Java集合类存放于Java.util包中，是一个存放对象的容器。

- 只存放对象，这个基本类型将会转为包装类
- 存放的是引用
- 可以存不同类型，不限数量的数据类型

集合又分为 set、list、map；Python中 set、list、dict 类似

Set:

① HashSet：使用Hash表存储，

- 无序
- 不重复　这里的不重复指的是HashCode不能相同；add重复值不会报错，但值没有加
- 不是线程安全
- 可有一个null

定义：

```
Set set = new HashCode();
```

增删查改：

```
set.add();
set.remove();
set.clear(); //清空
```

判断：

```
set.contains();
```

遍历集合：

```
//使用迭代器遍历集合
Iterator it = set.iterator();
while(it.hasNext()){
    System.out.println(it.next());
}

for(Object obj:set){
    System.out.println(obj);
}
```

获取大小：

```
set.size();
```

使用泛型：set存指定类型数据：

```
Set<String> set1 = new HashSet<String>();
```

② Treeset 可排序的集合

有2种排序方式：自然排序、自定义排序

定义：

```
Set<Integer> set = new TreeSet<Integer>();
```

增删查改.

创建

遍历

自定义类型的排序：

```
class Person implements Comparator<Person>{
    int age;
    String name;
    @Override
    public int compare(Person o1, Person o2){
        if(o1.age > o2.age){
            return 1;
        }else if(o1.age < o2.age){
            return -1;
        }else {
            return 0;
        }
    .....
    }
```

```
package Test;

import java.util.Comparator;
import java.util.Set;
import java.util.TreeSet;

public class Test2 {
    public static void main(String[] args) {
        Person p1=new Person(1,"张三");
        Person p2=new Person(2,"里斯");
        Person p3=new Person(3,"王五");
        Set<Person> treeSet=new TreeSet<Person>(new Person());
        treeSet.add(p1);
        treeSet.add(p2);
        treeSet.add(p3);
        for(Person p:treeSet){
            System.out.println(p.age);
        }
    }
}

class Person implements Comparator<Person> {
    int age;
```

```
    String name;

    public Person() {
        this(18, "lda");
    }

    public Person(int age, String name) {
        if (age > 0) {
            this.age = age;
        } else {
            this.age = 18;
        }
        this.name = name;
    }

    @Override
    public int compare(Person o1, Person o2) {
        if(o1.age>o2.age){
            return 1;
        }else if(o1.age<o2.age){
            return -1;
        }else {
            return 0;
        }
    }
}
```

List:

有序 可重复 可用索引访问

```
package Test;

import java.util.ArrayList;
import java.util.List;

public class Test3 {
    public static void main(String[] args) {
        List<String> list=new ArrayList<String>();
        list.add("a");
        list.add("b");
        list.add("c");
        list.add("d");
        System.out.println(list);
        System.out.println(list.get(2));
        list.add(0,"zero");
        System.out.println(list);
        List<String> list1=new ArrayList<String>();
```

```
        list1.add("a");
        list1.add("b");
        list1.add("c");
        list1.add("d");
        list.addAll(list1);
        System.out.println(list);
        System.out.println(list.indexOf("d"));
        System.out.println(list.lastIndexOf("a"));
        list.remove(0);
        System.out.println(list);
        list.set(1,"zero");
        System.out.println(list);
        List<String> sub=list.subList(2,4);  // 包前不包后
        System.out.println(sub);
    }
}
```

Map：{ HashMap
TreeMap：自然排序按照 key 排序

```
package Test;

import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class Test4 {
    public static void main(String[] args) {
        Map<String, Integer> map=new HashMap<String, Integer>();
        map.put("a",1);
        map.put("b",2);
        map.put("c",3);
        System.out.println(map);
        System.out.println(map.get("c"));
        map.remove("c");
        System.out.println(map);
        System.out.println(map.containsKey("a"));
        System.out.println(map.containsValue("a"));
        map.clear();
        map.put("a",1);
        map.put("b",2);
        map.put("c",3);
        System.out.println(map);
        Set<String> keys=map.keySet();
        System.out.println(map.values());
        for(String key : keys){
            System.out.println(map.get(key));
        }
```

```
    }
}
```

操作集合工具类：Collections

用多子类型真的太好用，可能这就是引类型独有的魅力吧！