Java中集合的迭代器原理

1. **Collection、List、Set接口**都继承了**Iterable接口**，所以也都继承了**Iterable接口**中的iterator方法，List集合和Set集合子类都实现了接口中的iterator方法。这些集合中的迭代器都利用内部类实现的，也就是，在集合类中创建一个实现Iterable接口、Iterator接口、或者ListIterator接口的迭代类，然后利用Iterator方法或者listIterator方法返回这个内部类创建的**迭代器对象，利用接口引用接收即可（多态）**。**对外无需提供集合内部创建的迭代类**。

**例如：****，Itr就是一个实现了接口Iterator的内部类，然后iterator方法：**

**返回内部类的一个对象即可**。

1. **迭代器iterator：可以获取集合中的元素，它是通过内部类实现的。**

**Collection中的iterator方法，就是为了获取集合中的迭代器对象，该对象必须依赖于具体容器，因为每一个容器中的数据结构都不同，所以该迭代器对象是在容器中进行内部类实现的。对于使用容器者而言，具体实现并不重要，只要通过容器获取到该实现的迭代器的对象即可，也就是iterator方法。Iterator 是一个接口，是对所有的容器（Collection）进行元素提取出的公共接口。**

Collection coll = new ArrayList();

Iterator it = coll.iterator();//coll: [abc1, abc2, abc3]

while (it.hasNext())

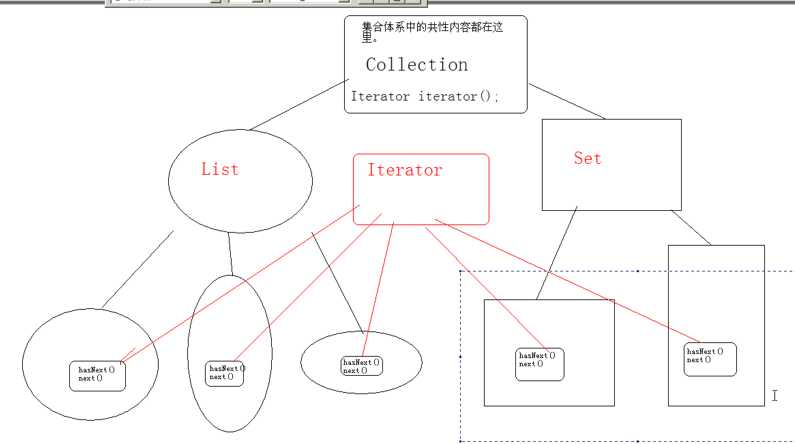
System.out.println(it.next()); //浪费内存，因为执行完后，迭代器仍然存在

**一般不用while（）语句，利用for循环语句，好处在于迭代器定义在for循环中，运行完之后就消失了。**

**for(Iterator it1 = coll.iterator(); it1.hasNext(); )**

**System.*out*.println(it1.next());**

1. **迭代器必须依赖于具体的集合容器，因为每个容器的数据结构都不相同，所以迭代器对象在容器中是利用内部类实现的。Iterator接口**是对各个集合容器进行**共性抽取**出来的**公共接口**。**对于使用者来说，迭代器的具体实现并不重要，只要通过容器获取到此集合容器的迭代器对象即可。**
2. 图解：



1. **迭代器对集合元素的输出方法是**：（前提：list：[89, abc1, abc2]）

**for(Iterator it = list.iterator();it.hasNext();)**

**System.out.println(it.next());**

由于List集合中的元素都具有索引，所以List 有一种自己特有的对集合元素输出的方法（Set不具备），通过**get直接**获取元素：

**for(int i = 0;i < list.size();i++)**

**System.out.println(list.get(i));**

1. **ListIterator接口：**
2. **ListIterator接口是Iterator接口的一个子接口，对其进行了功能扩展；Iterator接口中只有三个方法：hasNext、next、remove方法，而ListIterator接口中拓展了很多，如下所述。**
3. **ListIterator接口中的方法：**
4. **add方法：添加方法**



1. **remove方法：删除方法**



1. **hasNext和hasPrevious方法：**





1. **next方法和previous方法：**





1. **nextIndex和previousIndex方法：**





1. **set方法**



1. 下面的是出现异常的代码：

List list = **new** ArrayList();

list.add(89);list.add("abc1");list.add("abc2");

**for**(Iterator it = list.iterator();it.hasNext();) {

Object obj = it.next();

**if**(obj == "abc1")

list.add("abc6");

**//异常：java.util.ConcurrentModificationException由于并发操作集合出现此异常**

**else**

System.***out***.println(obj);

}

**解析：不要在使用迭代器的使用过程中，使用集合中的方法直接操作元素，会出现并发操作异常即ConcurrentModificationException，因为迭代过程中，迭代器正在操作此集合，如果再利用集合中的方法操作集合，就会出现并发操作，这是不允许的。**

**解决办法：**

1. **Iterator接口中提供的方法很少，需要添加元素操作，这时可以利用Iterator的子接口ListIterator中的方法，利用listIterator（）方法可以直接获取ListIterator子接口的实现对象。listIterator迭代器中有增删改查等的操作方法。如：**ListIterator lit = list.listIterator()；这句话是获取**列表迭代器对象，可以实现在迭代过程中对元素的增删改查等操作。**

注意：这种**获取listIterator子接口的迭代器对象**只有List拥有，其他接口无此功能。listIterator对集合的元素可以在两个方向上任意遍历，分别对应它的两个方法：**next（）向后走，previous（）向前走。hasNext（） 与hasPrevious（）对应。**

1. **利用索引进行遍历：由于List集合都具有索引，所以利用索引进行遍历。**
2. **传统for循环：**

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

list.set(i,i);

System.out.println(list.get(i));

}

1. **高级for循环（注意此方式只可以获取，不可以操作）：**

for (Object object : list) {

System.out.println(object);

}