迭代器(Iterator)是一个对象，它的工作是遍历并选择序列中的对象，**它提供了一种访问一个容器(container)对象中的各个元素，而又不必暴露该对象内部细节的方法**。通过迭代器，开发人员不需要了解容器底层的结构，就可以实现对容器的遍历。由于创建迭代器的代价小，因此迭代器通常被称为轻量级的容器。

**一、迭代器的使用主要有以下三个方面的注意事项：**

1. 使用容器的iterator()方法**返回一个Iterator**，然后**通过Iterator的next()方法返回第一个元素**。

Iterator<String> listIt = list.iterator();

String a = listIt.next();

1. **使用Iterator()的hasNext()方法判断容器中是否还有元素**，如果有，可以使用next()方法获取下一个元素。

while (listIt.hasNext()) {

Object obj = listIt.next();

if (obj.equals("张三3")) {

list.remove(obj);

}

}

3）可以**通过remove()方法删除迭代器返回的元素**。

**二、**

Iterator支持派生的兄弟成员。ListIterator只存在于List中，支持在迭代期间向List中添加或删除元素，并且可以在List中双向滚动。

Iterator的使用方法如下例所示：

package com.js;

/\*\*

\* Iterator使用Demo

\*/

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

public class Test {

public void main(String[] args){

List<String> list = new LinkedList<String>();

list.add("first");

list.add("second");

list.add("third");

list.add("fourth");

for(Iterator<String> listIt = list.iterator();listIt.hasNext();) {

String string = (String)listIt.next();

System.out.println(string);

}

}

}

运行结果为：

first

second

third

fourth

**三、**

使用iterator()方法时经常会遇到ConcurrentModificationException异常，这通常是由于在使用Iteraor遍历容器的同时又对容器做增加或删除操作所导致的，或者由于多线程操作导致，当一个线程使用迭代器遍历容器的同时，另外一个线程对这个容器进行增加或删除操作。下列主要介绍单线程抛出ConcurrentModificationException的情况：

package com.js;

/\*\*

\* Iterator使用Demo

\*/

import java.util.Iterator;

import java.util.LinkedList;

import java.util.List;

public class Test {

public void main(String[] args){

List<String> ll = new LinkedList<String>();

ll.add("first");

ll.add("second");

ll.add("third");

ll.add("fourth");

for(Iterator<String> iterator = ll.iterator();iterator.hasNext();)

{

String string = (String)iterator.next();

System.out.println(string);

if(string.equals("second"))

ll.add("fifth");

}

}

}

运行结果为：

first

second

Exception in thread "main" java.util.ConcurrentModificationException

at java.util.LinkedList$ListItr.checkForComodification(Unknown Source)

at java.util.LinkedList$ListItr.next(Unknown Source)

at com.js.Test.main(Test.java:17)

抛出上述异常的主要原因是当条用容器的iterator()方法返回Iterator对象时，把容器中包含对象的个数赋值给了一个变量expectedModCount，在调用next()方法时会比较变量expectedModCount与容器中实际对象的个数modCount的值是否相等，若二者不相等，则会抛出ConcurrentModificationException异常，因此在使用Iterator遍历容器的过程中，如果对容器进行增加或删除操作，就会改变容器中对象的数量，从而导致抛出异常。解决办法如下：**在遍历的过程中把需要删除的对象保存到一个集合中，等遍历结束后再调用removeAll()方法来删除，或者使用iterator.remove()方法**。

以上主要介绍了单线程的解决方案，那么多线程访问容器的过程中抛出ConcurrentModificationException异常了又该怎么解决呢？

1）在JDK1.5版本引入了线程安全的容器，比如ConcurrentHashMap和CopyOnWriteArrayList等。可以使用这些线程安全的容器来代替非线程安全的容器。

2）在使用迭代器遍历容器时对容器的操作放到synchronized代码块中，但是当引用程序并发程度比较高时，这会严重影响程序的性能。

引申：Iterator和ListIterator有什么区别？

Iterator只能正向遍历集合，适用于获取移除元素。**ListIterator继承自Iterator，专门针对List，可以从两个方向来遍历List，同时支持元素的修改**。