68从零开始学Java68之Set集合都有哪些特性

作者：孙玉昌，昵称【一一哥】，另外【壹壹哥】也是我哦

千锋教育高级教研员、CSDN博客专家、万粉博主、阿里云专家博主、掘金优质作者

前言

在上一篇文章中，**壹哥**带大家学习了List集合的用法和特性，尤其是对ArrayList和LinkedList了解的更多一些。但Java中还有Set和Map集合等待我们学习，所以接下来就请各位继续跟**壹哥**来学习今天的内容吧。在本文中，**壹哥**会详细地给大家介绍Set集合的定义、特点、常用方法和基本原理等内容。

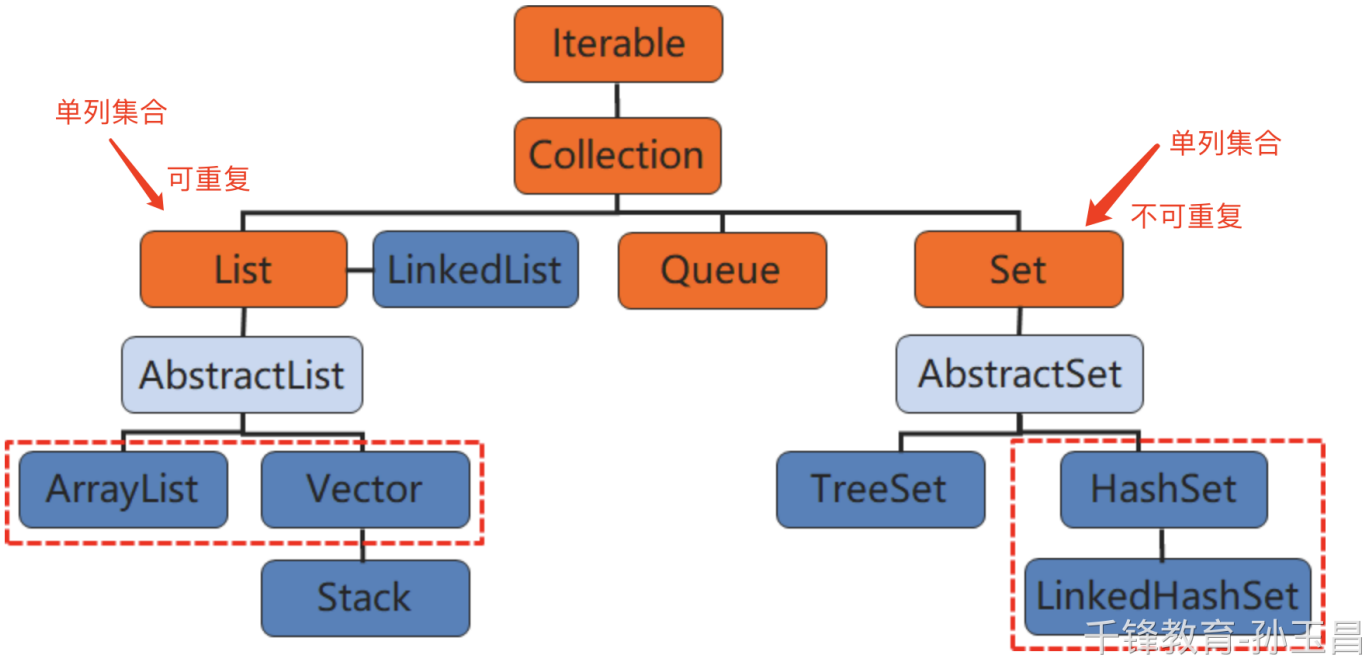
----------**前戏已做完，精彩即开始**----------

全文大约【4000】字，不说废话，只讲可以让你学到技术、明白原理的纯干货！本文带有丰富的案例及配图视频，让你更好地理解和运用文中的技术概念，并可以给你带来具有足够启迪的思考.....

配套开源项目资料

> Github：GitHub – SunLtd/LearnJava
>
> Gitee：一一哥/从零开始学Java

一. Set集合简介



1. Set定义

Set是Java的一种集合，继承自Collection接口，主要有两个常用的实现类**HashSet类和TreeSet类**。它没有固定的大小限制，可以动态地添加和删除元素。并且Set集合中的元素都是唯一的，不会有重复的元素，即使是null值也只能有一个。另外Set集合是无序的，不能记住元素的添加顺序，因为没有索引值，所以Set集合中的对象不会按特定的方式排序，它只是简单地把对象放到集合中。

从特性上来看，**Set相当于是一个只存储key、不存储value的Map**。我们可以把Set想象成是一个”特殊的Map“，这个Map只有key却没有value，所以我们可以用Set去除重复的元素。另外由于放入Set的元素和Map的key类似，需要正确地实现equals()和hashCode()方法，否则该元素就无法正确地放入Set。

2. Set特性

与其他集合不同，Set集合具有自己的一些特性：

> - Set集合中的元素都是唯一的，不允许有重复值，且最多只允许包含一个null元素；
> - Set集合中的元素没有顺序，我们无法通过索引来访问元素，但TreeSet是有序的；
> - Set集合没有固定的大小限制，可以动态地添加和删除元素；
> - Set集合提供了高效的元素查找和判断方法。

3. Set常用方法

Set集合给我们提供了一系列常用的方法，用于添加、删除、查找、遍历和获取集合元素等操作，下面是Set集合中常用方法的实现过程。

3.1 添加元素

我们可以使用add()方法进行元素的添加。

```java
public boolean add(E e)
```

该方法用于向Set集合添加元素，如果元素已经存在，则不会添加；如果添加成功，则返回true，否则返回false。该方法的示例代码如下：

```java
Set<String> set = new HashSet<>();
set.add("hello word");
set.add("java");
set.add("iOS");
System.out.println(set);
```

3.2 删除元素

我们可以使用remove()方法进行元素的删除。

```java
public boolean remove(Object o)
```

该方法用于从Set集合中删除指定的元素。如果元素存在且删除成功，则返回true，否则返回false。该方法的示例代码如下：

```java
Set<String> set = new HashSet<>();
set.add("hello word");
set.add("java");
set.remove("java");
System.out.println(set); // 输出结果为：[壹壹哥]
```

3.3 判断元素

我们可以使用contains()方法进行元素的判断。

```java
public boolean contains(Object o)
```

该方法用于判断Set集合中是否包含指定的元素。如果元素存在，则返回true，否则返回false。该方法的示例代码如下：

```java
Set<String> set = new HashSet<>();
set.add("hello word");
set.add("java");
System.out.println(set.contains("java")); // 输出结果为：true
System.out.println(set.contains("orange")); // 输出结果为：false
```

3.4 获取元素数量

我们可以使用size()方法判断集合的数量。

```java
public int size()
```

该方法的使用示例代码如下：

```java
Set<String> set = new HashSet<>();
set.add("hello word");
set.add("java");
System.out.println(set.size()); // 输出结果为：2
```

4. 配套视频

与本节内容配套的视频链接如下：

https://player.bilibili.com/player.html?bvid=BV1Ja411x7XB&p=153&page=153

二. HashSet集合

1. 简介

在Java的集合框架中，HashSet是一种非常常用的集合类型，它实现了Set接口，并继承了AbstractSet抽象类。HashSet集合的底层实现是一个哈希表，它使用哈希算法来存储和管理集合中的元素。HashSet集合中的元素没有顺序，且不允许重复。

如果我们想使用HashSet集合，一般要使用如下两个构造方法创建出HashSet对象：

> - **HashSet()**：构造一个新的空的Set集合对象；
> - **HashSet(Collection<? extends E> c)**：构造一个包含指定Collection集合元素的新Set集合。"< >"中的extends，表示这个Collection中的元素必须继承自HashSet的父类，该部分限定了Collection元素的类型。

2. HashSet特性

HashSet作为Set集合的具体子类，具有以下特点：

> - **HashSet的底层是基于HashMap来实现的；**
> - **HashSet中的元素是唯一的，内部不允许有重复的元素；**
> - **无序，不会记录插入元素的顺序，所以不能保证元素的排列顺序，获取顺序可能与添加顺序不同；**
> - **HashSet集合没有固定的大小限制，可以动态地添加和删除元素；**
> - **HashSet集合中的元素最多可以有一个null值；**
> - **HashSet不是线程安全的，默认线程不同步，如果有多个线程同时访问或修改同一个HashSet，必须通过代码来保证同步操作。**

3. 去重原理

从底层实现来看，**HashSet的底层其实就是一个值为Object的HashMap**，如下图所示：

```java
// Dummy value to associate with an Object in the backing Map
private static final Object PRESENT = new Object();

/**
 * Constructs a new, empty set; the backing {@code HashMap} instance has
 * default initial capacity (16) and load factor (0.75).
 */
public HashSet() {
    map = new HashMap<>();
}

 * @param e element to be added to this set
 * @return {@code true} if this set did not already contain the specified
 * element
 */
public boolean add(E e) {
    return map.put(e, PRESENT)==null;
}
```


所以**HashSet其实就是按照Hash算法来实现元素的查找和存储的**，具有很好的存取和查找性能。当我们向HashSet集合中存入一个元素时，HashSet会调用该对象的hashCode()方法来得到该对象的hashCode值，然后根据该hashCode值决定该对象在HashSet中的存储位置。此时如果有两个元素通过equals()方法进行比较，返回的结果为true，但它们的hashCode却不相等，HashSet也会把它们存储在不同的位置，我们依然可以添加成功。也就是说，**如果两个对象的hashCode值相等，且通过equals()方法比较返回的结果也为true， HashSet集合才会认为两个元素相等**。

与本节内容配套的视频链接如下：

https://player.bilibili.com/player.html?bvid=BV1Ja411x7XB&p=154&page=154

4. 使用案例

我们通过一个简单的案例，来看看HashSet的基本用法。

```java
import java.util.HashSet;

/**
 * @author 一一哥Sun
 */
public class Demo11 {

    public static void main(String[] args) {
        //创建HashSet集合
        HashSet<String> set = new HashSet<String>();
        set.add("一一哥");
        set.add("壹壹哥");
        set.add("java");
        //重复元素无法被添加进去
        set.add("java");
        System.out.println(set);

        //集合遍历
        Iterator<String> it = set.iterator();
        while (it.hasNext()) {
            //输出Set集合中的每个元素
            System.out.println("值="+it.next());
        }
    }
}
```

在上面的代码中，我们通过HashSet的构造方法创建了一个Set集合对象，并将几个元素对象存储到了这个Set集合中。然后我们使用HashSet类中的iterator()方法获取一个Iterator对象，并调用hasNext()方法遍历集合元素，再使用next()方法获取到下一个数据元素。但是HashSet输出的元素是无序的，输出时既不是添加元素的顺序，也不是String排序的顺序，在不同版本的JDK中，这个顺序可能也是不同的。另外因为Set是不可重复的，如果我们向Set集合中添加了两个相同的元素，则后添加的会覆盖前面添加的元素，所以Set集合中不会出现相同的元素。

5. 配套视频

与本节内容配套的视频链接如下：

https://player.bilibili.com/player.html?bvid=BV1Ja411x7XB&p=155&page=155

三. TreeSet集合

1. 简介

TreeSet是一种很常用的集合类型，它实现了Set和SortedSet接口，并且继承自AbstractSet抽象类。TreeSet集合中的元素也是唯一的，不允许重复。TreeSet集合的底层基于红黑树，可以使用自然排序或指定的比较器对集合中的元素进行排序。该类具有如下特点：

> - TreeSet集合中的元素是唯一的，不允许重复。
> - TreeSet集合中的元素是有序的，因为实现了SortedSet接口，具有字典顺序，可以通过迭代器按照升序或降序遍历。
> - TreeSet集合没有固定的大小限制，可以动态地添加和删除元素。
> - TreeSet集合提供了高效的元素查找和判断功能。

另外，SortedSet接口是Set接口的子接口，能够对集合进行自然排序，因此**TreeSet类默认情况下就是自然排序(升序)的**。但TreeSet只能对实现了Comparable接口的类对象进行排序，所以我们使用TreeSet集合存储对象时，该对象必须要实现Comparable接口。这是因为Comparable接口中有一个compareTo(Object o)方法，可以比较两个对象的大小。例如，a.compareTo(b)，如果 a 和 b 相等，则该方法会返回 0；如果 a 大于 b，则该方法返回大于 0 的正值；如果 a 小于 b，则该方法返回小于 0 的负值。

2. 常用方法

除了Set类中通用的方法之外，TreeSet类还有如下几个特有的方法：

方法名称	说明
E first()	返回该集合中的第一个元素，E表示返回元素的数据类型
E last()	返回该集合中的最后一个元素
E poolFirst()	获取并移除该集合中的第一个元素
E poolLast()	获取并移除该集合中的最后一个元素
SortedSet<E> subSet(E fromElement,E toElement)	返回一个新的集合，新集合会包含源集合fromElement与目标集合toElement之间的所有对象。结果会包含fromElement对象，但不包含toElement对象。
SortedSet<E> headSet<E toElement〉	返回一个新的集合，新集合包含原集合中toElement对象之前的所有对象，但不包含 toElement对象。

SortedSet<E> tailSet(E fromElement)	返回一个新的集合，新集合包含原集合中fromElement对象之后的所有对象，会包含fromElement对象。

因为TreeSet中的元素是有序的，所以增加了访问第一个、前一个、后一个、最后一个元素的相关方法，并提供了3个从 TreeSet中截取子TreeSet的方法。

3. 去重原理

当TreeSet集合在保存对象元素时，集合对象必须实现Comparable接口，并重写compareTo方法，该方法有如下两个作用：

> - **排序**: 返回值大于0表示升序，返回值小于0表示降序；
> - **去重(返回值为0)**：TreeSet认为返回0，表示两个对象是相同的对象。

所以我们利用TreeSet实现去重的原理就是：**如果compareTo()方法的返回值为0，则认为是相同的对象；如果compareTo()方法的返回大于0，则是升序排序；如果小于0，则是降序排序。**

4. 使用案例

接下来我们再通过一个案例来看看TreeSet的用法。

4.1 编写Person类

首先我们设计一个Person类，该类要实现Comparable接口。当TreeSet集合在保存对象元素时，集合中添加的元素对象必须实现Comparable接口，并重写compareTo方法。如果没有实现Comparable接口，那么创建TreeSet时必须传入一个Comparator对象。

Java | 复制代码

```java
/**
 * @author ——哥Sun
 * 实现Comparable接口，并重新compareTo()方法
 */
public class Person implements Comparable<Person>{

    private String username;
    private String password;

    public Person() {
    }

    public Person(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }

    @Override
    public String toString() {
        return "User [username=" + username + ", password=" + password + "]";
    }

    //重写compareTo()方法,对Person对象进行比较
    @Override
    public int compareTo(Person o) {
        if(!this.username.equals(o.username)) {
            //根据姓名及长度进行比较
            return this.username.length() - o.username.length();
        }else {
            //根据密码进行比较
            if(this.password.equals(o.password)) {
                return 0;
            }else {
                //比较姓名的长度
                return this.username.length() - o.username.length();
            }
        }
    }
}
```

与本节内容配套的视频链接如下：

https://player.bilibili.com/player.html?bvid=BV1Ja411x7XB&p=158&page=158

4.2 测试TreeSet排序功能

然后我们往TreeSet集合中添加若干个对象元素进行排序测试，代码如下：

Java | 复制代码

```java
import java.util.TreeSet;

/**
 * @author ——哥Sun
 */
public class Demo12 {

    public static void main(String[] args) {
        //TreeSet的去重原理
        TreeSet<Person> set = new TreeSet<Person>();
        set.add(new Person("admin","123"));
        set.add(new Person("yyg","bb"));
        set.add(new Person("jack","123"));
        set.add(new Person("rose123","123"));
        set.add(new Person("admin","123"));
        set.add(new Person("xksss6","abc"));

        //如果两个对象的用户名和密码都相等，则认为是两个相同的对象，且按照名字长度升序存放
        for (Person person : set) {
            System.out.println(person);
        }
    }
}
```

我们在遍历TreeSet时，输出的元素是有序的，这个顺序是元素的排序顺序。但是我们在使用TreeSet进行自然排序时，只能向 TreeSet 集合中添加相同数据类型的对象，否则会抛出ClassCastException异常。如果向 TreeSet集合中添加了一个 Double类型的对象，则后面只能添加 Double对象，不能再添加其他类型的对象，例如 String对象等。

5. 配套视频

与本节内容配套的视频链接如下：

https://player.bilibili.com/player.html?bvid=BV1Ja411x7XB&p=157&page=157

----------正片已结束，来根事后烟----------

四. 结语

至此，**壹哥**就带各位把Set集合及其子类学习完了，现在你学会了吗？本文的重点内容如下所示：

> - Set用于存储不重复的元素集合；
> - 放入HashSet的元素，与作为HashMap的key要求相同；
> - 放入TreeSet的元素，与作为TreeMap的Key要求相同；
> - 利用Set可以去除重复元素；
> - 遍历SortedSet时，可以按照元素的排序顺序进行遍历，我们也可以自定义排序算法；

另外如果你独自学习觉得有很多困难，可以加入**壹哥**的学习互助群，大家一起交流学习。

五. 今日作业

请实现如下需求：

有5名学生参加考试，老师录入每名学生的成绩后，程序会按照从低到高的排列顺序显示学生成绩。

另外老师还可以查询本次考试是否有满分、不及格的学生，及90分以上的学生有几名。