Java中Map集合

1. **Map接口**：存在于java.util包中，是集合体系中的**一个顶层接口**，和Collection接口并列。Map体系集合都是双列集合，而Collection体系集合都是单列集合。**Map接口**中主要有三个实现集合：**HashTable哈希表、HashMap、TreeMap**。
2. **Map 体系集合**： 一次添加**一对元素**。 Collection： 一次添加**一个元素**。

其中Map集合中存储的就是**键值对**，**Map集合中必须保证键的唯一性**。

**E 代表 Element；T 代表 Type； K 代表 Key； V代表 Value。**

1. **Map接口中有个内部接口：Map.Entry<K,V>**
2. **注意:集合也是一种对象，故集合中也可以存放集合。Map中的entrySet方法返回的Set集合中存放的就是Map.Entry集合。**
3. **在存在映射关系时，要优先考虑Map。 在查表法中，经常会利用到Map。**
4. Map的常用子类：
5. HashTable：内部结构是**哈希表**，是**同步的**，不允许null作为键和值。

其中有个子类**Properties**用来存储键值对型的配置文件的信息，可以与**IO技术**相结合。**Properties集合键单独介绍。**

1. HashMap：内部结构是**哈希表**，是不同步的（效率更高），允许null作为键和值。
2. TreeMap：内部结构是二叉树，不是同步的，可以对Map集合中的**键**进行排序。

（HashSet (的底层实际上就是一个HashMap) HashSet <== HashMap ; TreeSet的底层实际上就是一个TreeMap）TreeSet <== TreeMap）。

1. **Map<K,V>接口：**
2. **有个内部接口Map.Entry<K,V>，用于获取Map集合的映射关系视图。**



1. **方法：**
2. **添加方法：put（K，V）和putAll（Map）方法**

**put方法返回原来此键对应的值，如果原来不存在，则返回null。**



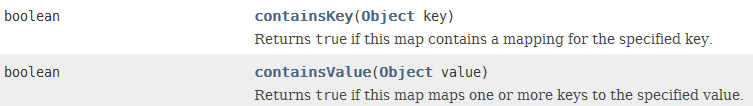


1. **删除方法：remove（K）和clear清空。**

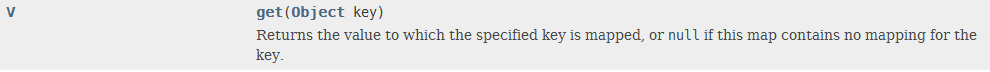




1. **是否包含判断 : containsKey和containsValue，返回Boolean值。**



1. **获取value：根据键获取值get（K）**



1. **获取大小：size（）方法：**



1. **判断是否为空：isEmpty方法**



1. **获取键视图、值视图、和映射关系视图的三个方法：**
2. **获取键视图:keyset（）方法：**

由于键值是唯一的，所以利用Set集合存储。



1. **获取值视图：values（）方法**

由于值可以重复，所以不能使用Set集合存储，而是利用List集合存储，利用Lis引用接收最好了。之所以利用Collection引用接收，是因为单列集合都可以存储。



1. **获取映射关系视图：返回一个Set集合，不过元素是Map.Entry<K,V>。**



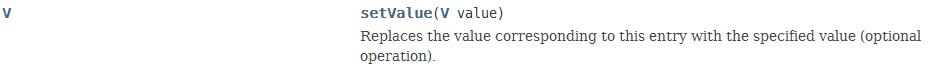
1. **三个方法的比较：keySet、values和entrySet方法：**
2. **keyset方法返回的是一个Set单列集合，集合中存放的是key值；values方法 返回一个Collection类的单列集合，集合中存放的是value值；entrySet方法返回的是一个Set单列集合，集合中存放的是Map.Entry集合，此集合利用getKey和getValue方法获取键和值。**entrySet 方法将键和值的映射关系，作为对象存储到了Set集合中，并且这个集合关系的类型就是Map.Entry 类型（相当于键和值的结婚证）。
3. **为什么values() 返回的值是在Collection 集合呢，而不是Set集合？** 因为在Map中键（key）是唯一的，不重复的，而value有可能不是唯一的，不能存到Set集合中（Set集合是保证元素唯一的）。
4. **Map.Entry<K,V> : 是一个Map接口的内部接口，存放的是Map集合的key和value的映射关系，其有几个主要方法:getKey()、getValue（）、setValue（）这个三个方法用于迭代过程。**
5. **getKey()方法：**



1. **getValue（）方法：**



1. **setValue（）方法：**



**为什么定义一个内部接口，直接定义在外部行吗？**

因为只有先有了Map，即定义了键和值，才会存在这种映射关系，而Entry正是把这种映射关系封装了起来，固然需要定义在Map内部。外部规则嵌套内部规则，内部规则就可以直接访问外部规则。

以后见到这种Xxxx.Yyy，即要想到嵌套，如内部类、内部接口。

在接口中定义一个接口，称为内部接口，如同内部类一样。

1. **如何遍历Map集合中的所有的键值对元素：**
2. **方法一：利用keySet方法**

**原理： 通过Map集合的keySet方法获取map中所有的键所在的Set集合，再通过Set的迭代器获取到每一个键，再通过map的get方法获取其对应的值。**

**例如：** Map<Integer,String> map = new HashMap<Integer,String>();

map.put(1,"I");map.put(2,"you");map.put(3,"they");

Set<Integer> keySet = map.keySet();

for(Iterator<Integer> it = keySet.iterator();it.hasNext();) {

Integer key = it.next();//获取元素

System.out.println("key: "+key+ " value: "+map.get(key));

}

1. **方法二：利用Map中values方法，直接获取value组成 的Collection集合。然后再利用Collection的迭代器从而访问出所有的值（value）。**

**例如：**Map<Integer,String> map = new HashMap<Integer,String>();

map.put(1,"I");map.put(2,"you");map.put(3,"they");

Collection<String> coll = map.values();

for(Iterator<String> it = coll.iterator();it.hasNext();) {

String value = it.next();//获取元素

System.out.println(value);

}

1. **方法三：利用Map的entrySet方法，将键和值的映射关系转换成Set集合，Set集合中存放的是Map.Entry<K,V>集合，集合中存放集合，通过Set集合的Map.Entry元素的getKey和getValue方法获取键值对。**

**例子：**Map<Integer,String> map = new HashMap<Integer,String>();

map.put(1,"I");map.put(2,"you");map.put(3,"they");

Set<Map.Entry<Integer,String>> mapEntrySet = map.entrySet();

for(Iterator<Entry<Integer, String>> it = mapEntrySet.iterator();it.hasNext();) {

Map.Entry<Integer,String> mapEntry = it.next();//获取集合的集合元素

Integer key = mapEntry.getKey();

String value = mapEntry.getValue();

System.out.println("key: "+key+ " value: "+value);

}

1. **HashMap集合：**
2. **构造方法：**
3. **默认大小和指定大小：**





1. **参数为Map 集合**



1. **参数为**



1. **特有方法：与Map差不多。**
2. **HashMap 集合：是无序的，也没有排序，利用hashCode和equals方法实现元素唯一性，但是子类LinkedHashMap可以实现有序，如同HashSet和LinkedHashSet一样。**
3. **LinkedHashSet集合：与父类HashMap差不多，仅仅是实现了有序性。**
4. **TreeMap集合：存在于java.util包中。**
5. **构造方法：**
6. **空参数：**



1. **参数为Map集合：**



1. **传进去比较器：**

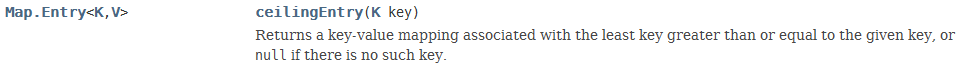


1. **参数为SortedMap集合：**

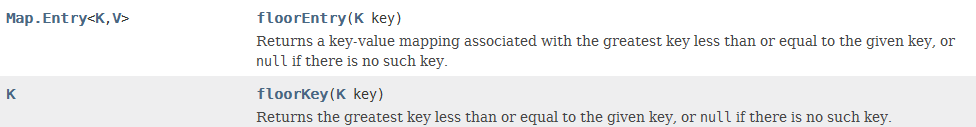


1. **特有方法：比较重要的还是Map中介绍的方法。**
2. **ceiling方法：ceilingKey、ceilingEntry**

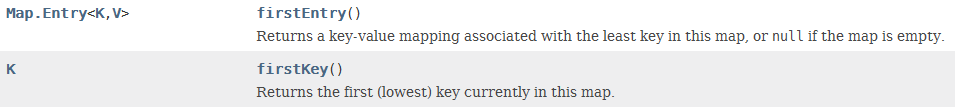




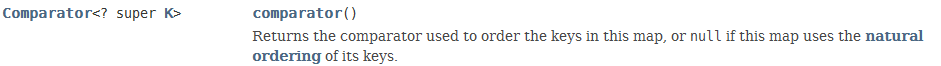
1. **floor方法：floorEntry和floorKey**



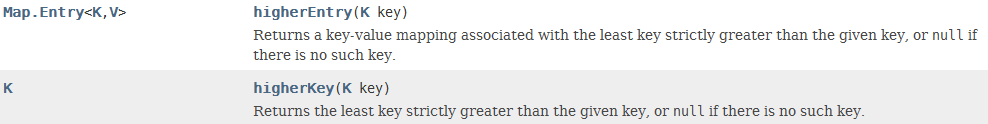
1. **first方法：firstEntry和firstKey**



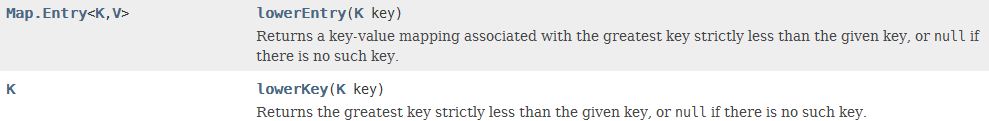
1. **comparator方法：**



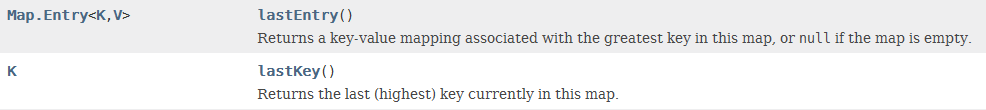
1. **higher方法：higherEntry和highKey**



1. **lower方法：lowerEntry和lowerKey**



1. **last方法：lastEntry和lastKey**



1. **TreeMap的排序方法和TreeSet完全一样，两种方法，只不过TreeSet本身就是单列集合，而TreeMap是双列集合。TreeMap排序是按照键进行排列的。**

**（1） 自然排序法。（Comparable接口中的compareTo方法）**

**（2） 比较器方法。（Comparator接口中的compare方法）**

1. **练习题：**

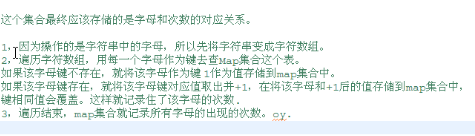


* 1. **分析：字母和次数之间存在映射关系，而且这种关系很多，所以就需要找个容器来存储一下，能存储映射关系的有数组和Map集合。**

**使用数组的条件：编号是有序的，所以需求映射一方是有序的，但是此处明显不符合。故不用数组。**

**Map集合的条件：映射一方必须能够保证唯一性，不能重复出现，可作为键。显然满足，abc不会重复，故选用Map集合。而且发现字母a、b、c 等有顺序，所以可以选用TreeMap集合。**

* 1. **步骤：**



**代码：**

**public class LianXi {**

**public static void main(String[] args) {**

**String str = "kdssdfs454sss";**

**System.out.println(str);**

**char[] cs = str.toCharArray();**

**String ok = findCharCount(cs);**

**System.out.println(ok);**

**}**

**private static String findCharCount(char[] cs) {**

**TreeMap<Character,Integer> tm = new TreeMap<Character, Integer>();**

**for (char c : cs) {**

**if(!((c <= 'z'&&c>='a')||(c<='Z'&&c>='A')))**

**continue;**

**Integer value = tm.get(c);**

**//第一种方法**

**int count = 0;**

**if(value != null)**

**count = value;**

**count++;**

**tm.put(c, count);**

//第二种方法

/\*if(value == null)

tm.put(c,1);

else

tm.put(c,value+1);\*/

**}**

**Set<Map.Entry<Character,Integer>> mape = tm.entrySet();**

**StringBuilder sb = new StringBuilder();**

**for(Iterator<Map.Entry<Character, Integer>> it = mape.iterator();it.hasNext();) {**

**Map.Entry<Character, Integer> m = it.next();**

**sb.append(m.getKey()+"("+m.getValue()+")");**

**}**

**sb.append(";");**

**return sb.toString();**

**}**

**}**

**结果为：**

1. **练习题：查表法的应用：**
2. **利用数组：局限性：键必须是数组角标；**
3. **利用Map集合：键可以任意，值也可以任意。**

**public class Weeks {**

**public static void main(String[] args) {**

**String week1 = getChineseWeek(1);**

**String week2 = getForeignWeek(1);**

**System.out.println("Chinese : "+week1+" Foreign : "+week2);**

**}**

**private static String getForeignWeek(int i) {**

**Map<Integer,String> map = new HashMap<Integer, String>();**

**map.put(1,"Monday");map.put(2,"Tuesday");**

**map.put(3,"Wednesday");map.put(4,"Thursday");**

**map.put(5,"Friday");map.put(6,"Saturday");**

**map.put(7,"Sunday");**

**return map.get(i);**

**}**

**private static String getChineseWeek(int i) {**

**String[] weeks = new String[]{"","星期一","星期二","星期三",**

**"星期四","星期五","星期六","星期天",};**

**return weeks[i];**

**}**

**}**

**结果：**