Map类

2018年7月1日 14:58

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

/*
 * 学生根据学号区分，通过学号获取学生姓名
 * Map集合的特点：可以存储键值对的元素
 *              学号1---->姓名1
 * 键是唯一的，但多个键可以为同一个值
 *              将建映射到值的对象，一个映射不能包含重复的键，每个键最多能映射到一个
值
 *
 * Map与Collection的区别：
 *                      Map集合存储元素是成对出现的；
 *                      Collection集合存储元素单独出现
 *
 * 注意：Map集合的数据结构只针对键有效，与值无关
 * HashMap和TreeMap
 *
 * Map集合的功能：
 *              1.添加功能
 *                              V put(K key, V value):添加元素
 *                              如果键是第一次存储，就返回null
 *                              如果键不是第一次存在，就把值用新值覆盖，返回以前的值
 *              2.删除功能
 *                              void clear():移除所有的键值对元素
 *                              V remove(Object key):根据键删除键值对元素，并返回值
 *              3.判断功能
 *                              boolean containsKey(Object key)：判断集合是否包含指定键
 *                              boolean containsValue(Object value)：判断集合是否包含指
定值
 *                              boolean isEmpty():判断集合是否为空
 *              4.获取功能
 *                              Set<Map.Entry<K,V>> entrySet()
 *                              V get(Object key):根据键获取值
 *                              Set<K> KeySet():获取集合中所有键的集合
 *                              Collection<V> values(): 获取几何中所有值的集合
 *               5.长度功能
 *                              int size():返回集合中的键值对数
 *
 */
public class MapDemo {
        public static void main(String[] args) {
                Map<String, String> map = new HashMap<String, String>();

                // V put(K key, V value):添加元素
                // System.out.println("put="+map.put("文章","马伊琍"));//put=null
                // System.out.println("put="+map.put("文章","姚笛"));//put=马伊琍

                map.put("邓超", "孙俪");
                map.put("黄晓明", "Angelababy");
                map.put("周杰伦", "昆凌");
                map.put("刘恺威", "杨幂");

                // void clear()
                // map.clear();

                // V remove(Object key)
                System.out.println("remove=" + map.remove("黄晓明"));//
                remove=Angelababy
                System.out.println("remove=" + map.remove("黄晓"));// remove=null

                // boolean containsKey(Object key)
                System.out.println("contains=" + map.containsKey("黄晓明"));
                System.out.println("contains=" + map.containsKey("周杰伦"));

                // boolean isEmpty()
                System.out.println("isempty=" + map.isEmpty());// isempty=false

                // int size()
                System.out.println("size=" + map.size());// size=3

                // V get(Object key)
                System.out.println("get=" + map.get("周杰伦"));// get=昆凌
```

```
		// Set<K> KeySet()
		Set<String> set = map.keySet();
		for (String key : set) {
			System.out.println(key);
		}

		// Collection<V> values()
		Collection<String> col = map.values();
		for (String value : col) {
			System.out.println(value);
		}

		System.out.println("Map:" + map);// 无序
	}
}
```

Map的遍历1

2018年7月17日 22:40

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

/*
 * Map集合的遍历
 *              1.获取所有的键
 *              2.遍历键的集合，获取每一个键
 *              3.根据键去找值
 */
public class MapDemo2 {
        public static void main(String[] args) {
                Map<String, String> map = new HashMap<String, String>();

                map.put("邓超", "孙俪");
                map.put("黄晓明", "Angelababy");
                map.put("周杰伦", "昆凌");
                map.put("刘恺威", "杨幂");

                // 遍历
                Set<String> set = map.keySet();
                for (String key : set) {
                        String value = map.get(key);
                        System.out.println(key + "---" + value);
                }
        }
}
```

Map的遍历2

2018年7月17日 22:41

```
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
/*
 * Map集合的遍历
 * 思路：
 *          1.获取所有键值对对象的集合
 *          2.遍历键值对对象集合，得到一个键值对对象
 *          3.根据键值对对象获取键和值
 *
 * 获取键值对对象：
 *          Set<Map.Entry<K,V>> entrySet()
 *          Map.Entry<K,V>就是键值对对象
 *          Entry:实体
 */
public class MapDemo3 {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<String, String>();

        map.put("邓超", "孙俪");
        map.put("黄晓明", "Angelababy");
        map.put("周杰伦", "昆凌");
        map.put("刘恺威", "杨幂");

        Set<Map.Entry<String,String>> set=map.entrySet();
        for(Map.Entry<String,String> me:set) {
            //根据键值获取键和值
            String key = me.getKey();
            String value = me.getValue();
            System.out.println(key+"----"+value);
        }
    }
}
```

案例1

2018年7月17日 22:54

```
import java.util.HashMap;
import java.util.Set;

/*
 * HashMap:是基于哈希表的Map接口实现
 * 哈希表的作用是用来保证键的惟一性的
 */
public class HashMapDemo {
	public static void main(String[] args) {
		HashMap<String,String> hm=new HashMap<String,String>();

		String key1="it001";
		String value1="马云";
		hm.put(key1, value1);
		hm.put("it003","马化腾");
		hm.put("it004","乔布斯");
		hm.put("it005","张朝阳");
		hm.put("it002","裘伯君");

		Set<String> set=hm.keySet();
		for(String key:set) {
			String value=hm.get(key);
			System.out.println(key+"-----"+value);
		}
	}
}
```

案例2

2018年7月18日 9:43

```
import java.util.HashMap;
import java.util.Set;

//HashMap<Integer,String>
public class HashMapDemo2 {
	public static void main(String[] args) {
		HashMap<Integer, String> hm = new HashMap<Integer, String>();

		hm.put(27, "内马尔");
		hm.put(22, "王若潇");
		hm.put(33, "C罗");
		hm.put(31, "梅西");

		// 下面的写法是八进制，不能出现超过8的数据
		// hm.put(003,"hello");
		// hm.put(008,"hello");

		Set<Integer> set = hm.keySet();
		for (Integer key : set) {
			String value = hm.get(key);
			System.out.println(key + "----" + value);
		}
	}
}
```

案例3

2018年7月18日 9:55

```
import java.util.HashMap;
import java.util.Set;

//HashMap<String,Student>
public class HashMapDemo3 {
	public static void main(String[] args) {
		HashMap<String, Student> hm = new HashMap<>();

		Student s1 = new Student("周星驰", 58);
		Student s2 = new Student("刘德华", 55);
		Student s3 = new Student("梁朝伟", 60);
		Student s4 = new Student("刘嘉玲", 63);

		hm.put("9527", s1);
		hm.put("9522", s2);
		hm.put("9524", s3);
		hm.put("9529", s4);

		Set<String> set = hm.keySet();
		for (String key : set) {
			Student s = hm.get(key);
			System.out.println(key + "----" + s.getName() + "----" + s.getAge());
		}
	}
}
```

案例4

2018年7月18日 9:55

```
/*
 * HashMap<Student,String>
 */
public class HashMapDemo4 {
        public static void main(String[] args) {
                HashMap<Student,String> hm= new HashMap<>();

                Student s1 = new Student("周星驰", 58);
                Student s2 = new Student("刘德华", 55);
                Student s3 = new Student("梁朝伟", 60);
                Student s4 = new Student("刘嘉玲", 63);
                Student s5= new Student("刘德华", 55);

                hm.put(s1, "8888");
                hm.put(s2, "5555");
                hm.put(s3, "7777");
                hm.put(s4, "6666");
                hm.put(s5, "1111");

                Set<Student> set=hm.keySet();
                for(Student key:set) {
                        String value=hm.get(key);
                        System.out.println(key.getName()+"---"+key.getAge()+"---"+value);
                }
        }
}
```

$\xrightarrow{\text{重新Student}}$

```
@Override
        public int hashCode() {
                final int prime = 31;
                int result = 1;
                result = prime * result + age;
                result = prime * result + ((name == null) ? 0 :
                name.hashCode());
                return result;
        }

        @Override
        public boolean equals(Object obj) {
                if (this == obj)
                        return true;
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                Student other = (Student) obj;
                if (age != other.age)
                        return false;
                if (name == null) {
                        if (other.name != null)
                                return false;
                } else if (!name.equals(other.name))
                        return false;
                return true;
        }
```

LinkedHashMap

2018年7月18日　　10:07

```
import java.util.LinkedHashMap;
import java.util.Set;

/*
 * LinkedHashMap:是Map接口的哈希表和链表实现，具有可预知的迭代顺序
 * 有哈希表保证惟一性，由链表保证有序
 */
public class LinkedHashMapDemo {
	public static void main(String[] args) {
		LinkedHashMap<String, String> hm = new LinkedHashMap<>();

		hm.put("it003", "马化腾");
		hm.put("it004", "乔布斯");
		hm.put("it005", "张朝阳");
		hm.put("it002", "裘伯君");
		hm.put("it002", "比尔盖茨");

		Set<String> set = hm.keySet();
		for (String key : set) {
			String value = hm.get(key);
			System.out.println(key + "---" + value);// 有序
		}
	}
}
```

TreeMap

2018年7月18日 10:25

```
/*
 * TreeMap:是基于红黑树的Map接口实现
 */
public class TreeMapDemo {
	public static void main(String[] args) {
		TreeMap<String, String> tm = new TreeMap<String, String>();

		tm.put("it003", "马化腾");
		tm.put("it004", "乔布斯");
		tm.put("it005", "张朝阳");
		tm.put("it002", "裘伯君");
		tm.put("it002", "比尔盖茨");

		Set<String> set = tm.keySet();
		for (String key : set) {
			String value = tm.get(key);
			System.out.println(key + "----" + value);// 自然排序
		}
	}
}
```

键为对象

2018年7月18日 10:31

```
import java.util.Comparator;
import java.util.Set;
import java.util.TreeMap;

import cn.itcast_02.Student;

/*
 * TreeMap<Student,String>
 */
public class TreeMapDemo2 {
	public static void main(String[] args) {
		TreeMap<Student, String> tm = new TreeMap<Student, String>(new
		Comparator<Student>() {

			@Override
			public int compare(Student s1, Student s2) {
				int num = s2.getAge() - s1.getAge();// 年龄由大到小
				int num2 = num == 0 ?
				s1.getName().compareTo(s2.getName()) : num;
				return num2;
			}
		});

		Student s1 = new Student("周星驰", 58);
		Student s2 = new Student("刘德华", 55);
		Student s3 = new Student("梁朝伟", 60);
		Student s4 = new Student("刘嘉玲", 63);
		Student s5 = new Student("刘德华", 55);

		tm.put(s1, "香港");
		tm.put(s2, "香港");
		tm.put(s3, "澳门");
		tm.put(s4, "大陆");
		tm.put(s5, "香港");

		Set<Student> set = tm.keySet();
		for (Student key : set) {
			String value = tm.get(key);
			System.out.println(key.getName() + "---" + key.getAge() + "---" +
			value);
		}
	}
}
```

练习

2018年7月18日 10:42

```
import java.util.Scanner;
import java.util.Set;
import java.util.TreeMap;

/*
 * 需求：
 *              "aabbabcabcdabcde" 转换为 a(5)b(4)c(3)d(2)e(1)
 */
public class TreeMapDemo {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("请输入一个字符串");
		String line = sc.nextLine();

		TreeMap<Character, Integer> tm = new TreeMap<>();
		char[] chs = line.toCharArray();
		for (char ch : chs) {
			Integer i = tm.get(ch);
			if (i == null) {
				tm.put(ch, 1);
			} else {
				i++;//Integer自动拆装箱
				tm.put(ch, i);
			}
		}

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

		String result = sb.toString();
		System.out.println("result=" + result);
		// result=a(5)b(5)c(3)d(2)e(1)
```

```
          }
      }
```

HashMap嵌套

2018年7月18日 11:29

```
import java.util.HashMap;
import java.util.Set;

/*
 * HashMap嵌套HashMap
 */
public class HashMapDemo {
	public static void main(String[] args) {
		HashMap<String, HashMap<String, Integer>> czbkMap = new
		HashMap<>();

		HashMap<String, Integer> jcMap = new HashMap<>();
		jcMap.put("陈玉龙", 20);
		jcMap.put("高越", 22);
		czbkMap.put("jc", jcMap);

		HashMap<String, Integer> jyMap = new HashMap<>();
		jyMap.put("李杰", 21);
		jyMap.put("曹石磊", 26);
		czbkMap.put("jy", jyMap);

		Set<String> czbkMapSet = czbkMap.keySet();
		for (String key : czbkMapSet) {
			System.out.println(key);
			HashMap<String, Integer> czbkMapValue = czbkMap.get(key);
			Set<String> czbkMapValueSet = czbkMapValue.keySet();
			for (String czbkMapValueKey : czbkMapValueSet) {
				Integer czbkMapValueValue =
				czbkMapValue.get(czbkMapValueKey);
				System.out.println(czbkMapValueKey + "----" +
				czbkMapValueValue);
			}
		}
	}
}
```

Hashtable

2018年7月18日 14:04

```
/*
 * Hashtable与HashMap的区别
 * Hashtable：线程安全，效率低，不允许null键和null值
 * HashMap：线程不安全，效率高，允许null键和null值
 */
public class HashTableDemo {
        public static void main(String[] args) {
                Hashtable<String,String> ht = new Hashtable<String,String>();

                ht.put("it001","马云");
                ht.put("it002","马化腾");
                ht.put("it003","涂磊");
        }
}
```

面试题

2018年7月18日 15:16

List，Set，Map是否都继承自Map接口

List和Set不是继承自Map接口，它继承自Collection接口
Map本身就是一个顶层接口

Collections类

2018年7月18日 15:51

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/*
 * Collections是针对集合进行操作的工具类，是静态方法
 *
 * Collection与Collections的区别：
 * Collection是单列集合的顶层接口，有子接口List和Set
 * Collections：是针对集合进行操作的工具类，有对集合进行排序和二分查找的方法
 *
 * 方法：
 *              public static <T> void sort(List<T> list):自然排序
 *              public static <T> int binarySearch<List<?> list,T key>:二分查找
 *              public static <T> T max(Collection<?> coll):最大值
 *              public static <T> void reverse (List<T> list):反转
 *              public static <T> void shuffle(List<?> list):随机置换
 */
public class CollectionsDemo {
        public static void main(String[] args) {
                //创建集合对象
                List<Integer> list = new ArrayList<Integer>();

                list.add(30);
                list.add(20);
                list.add(50);
                list.add(10);
                list.add(40);

                System.out.println("list="+list);//list=[30, 20, 50, 10, 40]
                //public static <T> void sort(List<T> list)
                Collections.sort(list);
                System.out.println("list="+list);//list=[10, 20, 30, 40, 50]

                //public static <T> int binarySearch<List<?> list,T key>
                System.out.println("binarySearch="+Collections. binarySearch(list,30));//2
        System.out.println("binarySearch="+Collections. binarySearch(list,300));//-6

        //public static <T> T max(Collection<?> coll)
        System.out.println("max="+Collections.max(list));//max=50

        //public static <T> void reverse (List<T> list)
        Collections.reverse(list);
        System.out.println("list="+list);//list=[50, 40, 30, 20, 10]

        //public static <T> void shuffle(List<?> list)
        Collections.shuffle(list);
        System.out.println("list="+list);//随机变换位置

    }
}
```

排序

2018年7月18日 15:51

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

/*
 * Collections对自定义对象排序
 */
public class CollectionsDemo2 {
	public static void main(String[] args) {
		List<Student> list = new ArrayList<Student>();

		Student s1 = new Student("周星驰", 58);
		Student s2 = new Student("刘德华", 55);
		Student s3 = new Student("梁朝伟", 60);
		Student s4 = new Student("刘嘉玲", 63);
		Student s5 = new Student("刘德华", 55);

		list.add(s1);
		list.add(s2);
		list.add(s3);
		list.add(s4);
		list.add(s5);

		// Collections.sort(list);
		// 比较器排序
		Collections.sort(list, new Comparator<Student>() {

			@Override
			public int compare(Student s1, Student s2) {
				return 0;
			}
		});
		// 如果同时有自然排序和比较器排序，以比较器排序为主

		for (Student s : list) {
			System.out.println(s.getName() + "---" + s.getAge());
		}
	}
}
```

```
public class Student implements Comparable<Student> {
	private String name;
	private int age;

	public Student() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Student(String name, int age) {
		super();
		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 String toString() {
		return "Student [Name=" + name + ", age=" + age + "]";
	}

	@Override
	public int compareTo(Student s) {
		// return 0;
		int num = this.age - s.age;
		int num2 = num == 0 ? this.name.compareTo(s.name) : num;
		return num2;
	}
}
```

斗地主

2018年7月18日　　17:33

```
package cn.itcast_08;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.TreeSet;

/*
 * 思路:
 *              1.创建一个HashMap集合
 *              2.创建一个ArrayList集合
 *              3.创建花色数组和点数数组
 *              4.从0开始往HashMap中储存编号和对应的牌
 *              5.洗牌（洗编号）
 *              6.发牌（发编号）
 *              7.看牌（获取编号，从TreeMap中找对应值）
 */
public class PokerDemo2 {
	public static void main(String[] args) {
		HashMap<Integer, String> hm = new HashMap<Integer, String>();

		ArrayList<Integer> array = new ArrayList<Integer>();
		// 定义一个花色数组
		String[] colors = { "♠", "♥", "♣", "♦" };
		// 定义一个点数数组
		String[] numbers = { "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A",
		"2", };

		int index = 0;
		for (String number : numbers) {
			for (String color : colors) {
				String poker = color.concat(number);
				hm.put(index, poker);
				array.add(index);
				index++;
			}
```

```
        }
        hm.put(index, "小王");
        array.add(index);
        index++;
        hm.put(index, "大王");
        array.add(index);

        // 洗牌
        Collections.shuffle(array);

        // 发牌
        TreeSet<Integer> wang = new TreeSet<>();
        TreeSet<Integer> ruo = new TreeSet<>();
        TreeSet<Integer> xiao = new TreeSet<>();
        TreeSet<Integer> dipai = new TreeSet<>();

        for (int x = 0; x < array.size(); x++) {
            if (x >= array.size() - 3) {
                dipai.add(array.get(x));
            } else if (x % 3 == 0) {
                wang.add(array.get(x));
            } else if (x % 3 == 1) {
                ruo.add(array.get(x));
            } else {
                xiao.add(array.get(x));
            }
        }

        // 看牌
        lookPoker("王", wang, hm);
        lookPoker("若", ruo, hm);
        lookPoker("潇", xiao, hm);
        lookPoker("底牌", dipai, hm);
    }

    public static void lookPoker(String name, TreeSet<Integer> ts,
    HashMap<Integer, String> hm) {
        System.out.println(name + "的牌是：");
        for (Integer key : ts) {
            String value = hm.get(key);
```

```
                System.out.print(value + "\t");
            }
            System.out.println();
        }
    }
```