

【Java面试题】List如何一边遍历，一边删除？

2020-03-20 12:04 申城异乡人 阅读(5655) 评论(4) 编辑 收藏

这是最近面试时被问到的1道面试题，本篇博客对此问题进行总结分享。

1. 新手常犯的错误

可能很多新手（包括当年的我，哈哈）第一时间想到的写法是下面这样的：

```
public static void main(String[] args) {
    List<String> platformList = new ArrayList<>();
    platformList.add("博客园");
    platformList.add("CSDN");
    platformList.add("掘金");

    for (String platform : platformList) {
        if (platform.equals("博客园")) {
            platformList.remove(platform);
        }
    }

    System.out.println(platformList);
}
```

然后满怀信心的去运行，结果竟然抛

`java.util.ConcurrentModificationException` 异常了，翻译成中文就是：并发修改异常。

```
C:\Program Files\Java\jdk1.8.0_191\bin\java.exe ...
Exception in thread "main" java.util.ConcurrentModificationException
    at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:909)
    at java.util.ArrayList$Itr.next(ArrayList.java:859)
    at com.zwwhnly.springbootaction.ListDemo.main(ListDemo.java:15)

Process finished with exit code 1
```

是不是很懵，心想这是为什么呢？

让我们首先看下上面这段代码生成的字节码，如下所示：

```
2 public class ListDemo {
3     public ListDemo() {
4     }
5
6     public static void main(String[] args) {
7         List<String> platformList = new ArrayList<>();
8         platformList.add("博客园");
9         platformList.add("CSDN");
10        platformList.add("掘金");
11        Iterator var2 = platformList.iterator();
12
13        while(var2.hasNext()) {
14            String platform = (String)var2.next();
15            if (platform.equals("博客园")) {
16                platformList.remove(platform);
17            }
18        }
19
20        System.out.println(platformList);
21    }
22 }
```

由此可以看出，foreach循环在实际执行时，其实使用的是 `Iterator`，使用的核心方法是 `hasnext()` 和 `next()`。

然后再来看下ArrayList类的Iterator是如何实现的呢？

About

昵称: [申城异乡人](#)
园龄: [8年9个月](#)
粉丝: [219](#)
关注: [3](#)
[+加关注](#)

最新随笔

- Java Optional使用指南
- 为什么要谨慎使用Arrays.asList、ArrayList的subList？
- 【踩坑系列】使用long类型处理金额，科学计数法导致金额转大写异常
- 【踩坑系列】MySQL only_full_group_by配置，竟导致所有应用报错？
- 使用Java Stream，提取集合中的某一列/按条件过滤集合/求和/最大值/最小值/平均值
- Java BigDecimal使用指南
- 【深度思考】如何优雅告知用户，网站正在升级维护？
- Redis系列(九)：Redis的事务机制
- Redis系列(八)：发布与订阅
- 【深度思考】JDK8中日期类型该如何使用？

随笔分类

随笔档案

IDE(2)	2021年1月(1)
Java(19)	2020年11月(1)
MyBatis(14)	2020年10月(2)
MySQL(2)	2020年9月(1)
Nacos(1)	2020年8月(2)
Nginx(1)	2020年7月(1)
Other(3)	2020年6月(3)
RabbitMQ(5)	2020年5月(2)
Redis(9)	2020年4月(2)
Spring(15)	2020年3月(2)
Spring Boot(2)	2020年2月(1)
多线程(2)	2020年1月(2)
面试题(3)	2019年12月(1)

阅读排行榜

1. 数据库管理工具DataGrip使用总结(一)(76471)
2. MyBatis系列(六)：MyBatis动态Sql之i标签的用法(15665)
3. RabbitMQ使用教程（二）RabbitMQ用户管理、角色管理及权限设置(14546)
4. Spring入门(四)：使用Maven管理Spring项目(13117)

更多

积分与排名

积分 - 190528
排名 - 3999

推荐排行榜

```

@NotNull public Iterator<E> iterator() {
    return new Itr();
}

/**
 * An optimized version of AbstractList.Itr
 */
private class Itr implements Iterator<E> {
    int cursor; // index of next element to return
    int lastRet = -1; // index of last element returned; -1 if no such
    int expectedModCount = modCount;

    Itr() {}

    public boolean hasNext() {
        return cursor != size;
    }

    @Override
    public E next() {
        checkForComodification();
        int i = cursor;
        if (i >= size)
            throw new NoSuchElementException();
        Object[] elementData = ArrayList.this.elementData;
        if (i >= elementData.length)
            throw new ConcurrentModificationException();
        cursor = i + 1;
        return (E) elementData[lastRet = i];
    }

    public void remove() {
        if (lastRet < 0)
            throw new IllegalStateException();
        checkForComodification();

        try {
            ArrayList.this.remove(lastRet);
            cursor = lastRet;
            lastRet = -1;
            expectedModCount = modCount;
        } catch (IndexOutOfBoundsException ex) {
            throw new ConcurrentModificationException();
        }
    }

    @Override
    @Override
    public void forEachRemaining(Consumer<? super E> consumer) {...}

    final void checkForComodification() {
        if (modCount != expectedModCount)
            throw new ConcurrentModificationException();
    }
}

```

可以看出，调用 `next()` 方法获取下一个元素时，第一行代码就是调用了

`checkForComodification()`，而该方法的核心逻辑就是比较

`modCount` 和 `expectedModCount` 这2个变量的值。

在上面的例子中，刚开始 `modCount` 和 `expectedModCount` 的值都为3，所以第1次获取元素"博客园"是没问题的，但是当执行完下面这行代码时：

```
platformList.remove(platform);
```

`modCount` 的值就被修改成了4。

```

public boolean remove(Object o) {
    if (o == null) {
        for (int index = 0; index < size; index++)
            if (elementData[index] == null) {
                fastRemove(index);
                return true;
            }
    } else {
        for (int index = 0; index < size; index++)
            if (o.equals(elementData[index])) {
                fastRemove(index);
                return true;
            }
    }
    return false;
}

/**
 * Private remove method that skips bounds checking and does not
 * return the value removed.
 */
private void fastRemove(int index) {
    modCount++;
    int numMoved = size - index - 1;
    if (numMoved > 0)
        System.arraycopy(elementData, index+1, elementData, index,
            numMoved);
    elementData[--size] = null; // clear to let GC do its work
}

```

所以在第2次获取元素时，`modCount` 和 `expectedModCount` 的值就不相等了，所以抛出了 `java.util.ConcurrentModificationException` 异常。

5. Spring入门(一): 创建Spring项目(103 59)

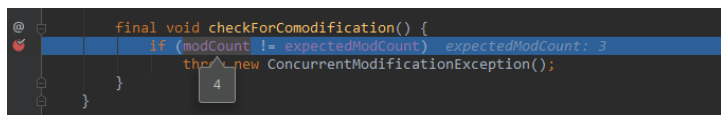
1. 程序员如何巧用Excel提高工作效率(15)

2. 【Redis面试题】如何使用Redis实现微信步数排行榜？(11)

3. 【Java面试题】List如何一边遍历，一边删除？(7)

4. 2019年终总结 | 我的写博元年及技术成长之路(7)

5. 【深度思考】如何优雅告知用户，网站正在升级维护？(6)



既然不能使用foreach来实现，那么我们该如何实现呢？

主要有以下3种方法：

1. 使用Iterator的remove()方法
2. 使用for循环正序遍历
3. 使用for循环倒序遍历

接下来——讲解。

2. 使用Iterator的remove()方法

使用Iterator的remove()方法的实现方式如下所示：

```
public static void main(String[] args) {
    List<String> platformList = new ArrayList<>();
    platformList.add("博客园");
    platformList.add("CSDN");
    platformList.add("掘金");

    Iterator<String> iterator = platformList.iterator();
    while (iterator.hasNext()) {
        String platform = iterator.next();
        if (platform.equals("博客园")) {
            iterator.remove();
        }
    }

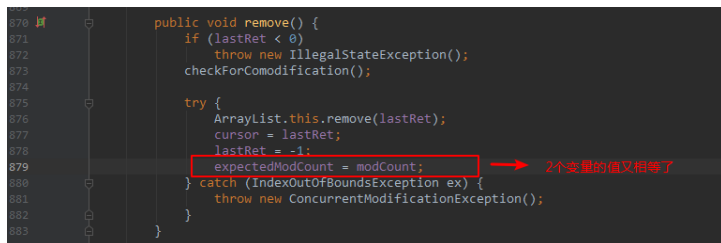
    System.out.println(platformList);
}
```

输出结果为：

[CSDN, 掘金]

为什么使用 `iterator.remove()` 就可以呢？

让我们看下它的源码：



可以看出，每次删除一个元素，都会将 `modCount` 的值重新赋值给

`expectedModCount`，这样2个变量就相等了，不会触发

`java.util.ConcurrentModificationException` 异常。

3. 使用for循环正序遍历

使用for循环正序遍历的实现方式如下所示：

```
public static void main(String[] args) {
    List<String> platformList = new ArrayList<>();
    platformList.add("博客园");
    platformList.add("CSDN");
    platformList.add("掘金");
}
```

```
for (int i = 0; i < platformList.size(); i++) {  
    String item = platformList.get(i);  
  
    if (item.equals("博客园")) {  
        platformList.remove(i);  
        i = i - 1;  
    }  
}  
  
System.out.println(platformList);  
}
```

这种实现方式比较好理解，就是通过数组的下标来删除，不过有个注意事项就是删除元素后，要修正下下标的值：

```
i = i - 1;
```

为什么要修正下标的值呢？

因为刚开始元素的下标是这样的：

0	博客园
1	CSDN
2	掘金

第1次循环将元素"博客园"删除后，元素的下标变成了下面这样：

0	CSDN
1	掘金

第2次循环时i的值为1，也就是取到了元素"掘金"，这样就导致元素"CSDN"被跳过检查了，所以删除完元素后，我们要修正下下标，这也是上面代码中

```
i = i - 1;
```

 的用途。

4. 使用for循环倒序遍历

使用for循环倒序遍历的实现方式如下所示：

```
public static void main(String[] args) {  
    List<String> platformList = new ArrayList<>();  
    platformList.add("博客园");  
    platformList.add("CSDN");  
    platformList.add("掘金");  
  
    for (int i = platformList.size() - 1; i >= 0; i--) {  
        String item = platformList.get(i);  
  
        if (item.equals("掘金")) {  
            platformList.remove(i);  
        }  
    }  
  
    System.out.println(platformList);  
}
```

这种实现方式和使用for循环正序遍历类似，不过不用再修正下标，因为刚开始元素的下标是这样的：

0	博客园
1	CSDN
2	掘金

第1次循环将元素"掘金"删除后，元素的下标变成了下面这样：

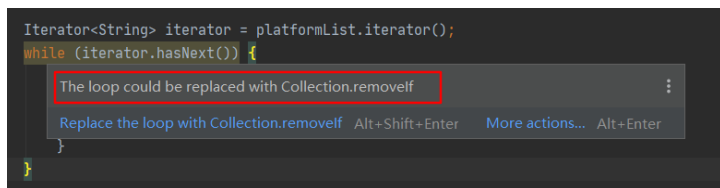
0	博客园
1	CSDN

第2次循环时i的值为1，也就是取到了元素"CSDN"，不会导致跳过元素，所以不需要修正下标。

5. 评论区释疑(2020-06-15更新)

5.1 使用removeIf()方法(推荐)

从JDK1.8开始，可以使用 `removeIf()` 方法来代替 `Iterator` 的 `remove()` 方法实现一边遍历一边删除，其实，IDEA中也会提示：



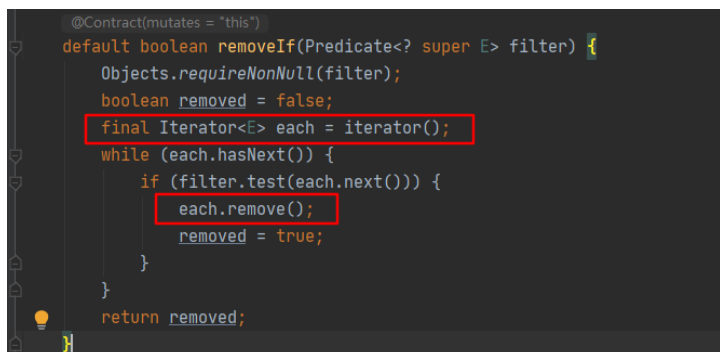
所以原来的代码：

```
Iterator<String> iterator = platformList.iterator();
while (iterator.hasNext()) {
    String platform = iterator.next();
    if (platform.equals("博客园")) {
        iterator.remove();
    }
}
```

就可以简化为如下所示的1行代码，非常简洁：

```
platformList.removeIf(platform -> "博客园".equals(platform));
```

看下removeIf()方法的源码，会发现其实底层也是用的 `Iterator` 的 `remove()` 方法：



5.2 使用for循环正序遍历，是否需要修正下标？

先说结论：需要。

不过之前文中举得例子不是太好，所以好多读者看完认为不修正下标也是可以的，其实不是，我们换个例子来理解：

```
List<String> platformList = new ArrayList<>();
platformList.add("博客园");
platformList.add("博客园");
platformList.add("CSDN");
platformList.add("掘金");

for (int i = 0; i < platformList.size(); i++) {
    String item = platformList.get(i);
    if ("博客园".equals(item)) {
        platformList.remove(i);
    }
}

System.out.println(platformList);
```

输出结果：

[博客园, CSDN, 掘金]

可以发现，如果不修正下标，第2个元素“博客园”在循环遍历时被跳过了，也就无法删除，所以一定要修正下标：

```
for (int i = 0; i < platformList.size(); i++) {
    String item = platformList.get(i);
    if ("博客园".equals(item)) {
        platformList.remove(i);
        i = i - 1;
    }
}
```

6. 参考

[Java集合怎么一边删除一边遍历](#)

[java 为什么遍历的时候不能删除元素](#)

好文要顶

关注我

收藏该文



申城异乡人

关注 - 3

粉丝 - 219

+加关注

7

0

- « 上一篇： [【Java面试题】如何判断一个字符串中某个字符出现的次数？](#)
- » 下一篇： [Redis系列\(四\)：Redis的复制机制\(主从复制\)](#)

分类 [Java](#) , [面试题](#)

标签 [Java](#) , [面试题](#)

刷新评论

刷新页面

返回顶部

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

- 园子动态：
- 致园友们的一封检讨书：都是我们的错
 - 数据库实例 CPU 100% 引发全站故障
 - 发起一个开源项目：博客引擎 fluss

最新新闻：

- 182亿背后的沉默和喧嚣、爱护和警醒
 - 对标特斯拉Semi？吉利回应开发纯电重卡传闻：不予置评
 - 阿里：品牌在其他平台开店，不会对公司造成实质负面影响
 - 无孔化就是手机的未来？还有很多问题需要解决
 - 喜马拉雅秘密提交美国IPO申请 估值50亿美元
- » 更多新闻...