

# 1.Map集合

## 1.1Map集合概述和特点【理解】

- Map集合概述

```
1 interface Map<K,V>  K : 键的类型 ; V : 值的类型
```

- Map集合的特点
  - 键值对映射关系
  - 一个键对应一个值
  - 键不能重复，值可以重复
  - 元素存取无序
- Map集合的基本使用

```
1 public class MapDemo01 {
2     public static void main(String[] args) {
3         //创建集合对象
4         Map<String,String> map = new HashMap<String,String>();
5
6         //V put(K key, V value) 将指定的值与该映射中的指定键相关联
7         map.put("itheima001","林青霞");
8         map.put("itheima002","张曼玉");
9         map.put("itheima003","王祖贤");
10        map.put("itheima003","柳岩");
11
12        //输出集合对象
13        System.out.println(map);
14    }
15 }
```

## 1.2Map集合的基本功能【应用】

- 方法介绍

方法名	说明
V put(K key,V value)	添加元素
V remove(Object key)	根据键删除键值对元素
void clear()	移除所有的键值对元素
boolean containsKey(Object key)	判断集合是否包含指定的键
boolean containsValue(Object value)	判断集合是否包含指定的值
boolean isEmpty()	判断集合是否为空
int size()	集合的长度，也就是集合中键值对的个数

- 示例代码

```

1  public class MapDemo02 {
2      public static void main(String[] args) {
3          //创建集合对象
4          Map<String,String> map = new HashMap<String,String>();
5
6          //V put(K key,V value): 添加元素
7          map.put("张无忌","赵敏");
8          map.put("郭靖","黄蓉");
9          map.put("杨过","小龙女");
10
11         //V remove(Object key): 根据键删除键值对元素
12         //      System.out.println(map.remove("郭靖"));
13         //      System.out.println(map.remove("郭襄"));
14
15         //void clear(): 移除所有的键值对元素
16         //      map.clear();
17
18         //boolean containsKey(Object key): 判断集合是否包含指定的键
19         //      System.out.println(map.containsKey("郭靖"));
20         //      System.out.println(map.containsKey("郭襄"));
21
22         //boolean isEmpty(): 判断集合是否为空
23         //      System.out.println(map.isEmpty());
24
25         //int size(): 集合的长度，也就是集合中键值对的个数
26         System.out.println(map.size());
27
28
29         //输出集合对象
30         System.out.println(map);
31     }
32 }

```

## 1.3Map集合的获取功能【应用】

- 方法介绍

方法名	说明
V get(Object key)	根据键获取值
Set keySet()	获取所有键的集合
Collection values()	获取所有值的集合
Set<Map.Entry<K,V>> entrySet()	获取所有键值对对象的集合

- 示例代码

```
1 public class MapDemo03 {
2     public static void main(String[] args) {
3         //创建集合对象
4         Map<String, String> map = new HashMap<String, String>();
5
6         //添加元素
7         map.put("张无忌", "赵敏");
8         map.put("郭靖", "黄蓉");
9         map.put("杨过", "小龙女");
10
11         //V get(Object key):根据键获取值
12         //     System.out.println(map.get("张无忌"));
13         //     System.out.println(map.get("张三丰"));
14
15         //Set<K> keySet():获取所有键的集合
16         //     Set<String> keySet = map.keySet();
17         //     for(String key : keySet) {
18         //         System.out.println(key);
19         //     }
20
21         //Collection<V> values():获取所有值的集合
22         Collection<String> values = map.values();
23         for(String value : values) {
24             System.out.println(value);
25         }
26     }
27 }
```

## 1.4Map集合的遍历(方式1)【应用】

- 遍历思路

- 我们刚才存储的元素都是成对出现的，所以我们把Map看成是一个夫妻对的集合
  - 把所有的丈夫给集中起来
  - 遍历丈夫的集合，获取到每一个丈夫
  - 根据丈夫去找对应的妻子

- 步骤分析

- 获取所有键的集合。用keySet()方法实现

- 遍历键的集合，获取到每一个键。用增强for实现
- 根据键去找值。用get(Object key)方法实现
- 代码实现

```
1 public class MapDemo01 {
2     public static void main(String[] args) {
3         //创建集合对象
4         Map<String, String> map = new HashMap<String, String>();
5
6         //添加元素
7         map.put("张无忌", "赵敏");
8         map.put("郭靖", "黄蓉");
9         map.put("杨过", "小龙女");
10
11        //获取所有键的集合。用keySet()方法实现
12        Set<String> keySet = map.keySet();
13        //遍历键的集合，获取到每一个键。用增强for实现
14        for (String key : keySet) {
15            //根据键去找值。用get(Object key)方法实现
16            String value = map.get(key);
17            System.out.println(key + "," + value);
18        }
19    }
20 }
```

## 1.5 Map集合的遍历(方式2)【应用】

- 遍历思路
  - 我们刚才存储的元素都是成对出现的，所以我们把Map看成是一个夫妻对的集合
    - 获取所有结婚证的集合
    - 遍历结婚证的集合，得到每一个结婚证
    - 根据结婚证获取丈夫和妻子
- 步骤分析
  - 获取所有键值对对象的集合
    - Set<Map.Entry<K,V>> entrySet()：获取所有键值对对象的集合
  - 遍历键值对对象的集合，得到每一个键值对对象
    - 用增强for实现，得到每一个Map.Entry
  - 根据键值对对象获取键和值
    - 用getKey()得到键
    - 用getValue()得到值
- 代码实现

```
1 public class MapDemo02 {
2     public static void main(String[] args) {
3         //创建集合对象
4         Map<String, String> map = new HashMap<String, String>();
5
6         //添加元素
```

```

7      map.put("张无忌", "赵敏");
8      map.put("郭靖", "黄蓉");
9      map.put("杨过", "小龙女");
10
11     //获取所有键值对对象的集合
12     Set<Map.Entry<String, String>> entrySet = map.entrySet();
13     //遍历键值对对象的集合，得到每一个键值对对象
14     for (Map.Entry<String, String> me : entrySet) {
15         //根据键值对对象获取键和值
16         String key = me.getKey();
17         String value = me.getValue();
18         System.out.println(key + "," + value);
19     }
20 }
21 }

```

## 1.6Map集合的案例【应用】

### 1.6.1HashMap集合练习之键是String值是Student

- 案例需求

创建一个HashMap集合，键是学号(String)，值是学生对象(Student)。存储三个键值对元素，并遍历

- 代码实现

- 学生类

```

1  public class Student {
2      private String name;
3      private int age;
4
5      public Student() {
6      }
7
8      public Student(String name, int age) {
9          this.name = name;
10         this.age = age;
11     }
12
13     public String getName() {
14         return name;
15     }
16
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public int getAge() {
22         return age;
23     }
24
25     public void setAge(int age) {
26         this.age = age;

```

```
27     }
28 }
```

#### o 测试类

```
1  /*
2      需求：
3          创建一个HashMap集合，键是学号(String)，值是学生对象(Student)。存储三个键值对
4          元素，并遍历
5
6      思路：
7          1:定义学生类
8          2:创建HashMap集合对象
9          3:创建学生对象
10         4:把学生添加到集合
11         5:遍历集合
12             方式1：键找值
13             方式2：键值对对象找键和值
14     */
15     public class HashMapDemo {
16         public static void main(String[] args) {
17             //创建HashMap集合对象
18             HashMap<String, Student> hm = new HashMap<String, Student>();
19
20             //创建学生对象
21             Student s1 = new Student("林青霞", 30);
22             Student s2 = new Student("张曼玉", 35);
23             Student s3 = new Student("王祖贤", 33);
24
25             //把学生添加到集合
26             hm.put("itheima001", s1);
27             hm.put("itheima002", s2);
28             hm.put("itheima003", s3);
29
30             //方式1：键找值
31             Set<String> keySet = hm.keySet();
32             for (String key : keySet) {
33                 Student value = hm.get(key);
34                 System.out.println(key + "," + value.getName() + "," +
35                                     value.getAge());
36             }
37             System.out.println("-----");
38
39             //方式2：键值对对象找键和值
40             Set<Map.Entry<String, Student>> entrySet = hm.entrySet();
41             for (Map.Entry<String, Student> me : entrySet) {
42                 String key = me.getKey();
43                 Student value = me.getValue();
44                 System.out.println(key + "," + value.getName() + "," +
45                                     value.getAge());
46             }
47         }
48     }
49 }
```

## 1.6.2 HashMap集合练习之键是Student值是String

- 案例需求
  - 创建一个HashMap集合，键是学生对象(Student)，值是居住地(String)。存储多个元素，并遍历。
  - 要求保证键的唯一性：如果学生对象的成员变量值相同，我们就认为是同一个对象
- 代码实现
  - 学生类

```
1  public class Student {
2      private String name;
3      private int age;
4
5      public Student() {
6      }
7
8      public Student(String name, int age) {
9          this.name = name;
10         this.age = age;
11     }
12
13     public String getName() {
14         return name;
15     }
16
17     public void setName(String name) {
18         this.name = name;
19     }
20
21     public int getAge() {
22         return age;
23     }
24
25     public void setAge(int age) {
26         this.age = age;
27     }
28
29     @Override
30     public boolean equals(Object o) {
31         if (this == o) return true;
32         if (o == null || getClass() != o.getClass()) return false;
33
34         Student student = (Student) o;
35
36         if (age != student.age) return false;
37         return name != null ? name.equals(student.name) : student.name ==
null;
38     }
39
40     @Override
41     public int hashCode() {
42         int result = name != null ? name.hashCode() : 0;
```

```

43         result = 31 * result + age;
44         return result;
45     }
46 }

```

#### ◦ 测试类

```

1  public class HashMapDemo {
2      public static void main(String[] args) {
3          //创建HashMap集合对象
4          HashMap<Student, String> hm = new HashMap<Student, String>();
5
6          //创建学生对象
7          Student s1 = new Student("林青霞", 30);
8          Student s2 = new Student("张曼玉", 35);
9          Student s3 = new Student("王祖贤", 33);
10         Student s4 = new Student("王祖贤", 33);
11
12         //把学生添加到集合
13         hm.put(s1, "西安");
14         hm.put(s2, "武汉");
15         hm.put(s3, "郑州");
16         hm.put(s4, "北京");
17
18         //遍历集合
19         Set<Student> keySet = hm.keySet();
20         for (Student key : keySet) {
21             String value = hm.get(key);
22             System.out.println(key.getName() + "," + key.getAge() + "," +
value);
23         }
24     }
25 }

```

### 1.6.3 集合嵌套之ArrayList嵌套HashMap

- 案例需求
  - 创建一个ArrayList集合，存储三个元素，每一个元素都是HashMap
  - 每一个HashMap的键和值都是String，并遍历。
- 代码实现

```

1  public class ArrayListIncludeHashMapDemo {
2      public static void main(String[] args) {
3          //创建ArrayList集合
4          ArrayList<HashMap<String, String>> array = new
ArrayList<HashMap<String, String>>();
5
6          //创建HashMap集合，并添加键值对元素
7          HashMap<String, String> hm1 = new HashMap<String, String>();
8          hm1.put("孙策", "大乔");
9          hm1.put("周瑜", "小乔");

```



```

10 //把HashMap作为元素添加到ArrayList集合
11 array.add(hm1);
12
13 HashMap<String, String> hm2 = new HashMap<String, String>();
14 hm2.put("郭靖", "黄蓉");
15 hm2.put("杨过", "小龙女");
16 //把HashMap作为元素添加到ArrayList集合
17 array.add(hm2);
18
19 HashMap<String, String> hm3 = new HashMap<String, String>();
20 hm3.put("令狐冲", "任盈盈");
21 hm3.put("林平之", "岳灵珊");
22 //把HashMap作为元素添加到ArrayList集合
23 array.add(hm3);
24
25 //遍历ArrayList集合
26 for (HashMap<String, String> hm : array) {
27     Set<String> keySet = hm.keySet();
28     for (String key : keySet) {
29         String value = hm.get(key);
30         System.out.println(key + "," + value);
31     }
32 }
33 }
34 }

```

#### 1.6.4集合嵌套之HashMap嵌套ArrayList

- 案例需求
  - 创建一个HashMap集合，存储三个键值对元素，每一个键值对元素的键是String，值是ArrayList
  - 每一个ArrayList的元素是String，并遍历。
- 代码实现

```

1 public class HashMapIncludeArrayListDemo {
2     public static void main(String[] args) {
3         //创建HashMap集合
4         HashMap<String, ArrayList<String>> hm = new HashMap<String,
5 ArrayList<String>>();
6
7         //创建ArrayList集合，并添加元素
8         ArrayList<String> sggy = new ArrayList<String>();
9         sggy.add("诸葛亮");
10        sggy.add("赵云");
11        //把ArrayList作为元素添加到HashMap集合
12        hm.put("三国演义", sggy);
13
14        ArrayList<String> xyj = new ArrayList<String>();
15        xyj.add("唐僧");
16        xyj.add("孙悟空");
17        //把ArrayList作为元素添加到HashMap集合
18        hm.put("西游记", xyj);

```

```

19     ArrayList<String> shz = new ArrayList<String>();
20     shz.add("武松");
21     shz.add("鲁智深");
22     //把ArrayList作为元素添加到HashMap集合
23     hm.put("水浒传", shz);
24
25     //遍历HashMap集合
26     Set<String> keySet = hm.keySet();
27     for(String key : keySet) {
28         System.out.println(key);
29         ArrayList<String> value = hm.get(key);
30         for(String s : value) {
31             System.out.println("\t" + s);
32         }
33     }
34 }
35 }

```

## 1.6.5统计字符串中每个字符出现的次数

- 案例需求
  - 键盘录入一个字符串，要求统计字符串中每个字符串出现的次数。
  - 举例：键盘录入“aababcbcdabcde”在控制台输出：“a(5)b(4)c(3)d(2)e(1)”
- 代码实现

```

1 public class HashMapDemo {
2     public static void main(String[] args) {
3         //键盘录入一个字符串
4         Scanner sc = new Scanner(System.in);
5         System.out.println("请输入一个字符串：");
6         String line = sc.nextLine();
7
8         //创建HashMap集合，键是Character，值是Integer
9         // HashMap<Character, Integer> hm = new HashMap<Character, Integer>();
10        TreeMap<Character, Integer> hm = new TreeMap<Character, Integer>();
11
12        //遍历字符串，得到每一个字符
13        for (int i = 0; i < line.length(); i++) {
14            char key = line.charAt(i);
15
16            //拿得到的每一个字符作为键到HashMap集合中去找对应的值，看其返回值
17            Integer value = hm.get(key);
18
19            if (value == null) {
20                //如果返回值是null：说明该字符在HashMap集合中不存在，就把该字符作为键，1
                //作为值存储
21                hm.put(key, 1);
22            } else {
23                //如果返回值不是null：说明该字符在HashMap集合中存在，把该值加1，然后重新
                //存储该字符和对应的值
24                value++;
25                hm.put(key, value);

```

```

26         }
27     }
28
29     //遍历HashMap集合，得到键和值，按照要求进行拼接
30     StringBuilder sb = new StringBuilder();
31
32     Set<Character> keySet = hm.keySet();
33     for(Character key : keySet) {
34         Integer value = hm.get(key);
35         sb.append(key).append("(").append(value).append(")");
36     }
37
38     String result = sb.toString();
39
40     //输出结果
41     System.out.println(result);
42 }
43 }

```

## 2.Collections集合工具类

### 2.1Collections概述和使用【应用】

- Collections类的作用  
是针对集合操作的工具类
- Collections类常用方法

方法名	说明
public static void sort(List list)	将指定的列表按升序排序
public static void reverse(List<?> list)	反转指定列表中元素的顺序
public static void shuffle(List<?> list)	使用默认的随机源随机排列指定的列表

- 示例代码

```

1 public class CollectionsDemo01 {
2     public static void main(String[] args) {
3         //创建集合对象
4         List<Integer> list = new ArrayList<Integer>();
5
6         //添加元素
7         list.add(30);
8         list.add(20);
9         list.add(50);
10        list.add(10);
11        list.add(40);
12
13        //public static <T extends Comparable<? super T>> void sort(List<T>
list): 将指定的列表按升序排序

```

```

14 //      Collections.sort(list);
15
16 //public static void reverse(List<?> list) : 反转指定列表中元素的顺序
17 //      Collections.reverse(list);
18
19 //public static void shuffle(List<?> list) : 使用默认的随机源随机排列指定的列表
20 Collections.shuffle(list);
21
22 System.out.println(list);
23 }
24 }

```

## 2.2 ArrayList集合存储学生并排序【应用】

- 案例需求
  - ArrayList存储学生对象，使用Collections对ArrayList进行排序
  - 要求：按照年龄从小到大排序，年龄相同时，按照姓名的字母顺序排序
- 代码实现
  - 学生类

```

1 public class Student {
2     private String name;
3     private int age;
4
5     public Student() {
6     }
7
8     public Student(String name, int age) {
9         this.name = name;
10        this.age = age;
11    }
12
13    public String getName() {
14        return name;
15    }
16
17    public void setName(String name) {
18        this.name = name;
19    }
20
21    public int getAge() {
22        return age;
23    }
24
25    public void setAge(int age) {
26        this.age = age;
27    }
28 }

```

- 测试类

```

1 public class CollectionsDemo02 {
2     public static void main(String[] args) {
3         //创建ArrayList集合对象
4         ArrayList<Student> array = new ArrayList<Student>();
5
6         //创建学生对象
7         Student s1 = new Student("linqingxia", 30);
8         Student s2 = new Student("zhangmanyu", 35);
9         Student s3 = new Student("wangzuxian", 33);
10        Student s4 = new Student("liuyan", 33);
11
12        //把学生添加到集合
13        array.add(s1);
14        array.add(s2);
15        array.add(s3);
16        array.add(s4);
17
18        //使用Collections对ArrayList集合排序
19        //sort(List<T> list, Comparator<? super T> c)
20        Collections.sort(array, new Comparator<Student>() {
21            @Override
22            public int compare(Student s1, Student s2) {
23                //按照年龄从小到大排序，年龄相同时，按照姓名的字母顺序排序
24                int num = s1.getAge() - s2.getAge();
25                int num2 = num == 0 ? s1.getName().compareTo(s2.getName())
: num;
26                return num2;
27            }
28        });
29
30        //遍历集合
31        for (Student s : array) {
32            System.out.println(s.getName() + "," + s.getAge());
33        }
34    }
35 }

```

## 3.斗地主案例

### 3.1模拟斗地主案例-普通版本【应用】

- 案例需求

通过程序实现斗地主过程中的洗牌，发牌和看牌

- 代码实现

```

1 public class PokerDemo {
2     public static void main(String[] args) {
3         //创建一个牌盒，也就是定义一个集合对象，用ArrayList集合实现
4         ArrayList<String> array = new ArrayList<String>();
5

```

```

6      //往牌盒里面装牌
7      /*
8          ♦2,♦3,♦4...♦K,♦A
9          ♣2,...
10         ♥2,...
11         ♠2,...
12         小王,大王
13     */
14     //定义花色数组
15     String[] colors = {"♦", "♣", "♥", "♠"};
16     //定义点数数组
17     String[] numbers = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J",
    "Q", "K", "A"};
18     for (String color : colors) {
19         for (String number : numbers) {
20             array.add(color + number);
21         }
22     }
23     array.add("小王");
24     array.add("大王");
25
26     //洗牌,也就是把牌打撒,用Collections的shuffle()方法实现
27     Collections.shuffle(array);
28
29     //      System.out.println(array);
30
31     //发牌,也就是遍历集合,给三个玩家发牌
32     ArrayList<String> lqxArray = new ArrayList<String>();
33     ArrayList<String> lyArray = new ArrayList<String>();
34     ArrayList<String> fgyArray = new ArrayList<String>();
35     ArrayList<String> dpArray = new ArrayList<String>();
36
37     for (int i = 0; i < array.size(); i++) {
38         String poker = array.get(i);
39         if (i >= array.size() - 3) {
40             dpArray.add(poker);
41         } else if (i % 3 == 0) {
42             lqxArray.add(poker);
43         } else if (i % 3 == 1) {
44             lyArray.add(poker);
45         } else if (i % 3 == 2) {
46             fgyArray.add(poker);
47         }
48     }
49
50     //看牌,也就是三个玩家分别遍历自己的牌
51     lookPoker("林青霞", lqxArray);
52     lookPoker("柳岩", lyArray);
53     lookPoker("风清扬", fgyArray);
54     lookPoker("底牌", dpArray);
55 }
56
57 //看牌的方法

```

```

58     public static void LookPoker(String name, ArrayList<String> array) {
59         System.out.print(name + "的牌是：");
60         for (String poker : array) {
61             System.out.print(poker + " ");
62         }
63         System.out.println();
64     }
65 }

```

## 3.2模拟斗地主案例-升级版【应用】

- 案例需求

通过程序实现斗地主过程中的洗牌，发牌和看牌。要求：对牌进行排序

- 代码实现

```

1  public class PokerDemo {
2      public static void main(String[] args) {
3          //创建HashMap，键是编号，值是牌
4          HashMap<Integer, String> hm = new HashMap<Integer, String>();
5
6          //创建ArrayList，存储编号
7          ArrayList<Integer> array = new ArrayList<Integer>();
8
9          //创建花色数组和点数数组
10         String[] colors = {"♦", "♣", "♥", "♠"};
11         String[] numbers = {"3", "4", "5", "6", "7", "8", "9", "10", "J", "Q",
12             "K", "A", "2"};
13
14         //从0开始往HashMap里面存储编号，并存储对应的牌。同时往ArrayList里面存储编号
15         int index = 0;
16
17         for (String number : numbers) {
18             for (String color : colors) {
19                 hm.put(index, color + number);
20                 array.add(index);
21                 index++;
22             }
23         }
24         hm.put(index, "小王");
25         array.add(index);
26         index++;
27         hm.put(index, "大王");
28         array.add(index);
29
30         //洗牌(洗的是编号)，用Collections的shuffle()方法实现
31         Collections.shuffle(array);
32
33         //发牌(发的也是编号，为了保证编号是排序的，创建TreeSet集合接收)
34         TreeSet<Integer> lqxSet = new TreeSet<Integer>();
35         TreeSet<Integer> lySet = new TreeSet<Integer>();
36         TreeSet<Integer> fqySet = new TreeSet<Integer>();
37         TreeSet<Integer> dpSet = new TreeSet<Integer>();

```

```

37
38     for (int i = 0; i < array.size(); i++) {
39         int x = array.get(i);
40         if (i >= array.size() - 3) {
41             dpSet.add(x);
42         } else if (i % 3 == 0) {
43             lqxSet.add(x);
44         } else if (i % 3 == 1) {
45             lySet.add(x);
46         } else if (i % 3 == 2) {
47             fqySet.add(x);
48         }
49     }
50
51     //调用看牌方法
52     lookPoker("林青霞", lqxSet, hm);
53     lookPoker("柳岩", lySet, hm);
54     lookPoker("风清扬", fqySet, hm);
55     lookPoker("底牌", dpSet, hm);
56 }
57
58 //定义方法看牌(遍历TreeSet集合, 获取编号, 到HashMap集合找对应的牌)
59 public static void lookPoker(String name, TreeSet<Integer> ts,
60 HashMap<Integer, String> hm) {
61     System.out.print(name + "的牌是:");
62     for (Integer key : ts) {
63         String poker = hm.get(key);
64         System.out.print(poker + " ");
65     }
66     System.out.println();
67 }

```