接口定义的方法：

public interface Iterable<T> {

    Iterator<T> iterator();

}

Collection接口中定义的方法：

package java.util;

public interface Collection<E> extends Iterable<E> {

    int size();

    boolean isEmpty();

    boolean contains(Object o);

    Iterator<E> iterator();

    Object[] toArray();

    <T> T[] toArray(T[] a);

    boolean add(E e);

    boolean remove(Object o);

    boolean containsAll(Collection<?> c);

    boolean addAll(Collection<? extends E> c);

    boolean removeAll(Collection<?> c);

    boolean retainAll(Collection<?> c);

    void clear();

    boolean equals(Object o);

    int hashCode();

}

Iterator是一个迭代器接口，专门用来迭代各种Collection集合，包括Set集合和List集合。

Iterator接口定义的方法：

|  |  |
| --- | --- |
| boolean | **[hasNext](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Iterator.html" \l "hasNext())**()            如果仍有元素可以迭代，则返回 true。 |
| [E](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Iterator.html" \o "Iterator 中的类型参数) | **[next](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Iterator.html" \l "next())**()            返回迭代的下一个元素。 |
| void | **[remove](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Iterator.html" \l "remove())**()            从迭代器指向的 collection 中移除迭代器返回的最后一个元素（可选操作）。 |

**3.List特性**

可以存放同一种类型的元素；内部维护元素之间的顺序,是有序集合；元素是可以重复的。

在Java中List接口有3个常用的实现类，分别是ArrayList、LinkedList、Vector。

二、ArrayList

     ArrayList内部存储的数据结构是数组存储。数组的特点：元素可以快速访问。每个元素之间是紧邻的不能有间隔，缺点：数组空间不够元素存储需要扩容的时候会开辟一个新的数组把旧的数组元素拷贝过去，比较消性能。从ArrayList中间位置插入和删除元素，都需要循环移动元素的位置，因此数组特性决定了数组的特点：**适合随机查找和遍历，不适合经常需要插入和删除操作。**

## 源码分析：

ArrayList类：

 public boolean add(E e) {

//确保数组存储空间是否已经满了，满了扩容

        ensureCapacityInternal(size + 1);  // Increments modCount!!

        //计算存储的位置

        elementData[size++] = e;

        return true;

    }

 public void ensureCapacity(int minCapacity) {

    modCount++;

    int oldCapacity = elementData.length;

    if (minCapacity > oldCapacity) {

        Object oldData[] = elementData;

        int newCapacity = (oldCapacity \* 3)/2 + 1;

            if (newCapacity < minCapacity)

        newCapacity = minCapacity;

            // minCapacity is usually close to size, so this is a win:

            elementData = Arrays.copyOf(elementData, newCapacity);

    }

    }

## ArrayList中什么时候触发克隆数组：

1. 构造参数是Collection集合子类可能触发。
2. trimToSize的时候触发。
3. grow的时候触发。
4. clone的时候触发。
5. toArray的时候触发。

三、Vector

     Vector内部实现和ArrayList一样都是数组存储，最大的不同就是它支持线程的同步，所以访问比ArrayList慢，但是数据安全，所以对元素的操作没有并发操作的时候用ArrayList比较快。

构造方法：

|  |
| --- |
| **[Vector](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Vector.html" \l "Vector())**()            构造一个空向量，使其内部数据数组的大小为 10，其标准容量增量为零。 |
| **[Vector](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Vector.html" \l "Vector(java.util.Collection))**([Collection](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Collection.html" \o "java.util 中的接口)<? extends [E](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Vector.html" \o "Vector 中的类型参数)> c)            构造一个包含指定 collection 中的元素的向量，这些元素按其 collection 的迭代器返回元素的顺序排列。 |
| **[Vector](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Vector.html" \l "Vector(int))**(int initialCapacity)            使用指定的初始容量和等于零的容量增量构造一个空向量。 |
| **[Vector](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Vector.html" \l "Vector(int, int))**(int initialCapacity, int capacityIncrement)            使用指定的初始容量和容量增量构造一个空的向量。 |

源码分析：

private void ensureCapacityHelper(int minCapacity) {

     int oldCapacity = elementData.length;

     if (minCapacity > oldCapacity) {

         Object[] oldData = elementData;

         int newCapacity = (capacityIncrement > 0) ?

        (oldCapacity + capacityIncrement) : (oldCapacity \* 2);

         if (newCapacity < minCapacity) {

        newCapacity = minCapacity;

         }

          elementData = Arrays.copyOf(elementData, newCapacity);

     }

 }

四、ArrayList和Vector主要区别如下:

（1）同步性：

Vector是线程安全的，也就是说是它的方法之间是线程同步的，而ArrayList是线程序不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用ArrayList，因为它不考虑线程安全，效率会高些；如果有多个线程会访问到集合，那最好是使用Vector，因为不需要我们自己再去考虑和编写线程安全的代码。

备注：对于Vector&ArrayList、Hashtable&HashMap，要记住线程安全的问题，记住Vector与Hashtable是旧的，是java一诞生就提供了的，它们是线程安全的，ArrayList与HashMap是java2时才提供的，它们是线程不安全的。

（2）数据增长：

ArrayList与Vector都有一个初始的容量大小，当存储进它们里面的元素的个数超过了容量时，就需要增加ArrayList与Vector的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要取得一定的平衡。Vector默认增长为原来两倍，而ArrayList的增长策略在文档中没有明确规定（从源代码看到的是增长为原来的1.5倍）。ArrayList与Vector都可以设置初始的空间大小，Vector还可以设置增长的空间大小，而ArrayList没有提供设置增长空间的方法。

    总结：即Vector增长原来的一倍，ArrayList增加原来的0.5倍。

五、LinkedList

     LinkedList内部存储用的数据结构是链表。链表的特点：适合动态的插入和删除。访问遍历比较慢。另外不支持get，remove，insertList方法。可以当做堆栈、队列以及双向队列使用。LinkedList是线程不安全的。所以需要同步的时候需要自己手动同步，比较费事，可以使用提供的集合工具类实例化的时候同步：具体使用List<String> springokList=Collections.synchronizedCollection(new 需要同步的类)。

源码分析：

|  |
| --- |
| **[LinkedList](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/LinkedList.html" \l "LinkedList())**()            构造一个空列表。 |
| **[LinkedList](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/LinkedList.html" \l "LinkedList(java.util.Collection))**([Collection](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/Collection.html" \o "java.util 中的接口)<? extends [E](mk:@MSITStore:C:\\Users\\张志鹏\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/mk:java/util/LinkedList.html" \o "LinkedList 中的类型参数)> c)            构造一个包含指定 collection 中的元素的列表，这些元素按其 collection 的迭代器返回的顺序排列。 |

 删除操作：

public boolean remove(Object o) {

        if (o==null) {

            for (Entry<E> e = header.next; e != header; e = e.next) {

                if (e.element==null) {

                    remove(e);

                    return true;

                }

            }

        } else {

            for (Entry<E> e = header.next; e != header; e = e.next) {

                if (o.equals(e.element)) {

                    remove(e);

                    return true;

                }

            }

        }

        return false;

    }

   private E remove(Entry<E> e) {

    if (e == header) throw new NoSuchElementException();

        E result = e.element;

    e.previous.next = e.next;

    e.next.previous = e.previous;

        e.next = e.previous = null;

        e.element = null;

    size--;

    modCount++;

        return result;

    }

六、ArrayList，LinkedList，Vector的区别

1.内部存储结构区别：

 ArrayList、Vector是数组存储。LinkedList是链表存储。

2.线程安全区别：

ArrayList、LinkedList线程不安全。Vector线程安全。

3.使用场景区别：

     使用线程同步的时候Vector类首选或者使用Collections工具类初始化时候同步。

需要经常删除、增加使用LinkedList(链表结构)、经常需要查询迭代使用ArrayList(数组结构)