一、基础概念

1. Set接口介绍

(1) Set表示的是没有重复元素、且不保证顺序的容器接口，它扩展了Collection，但没有定义任何新的方法，不过Set重写了Collection中的一些方法。

public interface Set<E> extends Collection<E> {

int size(); //set中存储值的数量

boolean isEmpty(); //是否为空的标志

boolean contains(Object o); //判断包含某个元素

Iterator<E> iterator(); //返回迭代器

Object[] toArray(); //将set变成数组返回

<T> T[] toArray(T[] a);

boolean add(E e); //添加一个元素

boolean remove(Object o); //删除一个对象

boolean containsAll(Collection<?> c); //是否包含某一个Collection

boolean addAll(Collection<? extends E> c); //添加某一个Collection

boolean retainAll(Collection<?> c);

boolean removeAll(Collection<?> c); //删除某一个Colletion

void clear(); //清空所有元素

boolean equals(Object o); //判断某个对象是否相等

int hashCode(); //获取hashCode值

}

(2) HashMap和HashSet都要求元素重写hashCode和equals方法，**且对两个对象，equals相同，则hashCode也必须相同**，如果元素是自定义的类，需要注意这一点。

2. HashSet底层实现

2.1 内部组成

(1) **HashSet内部是用HashMap实现的，它内部有一个HashMap实例变量**

private transient HashMap<E,Object> map;

(2) Map有键和值，HashSet相当于只有键，值都是相同的固定值。下面为固定值

private static final Object PRESENT = new Object();

2.2 构造方法

(1) 无参数构造方法

public HashSet() {

map = new HashMap<>(); //新建一个HashMap对象

}

(2) 传入一个固定容量

public HashSet(int initialCapacity) {

map = new HashMap<>(initialCapacity);//新建一个大小为iniu的HashMap

}

(3) 传入固定容量和加载因子

public HashSet(int initialCapacity, float loadFactor) {

//根据固定容量和加载因子新建HashMap

map = new HashMap<>(initialCapacity, loadFactor);

}

(4) 传入固定容量、加载因子、布尔值

HashSet(int initialCapacity, float loadFactor, boolean dummy) {

//利用LinkedHashMap实现

map = new LinkedHashMap<>(initialCapacity, loadFactor);

}

二、具体方法实现

1. 添加方法add

public boolean add(E e) {

return map.put(e, PRESENT) == null;

}

调用map的put方法，元素e用于键，值就是那个固定值PRESENT，**put返回null表示原来没有对应的键**，添加成功了。**HashMap中一个键只会保存一份，所以重复添加HashMap不会变化。**

2. 删除方法remove

public boolean remove(Object o) {

return map.remove(o)==PRESENT;

}

调用HashMap的remove方法，返回值为删除的那个value值表示成功删除，返回值为null表示删除失败。

3. 包含方法contain

public boolean contains(Object o) {

return map.containsKey(o);

}

调用HashMap的containKey方法，判断是否包含这个key，true表示包含，false表示不包含。

三、总结

1. 应用场景

(1) 排重，如果对排重后的元素没有顺序要求，则HashSet可以方便的用于排重。

(2) 保存特殊值，Set可以用于保存各种特殊值，程序处理用户请求或数据记录时，根据是否为特殊值，进行特殊处理，比如保存IP地址的黑名单或白名单。

(3) 集合运算，使用Set可以方便的进行数学集合中的运算，如交集、并集等运算，这些运算有一些很现实的意义。比如用户标签计算，每个用户都有一些标签，两个用户的标签交集就表示他们的共同特征，交集大小除以并集大小可以表示他们的相似长度。

2. HashSet特点 – 底层实现为HashMap

(1) 没有重复元素，添加重复元素会返回false。

原理：

HashMap中key值为唯一的，所以不会出现重复元素。

(2) 可以高效的添加、删除元素、判断元素是否存在，效率都为O(1)。

原理：

根据HashMap的添加方法或者删除方法，所以效率和HashMap差不多。

(3) 没有顺序

原理：HashMap和HashSet里面的元素都没有顺序。