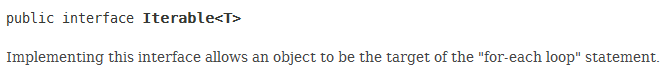
JDK中的枚举及迭代器&for-each语句

# Java中的集合框架都存在于java.util包中，所有的迭代器也存在于java.util包中。

# Iterable<T>接口：java.lang.Iterable<T>



实现了**Iterable接口的对象**可以利用**for-each循环**。

## 简单介绍

**存在于java.lang包中**。**各个集合的iterator方法**是通过实现Iterable接口中的iterator方法来具体实现的。

**java.util.Collection接口继承了java.lang.Iterable接口**。

## 方法介绍

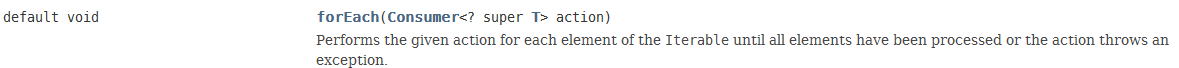
### iterator方法：获取该对象的迭代器。（最重要）



### spliterator方法



### forEach方法



## Iterable接口与Iterator接口的区别

**Iterable是java.lang包中的，而Iterator是java.util包中的。**

**实现Iterable接口**说明该类是可以迭代的，必须实现接口的iterator方法实现该类的**迭代器对象**。而Iterator只是该类对象迭代器对象的一个接口或超接口。

# JDK中提供了Iterator接口：接口介绍

## 简单介绍

存在于java.util包中，是**Java集合**框架中的一个接口，一般的集合都会实现Iterable接口，该集合是可以迭代的。

## 子接口：ListIterator<E>等；见后面介绍。

## 直接实现类：java.util.Scanner。

**Scanner** 类中实现了Iterator中定义的hasNext、next、remove方法，当然还有很多拓展的方法。

注意：Scanner由于不支持remove的真实的删除操作，remove是利用抛出异常来实现的，如果调用了该方法，则就会抛出UnsupportedOperationException。

这里体现了一种思想，如果一个类需要实现一个接口的大部分方法，但是少部分方法并不支持操作，可以通过抛出**运行时异常**如UnsupportedOperationException、或空操作或返回NULL（需要返回值）等假实现。所以再调用这些方法时，优秀的程序员需要考虑到这些情况。

## Iterator接口的方法：

### hasNext方法



### next方法



### remove方法：

如果具体的迭代器不具备删除删除功能，可以直接抛出运行时异常，如**UnsupportedOperationException**。程序员调用该方法时，一定要检查是否会发生这个异常。



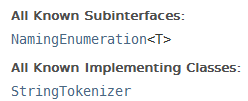
### forEachRemaining(Consumer<? super E> action)：default方法



# for-each语句

迭代器的主要作用是**遍历集合**，现在**for-each**提供了更为**简便的操作**。

# JDK中的Enumeration接口：接口介绍

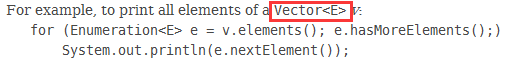


## 简单介绍

存在于**java.util**包中，是Java集合框架的一个接口。

## 功能：

**An object that implements the Enumeration interface** generates a series of elements, one at a time. Successive calls to the **nextElement** method return successive elements of the series.



Methods are provided to enumerate through the elements of a vector, the keys of a hashtable, and the values in a hashtable. **Enumerations are also used to specify the input streams to a SequenceInputStream.**

**Enumeration不仅用于遍历Vector、HashTable的keys和valuse，也可以用于指定一个序列输入流。**

## Enumeration接口的方法：只有两个

### hasMoreElements()：判断是否结束。与Iterator的hasNext()功能一样。



### nextElement():获取下一个元素。与Iterator的next()功能一样。



## Enumeration被Iterator取代

现在逐渐被**Iterator接口**取代，Iterator除了具有Enumeration接口的功能外，还具有remove方法。

Enumeration：The functionality of this interface is duplicated by the Iterator interface. In addition, Iterator adds **an optional remove operation**, and has shorter method names. New implementations should consider using Iterator in preference to Enumeration.

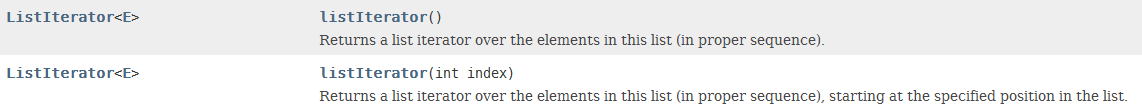
**Enumeration中的名字较长，Iterator名字较短，nextElement变为next，hasMoreElements变为hasNext**。Iterator中还定义了一个remove方法。

## 示例

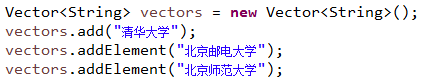
**Vector**可以通过elements方法返回一个Enumeration对象，当然也可以通过iterator方法获取一个Iterator迭代器，通过listIterator方法获取一个ListIterator对象。



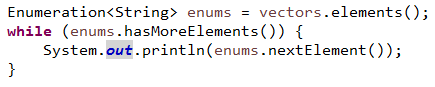




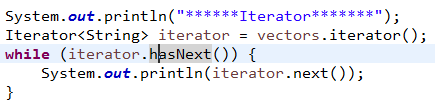
创建一个Vector：



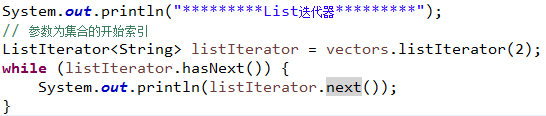
**获取Enumeration**：



**获取Iterator**：



**获取ListIterator**：



# ListIterator接口：

## 简单介绍



**ListIterator<E>**接口是**Iterator<E>接口**的子接口。在Iterator接口的基础上，又拓展了很多新的功能。

## 方法介绍

### hasNext、next、nextIndex、remove









### 向前：hasPrevious、previous、previousIndex







### add(E e):添加一个元素。



### set(E e):替换next或previous返回的元素。

