Stack--栈

Stack是栈，它的特性是：先进后出。

Stack继承于Vector（矢量队列）的，因为Vector是通过数组实现的，所以stack也是通过数组来实现的。但是现在基本上已经遗弃了Vector和stack，所以少用。LinkedList可以当作栈来用，内部是通过双向链表来实现的。

构造函数：只有一个构造函数，默认构造函数。

Stack stack = new Stack();

方法：

1.push 入栈

2.peek 取栈顶元素，返回当前栈顶的元素

3.pop 取出栈顶元素，删除当前栈顶元素并返回该元素

Vector—矢量队列

Vector是矢量队列。继承于AbstractList，实现了List, RandomAccess, Cloneable这些接口。

继承了AbstractList，实现了List，所以它是个队列，支持相关的添加、删除、修改、遍历的操作。

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。

Vector不支持序列化，即没有实现java.io.Serializable接口

和ArrayList不同，Vector中的操作是线程安全的。

Vector内部的数据结构很ArrayList类似包含了一个Object类型的数组和当前数据的长度，多的是还有一个增长系数，这个增长系数可以通过构造函数自己指定，默认的时候是将增长1倍。

Vector支持四种遍历

（1）Iterator

Iterator<String> it = vector.iterator();

while(it.hasNext()){

String s = it.next();

System.out.println(s);

}

（2）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。例如想访问第二个元素，就直接Vector[2]就可以

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

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

}

（3）for循环

for(String s : vector)

（4）Enumeration遍历

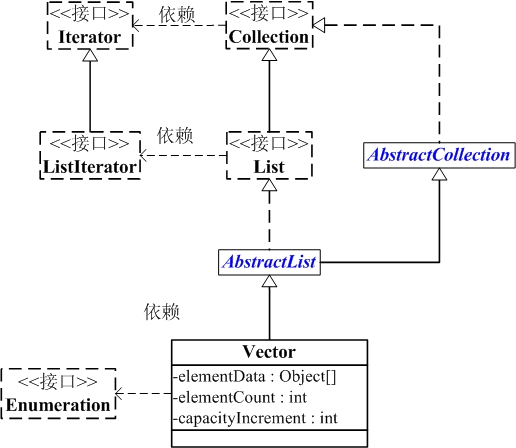
Enumeration enu = vec.elements(); //类似Iterator

while (enu.hasMoreElements()) {

Integer value = (Integer)enu.nextElement();}

建议使用第二种方法，随机访问方法访问数据元素。

继承的关系图：



为什么不推荐使用vector：

①Vector所有方法都是同步，有性能损失。  
②Vector早期版本出现的。  
③Vector初始length是10 超过length时 以100%比率增长，相比于ArrayList更多消耗内存。

现在一般都用ArrayList代替了vector。