Java集合与框架中的集合(List, Set)及实现类(ArrayList, HashSet)

2023.1.25, 1.27

Cs. Wust

在Java集合与框架中，规范要求：

(1)如果两个对象的Hash值相同，也不一定相等，会接着比较它们的equals()方法，看对象的内容是否相同;

(2)两个对象内容相同, 则它们一定有相同的Hash值。

集合所说的序，是指对象存入集合的顺序，当对象存储顺序和取出顺序一致时就是有序，否则就是无序。并不是说存储数据的时候无序，没有规则!

1. List接口/ArrayList类和Set接口/HashSet类, LinkedHashSet类

Set集合是最简单的一种集合。集合中的对象不按特定的方式排序，并且没有重复项。Set接口主要实现了三个具体类：HashSet类按照哈希算法来存取集合中的对象，存取速度比较快；LinkedHashSet类实现了数组和双向链表，TreeSet类实现了SortedSet接口，能够对集合中的对象进行排序。

(1)Set对象是无序的，且对象不可重复。Set的实现类HashSet底层是基于HashMap实现的，HashSet的对象存放在HashMap的key上面，因为HashMap不能存储重复的Key，所以HashSet不能存放重复对象。

(2)如何计算对象的hashCode值? HashSet集合是通过内部HashCode()方法计算Hash值来决定对象的存放位置：将对象加入到HashSet集时，会首先调用HashCode()方法计算出对象的hashCode值，接着根据此hashCode值调用HashMap中的hash()方法，得到扰动后的hash值，具体是：(h=key.hashCode() ^ (h>>>16))，key为待存储的字符串对象，再将此hash 值与当前哈希表的长度-1进行位与运算，即：hash & (tab.length-1) 得到该对象在hashMap条目中的保存位置的索引，接着找到数组中该索引位置保存的对象，并调用equals方法比较这两个对象是否相等，如果相等则不添加。

注意：1)HashSet的底层数据结构是哈希表，它是由数组+链表+红黑树组成的，所以当哈希值相同的时候，也只是意味着它们处于同一个链表或红黑树中，仅仅根据哈希值还不不能分辨出它们是否是同一个对象，所以HashSet中还要执行equals方法，Object中的equals是比较它们的地址值是否相等。（注意，HashSet中的链表处于同一个存储桶，但是它们的地址值是不同的）。如果哈希值不同，那么它就不会执行equals()方法，因为哈希值不同的一定是不同的对象。

2)HashSet采用哈希算法，底层用数组存储数据。默认初始化容量16，加载因子0.75，即超过12个时就会扩容，默认扩容倍数为2倍。

3)所有要存入HashSet的集合对象中的自定义类必须覆盖hashCode()、equals()两个方法，才能保证集合中对象不重复。

Integer的哈希值是它本身，而String的哈希值是经过计算打乱的； 对于单个字符而言，这些HashCode是按照ASCII码计算和排序的。例如，String类中重写的HashCode()方法如下，可以总结出一个String类中计算HashCode值的公式s[0]\*31^(n-1) + s[1]\*31^(n-2) + ... + s[n-1]。

其中s[i]是字符串的第i个字符，首先将字符串转换成一个char类型的数组，^表示求幂;

(3)TreeSet的底层使用TreeMap实现的，set的对象存放在map的key上面, TreeMap又是用红-黑树进行实现，红黑树是一种自平衡二叉查找树，它满足对象的有序性, 因此treeset的对象具有有序性；

TreeSet会调用集合对象的compareTo(Object obj)方法来比较对象之间的大小关系,然后将集合的对象按升序排列，即自然排序，默认按升序排列;

与TreeSet集合类似的是，TreeMap也是基于红黑树对TreeMap中的所有key进行排序，从而保证TreeMap总所有的key-value处于有序状态。

对HashSet对象的测试结果如下:

HashSet, 单个数字字符，按其ASCII码放入并输出；否则按对象的哈希码到其哈希值，再将其与哈希表的长度-1进行异或运算后得到的值作为存储位置的索引，进行散列存储；

LinkedHashSet, 存储和读取顺序是相同的, 即有序;

TreeSet, 会以自然顺序排列输出。

测试结果解释了HashSet对象的输出顺序为什么是这样：





Java集合与框架中的2组常用接口分析

2023.1.22

Cs. Wust

1. Iterable接口和Iterator接口

为何要设计这2个接口? 答案是数据解耦和方便用户类的元素访问!

Iterable接口：Java集合中的顶级接口之一，实现这个接口，可以视为具有了获取迭代器的能力。

Collection、List和Set都实现了Iterable接口。

Iterator接口：为了方便遍历集合中的所有元素，用于迭代访问集合中的元素。将在不同集合的遍历方式抽象出来，这样一来遍历时就不需要知道不同集合的内部结构，交给Iterator处理就可以。不同于Collection，List， Set和Map，它们主要是定义了如何存储元素！

而迭代器则交给程序员自行实现，可谓仁者见仁智者见智。

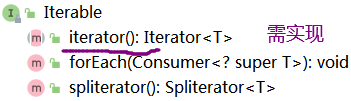
而如果不使用Iterator接口，就需要使用数组下标语法，例如：

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

T item = array[i];

}

一旦使用的数据结构变了，又要改变代码写法，不利于代码维护。

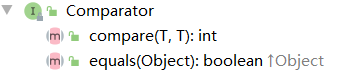
因此，Iterable接口主要是为了获取Iterator对象。

为什么Iterable接口和Iterator接口没有合二为一? 直观地，还是在于前者的抽象层次更高，同时，有些集合不止有一个Iterator内部类，可能有两个，比如ArrayList，可以获取不同的Iterator，执行不一样的操作。

对以上两个迭代接口的分析，可以平移到以下 “可比较”接口和“比较器”接口中。

2. Comparable接口和Comparator接口

效仿上述接口，出现了在类中实现Comparable接口和Comparator接口的设计！

共同点：Comparable接口和Comparator接口都是Java中用来比较和排序引用类型数据的接口，它们都可以自定义比较规则，并返回一个表示比较结果的整型值（两个对象相等，结果为0；前者小于后者，结果为-1；反之为1）。要实现比较，就要实现它们各自对应的compareTo(T)方法，或者compare(T, T)方法。其中，compareTo(T)方法是将当前对象与指定的对象进行比较以实现排序。而compare(T, T)方法可以将两个外部对象进行比较以实现排序。

Java中，8种包装器类型(Byte, Short, Integer, Character, Boolean, Long, Float, Double)都实现了Comparable接口中的抽象方法 compareTo(T)，这些类具备排序的功能。尽管没有实现Comparator接口，但它们都实现了compare(基本类型变量1，基本类型变量2)方法，等同于语句：

包装器类型.valueOf(基本类型变量1).compareTo(包装器类型.valueOf()基本类型变量2)方法

不同点：在类没有实现Comparable接口或对实现的排序不能满足需求时，将使用灵活性更好的Comparator接口，且在类的外部实现，并经常以匿名内部类的形式使用。如源码testComparableAndComparator.java所示。

由于Comparator将算法（比较规则的代码）和数据分离，因此更加灵活。它可以定义某个类的多个比较器，在排序时根据实际情况自由调用，且不用修改原先的类而是直接（匿名内部类）去实现一个新的比较器，因此Comparator的实际应用更加广泛。而Comparable接口的算法和数据出现在同一个类中，对类的修改将带来不便。

例如，Arrays.sort(T[] a, Comparator<? super T> c)实现了基于比较器排序的自定义比较规则：



Collections.sort(List<T> list, Comparator<? super T> c)实现了基于比较器排序的自定义比较规则：

