Map映射集

原理

用户存储键值类型数据的数据结构.

底层原理为Hash分桶算法.

特点

键不能重复,值可以重复

存储时顺序信息会丢失

存取效率很高,但是对存储空间有一定的浪费

API

方法定义	功能解释
V put(K key, V v)	将指定的值与此Map中的指定键关联。如果此映射以前包含一个该键的映射关系，则用指定值替换旧值
V get(Object key)	返回指定键所映射的值；如果此Map不包含该键的映射关系，则返回null。
int size()	返回此Map中的键-值映射关系数。
V remove(Object key)	如果存在一个键的映射关系，则将其从此Map中移除。
boolean containsKey(Object key)	如果此Map包含指定键的映射关系，则返回 true。
boolean containsValue(Object value)	如果此Map将一个或多个键映射到指定值，则返回 true。
boolean isEmpty()	如果此Map未包含键-值映射关系，则返回 true。
void clear()	从此Map中移除所有映射关系。

基础案例

```
package map2;

import java.util.HashMap;

/**
 * HashMap基础操作
 */
public class Demo01 {
```

```
    public static void main(String[] args) {
        //HashMap<String,Integer> map = new HashMap();
        HashMap map = new HashMap();
        //存数据
        map.put("k1","v1");
        map.put("k2","v2");
        map.put("k3","v3");
        //取数据
        String k2 = (String) map.get("k2");
        System.out.println(k2);
        //获取存储的数量
        int size = map.size();
        System.out.println(size);
        //移除数据
        map.remove("k2");
        //判断是否包含键
        System.out.println(map.containsKey("k2"));
        //判断是否包含值
        System.out.println(map.containsValue("v3"));
        //判断是否为空
        System.out.println(map.isEmpty());
        //清空Map
        map.clear();
        System.out.println(map);
    }
}
```

遍历案例

```
package map2;

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

/**
 * HashMap遍历
 */
public class Demo02 {
    public static void main(String[] args) {
        Map<String,Integer> map = new HashMap<>();
        map.put("数学",99);
        map.put("语文",87);
        map.put("英语",77);
        map.put("体育",66);

        //遍历1：获取所有键的集合,遍历出每一个键,通过键查找值
        Set<String> keys = map.keySet();
        for(String key : keys){
            Integer value = map.get(key);
            System.out.println(key + "#" + value);
        }
```

```
        //遍历2:获取所有键值对的集合,从键值对中获取键值
        Set<Map.Entry<String, Integer>> entries = map.entrySet();
        for(Map.Entry<String,Integer> entry : entries){
            System.out.println(entry.getKey()+"#"+entry.getValue());
        }

        //遍历3:直接对值进行遍历
        Collection<Integer> values = map.values();
        for(Integer v : values){
            System.out.println(v);
        }

        //遍历4:使用forEach方法,通过lambda表达式遍历
        map.forEach((k,v)-> System.out.println(k+"#"+v));
    }
}
```