Java集合比较

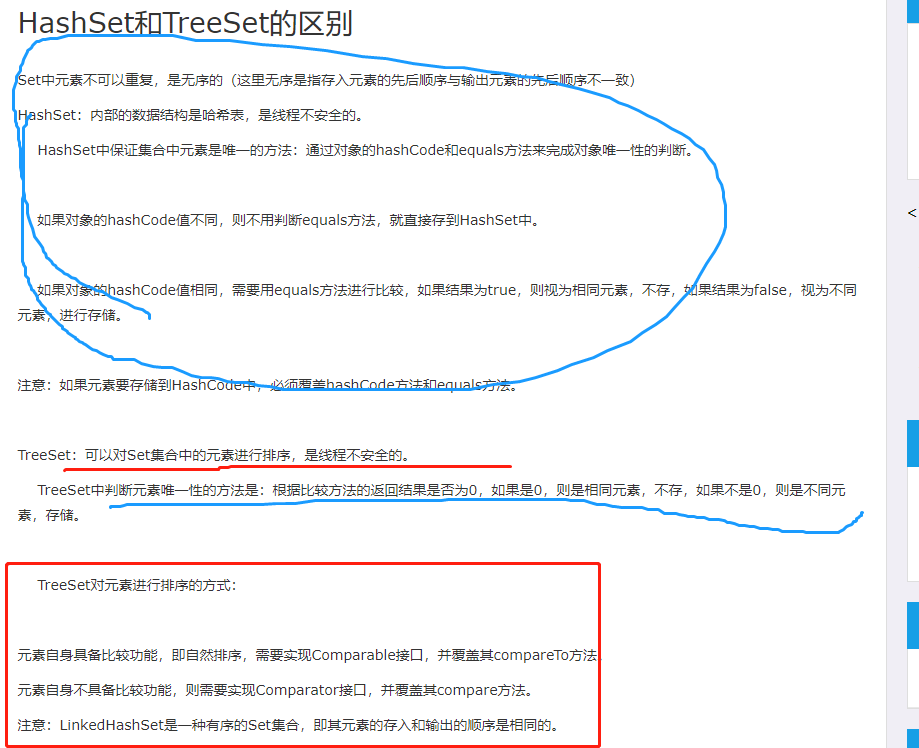


https://www.cnblogs.com/DDiamondd/p/10722278.html

public class HashMapDemo {  
 public static void main(String[] args) {  
 Map<String, String> map = new HashMap<String, String>();  
 map.put("1", "value1");  
 map.put("2", "value2");  
 map.put("3", "value3");  
 map.put("4", "value4");

// //第一种：普通使用，二次取值 keyset其实是遍历了两次，  
 // 第一次是得到iterator对象，第二次是从map中获取key对应的value  
 for (Object key : map.keySet()) {  
 System.*out*.println("key: " + key + ", value:" + map.get(key));  
 }  
 System.*out*.println("--------------------------------");  
 Set<String> strings = map.keySet();  
 Iterator<String> iterator = strings.iterator();  
 while (iterator.hasNext()) {  
 String key = iterator.next();  
 System.*out*.println("key: " + key + ", value:" + map.get(key));  
 }  
 //第二种  
 System.*out*.println("--------------------------------");  
 Set<Map.Entry<String, String>> entries = map.entrySet();  
 Iterator<Map.Entry<String, String>> iterator1 = entries.iterator();  
 while (iterator1.hasNext()) {  
 Map.Entry<String, String> next = iterator1.next();  
 String key = next.getKey();  
 String value = next.getValue();  
 System.*out*.println("key: " + key + ", value:" + value);  
 }  
 //第三种：推荐，尤其是容量大时  
 System.*out*.println("\n通过Map.entrySet遍历key和value");  
 for (Map.Entry<String, String> entry : map.entrySet()) {  
 System.*out*.println("key:" + entry.getKey() + ",value:" + entry.getValue());  
 }  
 //第四种  
 System.*out*.println("\n通过Map.values()遍历所有的value，但不能遍历key");  
 for (String v : map.values()) {  
 System.*out*.println("The value is " + v);  
 }  
 Collection<String> values = map.values();  
 Iterator<String> iterator2 = values.iterator();  
 while (iterator2.hasNext()) {  
 System.*out*.println("The value is " + iterator2.next());  
 }  
 }  
}







**HashSet底层原理：(面试过)**

HashSet实现Set接口，由哈希表（实际上是一个HashMap实例）支持。它不保证set 的迭代顺序；特别是它不保证该顺序恒久不变。此类允许使用null元素。HashSet的存储方式是把HashMap中的Key作为Set的对应存储项。该行的value就是一个Object类型的常量

（HashMap储存键值对 HashSet仅仅存储对象）

**为什么无重复？**

HashSet的存储方式是把HashMap中的Key作为Set的对应存储项。因为HashMap的key是不能有重复的

2. HashSet的实现：

对于HashSet而言，它是基于HashMap实现的，HashSet底层使用HashMap来保存所有元素，因此HashSet 的实现比较简单，相关HashSet的操作，基本上都是直接调用底层HashMap的相关方法来完成， （实际底层会初始化一个空的HashMap，并使用默认初始容量为16和加载因子0.75。）

HashSet的源代码

**对于HashSet中保存的对象，请注意正确重写其equals和hashCode方法，以保证放入的对象的唯一性。**

一定要覆盖自定义类的 equals 和 hashCode 方法，hashCode 方法是找到当前对象在 Node 数组的位置，而 equals 是比较当前对象与对应坐标链表中的对象是否相同。

插入

当有新值加入时，底层的HashMap会判断Key值是否存在（HashMap细节请移步深入理解HashMap），如果不存在，则插入新值，同时这个插入的细节会依照HashMap插入细节；如果存在就不插入。

**15，//现在要对学生进行排序,这时候就需要TreeMap了**

//第一种排序方法：让类实现Comparable接口，覆盖compareTo()方法

第二种排序方式二：让集合自身具备比较功能，定义一个类实现Comparator接口，覆盖compare方法。

16，**HashMap实现了Map接口，Map接口对键值对进行映射。Map中不允许重复的键**。Map接口有两个基本的实现，HashMap和TreeMap。TreeMap保存了对象的排列次序，而HashMap则不能。HashMap允许键和值为null。HashMap是非synchronized的，但collection框架提供方法能保证HashMap synchronized，这样多个线程同时访问HashMap时，能保证只有一个线程更改Map。public Object put(Object Key,Object value)方法用来将元素添加到map中。

17，**HashSet实现了Set接口，它不允许集合中有重复的值，当我们提到HashSet时**，第一件事情就是在将对象存储在HashSet之前，要先确保对象重写equals()和hashCode()方法，这样才能比较对象的值是否相等，以确保set中没有储存相等的对象。如果我们没有重写这两个方法，将会使用这个方法的默认实现。

**9，建立对象判断是否相同的依据。**

**|--TreeSet:可以对Set集合中的元素进行排序。是不同步的。**

判断元素唯一性的方式：就是根据比较方法的返回结果是否是0，是0，就是相同元素，不存。

TreeSet对元素进行排序的方式一：

让元素自身具备比较功能，元就需要实现Comparable接口。覆盖compareTo方法。

如果不要按照对象中具备的自然顺序进行排序。如果对象中不具备自然顺序。怎么办？

可以使用TreeSet集合第二种排序方式二：

让集合自身具备比较功能，定义一个类实现Comparator接口，覆盖compare方法。

将该类对象作为参数传递给TreeSet集合的构造函数。

**10，哈希表确定元素是否相同**

判断的是两个元素的哈希值是否相同，如果相同，在判断两个对象的内容是否相同。判断哈希值相同，其实判断的是对象的hashCode的方法。判断内容相同，用的是equals方法。

注意：如果哈希值不同，是不需要判断equals。