双向Map：BidiMap<K,V>

1. 双向Map：就是指的Key和Value都不可以重复，这样的话无论Key还是Value都是唯一的，都可以作为主键值，且Key和Value可以翻转。
2. **BidiMap<K,V>主要分为两种：**
3. **DualTreeBidiMap<K,V>：有序；**
4. **DualHashBidiMap<K,V>：无序。**
5. 接口BidiMap<K,V>:存在于接口org.apache.commons.collections4中，继承了接口IterableMap<K,V>，显然是可以迭代的。





1. **接口BidiMap**中定义的方法:
2. **inverseBidiMapMap（）：key和Value翻转，返回一个新的BidiMap对象。**



1. put添加



1. 即可以根据key获取value，也可以根据value获取key。

get方法、getKey方法。



1. 等等继承方法。
2. 既然双向Map中的Key和Value都不可以重复，那么，再添加元素时，只要是与已有的重复就相当于是修改原来的数据。

BidiMap<String, String> map1 = **new** DualHashBidiMap<String, String>();

map1.put("A","B");

map1.put("A","C"); //由A=B,修改为A=C.,并未添加新映射

System.***out***.println(map1);

BidiMap<String, String> map2 = **new** DualHashBidiMap<String, String>();

map2.put("A","B");

map2.put("C","B"); ////由A=B,修改为C=B.,并未添加新映射

System.***out***.println(map2);

输出：

1. **DualTreeBidiMap<K,V>**：**有序**（不是排序，是次序）\*；

System.***out***.println("---------有序输出----------");

BidiMap<Integer, String> bidiMap = **new** DualTreeBidiMap<Integer,String>();

bidiMap .put(1, "First");

bidiMap .put(2, "Second");

bidiMap .put(3, "Third");

BidiMap<String, Integer> inverseBidiMap = bidiMap.inverseBidiMap();

MapIterator<String ,Integer> mapIterator = inverseBidiMap .mapIterator();

**while**(mapIterator .hasNext()) {

String key = mapIterator .next();

Integer value = mapIterator .getValue();

System.***out***.println(key+"==="+value);

}

输出结果: 

1. **DualHashBidiMap<K,V>**：无序。

System.***out***.println("------无序-------");

BidiMap<Integer, String> bidiMap = **new** DualHashBidiMap<Integer,String>();

bidiMap .put(1, "First");

bidiMap .put(2, "Second");

bidiMap .put(3, "Third");

BidiMap<String, Integer> inverseBidiMap = bidiMap.inverseBidiMap();

MapIterator<String ,Integer> mapIterator = inverseBidiMap .mapIterator();

**while**(mapIterator .hasNext()) {

String key = mapIterator .next();

Integer value = mapIterator .getValue();

System.***out***.println(key+"==="+value);

}

输出结果：