

声明，本文使用的是JDK1.8

从今天开始正式去学习Java基础中最重要的东西--->集合

无论在开发中，在面试中这个知识点都是非常非常重要的，因此，我在此花费的时间也是很多，得参阅挺多的资料，下面未必就做到日更了...

当然了，如果讲得有错的地方还请大家多多包涵并不吝在评论去指正~

一、集合(Collection)介绍

1.1为什么需要Collection

1. Java是一门面向对象的语言，就免不了处理对象
2. 为了方便操作多个对象，那么我们就得把这多个对象存储起来
3. 想要存储多个对象(变量),很容易就能想到一个容器
4. 常用的容器我们知道有-->StringBuffered,数组(虽然有对象数组，但是数组的长度是不可变的！)
5. 所以，Java就为我们提供了集合(Collection)~

1.2数组和集合的区别

接下来，我们可以对数组和集合的区别来分析一下：

数组和集合的区别：

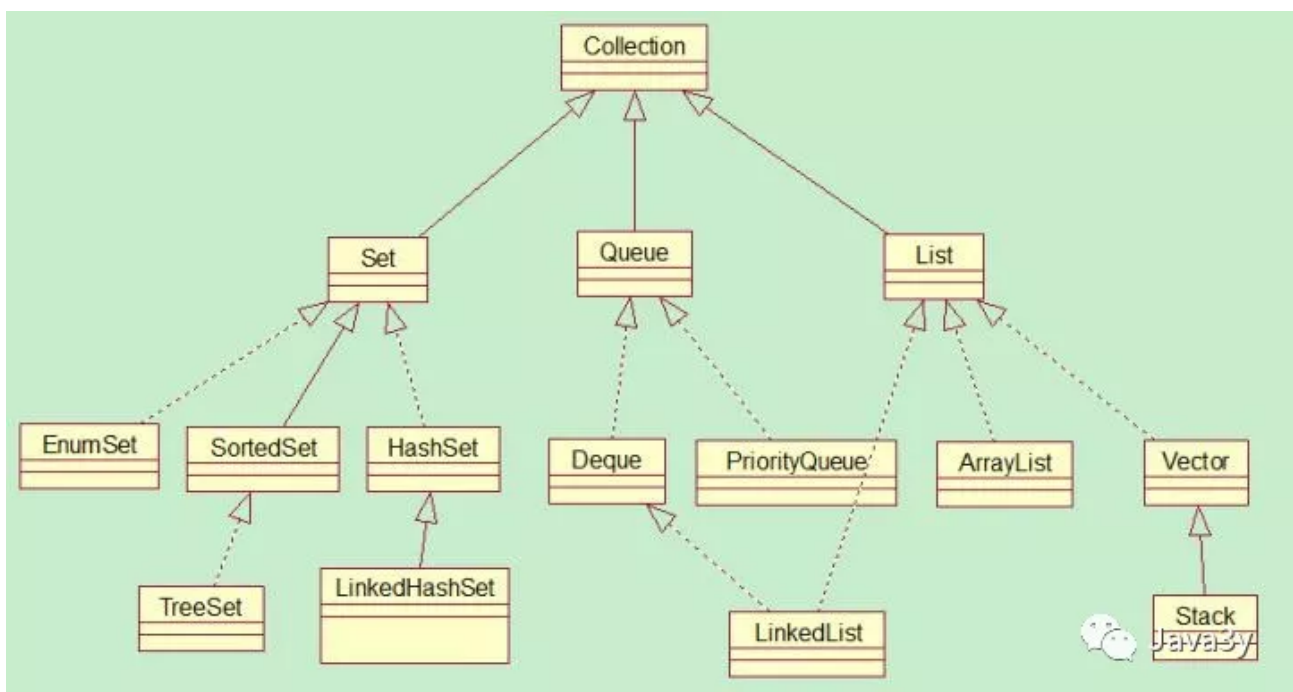
- 1:长度的区别
 - **数组的长度固定**
 - **集合的长度可变**
- 2:内容不容
 - 数组存储的是同一种类型的元素
 - 集合可以存储不同类型的元素(但是一般我们不这样干..)
- 3:元素的数据类型
 - 数组可以存储基本数据类型,也可以存储引用类型
 - **集合只能存储引用类型(你存储的是简单的int, 它会自动装箱成Integer)**

1.3Collection的由来与功能

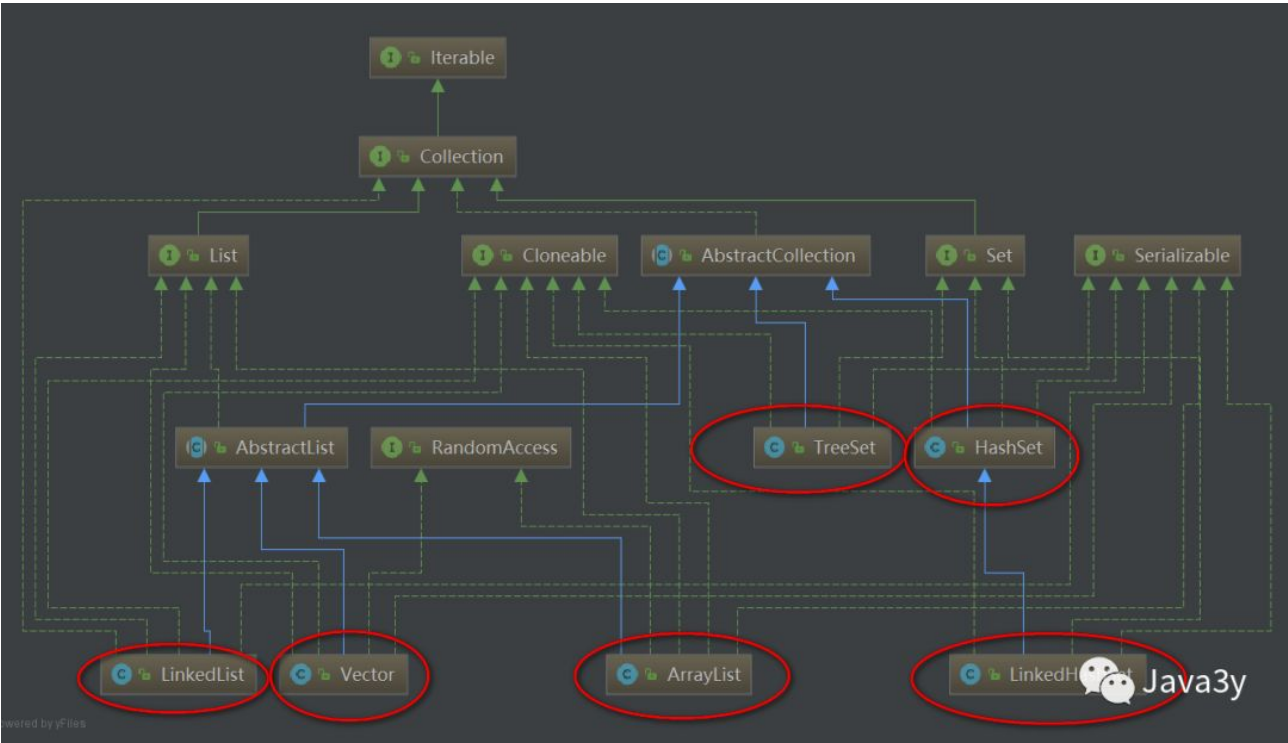
Collection的由来：

- 集合可以存储多个元素,但我们**对多个元素也有不同的需求**
 - 多个元素,不能有相同的
 - 多个元素,能够按照某个规则排序
- 针对不同的需求: java就提供了很多集合类, 多个集合类的数据结构不同。但是, 结构不重要, 重要的是**能够存储东西,能够判断,获取**
- 把集合**共性的内容不断往上提取**,最终形成集合的继承体系---->Collection

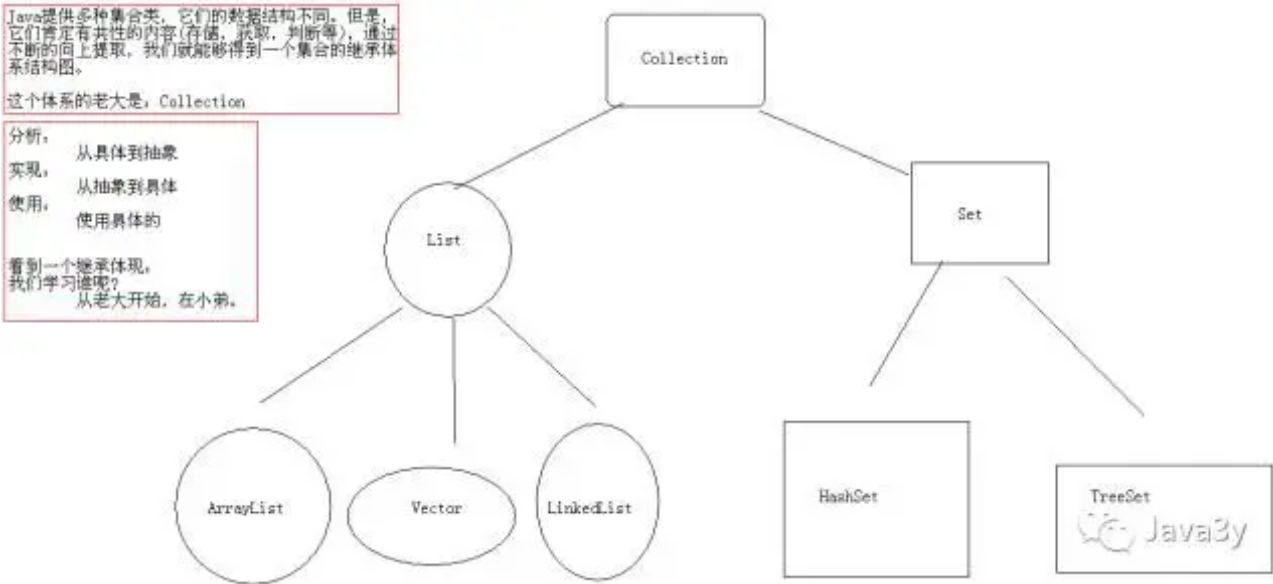
Collection的大致结构体系是这样的：



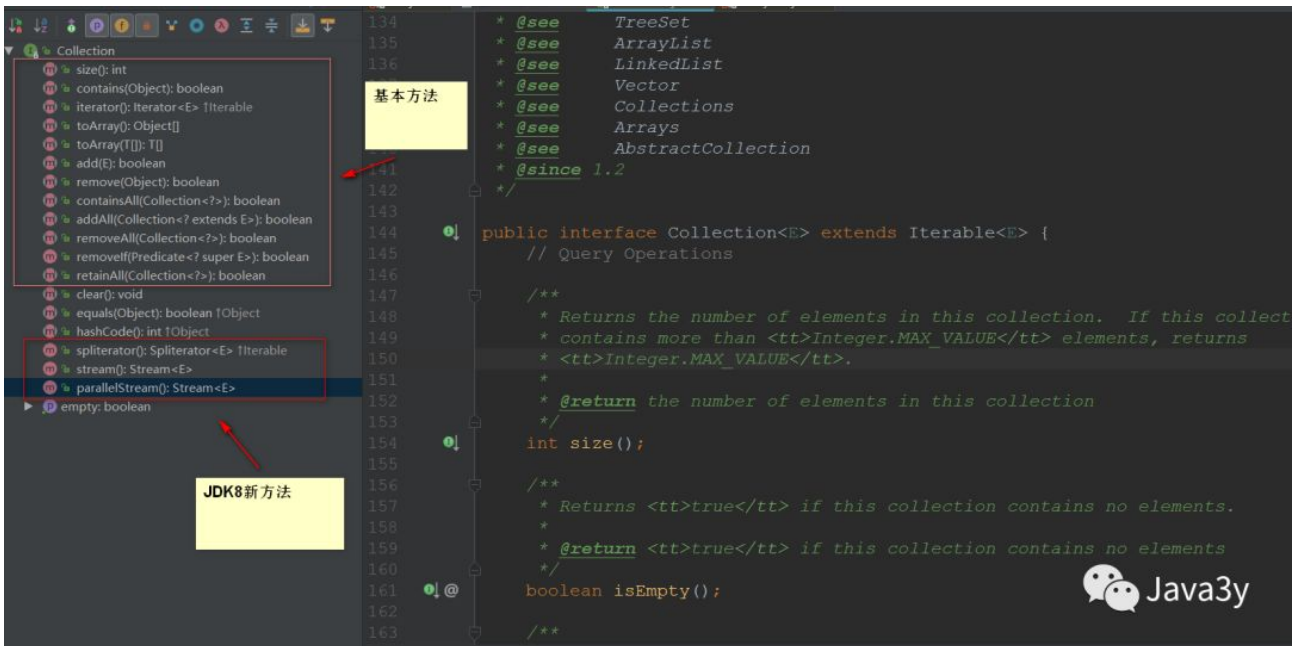
但是，一般我们要掌握的并不需要那么多，只需要掌握一些常用的集合类就行了。下面我圈出来的那些：



再次精减：



Collection的基础功能：



Collection的功能:

1: 添加功能

boolean add(Object obj): 添加一个元素
boolean addAll(Collection c): 添加一个集合的元素

2: 删除功能:

void clear(): 移除所有的元素
boolean remove(Object): 移除一个元素
boolean removeAll(Collection c): 移除一个集合的元素, 只要一个元素被移除了, 就返回true

3: 判断功能:

boolean contains(Object o): 判断集合是否包含该元素
boolean containsAll(Collection c): 判断集合中是否包含指定的集合元素, 只有包含所有的元素, 才叫包含
boolean isEmpty(): 判断集合是否为空

4: 获取功能:

Iterator<E> iterator(): 迭代器

5: 长度功能:

int size(): 元素的个数

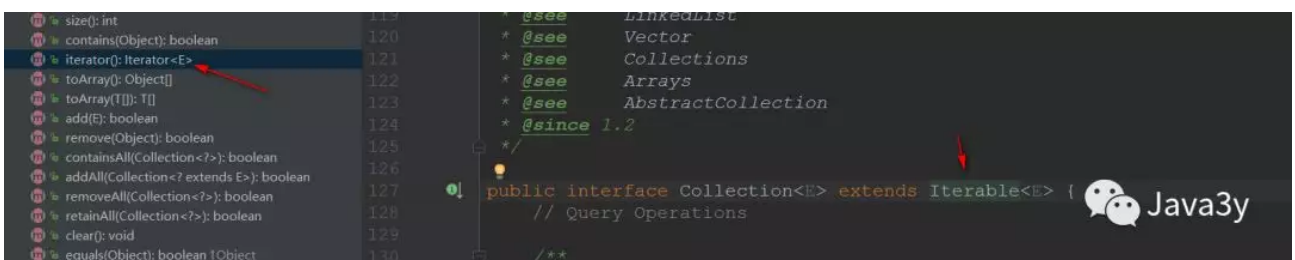
6: 交集功能:

boolean retainAll(Collection c): 移除此collection中未包含在指定collection中的所有元素。
集合A和集合B做交集, 最终的结果保存在集合A, 返回值表示的是A是否发生过变化。

Java3y

二、迭代器(Iterator)介绍

我们可以发现Collection的源码中继承了Iterable, 有iterator()这个方法...



点进去看了一下, Iterable是一个接口:

```
import java.util.Iterator;

/**
 * Implementing this interface allows an object to be the target of
 * the "foreach" statement.
 *
 * @param <T> the type of elements returned by the iterator
 *
 * @since 1.5
 */
public interface Iterable<T> {

    /**
     * Returns an iterator over a set of elements of type T.
     *
     * @return an Iterator.
     */
    Iterator<T> iterator();
}
```



它有iterator()这个方法，返回的是**Iterator**

再来看一下，Iterator也是一个接口，它只有三个方法：

- hasNext()
- next()
- remove()

```

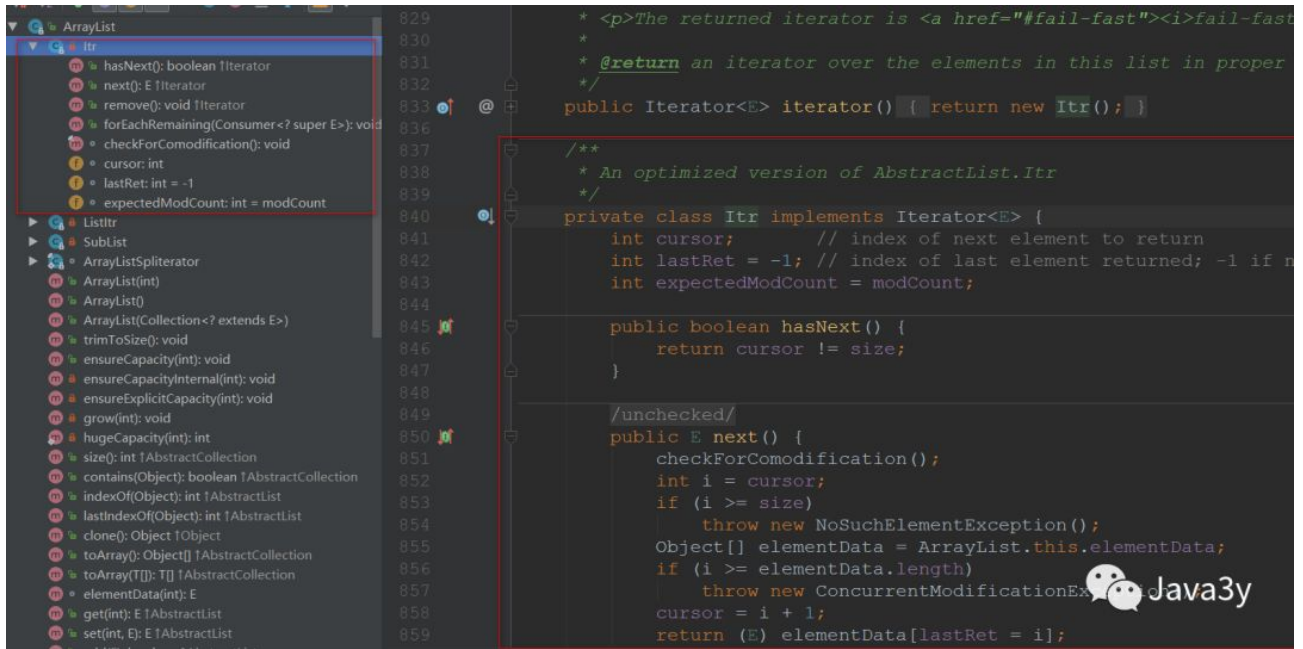
51  /**
52  * public interface Iterator<E> {
53  *     /**
54  *      * Returns (@code true) if the iteration has more elements.
55  *      * (In other words, returns (@code true) if (@link #next) would
56  *      * return an element rather than throwing an exception.)
57  *
58  *      * @return (@code true) if the iteration has more elements
59  *      */
60  *     boolean hasNext();
61  *
62  *     /**
63  *      * Returns the next element in the iteration.
64  *
65  *      * @return the next element in the iteration
66  *      * @throws NoSuchElementException if the iteration has no more elements
67  *      */
68  *     E next();
69  *
70  *     /**
71  *      * Removes from the underlying collection the last element returned
72  *      * by this iterator (optional operation). This method can be called
73  *      * only once per call to (@link #next). The behavior of an iterator
74  *      * is unspecified if the underlying collection is modified while the
75  *      * iteration is in progress in any way other than by calling this
76  *      * method.
77  *
78  *      * @throws UnsupportedOperationException if the (@code remove)
79  *      *      operation is not supported by this iterator
80  *
81  *      * @throws IllegalStateException if the (@code next) method has not
82  *      *      yet been called, or the (@code remove) method has already
83  *      *      been called after the last call to the (@code next)
84  *      *      method
85  *      */
86  *     void remove();
87  * }

```



可是，我们没能找到对应的实现方法，只能往Collection的子类下找找了，于是我们找到了--->ArrayList(该类后面会说)

于是，我们在ArrayList下找到了iterator实现的身影：它是在ArrayList以**内部类的方式**实现的！并且，从源码可知：**Iterator实际上就是在遍历集合**



所以说：我们**遍历集合(Collection)的元素都可以使用Iterator**，至于它的具体实现是以内部类的方式实现的！

迭代器：是遍历集合的一种方式。
迭代器是依赖于集合而存在的。
我有一个集合：Collection c = new ArrayList();
我给集合中添加元素，c.add("hello");c.add("world");c.add("java");

hello world java

通过集合获取迭代器对象，Iterator it = c.iterator();

hello world java

```
while(it.hasNext()) {
    String s = (String)it.next();
    System.out.println(s);
}
```

迭代器的方法，next(),hasNext();

集合的使用步骤：

- 创建集合对象
- 创建元素对象
- 把元素添加到集合
- 遍历集合
 - 通过集合对象获取迭代器对象
 - 通过迭代器对象的hasNext()方法判断是否有元素
 - 通过迭代器对象的next()方法获取元素并移动到下一个位置

迭代器为什么不定义成一个类，而是一个接口？

假设迭代器定义的是一个类，这样我们就可以创建该类的对象，调用该类的方法来实现集合的遍历。但是呢？我们想想，Java中提供了很多的集合类，而这些集合类的数据结构是不同的，所以，存储的方式和遍历的方式应该是不同的。进而它们的遍历方式也应该是不一样的。最终，就没有定义迭代器类的必要了。

而无论你是那种集合，你都应该具备获取元素的操作，并且，最好在辅助于判断功能，这样，在获取时，先判断，这样就更不容易出错。也就是说，判断功能和获取功能应该是一个集合遍历所具备的，而每种集合的方式又不大一样。所以，我们把这两个功能给提取出来，并不提供具体实现。这种方式就是接口。

那么，真正的具体的实现类在哪里呢？

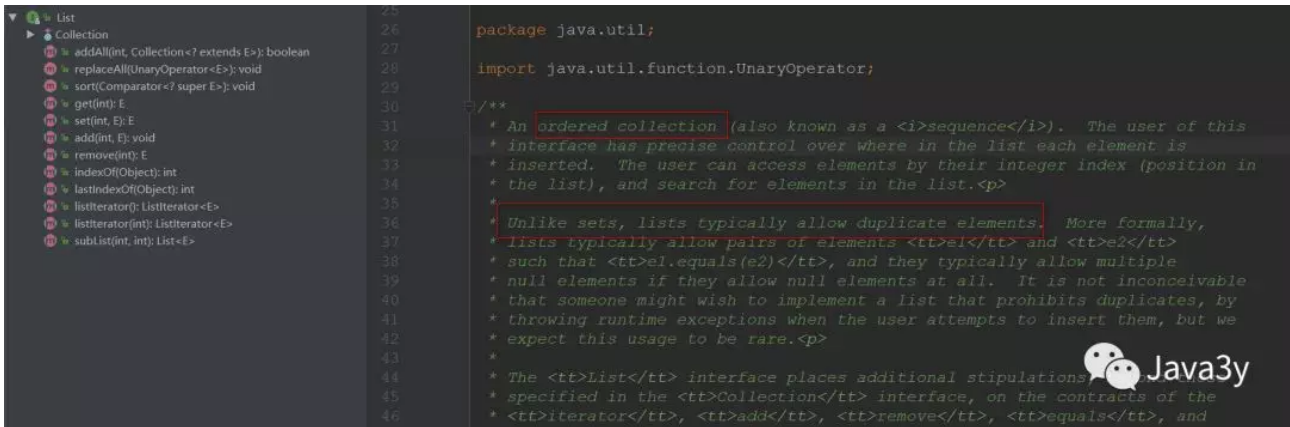
在真正的具体的子类中，以内部类的方式体现的。

三、List集合介绍

从上面已经可以看到了，Collection主要学习集合的类型两种：**Set和List**，这里主要讲解List！

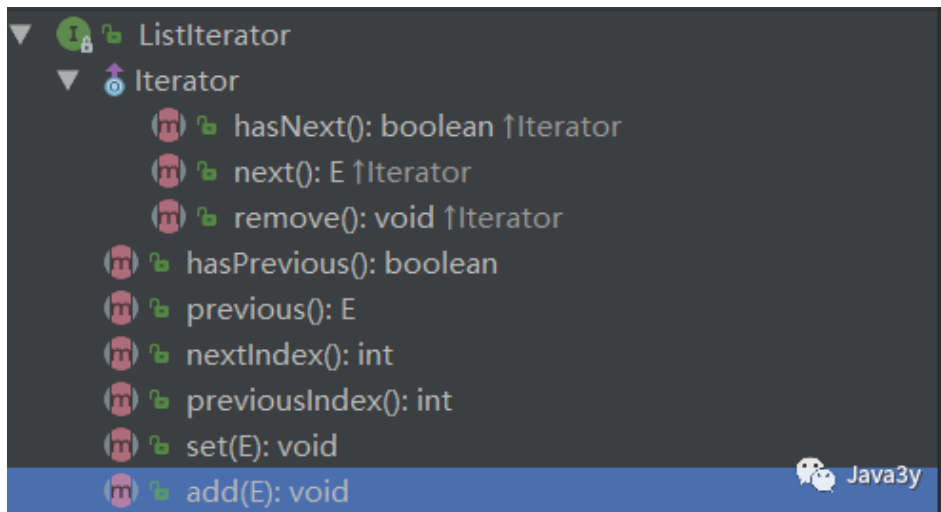
我们来看一下List接口的方法，比Collection多了一点点：

- List集合的**特点**就是：**有序(存储顺序和取出顺序一致),可重复**



Collection返回的是Iterator迭代器接口，而List中又有它自己对应的实现--> **ListIterator**接口

该接口比普通的Iterator接口多了几个方法：



从方法名就可以知道：**ListIterator**可以往前遍历，添加元素，设置元素

3.1 List集合常用子类

List集合常用的子类有三个：

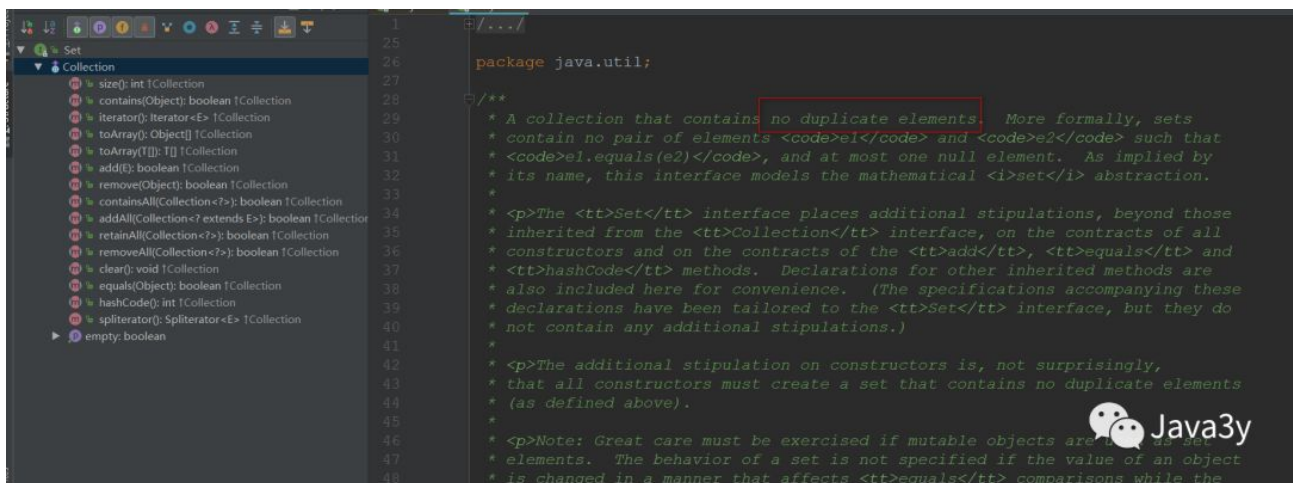
- ArrayList
 - 底层数据结构是数组。线程不安全
- LinkedList
 - 底层数据结构是链表。线程不安全
- Vector
 - 底层数据结构是数组。线程安全

现在知道有三个常用的集合类即可，后面会开新的文章来讲解的~

四、Set集合介绍

从Set集合的方法我们可以看到：方法没有比Collection要多

- Set集合的特点是：**元素不可重复**



4.1 Set集合常用子类

- HashSet集合
 - A:底层数据结构是哈希表(是一个元素为链表的数组)
- TreeSet集合
 - A:底层数据结构是红黑树(是一个自平衡的二叉树)
 - B:保证元素的排序方式
- LinkedHashSet集合
 - A: 底层数据结构由哈希表和链表组成。

