📖 青空的霞光的B站课程

📅 2024年4月17日

> ps:后面Map会更加详细

Set集合

⭐ `Set`集合接口也是继承自`Collection`，实现的方法也是大差不差，但是`Set`不允许出现重复的元素、不允许随机访问、不允许通过下标访问

```
public interface Set<E> extends Collection<E> {
    // Set集合中基本都是从Collection直接继承过来的方法，只不过对这些方法有更加特殊的定义
    int size();
    boolean isEmpty();
    boolean contains(Object o);
    Iterator<E> iterator();
    Object[] toArray();
    <T> T[] toArray(T[] a);

    //添加元素只有在当前Set集合中不存在此元素时才会成功，如果插入重复元素，那么会失败
    boolean add(E e);

    //这个同样是删除指定元素
    boolean remove(Object o);

    boolean containsAll(Collection<?> c);

    //同样是只能插入那些不重复的元素
    boolean addAll(Collection<? extends E> c);

    boolean retainAll(Collection<?> c);
    boolean removeAll(Collection<?> c);
    void clear();
    boolean equals(Object o);
    int hashCode();

    //这个方法我们同样会放到多线程中进行介绍
    @Override
    default Spliterator<E> spliterator() {
        return Spliterators.spliterator(this, Spliterator.DISTINCT);
    }
}
```

⭐ 它的实现之一的`HashSet`，它的底层就是采用了哈希表来实现，底层是`HashMap`(后面会了解)

```

public class HashSet<E>
    extends AbstractSet<E>
    implements Set<E>, Cloneable, java.io.Serializable
{
    ....
    private transient HashMap<E,Object> map; //哈希漫步
}
```

1️⃣ 在Set接口中并没有定义指定下标来删除和获取对象，只能通过输入对象来删除对象

```
public static void main(String[] args) {
    Set<String> set = new HashSet<>();
    System.out.println(set.add("AAA"));   //这里我们连续插入两个同样的字符串
    System.out.println(set.add("AAA"));
    System.out.println(set);   //可以看到，最后实际上只有一个成功插入了
}

//输出:
true
false
[AAA]
```

2️⃣ 底层是哈希表来实现的，之前我们说过哈希表会计算哈希值，然后用哈希值来计算存放在哈希表的哪个位置，所以说它是无法确定存放顺序的，当然如果你输入A B C D，那么肯定存放位置会是A B C D

```
    public static void main(String[] args) {
        Set<String> set = new HashSet<>(Arrays.asList("A","2","6","D"));
        System.out.println(set);
    }

//输出
[A, 2, D, 6]
```

⭐ 如果想让维持顺序的`Set`集合可以使用`LinkedHashSet`,它的底层不在是`HashMap`，而是`LinkedHashMap`,它能够在插入数据时利用链表自动维护顺序，因此这样就能够保证我们插入顺序和最后的迭代顺序一致了

```
    public static void main(String[] args) {
        Set<String> set = new LinkedHashSet<>(Arrays.asList("A","2","6","D"));
        System.out.println(set);
    }
//输出
[A, 2, 6, D]
```

⭐ 还有一种Set叫做`TreeSet`，它会在插入时就进行排序

```
    public static void main(String[] args) {
        Set<String> set = new TreeSet<>(Arrays.asList("A","2","6","D"));
        System.out.println(set);
    }
//输出
[2, 6, A, D]
```

1️⃣ 当然他也是使用的比较器

```
public static void main(String[] args) {
    TreeSet<Integer> set = new TreeSet<>((a, b) -> b - a);  //同样是一个Comparator，反过来排序
    set.add(1);
    set.add(3);
    set.add(2);
    System.out.println(set);
}
```