## 使用数组的局限性

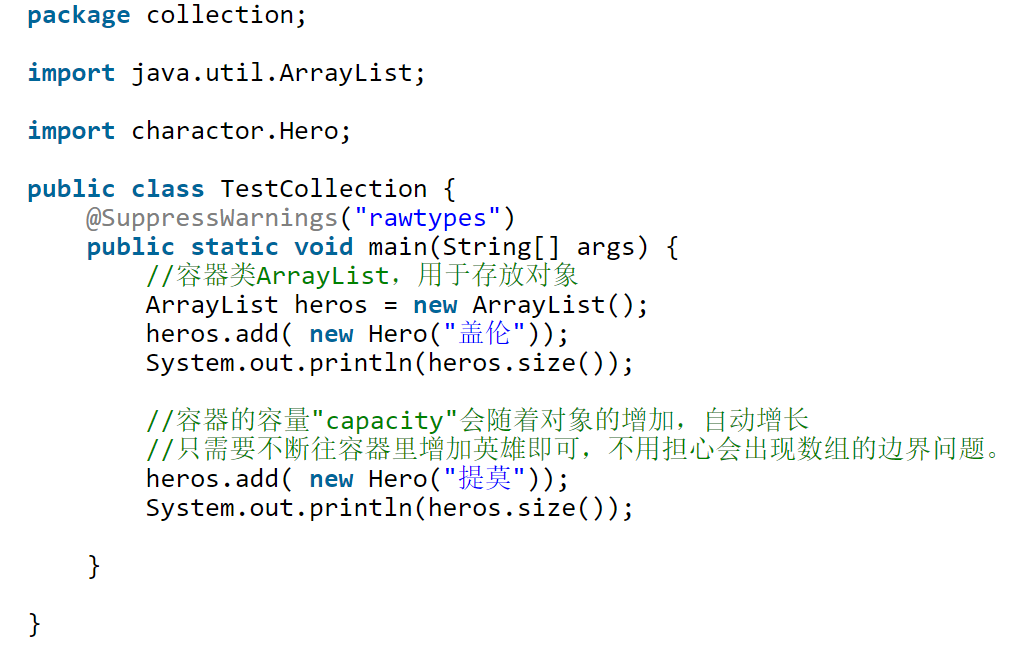
**[顶](https://how2j.cn/k/collection/collection-arraylist/363.html)[折](https://how2j.cn/k/collection/collection-arraylist/363.html" \l "nowhere" \o "折叠)**

如果要存放多个对象，可以使用数组，但是数组有局限性  
比如 声明长度是10的数组  
不用的数组就浪费了  
超过10的个数，又放不下

## ArrayList存放对象

**[顶](https://how2j.cn/k/collection/collection-arraylist/363.html)[折](https://how2j.cn/k/collection/collection-arraylist/363.html" \l "nowhere" \o "折叠)**

为了解决数组的局限性，引入容器类的概念。 最常见的容器类就是  
ArrayList  
[容器的容量](https://how2j.cn/k/number-string/number-string-stringbuilder/328.html" \l "step724)"capacity"会随着对象的增加，自动增长  
只需要不断往容器里增加英雄即可，不用担心会出现数组的边界问题。



常用方法

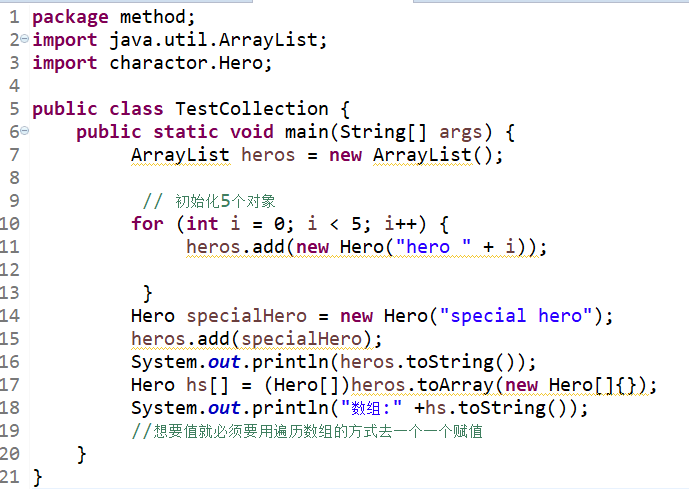
增加

****add****有两种用法  
第一种是直接add对象，把对象加在最后面

heros.add(new Hero("hero " + i));

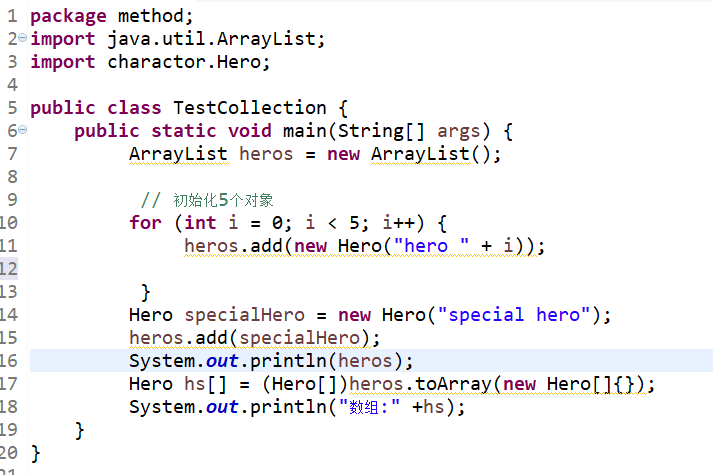
第二种是在指定位置加对象

heros.add(3, specialHero);



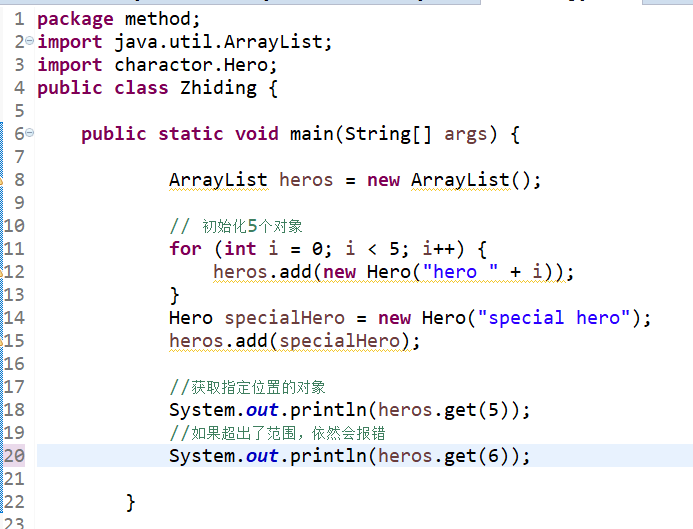
## 判断是否存在

通过方法****contains****判断一个对象是否在容器中  
判断标准： 是否是同一个对象，而不是name是否相同



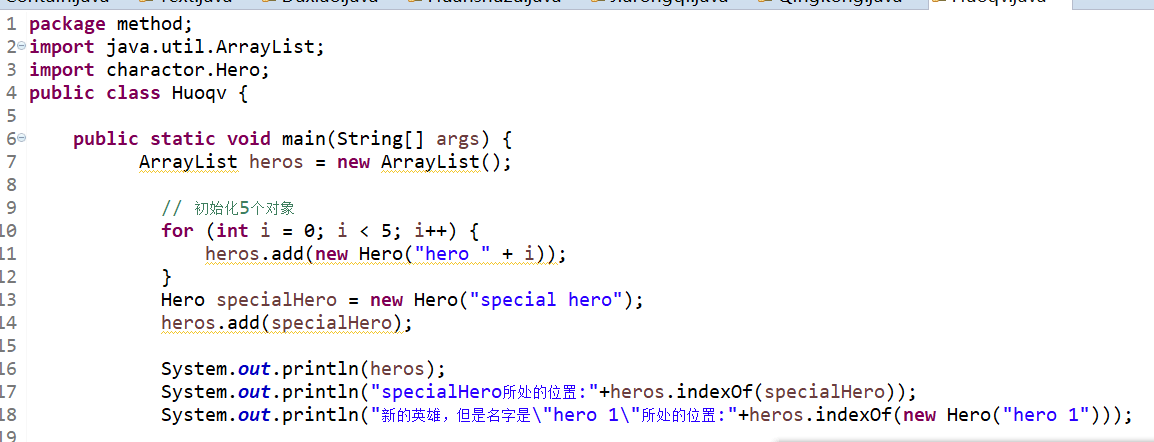
获取指定位置的对象

通过get获取指定位置的对象，如果输入的下标越界，一样会报错



取对象所处的位置

****indexOf****用于判断一个对象在ArrayList中所处的位置  
与[contains](https://how2j.cn/k/collection/collection-arraylist-method/685.html" \l "step2454)一样，判断标准是对象是否相同，而非对象的name值是否相等



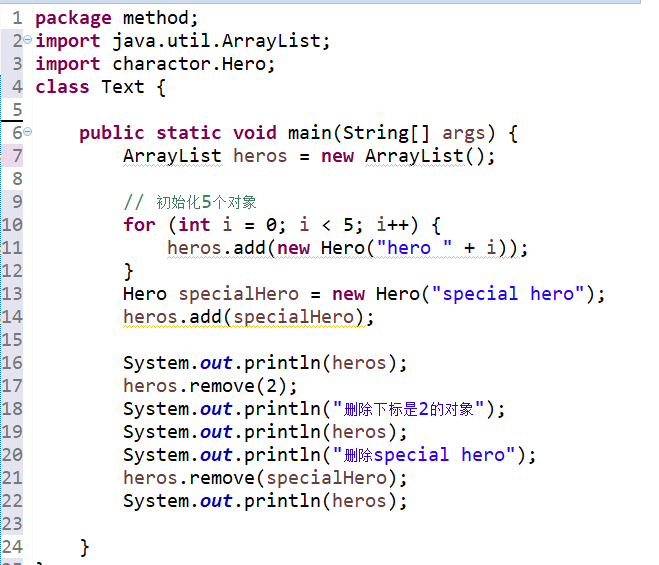
## 删除

****remove****用于把对象从ArrayList中删除  
remove可以根据下标删除ArrayList的元素

heros.remove(2);

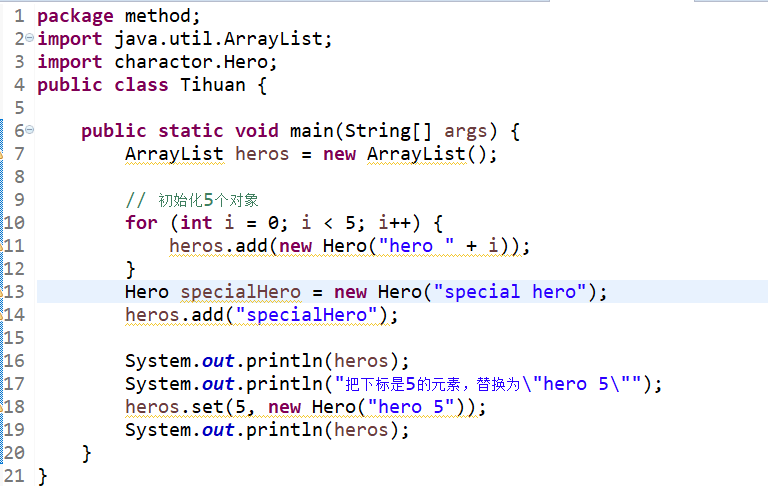
也可以根据对象删除

heros.remove(specialHero);



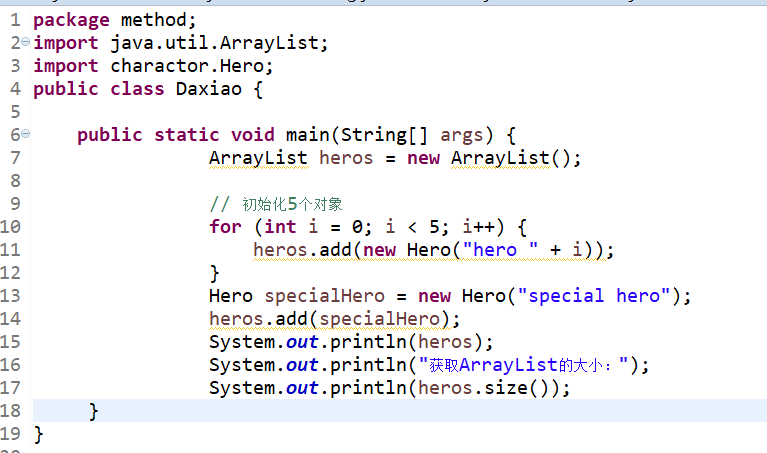
## 替换

****set****用于替换指定位置的元素



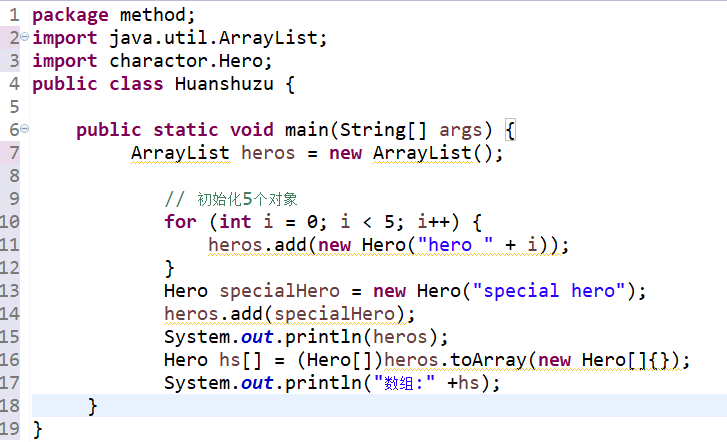
获取大小

****size**** 用于获取ArrayList的大小



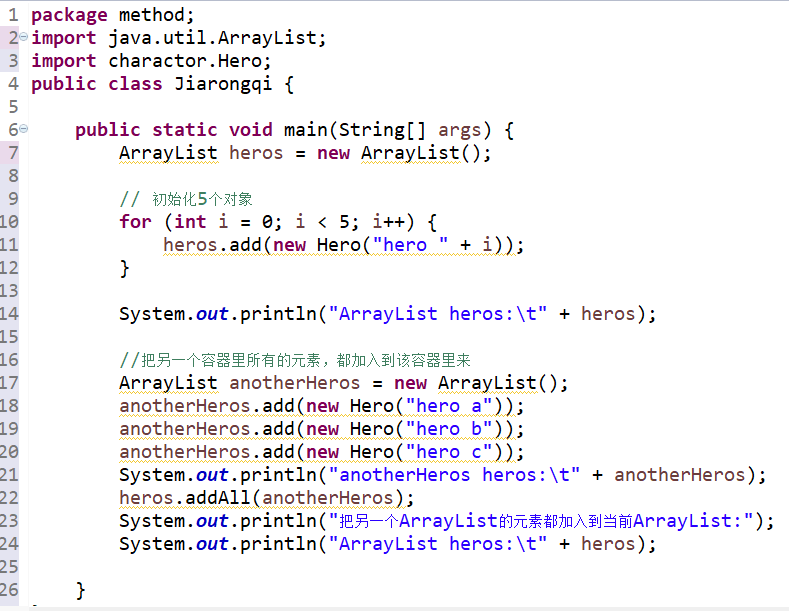
## 转换为数组

****toArray****可以把一个ArrayList对象转换为数组。  
需要注意的是，如果要转换为一个Hero数组，那么需要传递一个Hero数组类型的对象给toArray()，这样toArray方法才知道，你希望转换为哪种类型的数组，否则只能转换为Object数组



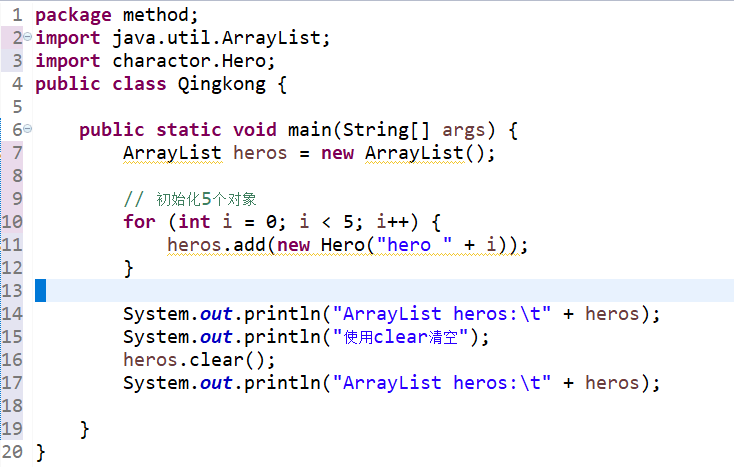
## 把另一个容器所有对象都加进当前容器

****addAll**** 把另一个容器所有对象都加进来



清空

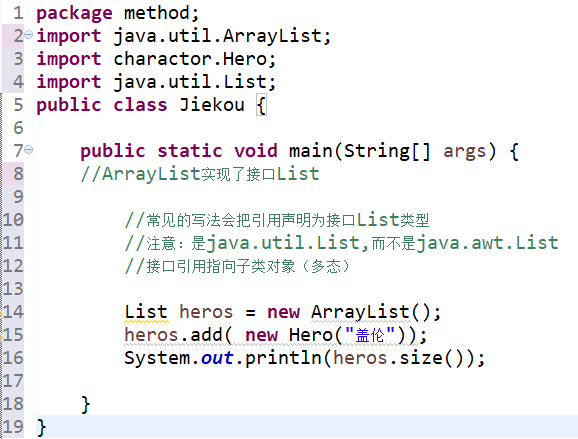
****clear**** 清空一个ArrayList



## List框架

## ArrayList和List（List为接口）

## ArrayList实现了接口List 常见的写法会把引用声明为接口List类型 注意：是****java.util.List****,而****不是****java.awt.List



## List接口的方法

因为ArrayList实现了List接口，所以List接口的方法ArrayList都实现了。import java.util.List

## 泛型 Generic

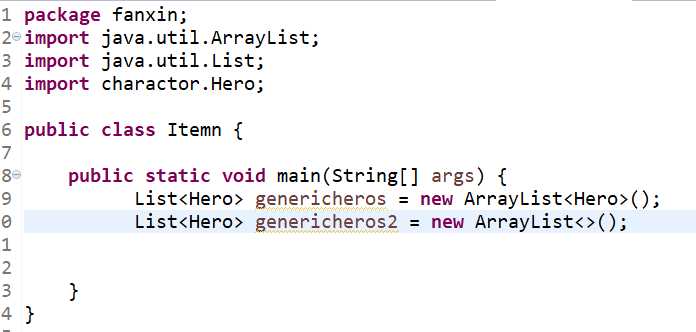
不指定泛型的容器，可以存放任何类型的元素  
指定了泛型的容器，只能存放指定类型的元素以及其子类

## 泛型的简写

为了不使编译器出现警告，需要前后都使用泛型，像这样：

List<Hero> genericheros = new ArrayList<Hero>();  
不过JDK7提供了一个可以略微减少代码量的泛型简写方式

List<Hero> genericheros2 = new ArrayList<>();  
后面的泛型可以用<>来代替

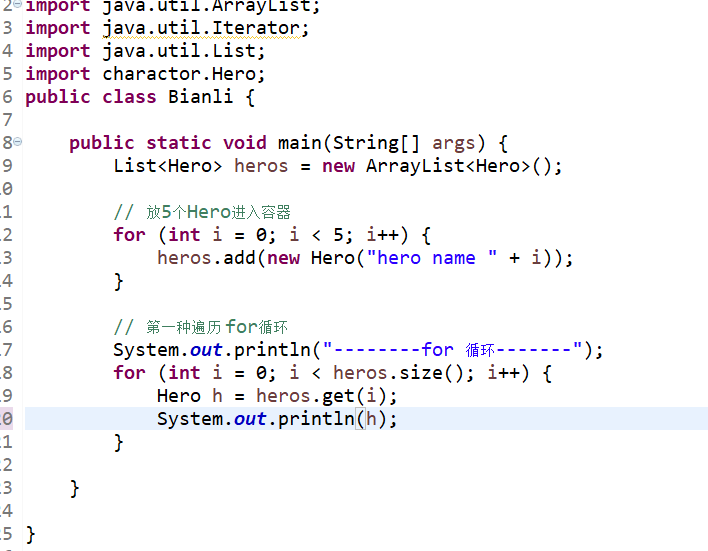


<Hero>指的是类类型呢

遍历

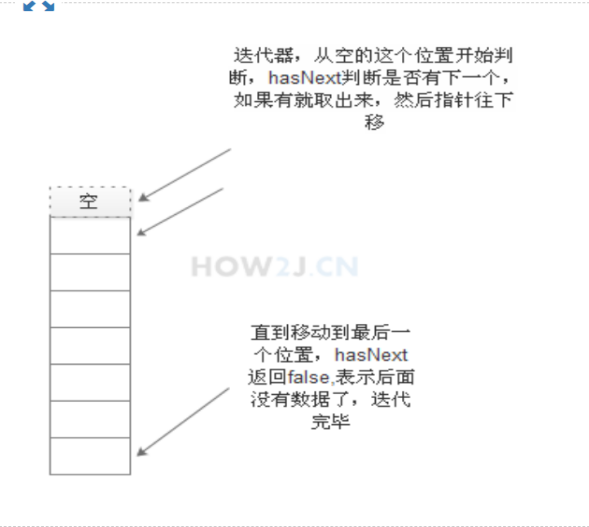
用for循环遍历

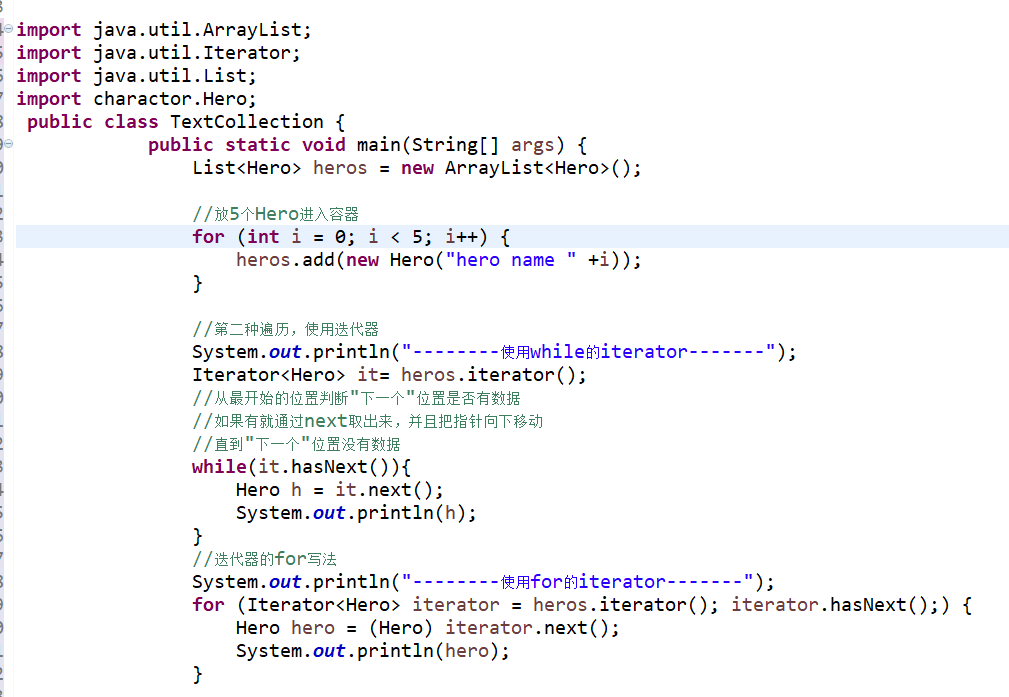
通过前面的学习，知道了可以用size()和get()分别得到大小，和获取指定位置的元素，结合for循环就可以遍历出ArrayList的内容



迭代器遍历

使用迭代器Iterator遍历集合中的元素





增强型for循环

使用增强型for循环可以非常方便的遍历ArrayList中的元素，这是很多开发人员的首选。  
不过增强型for循环也有不足：  
无法用来进行ArrayList的初始化

无法得知当前是第几个元素了，当需要只打印单数元素的时候，就做不到了。 必须再自定下标变量。

