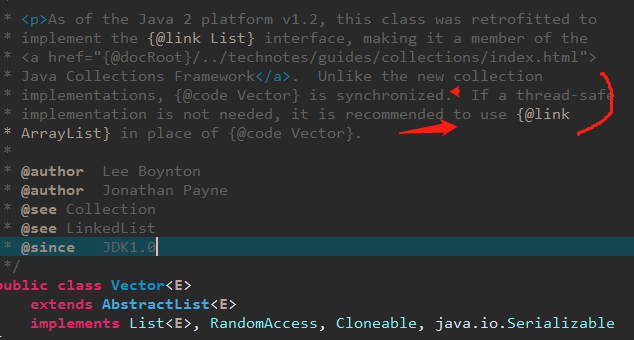
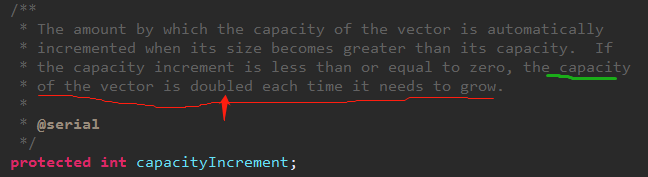
第1部分 Vector介绍

Vector简介



Vector 是**矢量队列**，它是JDK1.0版本添加的类。继承于AbstractList，实现了List, RandomAccess, Cloneable, serializable这些接口。  
Vector 继承了AbstractList，实现了List；所以，**它是一个队列，支持相关的添加、删除、修改、遍历等功能**。  
Vector 实现了RandmoAccess接口，即**提供了随机访问功能**。RandmoAccess是java中用来被List实现，为List提供快速访问功能的。在Vector中，我们即可以通过元素的序号快速获取元素对象；这就是快速随机访问。  
Vector 实现了Cloneable接口，即实现clone()函数。它能被克隆。  
  
和ArrayList不同，**Vector中的操作是线程安全的**。

Vector的构造函数：



[复制代码](javascript:void(0);)

Vector共有4个构造函数

Vector()

// 默认构造函数，capacity为10；

Vector(int capacity)

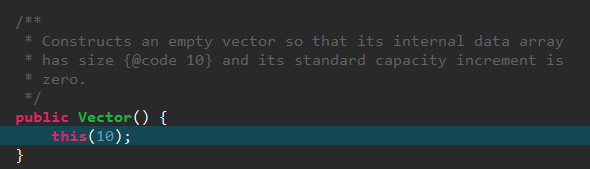
// capacity是Vector的默认容量大小。当由于增加数据导致容量增加时，每次容量会增加一倍。

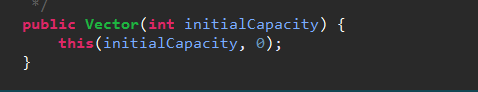
Vector(int capacity, int capacityIncrement)

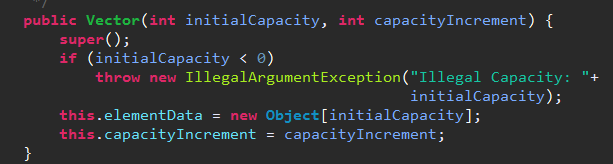
// capacity是Vector的默认容量大小，capacityIncrement是每次Vector容量增加时的增量值。

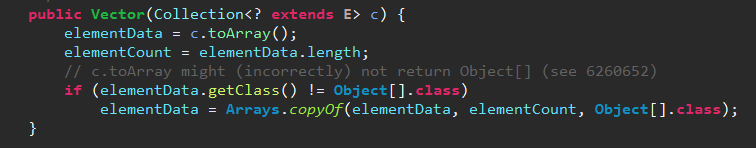
Vector(Collection<? extends E> collection)

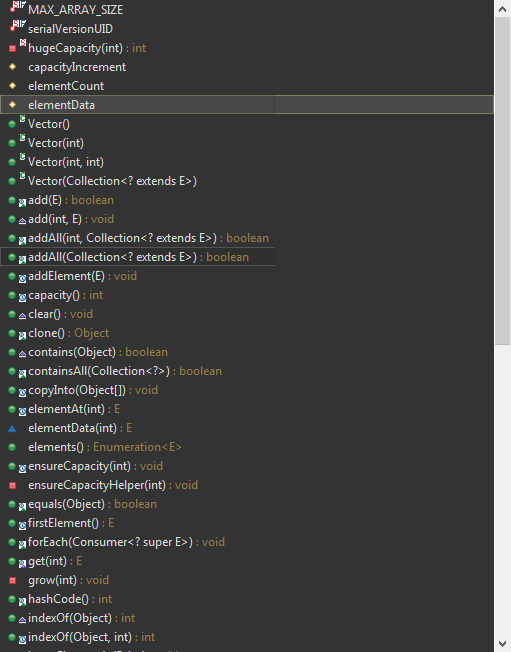
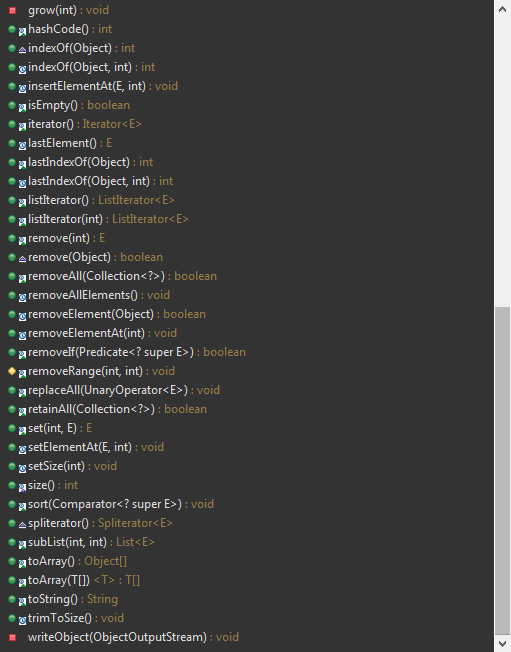
// 创建一个包含collection的Vector





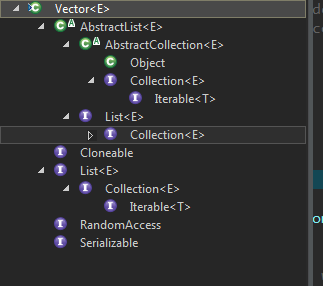




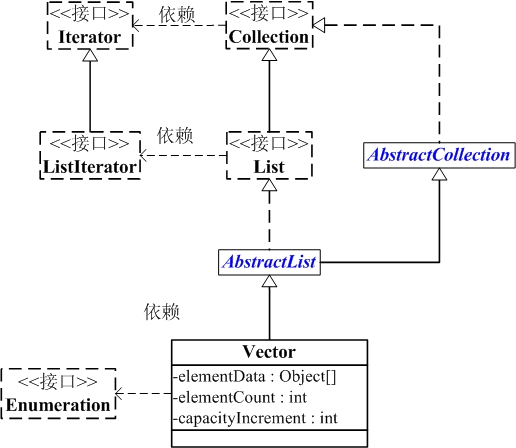
Vector的API

第2部分 Vector数据结构

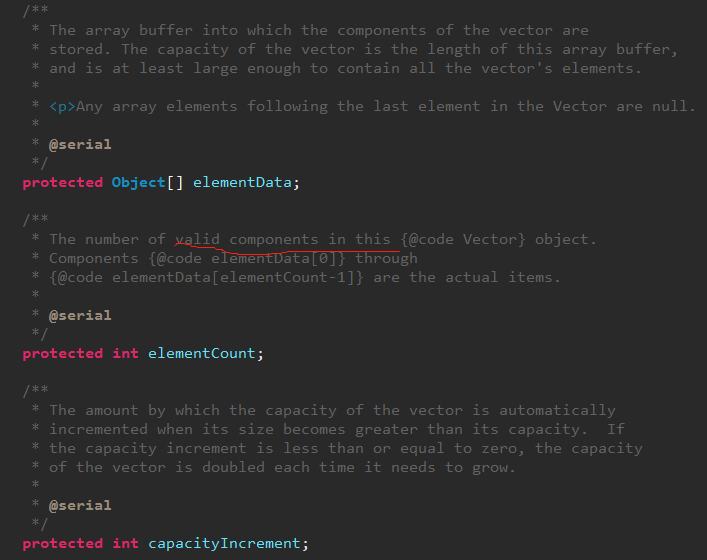
Vector的继承关系



**Vector与Collection关系如下图**：

[](https://images0.cnblogs.com/blog/497634/201401/272347229531613.jpg)

Vector的数据结构和[ArrayList](http://www.cnblogs.com/skywang12345/p/3308556.html)差不多，它包含了3个成员变量：elementData , elementCount， capacityIncrement。



(01) elementData 是"Object[]类型的数组"，它保存了添加到Vector中的元素。elementData是个动态数组，如果初始化Vector时，没指定动态数组的大小，则使用默认大小10。随着Vector中元素的增加，Vector的容量也会动态增长，capacityIncrement是与容量增长相关的增长系数，具体的增长方式，请参考源码分析中的ensureCapacity()函数。

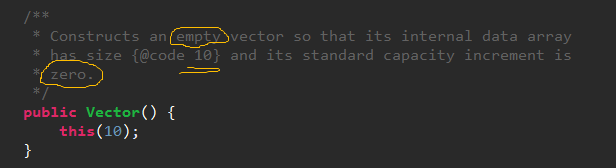
(02) elementCount 是动态数组的实际大小。

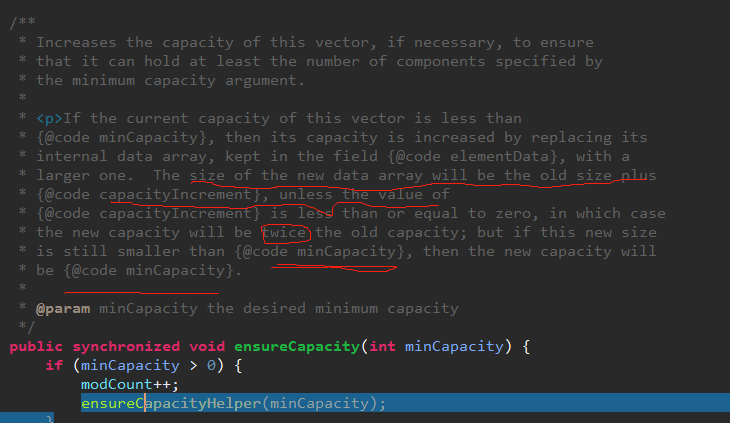
(03) capacityIncrement 是动态数组的增长系数。如果在创建Vector时，指定了capacityIncrement的大小；则，每次当Vector中动态数组容量增加，增加的大小都是capacityIncrement。

第3部分 Vector源码解析(基于JDK1.6.0\_45)

为了更了解Vector的原理，下面**对Vector源码代码作出分析**。

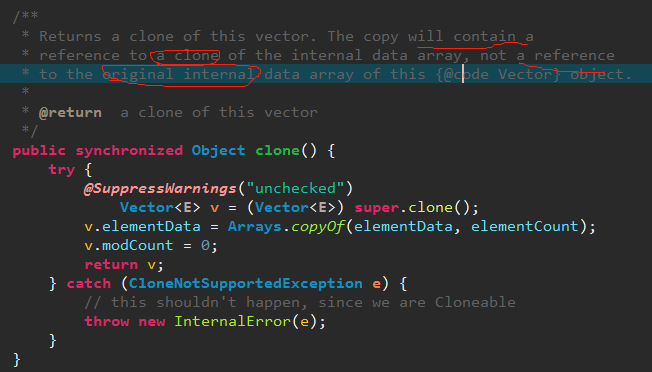
**总结**：  
(01) Vector实际上是通过一个**数组**去保存数据的。当我们构造Vecotr时；若使用默认构造函数，则Vector的**默认容量大小是10**。

  
(02) 当Vector容量不足以容纳全部元素时，Vector的容量会增加。**若容量增加系数 >0，则将容量的值增加“容量增加系数”；否则，将容量大小增加一倍。**



  
(03) Vector的克隆函数，即是将全部元素克隆到一个数组中。

 By convention, the object returned by this method should be independent of this object (which is being cloned).



第4部分 Vector遍历方式

Vector支持**4种遍历方式**。建议使用下面的第二种去遍历Vector，因为效率问题。

(01) 第一种，通过**迭代器**遍历。即通过Iterator去遍历。

Integer value = null;

int size = vec.size();

for (int i=0; i<size; i++) {

value = (Integer)vec.get(i);

}

(02) 第二种，**随机访问**，通过索引值去遍历。  
由于Vector实现了RandomAccess接口，它支持通过索引值去随机访问元素。

Integer value = null;

int size = vec.size();

for (int i=0; i<size; i++) {

value = (Integer)vec.get(i);

}

(03) 第三种，**另一种for循环**。如下：

Integer value = null;

for (Integer integ:vec) {

value = integ;

}

(04) 第四种，**Enumeration遍历**。如下：

Integer value = null;

Enumeration enu = vec.elements();

while (enu.hasMoreElements()) {

value = (Integer)enu.nextElement();

}

**总结**：遍历Vector，使用**索引的随机访问方**式最快，使用迭代器最慢。

总结

Vector和ArrayList的实现方式可以看出非常的类似，既然**Vector类建议尽量少的使用，还是最好不要用了，**通过上面的源码发现，每个方法中都添加了synchronized的关键字来保证同步，所以它是线程安全的，但正是这些方法的同步，让它的效率大大的降低了，比ArrayList的效率要慢。

给出如下几点总结：

　　1.Vector有四个不同的构造方法。无参构造方法的容量为默认值10，仅包含容量的构造方法则将容量增长量（从源码中可以看出容量增长量的作用，第二点也会对容量增长量详细说）明置为0。

　　2.注意扩充容量的方法ensureCapacityHelper。

与ArrayList相同，Vector在每次增加元素（可能是1个，也可能是一组）时，都要调用该方法来确保足够的容量。当容量不足以容纳当前的元素个数时，就先看构造方法中传入的容量增长量参数CapacityIncrement是否为0，如果不为0，就设置新的容量为就容量加上容量增长量，如果为0，就设置新的容量为旧的容量的2倍，如果设置后的新容量还不够，则直接新容量设置为传入的参数（也就是所需的容量），而后同样用Arrays.copyof()方法将元素拷贝到新的数组。

　　3.很多方法都加入了synchronized同步语句，来保证线程安全。

　　4.同样在查找给定元素索引值等的方法中，源码都将该元素的值分为null和不为null两种情况处理，Vector中也允许元素为null。

　　5.其他很多地方都与ArrayList实现大同小异，Vector现在已经基本不再使用。