1、Collection接口中的方法的使用

2、方法测试

1、Collection接口中的方法的使用

1. add(Object e):将元素e添加到集合coll中
2. size():获取添加的元素的个数
3. addAll(Collection colls):将colls集合中的元素添加到当前的集合中
4. clear():清空集合元素
5. isEmpty():判断当前集合是否为空

6.1 contains(Object obj):判断当前集合中是否包含obj，判断时会调用obj对象所在类的equals()。

6.2 containsAll(Collection coll1):判断形参coll1中的所有元素是否都存在于当前集合中。

7.1 remove(Object obj):从当前集合中移除obj元素。

7.2 removeAll(Collection colls):差集：从当前集合中移除colls中所有的元素。

8. retainAll(Collection coll1):交集：获取当前集合和coll1集合的交集，并返回给当前集合
9. equals(Object obj):要想返回true，需要当前集合和形参集合的元素都相同。
10. hashCode():返回当前对象的哈希值
11. 集合 --->数组：toArray()

拓展：数组 --->集合:调用Arrays类的静态方法asList()

12. iterator():返回Iterator接口的实例，用于遍历集合元素。

总结：向Collection接口的实现类的对象中添加数据obj时，要求obj所在类要重写equals().

2、方法测试

```
1  public class CollectionTest {
2
```

```
@Test
public void test1(){
Collection coll = new ArrayList();

//add(Object e):将元素e添加到集合coll中
coll.add("AA");
coll.add("BB");
 coll.add(123);//自动装箱
 coll.add(new Date());

 //size():获取添加的元素的个数
 System.out.println(coll.size());//4

 //addAll(Collection coll1):将coll1集合中的元素添加到当前的集合中
 Collection coll1 = new ArrayList();
 coll1.add(456);
 coll1.add("CC");
 coll.addAll(coll1);

 System.out.println(coll.size());//6
 System.out.println(coll);

 //clear():清空集合元素
 coll.clear();

 //isEmpty():判断当前集合是否为空
 System.out.println(coll.isEmpty());

 }
 }


```

```
@Test
public void test1(){
Collection coll = new ArrayList();
coll.add(123);
coll.add(456);
// Person p = new Person("Jerry",20);
// coll.add(p);
coll.add(new Person("Jerry",20));
```

```
 coll.add(new String("Tom"));
 coll.add(false);
 //1.contains(Object obj):判断当前集合中是否包含obj
 //我们在判断时会调用obj对象所在类的equals()。
 boolean contains = coll.contains(123);
 System.out.println(contains);
 System.out.println(coll.contains(new String("Tom")));
 // System.out.println(coll.contains(p));//true
 System.out.println(coll.contains(new Person("Jerry",20)));//false -->tr
ue

 //2.containsAll(Collection coll1):判断形参coll1中的所有元素是否都存在于当前
集合中。
 Collection coll1 = Arrays.asList(123,4567);
 System.out.println(coll.containsAll(coll1));
 }

 @Test
 public void test2(){
 //3.remove(Object obj):从当前集合中移除obj元素。
 Collection coll = new ArrayList();
 coll.add(123);
 coll.add(456);
 coll.add(new Person("Jerry",20));
 coll.add(new String("Tom"));
 coll.add(false);

 coll.remove(1234);
 System.out.println(coll);

 coll.remove(new Person("Jerry",20));
 System.out.println(coll);

 //4. removeAll(Collection coll1):差集：从当前集合中移除coll1中所有的元素。
 Collection coll1 = Arrays.asList(123,456);
 coll.removeAll(coll1);
 System.out.println(coll);


 }

```

```java
@Test
public void test3(){
Collection coll = new ArrayList();
coll.add(123);
coll.add(456);
coll.add(new Person("Jerry",20));
coll.add(new String("Tom"));
coll.add(false);

5.retainAll(Collection coll1):交集：获取当前集合和coll1集合的交集，并返回给当前集合
Collection coll1 = Arrays.asList(123,456,789);
coll.retainAll(coll1);
System.out.println(coll);

//6.equals(Object obj):要想返回true，需要当前集合和形参集合的元素都相同。
Collection coll1 = new ArrayList();
coll1.add(456);
coll1.add(123);
coll1.add(new Person("Jerry",20));
coll1.add(new String("Tom"));
coll1.add(false);

System.out.println(coll.equals(coll1));


}

@Test
public void test4(){
Collection coll = new ArrayList();
coll.add(123);
coll.add(456);
coll.add(new Person("Jerry",20));
coll.add(new String("Tom"));
coll.add(false);

//7.hashCode():返回当前对象的哈希值
System.out.println(coll.hashCode());

//8.集合 --->数组：toArray()
```

```java
Object[] arr = coll.toArray();
for(int i = 0;i < arr.length;i++){
System.out.println(arr[i]);
}

//拓展：数组 --->集合:调用Arrays类的静态方法asList()
List<String> list = Arrays.asList(new String[]{"AA", "BB", "CC"});
System.out.println(list);

List arr1 = Arrays.asList(new int[]{123, 456});
System.out.println(arr1.size());//1

 List arr2 = Arrays.asList(new Integer[]{123, 456});
 System.out.println(arr2.size());//2

 //9.iterator():返回Iterator接口的实例，用于遍历集合元素。
 }
```