MapDemo

```
package map;

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

/**
 * java.util.Map：查找表
 * Map体现的结构是一个多行两列的表格，左列称为key，右列称为value
 * Map总是以key-value对的形式保存数据，并根据key提取value。
 * 在Map中key是不允许重复的（用equals()来判断是否重复）。
 * <p>
 * Map是一个接口，规定了Map所有功能对应的方法定义，其中常用的实现类：
 * java.util.HashMap：散列表（哈希表），使用散列算法实现的Map，当今查询速度最快的数据结构。在缓存中被大量应用。
 */
public class MapDemo {
    public static void main(String[] args) {
        Map<String, Integer> map = new HashMap<>();

        /*
            V put(K k,V v)：
            将一组键值对存入Map中。
            由于Map要求key不能重复，所以如果使用已有的key存入Map，那么就是替换value操作，此时返回值为被替换的value。
            如果不是重复的key，那么返回值为null。
         */
        //int value1 = map.put("语文", 100);
        //System.out.println(value1);
        //返回的null不能拆箱为int，会有NullPointerException空指针异常，所以只能用Integer

        map.put("语文", 99);
        map.put("数学", 98);
        map.put("英语", 97);
        map.put("物理", 96);
        map.put("化学", 95);
        System.out.println(map);

        Integer value = map.put("数学", 77); //将数学原有的98替换为77并返回98
        System.out.println(map);
        System.out.println(value);

        /*
            V get(Object key)：
            根据给定的key获取对应的value。
            如果给定的key不存在则返回null。
         */
        value = map.get("英语");
        System.out.println(value);
        value = map.get("体育");
        System.out.println(value);
```

```
        int size = map.size();
        System.out.println("size: " + size);

        //map.clear();
        //System.out.println(map);

        /*
            V remove(Object key):
            删除当前Map中给定的key所对应的一组键值对，返回值为这个key对应的value
         */
        Integer value2 = map.remove("英语");
        System.out.println(map);
        System.out.println(value2);

        /*
            判断Map是否包含给定的key或value
         */
        boolean ck = map.containsKey("语文");
        boolean cv = map.containsValue(99);
        System.out.println("包含key: " + ck);
        System.out.println("包含value: " + cv);
    }
}
```

MapDemo2

```
package map;

import java.util.*;

/**
 * Map的遍历：
 * Map提供了三种遍历方式：
 * 1：遍历所有key
 * 2：遍历每一组键值对
 * 3：遍历所有value（相对不常用）
 */
public class MapDemo2 {
    public static void main(String[] args) {
        Map<String, Integer> map = new HashMap<>();
        map.put("语文", 99);
        map.put("数学", 98);
        map.put("英语", 97);
        map.put("物理", 96);
        map.put("化学", 99);
        System.out.println(map);

        /*
            遍历所有key：
            Set<K> keySet()：
            将当前Map中所有key以一个Set集合形式返回。
         */
        Set<String> keySet = map.keySet();
        for (String key : keySet) {
            System.out.println("key: " + key);
        }

        /*
            遍历每一组键值对：
            Set entrySet()：
            将当前Map中每一组键值对以一个Entry实例形式表示，并将所有键值对存入Set后返回。
            java.util.Map.Entry的每一个实例表示一组键值对。
            两个常用方法：
            K getKey()
            V getValue()
         */
        Set<Map.Entry<String, Integer>> entrySet = map.entrySet();
        for (Map.Entry<String, Integer> entry : entrySet) {
            String key = entry.getKey();
            Integer value = entry.getValue();
            System.out.println(key + ": " + value);
        }

        /*
            Collection values()：
            将当前Map中所有value以一个Collection形式返回。
         */
        //key不能重复，所以用Set。value可以重复，所以用Collection
        Collection<Integer> values = map.values();
        for (Integer value : values) {
```

```
            System.out.println("value: " + value);
        }

        //JDK8中集合和Map都提供了基于lambda表达式的遍历方式
        Collection<String> c = new ArrayList<>();
        c.add("one");
        c.add("two");
        c.add("three");
        c.add("four");
        c.add("five");

        for (String s : c) {
            System.out.println(s);
        }

        c.forEach(s -> System.out.println(s)); //消费者模式
        //@FunctionalInterface就是用来指定某个接口必须是函数式接口，函数式接口就是为
Lambda表达式准备的

        map.forEach((k, v) -> System.out.println(k + ": " + v));
    }
}
```