java的集合之Map

一、定义

Map的字面意思：地图。

在java中，Map集合与"地图"没有直接关系。

在java中，集合有两个顶级的接口：Collection和Map。

Collection有两种实现：List和Set

Map是一种独立的集合。

Map是一种键（key）值（value）的集合。

1、什么叫键值对？

一个键对应一个值，键和值是成对出现。如果想要获取某个值，必须要通过键来获取。

2、为什么要用键值对？

考试的时候，我们试卷上：

姓名：张三 （name = 张三）

班级：高级一班 （clazz = 高级一班）

阅卷老师在阅卷的时候，name就是键，张三是他的值。clazz是键，他的值是：高级一班。

老师阅卷就是根据这些信息来获取到试卷。

3、如何理解键值对？

一个键对应一个值。

在我们没有使用键值对的时候，以数组来作为对比:

数组其实也可以看做是一个键值对，为什么？

数组的下标，就是一个键，其内的元素，就是值。

我们从数组中获取数据的时候，arr[5],从数组中，取出下标为5的数据。

可以将数组也看做键值对：

比如，有数组：arr = {张三、李四、王五、赵六、麻子}

1：张三

2：李四

3：王五

4：赵六

5：麻子

其中，下标就是key，元素值就是value

4、Map的键值对

可以将Map视为一种特殊的”数组“。

特殊在哪里？该特殊数组的下标可以是任意的字符串。

map = {姓名：张三；性别：男；年龄：18}

从键值对中取数据：map.get(”姓名“)

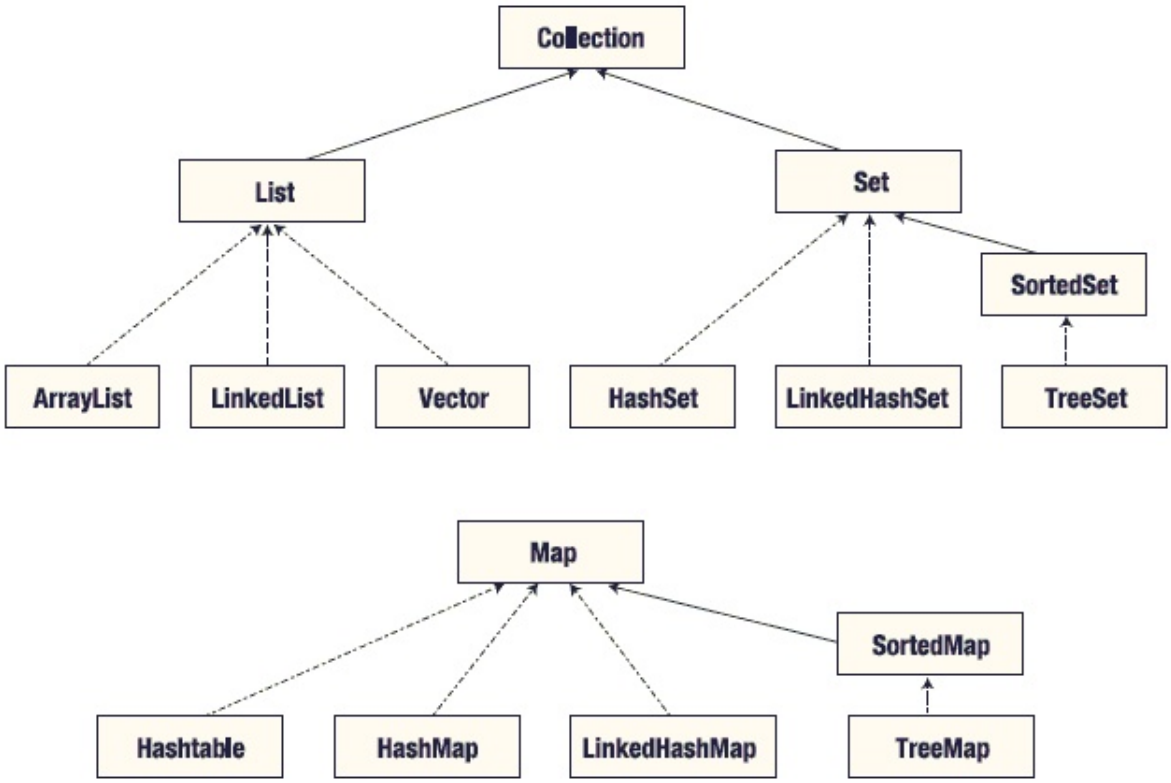
5、java中的Map具体实现

Map中的键不能重复，每一个键最多对应一个值。

java的Map集合的接口，位于java.util.Map

Map常用的实现类：HashMap（HashTable）、LinkedHashMap、TreeMap

6、java中集合的架构



二、java.util.Map接口的API

提供了数据的增删改查的相关API：

新增数据：put、putAll

修改数据（key是不能被修改的）：repalce、put（键必须是存在的）

删除数据：remove、clear

查询数据：get

元素数量：size

判断数据是否存在：

判断key：containsKey

判断value：containsValue

Map的迭代相关：

entrySet

keySet

分别返回key和value的集合：

keySet

values

三、HashMap

根据意思，采用hash算法来存储的键值对集合。

HashMap的默认初始容量为：16

也可以在创建Map的时候，指定初始容量

```
//实例
package com.psfd.util.map;

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

public class MapTest {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<>();
        // put方法，添加单个元素
        map.put("姓名", "张三");
        map.put("性别", "男");
        map.put("年龄", "18");
        map.put("家庭住址", "坪山");

        Map<String, String> map1 = new HashMap<>();
        map1.put("手机号码", "1626252");
        map1.put("民族", "汉");
        map1.put("家庭住址", "南山");
```

```
        System.out.println(map);
        System.out.println(map1);
        System.out.println("==================");

        //putAll 将另一个Map里的所有元素，添加到当前Map
        //如果有相同的key，则会覆盖原来的值
        map.putAll(map1);
        System.out.println(map);
        System.out.println(map1);

        System.out.println("==================");
        // replace 替换指定键的值。如果键不存在，不予理会。
//      map.replace("姓名", "麻子");
        // put 如果键存在，则替换。如果键不存在，则新增。
        map.put("姓名1", "小强");
        System.out.println(map);

        System.out.println("==================");
        //remove 通过key删除元素，如果key不存在，则不予理会。
        map.remove("姓名1");
        System.out.println(map);

        System.out.println("==================");
        //get 通过key获取value。如果键不存在，返回null
        String string = map.get("手机号码");
        System.out.println(string);

        System.out.println(map.size());
        System.out.println("==================");

        //判断map中是否包含某个键或者值
        System.out.println(map.containsKey("姓名"));
        System.out.println(map.containsValue("123"));
    }
}
```

1、Map的迭代

Map中并没有给直接提供迭代器，也不能使用foreach（增强for循环）来迭代。

Map的迭代有四种方式。

Map的迭代必须要使用Map提供的entrySet()和keySet()

所谓的四种方式：

foreach的entrySet

foreach的keySet

迭代器的entrySet

迭代器的keySet

```java
//实例
package com.psfd.util.map;

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

public class MapTest {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<>();
        map.put("姓名", "张三");
        map.put("性别", "男");
        map.put("年龄", "18");
        map.put("家庭住址", "坪山");
        map.put("手机号码", "1626252");
        map.put("民族", "汉");

        //Map的迭代
        //keyset()方法，得到Map中所有key的set集合。
        Set<String> keySet = map.keySet();
        for (String key : keySet) {
            System.out.println(String.format("%s:%s", key,map.get(key)));
        }

        System.out.println("=================");
        Iterator<String> iterator = keySet.iterator();
        while (iterator.hasNext()) {
            String key = iterator.next();
            String value = map.get(key);
            System.out.println(String.format("%s:%s", key,value));
        }

        System.out.println("=================");

        //entrySet方法。
        //Entry是Map中的一个内部类。
        Set<Entry<String, String>> entrySet = map.entrySet();
        for (Entry<String, String> entry : entrySet) {
            System.out.println(String.format("%s:%s", entry.getKey(),entry.getValue()));
        }

        System.out.println("=================");
        //迭代
        Iterator<Entry<String, String>> iterator2 = entrySet.iterator();
        while (iterator2.hasNext()) {
            Entry<String, String> entry = iterator2.next();
            System.out.println(String.format("%s:%s", entry.getKey(),entry.getValue()));
        }
    }
}
```

2、测试各种迭代方式的性能

```
package com.psfd.util.map;

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

public class MapTest {
    public static void main(String[] args) {
        Map<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < 8000000; i++) {
            map.put(i, i);
        }

        //Map的迭代
        //keyset()方法，得到Map中所有key的set集合。
        Set<Integer> keySet = map.keySet();

        long startTm = System.currentTimeMillis();
        for (Integer key : keySet) {
            map.get(key);
        }
        long endTm = System.currentTimeMillis();
        System.out.println("keySet的foreach耗时 " + (endTm - startTm));

        startTm = System.currentTimeMillis();
        Iterator<Integer> iterator = keySet.iterator();
        while (iterator.hasNext()) {
            map.get(iterator.next());
        }
        endTm = System.currentTimeMillis();
        System.out.println("keySet的迭代器耗时 " + (endTm - startTm));


        //entrySet方法。
        //Entry是Map中的一个内部类。
        Set<Entry<Integer, Integer>> entrySet = map.entrySet();
        startTm = System.currentTimeMillis();
        for (Entry<Integer, Integer> entry : entrySet) {
            entry.getKey();
            entry.getValue();
        }
        endTm = System.currentTimeMillis();
        System.out.println("entrySet的foreach耗时 " + (endTm - startTm));

        //迭代
        startTm = System.currentTimeMillis();
```

```
        Iterator<Entry<Integer, Integer>> iterator2 = entrySet.iterator();
        while (iterator2.hasNext()) {
            Entry<Integer, Integer> entry = iterator2.next();
            entry.getKey();
            entry.getValue();
        }
        endTm = System.currentTimeMillis();
        System.out.println("entrySet的迭代器耗时 " + (endTm - startTm));
    }
}

--执行四次，取平均值

keySet的foreach耗时 91
keySet的迭代器耗时 126
entrySet的foreach耗时 73
entrySet的迭代器耗时 74


keySet的foreach耗时 97
keySet的迭代器耗时 131
entrySet的foreach耗时 77
entrySet的迭代器耗时 77


keySet的foreach耗时 97
keySet的迭代器耗时 128
entrySet的foreach耗时 77
entrySet的迭代器耗时 78


keySet的foreach耗时 101
keySet的迭代器耗时 133
entrySet的foreach耗时 82
entrySet的迭代器耗时 82


 //取平均值
keySet-foreach   96
keySet-迭代器    130
entrySet-foreach 77
entrySet-迭代器  77
```

从测试结果上可以看出，entrySet的性能比keyset的性能高1.5倍。

因此，Map的迭代，使用entrySet，不要使用keySet

3、HashSet和HashMap的关系

看HashSet的源码：

```
//HashSet的add方法
    private transient HashMap<E,Object> map;

    private static final Object PRESENT = new Object();

    public boolean add(E e) {
        return map.put(e, PRESENT)==null;
    }
```

从上面的源码可以看到，HashSet其内部就是使用了HashMap，当向hashset中保存元素的的时候，会将元素放到HashMap的Key上。value是没有实际意义的一个new Object()；

```
//HashSet的迭代
 public Iterator<E> iterator() {
        return map.keySet().iterator();
    }
```

HashSet的迭代，其底层是：map.keySet()获取到所有key的Set集合。

所以，hashSet和HashMap没有实质的区别。

他们的区别仅仅在于Set是一个单个元素的集合，而HahsMap保存的是键值对。

四、LinkedHashMap

LinkedHahshMap是HashMap的子类，LinkedHahshMap与HashMap的区别仅仅在与：

LinkedHahshMap会维护元素的顺序，而hashMap不会维护。

如果查询多，用ArrayList、HashSet、HashMap。

如果增删多，用LinkedList、LinkedHashSet、LinkedHashMap。

五、HashTable

HashTable与HashMap一样，都是Map接口的实现类。

他们都是基于哈希表的键值对的集合。

HashTable与HashMap的区别：

1、HashMap的键和值，允许一个null键，值可以多个null。

HashTable的键和值，都不允许为null；

```
Hashtable<String, String> hashtable = new Hashtable<>();
        hashtable.put("name", "张三");
        hashtable.put("age", "18");
        hashtable.put("sex", "nan");
        hashtable.put("phone", "1812313");
        hashtable.put(null, null);
        System.out.println(hashtable);

Exception in thread "main" java.lang.NullPointerException
    at java.util.Hashtable.put(Unknown Source)
    at com.psfd.util.map.MapTest.main(MapTest.java:39)
```

2、HashMap是一个线程不安全的集合。

HashTable是一个线程安全的集合。（是一个“同步”集合）

HashMap的性能比HashTable的性能要好。

从jdk1.5开始，出现了java.util.concurrent包，这个包提供的java多线程编程的相关支持。

其内有一个Map集合：ConcurrentHashMap,这个Map集合是一个线程安全的集合，但是它比hashTable更佳灵活和更具有扩展性。

所有，HashTable在某种程度上，被ConcurrentHashMap取代了。

六、TreeMap

采用红黑树（自平衡的树）来保存数据。

TreeMap与TreeSet的基本原理一样的。

TreeSet中保存数据，也是放在一个TreeMap的key中。

因此，TreeMap的排序，与TreeSet的排序，一个套路。

TreeSet的排序分为：自然排序和定制排序。

自然排序：排序的对象要实现java.lang.Comparable接口，重写compareTo方法

定制排序：自己定义一个比较器，java.util.Compartor.（1、自己写一个类，2、采用匿名内部类的方式）

```
//采用匿名内部类的比较器进行比较
Map<String, String> map = new TreeMap<String, String>(new Comparator<String>() {
    public int compare(String obj1, String obj2) {
        return obj2.compareTo(obj1);// 降序排序
    }
});
map.put("a", "c");
map.put("b", "b");
map.put("c", "a");

for (String key : map.keySet()) {
    System.out.println(key + "  :" + map.get(key));
}
```

七、java的集合对null值的处理

1、List集合允许元素的值为null

2、Set集合，除了TreeSet不能允许null值，其他允许。

3、Map中，HashMap允许null值和null键。

八、集合的总结