1.Map集合

1.1Map集合概述和特点【理解】

- Map集合概述

```
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

```
public class MapDemo01 {
    public static void main(String[] args) {
        //创建集合对象
        Map<String,String> map = new HashMap<String,String>();

        //V put(K key, V value) 将指定的值与该映射中的指定键相关联
        map.put("itheima001","林青霞");
        map.put("itheima002","张曼玉");
        map.put("itheima003","王祖贤");
        map.put("itheima003","柳岩");

        //输出集合对象
        System.out.println(map);
    }
}
```

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

```
public class MapDemo02 {
    public static void main(String[] args) {
        //创建集合对象
        Map<String,String> map = new HashMap<String,String>();

        //V put(K key,V value)：添加元素
        map.put("张无忌","赵敏");
        map.put("郭靖","黄蓉");
        map.put("杨过","小龙女");

        //V remove(Object key)：根据键删除键值对元素
//        System.out.println(map.remove("郭靖"));
//        System.out.println(map.remove("郭襄"));

        //void clear()：移除所有的键值对元素
//        map.clear();

        //boolean containsKey(Object key)：判断集合是否包含指定的键
//        System.out.println(map.containsKey("郭靖"));
//        System.out.println(map.containsKey("郭襄"));

        //boolean isEmpty()：判断集合是否为空
//        System.out.println(map.isEmpty());

        //int size()：集合的长度，也就是集合中键值对的个数
        System.out.println(map.size());


        //输出集合对象
        System.out.println(map);
    }
}
```

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

```
public class MapDemo03 {
    public static void main(String[] args) {
        //创建集合对象
        Map<String, String> map = new HashMap<String, String>();

        //添加元素
        map.put("张无忌", "赵敏");
        map.put("郭靖", "黄蓉");
        map.put("杨过", "小龙女");

        //V get(Object key):根据键获取值
//        System.out.println(map.get("张无忌"));
//        System.out.println(map.get("张三丰"));

        //Set<K> keySet():获取所有键的集合
//        Set<String> keySet = map.keySet();
//        for(String key : keySet) {
//            System.out.println(key);
//        }

        //Collection<V> values():获取所有值的集合
        Collection<String> values = map.values();
        for(String value : values) {
            System.out.println(value);
        }
    }
}
```

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

```
public class MapDemo01 {
    public static void main(String[] args) {
        //创建集合对象
        Map<String, String> map = new HashMap<String, String>();

        //添加元素
        map.put("张无忌", "赵敏");
        map.put("郭靖", "黄蓉");
        map.put("杨过", "小龙女");

        //获取所有键的集合。用keySet()方法实现
        Set<String> keySet = map.keySet();
        //遍历键的集合，获取到每一个键。用增强for实现
        for (String key : keySet) {
            //根据键去找值。用get(Object key)方法实现
            String value = map.get(key);
            System.out.println(key + "," + value);
        }
    }
}
```

1.5Map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

```
public class MapDemo02 {
    public static void main(String[] args) {
        //创建集合对象
        Map<String, String> map = new HashMap<String, String>();

        //添加元素
```

```
        map.put("张无忌", "赵敏");
        map.put("郭靖", "黄蓉");
        map.put("杨过", "小龙女");

        //获取所有键值对对象的集合
        Set<Map.Entry<String, String>> entrySet = map.entrySet();
        //遍历键值对对象的集合，得到每一个键值对对象
        for (Map.Entry<String, String> me : entrySet) {
            //根据键值对对象获取键和值
            String key = me.getKey();
            String value = me.getValue();
            System.out.println(key + "," + value);
        }
    }
}
```

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

```
public class Student {
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
```

```
    }
}
```

- 测试类

```
/*
    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

    思路：
        1:定义学生类
        2:创建HashMap集合对象
        3:创建学生对象
        4:把学生添加到集合
        5:遍历集合
            方式1：键找值
            方式2：键值对对象找键和值
 */
public class HashMapDemo {
    public static void main(String[] args) {
        //创建HashMap集合对象
        HashMap<String, Student> hm = new HashMap<String, Student>();

        //创建学生对象
        Student s1 = new Student("林青霞", 30);
        Student s2 = new Student("张曼玉", 35);
        Student s3 = new Student("王祖贤", 33);

        //把学生添加到集合
        hm.put("itheima001", s1);
        hm.put("itheima002", s2);
        hm.put("itheima003", s3);

        //方式1：键找值
        Set<String> keySet = hm.keySet();
        for (String key : keySet) {
            Student value = hm.get(key);
            System.out.println(key + "," + value.getName() + "," + value.getAge());
        }
        System.out.println("--------");

        //方式2：键值对对象找键和值
        Set<Map.Entry<String, Student>> entrySet = hm.entrySet();
        for (Map.Entry<String, Student> me : entrySet) {
            String key = me.getKey();
            Student value = me.getValue();
            System.out.println(key + "," + value.getName() + "," + value.getAge());
        }
    }
}
```

1.6.2HashMap集合练习之键是Student值是String

- 案例需求　　重写Student的HashSet&equal
  - 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

```
public class Student {
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Student student = (Student) o;

        if (age != student.age) return false;
        return name != null ? name.equals(student.name) : student.name ==
null;
    }

    @Override
    public int hashCode() {
        int result = name != null ? name.hashCode() : 0;
```

```
        result = 31 * result + age;
        return result;
    }
}
```

- 测试类

```
public class HashMapDemo {
    public static void main(String[] args) {
        //创建HashMap集合对象
        HashMap<Student, String> hm = new HashMap<Student, String>();

        //创建学生对象
        Student s1 = new Student("林青霞", 30);
        Student s2 = new Student("张曼玉", 35);
        Student s3 = new Student("王祖贤", 33);
        Student s4 = new Student("王祖贤", 33);

        //把学生添加到集合
        hm.put(s1, "西安");
        hm.put(s2, "武汉");
        hm.put(s3, "郑州");
        hm.put(s4, "北京");

        //遍历集合
        Set<Student> keySet = hm.keySet();
        for (Student key : keySet) {
            String value = hm.get(key);
            System.out.println(key.getName() + "," + key.getAge() + "," +
value);
        }
    }
}
```

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

```
public class ArrayListIncludeHashMapDemo {
    public static void main(String[] args) {
        //创建ArrayList集合
        ArrayList<HashMap<String, String>> array = new
ArrayList<HashMap<String, String>>();

        //创建HashMap集合，并添加键值对元素
        HashMap<String, String> hm1 = new HashMap<String, String>();
        hm1.put("孙策", "大乔");
        hm1.put("周瑜", "小乔");
```

```
        //把HashMap作为元素添加到ArrayList集合
        array.add(hm1);

        HashMap<String, String> hm2 = new HashMap<String, String>();
        hm2.put("郭靖", "黄蓉");
        hm2.put("杨过", "小龙女");
        //把HashMap作为元素添加到ArrayList集合
        array.add(hm2);

        HashMap<String, String> hm3 = new HashMap<String, String>();
        hm3.put("令狐冲", "任盈盈");
        hm3.put("林平之", "岳灵珊");
        //把HashMap作为元素添加到ArrayList集合
        array.add(hm3);

        //遍历ArrayList集合
        for (HashMap<String, String> hm : array) {
            Set<String> keySet = hm.keySet();
            for (String key : keySet) {
                String value = hm.get(key);
                System.out.println(key + "," + value);
            }
        }
    }
}
```

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

```
public class HashMapIncludeArrayListDemo {
    public static void main(String[] args) {
        //创建HashMap集合
        HashMap<String, ArrayList<String>> hm = new HashMap<String,
ArrayList<String>>();

        //创建ArrayList集合，并添加元素
        ArrayList<String> sgyy = new ArrayList<String>();
        sgyy.add("诸葛亮");
        sgyy.add("赵云");
        //把ArrayList作为元素添加到HashMap集合
        hm.put("三国演义",sgyy);

        ArrayList<String> xyj = new ArrayList<String>();
        xyj.add("唐僧");
        xyj.add("孙悟空");
        //把ArrayList作为元素添加到HashMap集合
        hm.put("西游记",xyj);

```

```
        ArrayList<String> shz = new ArrayList<String>();
        shz.add("武松");
        shz.add("鲁智深");
        //把ArrayList作为元素添加到HashMap集合
        hm.put("水浒传",shz);

        //遍历HashMap集合
        Set<String> keySet = hm.keySet();
        for(String key : keySet) {
            System.out.println(key);
            ArrayList<String> value = hm.get(key);
            for(String s : value) {
                System.out.println("\t" + s);
            }
        }
    }
}
```

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“aababcabcdabcde” 在控制台输出：“a(5)b(4)c(3)d(2)e(1)”
- 代码实现

```
public class HashMapDemo {
    public static void main(String[] args) {
        //键盘录入一个字符串
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入一个字符串：");
        String line = sc.nextLine();

        //创建HashMap集合，键是Character，值是Integer
//        HashMap<Character, Integer> hm = new HashMap<Character, Integer>();
        TreeMap<Character, Integer> hm = new TreeMap<Character, Integer>();
                                                    改为TreeMap数据结构，则按照“键”有序
        //遍历字符串，得到每一个字符                        联想HashSet和TreeSet
        for (int i = 0; i < line.length(); i++) {
            char key = line.charAt(i);

            //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
            Integer value = hm.get(key);

            if (value == null) {
                //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
                hm.put(key,1);
            } else {
                //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
                value++;
                hm.put(key,value);
```

```
            }
        }

        //遍历HashMap集合，得到键和值，按照要求进行拼接
        StringBuilder sb = new StringBuilder();

        Set<Character> keySet = hm.keySet();
        for(Character key : keySet) {
            Integer value = hm.get(key);
            sb.append(key).append("(").append(value).append(")");
        }

        String result = sb.toString();

        //输出结果
        System.out.println(result);
    }
}
```

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

```
public class CollectionsDemo01 {
    public static void main(String[] args) {
        //创建集合对象
        List<Integer> list = new ArrayList<Integer>();

        //添加元素
        list.add(30);
        list.add(20);
        list.add(50);
        list.add(10);
        list.add(40);

        //public static <T extends Comparable<? super T>> void sort•(List<T>
list)：将指定的列表按升序排序
```

```
//        Collections.sort(list);

        //public static void reverse•(List<?> list)：反转指定列表中元素的顺序
//        Collections.reverse(list);

        //public static void shuffle•(List<?> list)：使用默认的随机源随机排列指定的列表
        Collections.shuffle(list);

        System.out.println(list);
    }
}
```

2.2ArrayList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

```
public class Student {
    private String name;
    private int age;

    public Student() {
    }

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

  - 测试类

```java
public class CollectionsDemo02 {
    public static void main(String[] args) {
        //创建ArrayList集合对象
        ArrayList<Student> array = new ArrayList<Student>();

        //创建学生对象
        Student s1 = new Student("linqingxia", 30);
        Student s2 = new Student("zhangmanyu", 35);
        Student s3 = new Student("wangzuxian", 33);
        Student s4 = new Student("liuyan", 33);

        //把学生添加到集合
        array.add(s1);
        array.add(s2);
        array.add(s3);
        array.add(s4);

        //使用Collections对ArrayList集合排序
        //sort•(List<T> list, Comparator<? super T> c)
        Collections.sort(array, new Comparator<Student>() {
            @Override
            public int compare(Student s1, Student s2) {
                //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
                int num = s1.getAge() - s2.getAge();
                int num2 = num == 0 ? s1.getName().compareTo(s2.getName())
: num;
                return num2;
            }
        });

        //遍历集合
        for (Student s : array) {
            System.out.println(s.getName() + "," + s.getAge());
        }
    }
}
```

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

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

```

```
        //往牌盒里面装牌
        /*
            ♦2,♦3,♦4...♦K,♦A
            ♣2,...
            ♥2,...
            ♠2,...
            小王，大王
         */
        //定义花色数组
        String[] colors = {"♦", "♣", "♥", "♠"};
        //定义点数数组
        String[] numbers = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "J",
"Q", "K", "A"};
        for (String color : colors) {
            for (String number : numbers) {
                array.add(color + number);
            }
        }
        array.add("小王");
        array.add("大王");

        //洗牌，也就是把牌打撒，用Collections的shuffle()方法实现
        Collections.shuffle(array);

//        System.out.println(array);

        //发牌，也就是遍历集合，给三个玩家发牌
        ArrayList<String> lqxArray = new ArrayList<String>();
        ArrayList<String> lyArray = new ArrayList<String>();
        ArrayList<String> fqyArray = new ArrayList<String>();
        ArrayList<String> dpArray = new ArrayList<String>();

        for (int i = 0; i < array.size(); i++) {
            String poker = array.get(i);
            if (i >= array.size() - 3) {
                dpArray.add(poker);
            } else if (i % 3 == 0) {
                lqxArray.add(poker);
            } else if (i % 3 == 1) {
                lyArray.add(poker);
            } else if (i % 3 == 2) {
                fqyArray.add(poker);
            }
        }

        //看牌，也就是三个玩家分别遍历自己的牌
        lookPoker("林青霞", lqxArray);
        lookPoker("柳岩", lyArray);
        lookPoker("风清扬", fqyArray);
        lookPoker("底牌", dpArray);
    }

    //看牌的方法
```

```
    public static void lookPoker(String name, ArrayList<String> array) {
        System.out.print(name + "的牌是：");
        for (String poker : array) {
            System.out.print(poker + " ");
        }
        System.out.println();
    }
}
```

3.2模拟斗地主案例-升级版本【应用】

- 案例需求

  （需求排序，所以牌要有序，要改数据结构）

  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

```
public class PokerDemo {
    public static void main(String[] args) {
        //创建HashMap，键是编号，值是牌
        HashMap<Integer, String> hm = new HashMap<Integer, String>();

        //创建ArrayList，存储编号
        ArrayList<Integer> array = new ArrayList<Integer>();

        //创建花色数组和点数数组
        String[] colors = {"♦", "♣", "♥", "♠"};
        String[] numbers = {"3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A", "2"};

        //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
        int index = 0;

        for (String number : numbers) {
            for (String color : colors) {
                hm.put(index, color + number);
                array.add(index);
                index++;
            }
        }
        hm.put(index, "小王");
        array.add(index);
        index++;
        hm.put(index, "大王");
        array.add(index);

        //洗牌(洗的是编号)，用Collections的shuffle()方法实现
        Collections.shuffle(array);

        //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)
        TreeSet<Integer> lqxSet = new TreeSet<Integer>();
        TreeSet<Integer> lySet = new TreeSet<Integer>();
        TreeSet<Integer> fqySet = new TreeSet<Integer>();
        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,
    HashMap<Integer, String> hm) {
60          System.out.print(name + "的牌是：");
61          for (Integer key : ts) {
62              String poker = hm.get(key);
63              System.out.print(poker + " ");
64          }
65          System.out.println();
66      }
67  }
```