Set、HashSet和TreeSet

Djbfifjd

一、Set：无序、不可重复

1. 无序：指的是元素的添加顺序和迭代出来的顺序不一定相等。
2. 不可重复：内部通过 equals 方法来判断元素是否相等。
3. add() 返回 boolean 值。

```
Set<String> set = new HashSet<>();
System.out.println("添加第一个元素返回值:" + set.add("a"));//
true
System.out.println("添加的第二个元素返回值:" + set.add("b"));//
true
System.out.println("添加重复元素后的返回值:" + set.add("b"));//
false
System.out.println("当插入空值的时候返回什么?" +
set.add(""));//true
System.out.println("当插入重复的空值的时候返回什么?" +
set.add(""));//false
Iterator it = set.iterator();
System.out.print("set遍历输出:");
while (it.hasNext()) {
   System.out.print(it.next() + ",");
}
```

二、HashSet：无序、不可重复

1️⃣什么是 HashSet？操作过程是怎么样的？

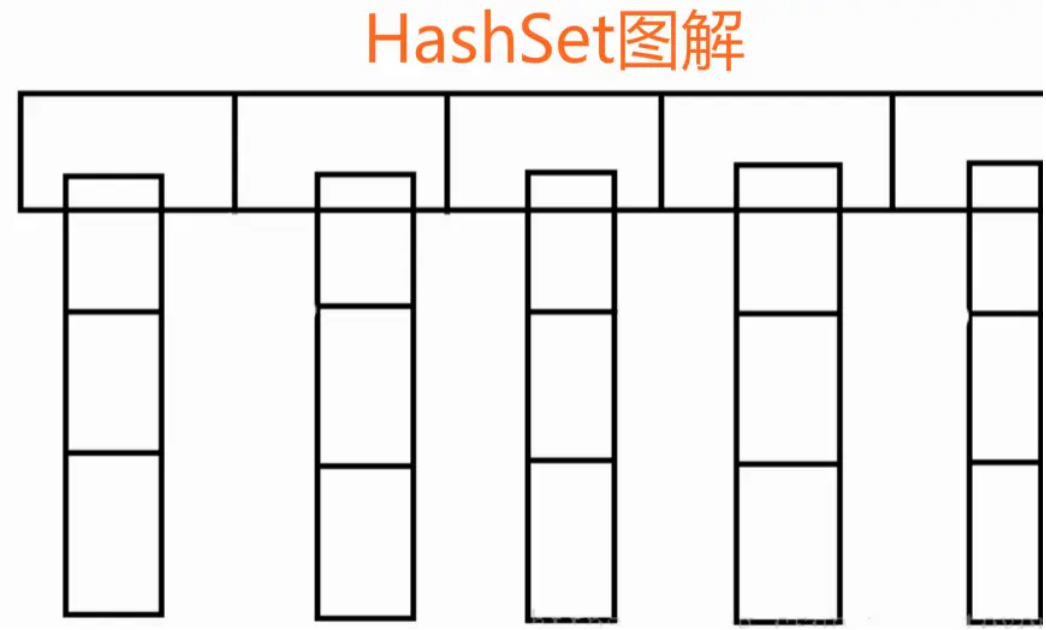
1. HashSet 底层实际上是一个 HashMap，也是采用了哈希表数据结构。
2. 哈希表又叫做散列表，哈希表底层是一个数组，这个数组中每一个元素是一个单向链表，每个单向链表都有一个独一无二的 hash 值，代表数组的下标。在某个单向链表中的每一个节点上的 hash 值是相同的。hash 值实际上是 key 调用 hashCode 方法，再通过"hash function"转换成的值。
3. 如何向哈希表中添加元素？

   先调用被存储的 key 的 hashCode 方法，经过某个算法得出 hash 值，如果在这个哈希表中不存在这个 hash 值，则直接加入元素。如果该 hash 值已经存在，继续调用 Key 之间的 equals 方法，如果 equals 方法返回 false，则将该元素添加。如果 equals 方法返回 true，则放弃添加该元素。

```
4. HashMap 和 HashSet 初始化容量是16，默认加载因子是0.75。
```

   HashSet 完全继承了 Set、Collection 里的方法实现 add、addAll、clear、isEmpty、size、contains、iteretor、remove 等。HashSet 是对 HashMap 的一层封装，类似一维数组，而一维数组里的元素又是一个单

链表，如图：



2️⃣HashMap 是如何保证元素唯一的呢?

根据元素的 hash 值以及调用 equals 方法实现的。

HashMap 在添加元素时，首先会先计算元素的 hash 值，得到的结果作为一维数组的索引。然后会去调用 equals 方法，来比较元素的值是否相同，如果相同，则表示同一个对象，不再添加进去，这样就保证了 HashMap 的唯一性。对 HashSet 的函数调用都会转换成合适的 HashMap 方法，同理。

注：这里计算处理的 hash 值只是类似于数组的索引，不一定就是数组的索引，这样只是便于理解。在 new HashMap 的时候，建议指定大小，因为如果使用默认的大小，当元素填满时，会自动扩容，这时会重新计算 hash 值，占用大量资源，影响效率。

三、TreeSet：有序，不可重复

TreeSet 是一种树形结构，是一种红黑树，这是一种近似平衡的二叉树。TreeSet 除了具有 HashSet 的唯一性之外，它还具有有序性。同样，TreeSet 是对 TreeMap 的一层封装。TreeMap 的原理如下：

TreeMap 保证元素唯一的特点与 HashMap 相似。有序是如何做到的？TreeMap 是一个红黑树，在添加元素时，有可能会破坏树的平衡，这时会自动的做相应的旋转，来保持树的平衡。就类似于天平那样，要保证这棵树平衡。保证平衡的规则就是：父节点的值要比左子树的值大，比右子树的值小。如果添加进来的一个元素破坏了这种平衡，就会自动做相应的调整，从而保证元素的有序性。对 TreeSet 的函数调用都会转换成合适的 TreeMap 方法，同理。

注：TreeSet 在添加元素时，元素必须有可比较性。由于基本类型的包装类以及 String 类已经重写了 hashCode() 和 equals()，所以可以直接添加。如果添加的是自定义实体类的话，必须要实现 Comparable 接口，或者自定义一个比较器，实现 Comparator 接口，才能添加进去。

TreeSet 对 Set 接口功能做了极大扩展，并且具有排序功能。新增了很多方法：

- first(取第一个元素)

- last(取最后一个元素)
- pollFirst(获取并移除第一个元素)
- pollLast(获取并移除最后一个元素)
- ceiling(获取该Set中大于等于指定值的最小元素)
- floor(获取该Set中小于等于指定值的最大元素)
- higher(获取该Set中严格大于指定值的最小元素)
- lower(获取该Set中严格小于指定值的最大元素)
- headSet(开头一段子集)
- tailSet(结尾一段子集)
- subSet(中间一段子集)

四、HashSet、HashMap

1️⃣问题

1. HashSet 和 HashMap 一直都是 JDK 中最常用的两个类，HashSet 要求不能存储相同的对象、无序、线程非同步，底层是哈希表结构。但它是怎么做到的？什么是散列表数据结构 (哈希表)？有什么特性？
2. HashMap 要求不能存储相同的键。
3. Java 运行时环境是如何判断 HashSet 中相同对象、HashMap 中相同键的呢？当存储了“相同的东西”之后，又将如何维护？

2️⃣分析

在研究这些问题之前，首先说明一下 JDK 对 equals(Object obj) 和 hashCode() 这两个方法的定义和规范：

1. Java 中任何一个对象都具备 equals(Object obj) 和 hashcode() 这两个方法，因为它们是在 Object 类中定义的。
2. equals(Object obj) 用来判断两个对象是否“相同”，如果“相同”则返回 true，否则返回 false。
3. hashcode() 返回一个 int 数，在 Object 类中的默认实现是“将该对象的内部地址转换成一个整数返回”。

3️⃣两个关于这两个方法的重要规范

【规范一】`若重写 equals(Object obj)，有必要重写 hashcode()，确保通过 equals(Object obj) 判断结果为 true 的两个对象具备相等的 hashcode() 返回值。`简言之：“如果两个对象相同，那么它们的 hashcode 应该相等”。不过请注意：这个只是规范，如果非要写一个类让 equals(Object obj) 返回 true 而 hashcode() 返回两个不相等的值，编译和运行都是不会报错的。不过这样违反了 Java 规范，程序也就埋下了 BUG。

【规范 2】如果 equals(Object obj) 返回 false，即两个对象“不相同”，并不要求对这两个对象调用 hashcode() 得到两个不相同的数。简言之：“如果两个对象不相同，它们的 hashcode 可能相同”。

根据这两个规范，可以得到如下推论：

1. 如果两个对象 equals，Java 运行时环境会认为它们的 hashcode 一定相等。
2. 如果两个对象不 equals，它们的 hashcode 有可能相等。
3. 如果两个对象 hashcode 相等，它们不一定 equals。
4. 如果两个对象 hashcode 不相等，它们一定不 equals。

这样就可以推断 Java 运行时环境是怎样判断 HashSet 和 HastMap 中的两个对象相同或不同了。(先判断 hashcode 是否相等，再判断是否 equals)

再看 HashSet 的源码，首先看默认构造器：

```
[java]
public HashSet() {
 map = new HashMap<E,Object>();
}
// 构造器中new了一个HashMap。key使用了泛型，value使用Object。
```

再看 add 方法源码：

```
[java]
private static final Object PRESENT = new Object();
public boolean add(E e) {
    return map.put(e, PRESENT)==null;
```

```
}
// PRESENT是一个Object类型的常量，用来当做map的value。也就是说，
//在HashSet中存储的元素都是HashMap中key，value全部使用Object。
```

HashMap 的 key 是不可以重复的，保证元素唯一的依据是对象的 hashCode 跟 equals。
而 HashSet 不就是用 HashMap 的 key 来存储元素嘛，也就保证了元素的唯一性。包括迭代器也是 HashMap 中 keySet 方法取得的 iterator。

```
[java]
public Iterator<E> iterator() {
    return map.keySet().iterator();
}
public Iterator<E> iterator() {
 return map.keySet().iterator();
}
```

由此得知，HashSet 底层是用了 HashMap。要想知道怎么做到快速存取的，直接看 HashMap 就好了。

五、ArrayList 和 HashSet 的区别

1️⃣ArrayList 是用数组实现的。HashSet 的底层是用 HashMap 实现的。调用 ArrayList 的 add 方法时在数组的下一个位置添加了一个对象。调用 HashSet 的 add 方法时，实际上是向 HashMap 中增加了一行 (key-value 对)，该行的

key 就是向 HashSet 增加的那个 对象，该行的 value 就是一个 Object 类型的常量。

2️⃣ArrayList 中保存的数据是有序的。HashSet 中保存的数据是无序的。

3️⃣ArrayList 可以保存重复的数据，而 HashSet 不能。保证 HashSet 的数据是唯一的要重写 equals() 和 hashCode()。