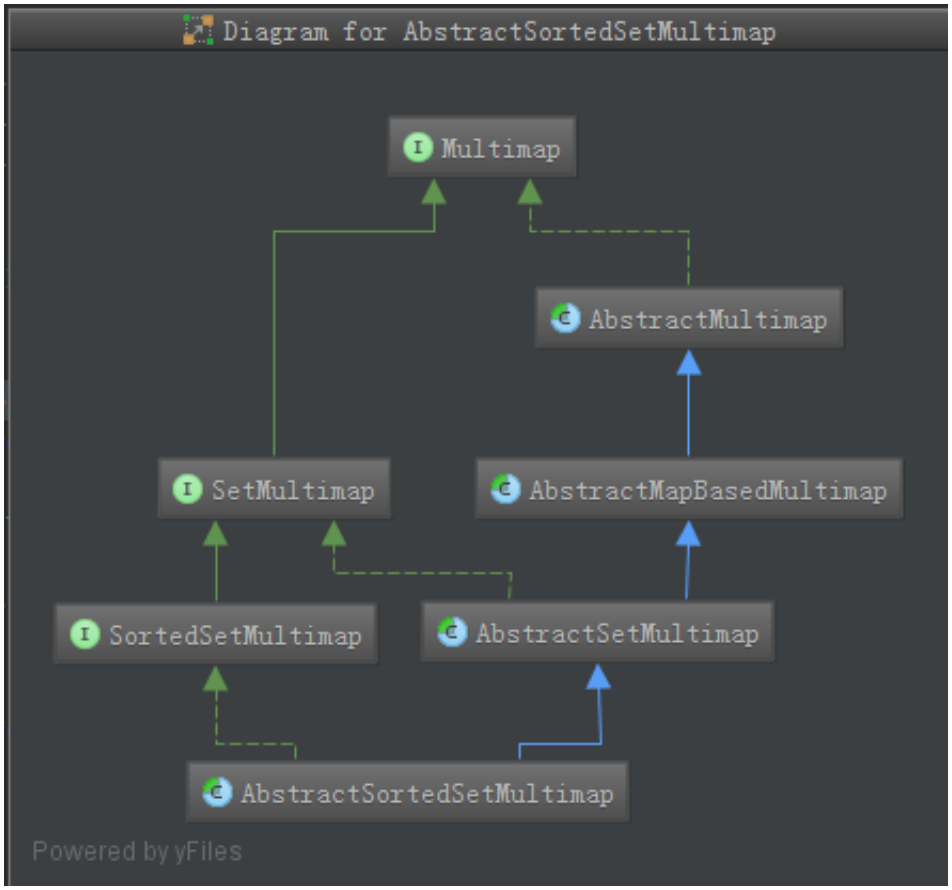
Guava学习之AbstractSortedSetMultimap



Guava学习之AbstractSortedSetMultimap

AbstractSortedSetMultimap是一个抽象类，其继承关系如上所示，关于AbstractSetMultimap和SortedSetMultimap的介绍分别在《Guava学习之AbstractSetMultimap》和《Gauva学习之SortedSetMultimap》，这里就不再介绍了。AbstractSortedSetMultimap类是SortedSetMultimap的基本实现，不过AbstractSortedSetMultimap类的很多函数的实现都是通过封装AbstractMapBasedMultimap类来实现的：将AbstractMapBasedMultimap类中相应的函数返回类型为collections的改变成SortedSet。

AbstractSortedSetMultimap类将AbstractSetMultimap类中的createCollection函数返回类型变成SortedSet，原型如下：

```
@Override
abstract SortedSet<V> createCollection();
```

而其具体的实现得等到其子类（这里指的是TreeMultimap）才去实现。
AbstractSortedSetMultimap类重写了AbstractSetMultimap类中的createUnmodifiableEmptyCollection函数，其实现如下：

```
@Override
SortedSet<V> createUnmodifiableEmptyCollection() {
        Comparator<? super V> comparator = valueComparator();
        if (comparator == null) {
            return Collections.unmodifiableSortedSet(createCollection());
        } else {
            return ImmutableSortedSet.emptySet(valueComparator());
        }
}
```

在SortedSetMultimap接口的介绍中我们就说了valueComparator()函数的含义，它是返回value的比较器。我们也说过，value比较器可能为空，这时候将会按照自然的顺序去对同一个key中的不同value进行排序。所以在createUnmodifiableEmptyCollection()函数中通过判断comparator的值是否为空而进行不同的处理。最后得到的都是一个空的不可改变的SortedSet。
在AbstractSortedSetMultimap类中比较有趣的函数是values()函数，其实现如下：

```
@Override
public Collection<V> values() {
        return super.values();
}
```

调用了AbstractSetMultimap类中的values()方法。values()函数的功能是将AbstractSortedSetMultimap子类所有key对应的value存储到Collection集合中。但是需要注意的是，这里并没有返回SortedSet而是Collection；这也就是说，通过values()函数得到的所有value的集合不是有序的，而是相同key中的value是有序的；不同key中的value顺序不能保证。如下程序：

```
TreeMultimap treeMultimap = TreeMultimap.create();
treeMultimap.put("c", "a");
treeMultimap.put("c", "1");
treeMultimap.put("a", "a");
treeMultimap.put("a", "b");
treeMultimap.put("a", "3");
treeMultimap.put("b", "a");
treeMultimap.put("b", "b");
treeMultimap.put("a", "c");
treeMultimap.put("a", "1");
```

```
treeMultimap.put("a", "2");

Set<Map.Entry> entries1 = treeMultimap.entries();
System.out.println(entries1);

Collection values = treeMultimap.values();
System.out.println(values);
```

程序运行的结果如下：

```
[a=1, a=2, a=3, a=a, a=b, a=c, b=a, b=b, c=1, c=a]
[1, 2, 3, a, b, c, a, b, 1, a]
```

可以看出values()函数返回的元素不是全部有序，而是部分有序的：按照key的先后顺序将其value输出，而同一个key中value是有序的。（完）