一、List接口框架

|----Collection接口：单列集合，用来存储一个一个的对象

|----List接口：存储有序的、可重复的数据。 -->“动态”数组,替换原有的数组

|----ArrayList：作为List接口的主要实现类；线程不安全的，效率高；底层使用Object[] elementData存储

|----LinkedList：对于频繁的插入、删除操作，使用此类效率比ArrayList高；底层使用双向链表存储

|----Vector：作为List接口的古老实现类；线程安全的，效率低；底层使用Object[] elementData存储

二、ArrayList的源码分析

1、jdk 7的情况下

```
ArrayList list = new ArrayList();//底层创建了长度是10的Object[]数组elementData
list.add(123);//elementData[0] = new Integer(123);
...
list.add(11);//如果此次的添加导致底层elementData数组容量不够，则扩容。
```

默认情况下，扩容为原来的容量的1.5倍，同时需要将原有数组中的数据复制到新的数组中。

结论：建议开发中使用带参的构造器：ArrayList list = new ArrayList(int capacity)

2、jdk 8中ArrayList的变化：

ArrayList list = new ArrayList(); //底层Object[] elementData初始化为{}.并没有创建长度为10的数组

list.add(123);//第一次调用add()时，底层才创建了长度10的数组，并将数据123添加到elementData[0]

...

后续的添加和扩容操作与jdk 7 无异

3、小结

jdk7中的ArrayList的对象的创建类似于单例的饿汉式，而jdk8中的ArrayList的对象

的创建类似于单例的懒汉式，延迟了数组的创建，节省内存。

三、LinkedList的源码分析

```
1  LinkedList list = new LinkedList(); 内部声明了Node类型的first和la
st属性，默认值为null
2  list.add(123);//将123封装到Node中，创建了Node对象。
3
4  其中，Node定义为：体现了LinkedList的双向链表的说法
5  private static class Node<E> {
6  E item;
7  Node<E> next;
8  Node<E> prev;
9
10  Node(Node<E> prev, E element, Node<E> next) {
11  this.item = element;
12  this.next = next;
13  this.prev = prev;
14  }
15  }
```

四、Vector的源码分析

1、jdk7和jdk8中通过Vector()构造器创建对象时，底层都创建了长度为10的数组。

2、在扩容方面，默认扩容为原来的数组长度的2倍。

五、List接口中的常用方法

```
1  void add(int index, Object ele):在index位置插入ele元素
2  boolean addAll(int index, Collection eles):从index位置开始将eles
中的所有元素添加进来
3  Object get(int index):获取指定index位置的元素
4  int indexOf(Object obj):返回obj在集合中首次出现的位置
5  int lastIndexOf(Object obj):返回obj在当前集合中末次出现的位置
6  Object remove(int index):移除指定index位置的元素，并返回此元素
7  Object set(int index, Object ele):设置指定index位置的元素为ele
8  List subList(int fromIndex, int toIndex):返回从fromIndex到toInde
x位置的子集合
9
10  总结：常用方法
11  增：add(Object obj)
12  删：remove(int index) / remove(Object obj)
13  改：set(int index, Object ele)
14  查：get(int index)
15  插：add(int index, Object ele)
16  长度：size()
17  遍历：① Iterator迭代器方式
18  ② 增强for循环
19  ③ 普通的循环
```

六、代码举例

```
1  @Test
2  public void test1(){
3  ArrayList list = new ArrayList();
4  list.add(123);
5  list.add(456);
6  list.add("AA");
7  list.add(new Person("Tom",12));
8  list.add(456);
9
```

```
10  System.out.println(list);
11
12  //void add(int index, Object ele):在index位置插入ele元素
13  list.add(1,"BB");
14  System.out.println(list);
15
16  //boolean addAll(int index, Collection eles):从index位置开始将e
es中的所有元素添加进来
17  List list1 = Arrays.asList(1, 2, 3);
18  list.addAll(list1);
19  // list.add(list1);
20  System.out.println(list.size());//9
21
22  //Object get(int index):获取指定index位置的元素
23  System.out.println(list.get(0));
24
25  }
```

```
1  @Test
2  public void test2(){
3  ArrayList list = new ArrayList();
4  list.add(123);
5  list.add(456);
6  list.add("AA");
7  list.add(new Person("Tom",12));
8  list.add(456);
9  //int indexOf(Object obj):返回obj在集合中首次出现的位置。如果不存
在，返回-1.
10  int index = list.indexOf(4567);
11  System.out.println(index);
12
13  //int lastIndexOf(Object obj):返回obj在当前集合中末次出现的位置。
如果不存在，返回-1.
14  System.out.println(list.lastIndexOf(456));
15
16  //Object remove(int index):移除指定index位置的元素，并返回此元素
17  Object obj = list.remove(0);
```

```
18  System.out.println(obj);
19  System.out.println(list);
20
21  //Object set(int index, Object ele):设置指定index位置的元素为ele
22  list.set(1,"CC");
23  System.out.println(list);
24
25  //List subList(int fromIndex, int toIndex):返回从fromIndex到toI
 ndex位置的左闭右开区间的子集合
26  List subList = list.subList(2, 4);
27  System.out.println(subList);
28  System.out.println(list);
29
30
31  }
```

```
1  @Test
2  public void test3(){
3  ArrayList list = new ArrayList();
4  list.add(123);
5  list.add(456);
6  list.add("AA");
7
8  //方式一：Iterator迭代器方式
9  Iterator iterator = list.iterator();
10  while(iterator.hasNext()){
11  System.out.println(iterator.next());
12  }
13
14  System.out.println("***************");
15
16  //方式二：增强for循环
17  for(Object obj : list){
18  System.out.println(obj);
19  }
20
```

```
21  System.out.println("***************");
22
23  //方式三：普通for循环
24  for(int i = 0;i < list.size();i++){
25  System.out.println(list.get(i));
26  }
27
28
29
30  }
```

七、面试题

1、ArrayList、LinkedList、Vector三者的异同？

同：三个类都是实现了List接口，存储数据的特点相同：存储有序的、可重复的数据

1、ArrayList和LinkedList二者都线程不安全，相对线程安全的Vector，执行效率高。

2、Vector和ArrayList几乎是完全相同的

不同：

1、ArrayList是实现了基于动态数组的数据结构，LinkedList基于链表的数据结构。对于随机访问get和set，ArrayList觉得优于LinkedList，因为LinkedList要移动指针。对于新增和删除操作add(特指插入)和remove，LinkedList比较占优势，因为ArrayList要移动数据。

2、Vector和ArrayList几乎是完全相同的,唯一的区别在于Vector是同步类(synchronized)，属于强同步类。因此开销就比ArrayList要大，访问要慢。正常情况下,大多数的Java程序员使用 ArrayList而不是Vector,因为同步完全可以由程序员自己来控制。Vector每次扩容请求其大小的2倍空间，而ArrayList是1.5倍。Vector还有一个子类Stack。