Collection接口下方法

实现对象：

```java
Collection<String> c1 = new ArrayList<>();
```

打印效果：

［元素1，元素2.。。。］

```java
System.out.println(c1);
```

添加元素：

```java
c1.add("亚瑟");
c1.add("鲁班");
c1.add("狂铁");
System.out.println(c1);
```

元素个数：

```java
System.out.println(c1.size());
```

添加多个元素：

```java
Collection<String> c2 = new ArrayList<>();
c2.addAll(c1);
System.out.println(c2);
```

判断是否包含多个元素

```java
System.out.println(c1.containsAll(c2));
```

清空元素

```java
c2.clear();
System.out.println(c2);
```

判断是否包含某个元素

```java
System.out.println(c1.contains("亚瑟"));
System.out.println(c1.contains("貂蝉"));
```

判断是否为空

```java
System.out.println(c1.isEmpty());
System.out.println(c2.isEmpty());
```

删除某个元素

```
c1.remove("鲁班");
System.out.println(c1);
```

删除多个元素

```
c1.add("孙尚香");
c1.add("大乔");
c1.add("小乔");
System.out.println(c1);
Collection<String> c3 = new ArrayList<>();
c3.add("亚瑟");
c3.add("狂铁");
c1.removeAll(c3);
System.out.println(c1);
```

保留某些元素

```
Collection<String> c4 = new ArrayList<>();
c4.add("孙尚香");
c1.retainAll(c4);
System.out.println(c1);
```

集合转数组

```
Object[] array1 = c3.toArray();
System.out.println(Arrays.toString(array1));
String[] array2 = new String[2];
c3.toArray(array2);
System.out.println(Arrays.toString(array2));
```

使用迭代器

```
Iterator<String> iterator = c3.iterator();
```

获取下一个元素

```
String s = iterator.next();
System.out.println(s);
s =iterator.next();
System.out.println(s);
System.out.println(iterator.hasNext());
```

遍历集合

方法1

```
Iterator<String> iterator1 = c3.iterator();
for (int i = 1; i <= c3.size();i++ ){
 System.out.println(iterator1.next());
}
```

方法2

```
//快捷键 itit
Iterator<String> iterator2 = c3.iterator();
while (iterator2.hasNext()){
 System.out.println(iterator2.next());
}//快捷键 itco
for (Iterator<String> stringIterator = c3.iterator(); stringIterator.hasNext(); ) {
 String next = stringIterator.next();
}
```

方式3 快速枚举

```
//缺点：不能对集合或数组的个数有影响的操作
//快捷键 iter
for (String s1 : c3) {
 System.out.println(s1);
}
```