ArrayList（动态数组）

一、基本介绍

1、特点

底层基于数组实现容量大小动态变化。

允许 null 的存在。

支持快速访问、复制、序列化的。

2、构造方法

- new ArrayList();
- new ArrayList(int 大小);//制定底层数组的大小
- new ArrayList(另外一个集合)//使用另外一个集合创建一个新集合，新集合中包含了参数的所有元素

3、常用方法

contains：判断集合是否包含制定元素

set：修改指定位置的元素

size:获取链表大小

二、ArrayList线程不安全问题

举个例子：

一个 ArrayList ，在添加一个元素的时候，它可能会有两步来完成：

1、在 Items[Size] 的位置存放此元素；

2、增大 Size 的值。 增大 Size 的值。

ArrayList.add的内部原理：

```
public class ArrayList<E>
{
   private Object[] elementData;      // 存储元素的数组。其分配的空间长度是capacity。
   private int size;                  // elementData存储了多少个元素。
   public ArrayList(){this(10);};     // 默认capacity是10
   boolean add(E e)
```

```
    {
        ensureCapacityInternal(size + 1);  // capacity至少为 size+1
        elementsData[size++]=e;            // size++
        return true;
    }
    void ensureCapacityInternal(int minCapacity){
        if(minCapacity > elementData.length)     // 扩容
            grow(minCapacity);
    }
    void grow(int minCapacity){
        int oldCapacity = elementData.length;
        int newCapacity = oldCapacity + (oldCapacity >> 1);     // 约是原先的1.5倍。
        elementData = Arrays.copyOf(elementData,newCapacity );
    }
}
```

在单线程运行的情况下，如果 Size = 0，添加一个元素后，此元素在位置 0，而且 Size=1； 而如果是在多线程情况下，比如有两个线程，线程 A 先将元素存放在位置 0。但是此时 CPU 调度线程A暂停，线程 B 得到运行的机会。线程B也向此 ArrayList 添加元素，因为此时 Size 仍然等于 0 （注意哦，我们假设的是添加一个元素是要两个步骤哦，而线程A仅仅完成了步骤1），所以线程B也将元素存放在位置0。然后线程A和线程B都继续运行，都增加 Size 的值。现在看看 ArrayList 的情况，元素实际上只有一个，存放在位置 0，而 Size 却等于 2。这就是“线程不安全”了。

虽然ArrayList是非线程安全的，要想实现线程安全的ArrayList，可在ArrayList的基础上通过同步块来实现，或者使用同步包装器（Collections.synchronizedList），还可以使用J.U.C中的CopyOnWriteArrayList。

```
    final ArrayList<String> list = new ArrayList<String>();
#1:自己手动同步
    public static List<E> list = ... ;
    lock.lock();
    list.add();
    lock.unlock();
    #2:使用同步包装器
    List<E> syncList = Collections.synchronizedList(new ArrayList<E>());
    迭代时，需要包含在同步块当中
    synchronized(syncList){
       while(Iterator<E> iter = syncList.iterator();iter.hasNext();){}
    }
    #3:使用J.U.C中的CopyOnWriteArrayList。
```

三、知识点

1、arraylist的扩容增长率为目前数组长度的1.5倍，当数组大小不够且将要添加数据时扩容