第12天 集合

今日内容介绍

* Set集合特点和HashSet集合类
* Map集合
* HashMap集合的相关案例

# Set集合特点和HashSet集合类

## Set集合的特点

Set:元素唯一,存储元素无序

一个不包含重复元素的 collection

HashSet:

它不保证 set 的迭代顺序；特别是它不保证该顺序恒久不变

### 案例代码一:

package com.itheima\_01;

import java.util.HashSet;

import java.util.Set;

/\*

\* Set:

\* 一个不包含重复元素的 collection

\*

\* HashSet:

\* 它不保证 set 的迭代顺序；特别是它不保证该顺序恒久不变

\*/

public class SetDemo {

public static void main(String[] args) {

//创建集合对象

Set<String> set = new HashSet<String>();

//添加元素

set.add("hello");

set.add("world");

set.add("java");

//唯一

set.add("world");

//遍历集合

for(String s : set) {

System.out.println(s);

}

}

}

## HashSet集合类

HashSet:

它不保证 set 的迭代顺序；特别是它不保证该顺序恒久不变

### HashSet保证元素唯一性的原理

HashSet保证元素唯一性的原理?

通过查看add方法的源码，我们知道了添加功能的执行过程中，是进行了数据的判断的。

这个判断的流程是：

首先比较对象的哈希值是否相同，这个哈希值是根据对象的hashCode()计算出来的。

如果哈希值不同，就直接添加到集合中

如果哈希值相同，继续执行equals()进行比较，

返回的是true，说明元素重复，不添加。

返回的是false，说明元素不重复，就添加。

如果我们使用HashSet集合存储对象，你要想保证元素的唯一性，就必须重写hashCode()和equals()方法。

#### 案例代码二:

**package** com.itheima\_02;

**import** java.util.HashSet;

/\*

\* HashSet保证元素唯一性的原理?

\*

\* 通过查看add方法的源码，我们知道了添加功能的执行过程中，是进行了数据的判断的。

\* 这个判断的流程是：

\* 首先比较对象的哈希值是否相同，这个哈希值是根据对象的hashCode()计算出来的。

\* 如果哈希值不同，就直接添加到集合中

\* 如果哈希值相同，继续执行equals()进行比较，

\* 返回的是true，说明元素重复，不添加。

\* 返回的是false，说明元素不重复，就添加。

\*

\* 如果我们使用HashSet集合存储对象，你要想保证元素的唯一性，就必须重写hashCode()和equals()方法。

\*/

**public** **class** HashSetDemo {

**public** **static** **void** main(String[] args) {

//创建集合对象

HashSet<String> hs = **new** HashSet<String>();

//添加元素

hs.add("hello");

hs.add("world");

hs.add("java");

hs.add("world");

//遍历集合

**for**(String s : hs) {

System.***out***.println(s);

}

}

}

### HashSet集合的练习存储自定义对象并遍历

HashSet集合存储自定义对象并遍历

提示：自定义一个学生类，给出成员变量name和age。遍历集合的时候，在控制台输出学生对象的成员变量值。

两种方式遍历

迭代器

增强for

#### 案例代码三:

**package** com.itheima\_03;

**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;

}

}

#### 案例代码四:

package com.itheima\_03;

import java.util.HashSet;

import java.util.Iterator;

/\*

\* HashSet集合存储自定义对象并遍历

\* 提示：自定义一个学生类，给出成员变量name和age。遍历集合的时候，在控制台输出学生对象的成员变量值。

\* 两种方式遍历

\* 迭代器

\* 增强for

\*/

public class HashSetTest {

public static void main(String[] args) {

//创建集合对象

HashSet<Student> hs = new HashSet<Student>();

//创建元素对象

Student s1 = new Student("林青霞",30);

Student s2 = new Student("张曼玉",35);

Student s3 = new Student("王祖贤",33);

//把元素添加到集合

hs.add(s1);

hs.add(s2);

hs.add(s3);

//遍历

//迭代器

Iterator<Student> it = hs.iterator();

while(it.hasNext()){

Student s = it.next();

System.out.println(s.getName()+"---"+s.getAge());

}

System.out.println("------------------");

//增强for

for(Student s : hs) {

System.out.println(s.getName()+"---"+s.getAge());

}

}

}

### HashSet集合的练习存储自定义对象保证元素唯一性

HashSet集合存储自定义对象并遍历

要求：如果对象的成员变量值相同，就认为是同一个元素。

提示：自定义一个学生类，给出成员变量name和age。遍历集合的时候，在控制台输出学生对象的成员变量值。

两种方式遍历

迭代器

增强for

因为我们存储的元素所属的类没有重写hashCode()和equals()方法，所以保证不了元素的唯一性。

而我想保证，怎么办呢?重写这两个方法就可以了。

如何重写呢?自动生成就可以了。

#### 案例代码五:

**package** com.itheima\_04;

**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** **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**;

}

}

#### 案例代码六:

**package** com.itheima\_04;

**import** java.util.HashSet;

/\*

\* HashSet集合存储自定义对象并遍历

\* 要求：如果对象的成员变量值相同，就认为是同一个元素。

\* 提示：自定义一个学生类，给出成员变量name和age。遍历集合的时候，在控制台输出学生对象的成员变量值。

\* 两种方式遍历

\* 迭代器

\* 增强for

\*

\* 因为我们存储的元素所属的类没有重写hashCode()和equals()方法，所以保证不了元素的唯一性。

\* 而我想保证，怎么办呢?重写这两个方法就可以了。

\* 如何重写呢?自动生成就可以了。

\*/

**public** **class** HashSetTest {

**public** **static** **void** main(String[] args) {

//创建集合对象

HashSet<Student> hs = **new** HashSet<Student>();

//创建元素对象

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("林青霞",30);

Student s5 = **new** Student("张曼玉",35);

//把元素添加到集合

hs.add(s1);

hs.add(s2);

hs.add(s3);

hs.add(s4);

hs.add(s5);

//遍历集合

//增强for

**for**(Student s : hs) {

System.***out***.println(s.getName()+"---"+s.getAge());

}

}

}

# Map集合

Map:

将键映射到值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。

## 创建Map集合对象并添加元素

Map:

将键映射到值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。

举例：学生的学号和姓名

it001 林青霞

it002 张曼玉

it003 王祖贤

public interface Map<K,V>

### 案例代码六:

package com.itheima\_01;

import java.util.HashMap;

import java.util.Map;

/\*

\* Map:

\* 将键映射到值的对象。一个映射不能包含重复的键；每个键最多只能映射到一个值。

\*

\* 举例：学生的学号和姓名

\* it001 林青霞

\* it002 张曼玉

\* it003 王祖贤

\*

\* public interface Map<K,V>

\*

\*/

public class MapDemo {

public static void main(String[] args) {

//创建集合对象

Map<String,String> map = new HashMap<String,String>();

//添加元素

//put(K key,V value):添加元素。

map.put("it001", "林青霞");

map.put("it002", "张曼玉");

map.put("it003", "王祖贤");

//输出集合对象

System.out.println(map);

}

}

## Map集合的成员方法

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()：返回集合中的键值对的对数

### 案例代码七:

package com.itheima\_01;

import java.util.HashMap;

import java.util.Map;

/\*

\* 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()：返回集合中的键值对的对数

\*

\* Map集合中的实现类的数据结构只针对键有效。

\*/

public class MapDemo2 {

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

//如果键是第一次存储，就直接存储元素，返回null

//如果键不是第一次存储，就用值把以前的值替换，返回以前的值

// System.out.println("put:"+map.put("张无忌", "周芷若"));

// System.out.println("put:"+map.put("张无忌", "赵敏"));

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("remove:"+map.remove("郭靖"));

// System.out.println("remove:"+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("containsKey:"+map.containsKey("郭靖"));

// System.out.println("containsKey:"+map.containsKey("郭襄"));

//boolean containsValue(Object value):判断集合是否包含指定的值 自己练习

//boolean isEmpty()：判断集合是否为空

//System.out.println("isEmpty:"+map.isEmpty());

//int size()：返回集合中的键值对的对数

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

//输出集合对象

System.out.println(map);

}

}

## Map集合的获取功能测试

V get(Object key):根据键获取值

Set<K> keySet():获取所有键的集合

Collection<V> values():获取所有值的集合

### 案例代码八:

package com.itheima\_01;

import java.util.Collection;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

/\*

\* V get(Object key):根据键获取值

\* Set<K> keySet():获取所有键的集合

\* Collection<V> values():获取所有值的集合

\*/

public class MapDemo3 {

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("郭靖", "黄蓉");

map.put("杨过", "小龙女");

map.put("张无忌", "赵敏");

//V get(Object key):根据键获取值

System.out.println("get:"+map.get("张无忌"));

System.out.println("get:"+map.get("张三丰"));

System.out.println("--------------------");

//Set<K> keySet():获取所有键的集合

Set<String> set = map.keySet();

for(String key : set) {

System.out.println(key);

}

System.out.println("--------------------");

//Collection<V> values():获取所有值的集合

Collection<String> values = map.values();

for(String value : values) {

System.out.println(value);

}

}

}

## Map集合的遍历之键找值

Map集合的遍历

思路：Map看成是一个夫妻对的集合

A:把所有的丈夫给集中起来

B:遍历丈夫的集合，获取到每一个丈夫

C:根据丈夫去找对应的妻子

转换：

A:获取所有键的集合

B:遍历键的集合，获取到每一个键

C:根据键去找值

### 案例代码九:

package com.itheima\_02;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

/\*

\* Map集合的遍历

\*

\* 思路：Map看成是一个夫妻对的集合

\* A:把所有的丈夫给集中起来

\* B:遍历丈夫的集合，获取到每一个丈夫

\* C:根据丈夫去找对应的妻子

\*

\* 转换：

\* A:获取所有键的集合

\* B:遍历键的集合，获取到每一个键

\* C:根据键去找值

\*/

public class MapDemo {

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("郭靖","黄蓉");

map.put("杨过","小龙女");

map.put("张无忌","赵敏");

//获取所有键的集合

Set<String> set = map.keySet();

//遍历键的集合，获取到每一个键

for(String key : set) {

//根据键去找值

String value = map.get(key);

System.out.println(key+"---"+value);

}

}

}

## Map集合的遍历之键值对对象找键和值

Map集合遍历的方式2

思路：

A:获取所有结婚证的集合

B:遍历结婚证的集合，得到每一个结婚证

C:根据结婚证获取丈夫和妻子

转换：

A:获取所有键值对对象的集合

B:遍历键值对对象的集合，得到每一个键值对对象

C:根据键值对对象获取键和值

### 案例代码十:

package com.itheima\_02;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

/\*

\* Map集合遍历的方式2

\*

\* 思路：

\* A:获取所有结婚证的集合

\* B:遍历结婚证的集合，得到每一个结婚证

\* C:根据结婚证获取丈夫和妻子

\*

\* 转换：

\* A:获取所有键值对对象的集合

\* B:遍历键值对对象的集合，得到每一个键值对对象

\* C:根据键值对对象获取键和值

\*/

public class MapDemo2 {

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("郭靖","黄蓉");

map.put("杨过","小龙女");

map.put("张无忌","赵敏");

//获取所有键值对对象的集合

// Set<Map.Entry<K,V>> entrySet()

// 获取键值对对象的集合

Set<Map.Entry<String,String>> set = 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 : set) {

//根据键值对对象获取键和值

String key = me.getKey();

String value = me.getValue();

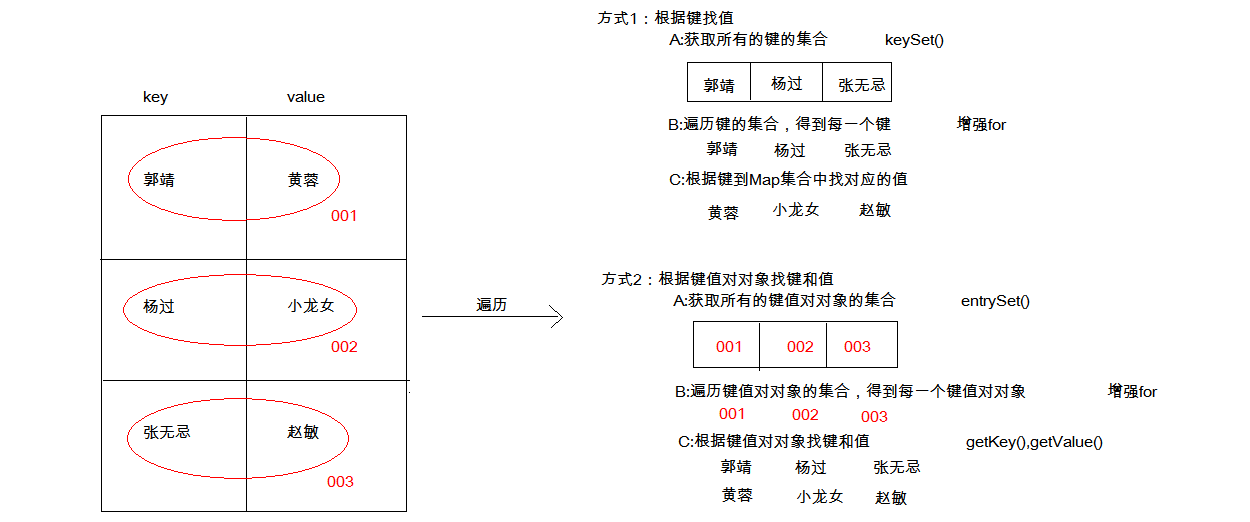
System.out.println(key+"---"+value);

}

}

}

## Map集合的两种遍历方式图解



# HashMap集合的相关案例

## HashMap集合的练习键是String值是Student

练习1：两种方式遍历

HashMap<String,Student>

键：String 学号

值：Student 学生对象

### 案例代码十一:

**package** com.itheima\_03;

**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;

}

}

### 案例代码十二:

package com.itheima\_03;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

/\*

\* 练习1：两种方式遍历

\* HashMap<String,Student>

\* 键：String 学号

\* 值：Student 学生对象

\*/

public class HashMapTest {

public static void main(String[] args) {

// 创建集合对象

HashMap<String, Student> hm = new HashMap<String, Student>();

// 创建元素对象

Student s1 = new Student("林青霞", 30);

Student s2 = new Student("张曼玉", 35);

Student s3 = new Student("王祖贤", 33);

// 添加元素到集合中

hm.put("it001", s1);

hm.put("it002", s2);

hm.put("it003", s3);

// 遍历

// 根据键找值

Set<String> set = hm.keySet();

for (String key : set) {

Student value = hm.get(key);

System.out.println(key + "---" + value.getName() + "---" + value.getAge());

}

System.out.println("---------------------");

// 根据键值对对象找键和值

Set<Map.Entry<String, Student>> set2 = hm.entrySet();

for (Map.Entry<String, Student> me : set2) {

String key = me.getKey();

Student value = me.getValue();

System.out.println(key + "---" + value.getName() + "---" + value.getAge());

}

}

}

## HashMap集合的练习键是Student值是String

练习2：任选一种方式遍历

HashMap<Student,String>

键：Student 学生对象

值：String 学生住址

要求：如果学生对象的成员变量值相同，就说明是同一个键。

### 案例代码十三:

**package** com.itheima\_04;

**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** **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**;

}

}

### 案例代码十四:

package com.itheima\_04;

import java.util.HashMap;

import java.util.Set;

/\*

\* 练习2：任选一种方式遍历

\* HashMap<Student,String>

\* 键：Student 学生对象

\* 值：String 学生住址

\* 要求：如果学生对象的成员变量值相同，就说明是同一个键。

\*/

public class HashMapTest {

public static void main(String[] args) {

//创建集合对象

HashMap<Student,String> hm = new HashMap<Student,String>();

//创建元素对象

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("张曼玉",35);

//添加元素到集合中

hm.put(s1, "北京");

hm.put(s2, "上海");

hm.put(s3, "西安");

hm.put(s4, "香港");

//键找值

Set<Student> set = hm.keySet();

for(Student key : set) {

String value = hm.get(key);

System.out.println(key.getName()+"---"+key.getAge()+"---"+value);

}

}

}

## 集合的嵌套练习之ArrayList嵌套HashMap

需求：ArrayList集合嵌套HashMap集合并遍历。

定义一个ArrayList集合，它包含三个元素，每一个元素都是HashMap类型的。

每一个HashMap集合的键和值都是String类型的，

键：String 丈夫的姓名

值：String 妻子的姓名

给出如下的字符串数据，请用代码实现需求。

第一个HashMap集合的元素：

孙策 大乔

周瑜 小乔

第二个HashMap集合的元素：

郭靖 黄蓉

杨过 小龙女

第三个HashMap集合的元素：

令狐冲 任盈盈

林平之 岳灵珊

### 案例代码十五:

package com.itheima;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Set;

/\*

\* 需求：ArrayList集合嵌套HashMap集合并遍历。

\* 定义一个ArrayList集合，它包含三个元素，每一个元素都是HashMap类型的。

\* 每一个HashMap集合的键和值都是String类型的，

\* 键：String 丈夫的姓名

\* 值：String 妻子的姓名

\* 给出如下的字符串数据，请用代码实现需求。

\* 第一个HashMap集合的元素：

\* 孙策 大乔

\* 周瑜 小乔

\* 第二个HashMap集合的元素：

\* 郭靖 黄蓉

\* 杨过 小龙女

\* 第三个HashMap集合的元素：

\* 令狐冲 任盈盈

\* 林平之 岳灵珊

\*/

public class ArrayListIncludeHashMapTest {

public static void main(String[] args) {

//创建集合对象

ArrayList<HashMap<String,String>> array = new ArrayList<HashMap<String,String>>();

//创建元素1

HashMap<String,String> hm1 = new HashMap<String,String>();

hm1.put("孙策","大乔");

hm1.put("周瑜","小乔");

//把元素添加到array中

array.add(hm1);

//创建元素2

HashMap<String,String> hm2 = new HashMap<String,String>();

hm2.put("郭靖","黄蓉");

hm2.put("杨过","小龙女");

//把元素添加到array中

array.add(hm2);

//创建元素3

HashMap<String,String> hm3 = new HashMap<String,String>();

hm3.put("令狐冲","任盈盈");

hm3.put("林平之","岳灵珊");

//把元素添加到array中

array.add(hm3);

//遍历ArrayList集合

for(HashMap<String,String> hm : array) {

Set<String> set = hm.keySet();

for(String key : set) {

String value = hm.get(key);

System.out.println(key+"---"+value);

}

System.out.println("-----------------");

}

}

}

## 集合的嵌套练习之HashMap嵌套ArrayList

需求：HashMap集合嵌套ArrayList集合并遍历。

定义一个HashMap集合，它包含三个元素，每一个元素的键是String类型，值是ArrayList类型。

键：String 人物来自哪部电视剧

值：ArrayList 人物的名称

每一个ArrayList集合的数据是String类型的。

给出如下的字符串数据，请用代码实现需求。

第一个ArrayList集合的元素：(三国演义)

诸葛亮

赵云

第二个ArrayList集合的元素：(西游记)

唐僧

孙悟空

第三个ArrayList集合的元素：(水浒传)

武松

鲁智深

### 案例代码十六:

package com.itheima;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Set;

/\*

\* 需求：HashMap集合嵌套ArrayList集合并遍历。

\* 定义一个HashMap集合，它包含三个元素，每一个元素的键是String类型，值是ArrayList类型。

\* 键：String 人物来自哪部电视剧

\* 值：ArrayList 人物的名称

\* 每一个ArrayList集合的数据是String类型的。

\* 给出如下的字符串数据，请用代码实现需求。

\* 第一个ArrayList集合的元素：(三国演义)

\* 诸葛亮

\* 赵云

\* 第二个ArrayList集合的元素：(西游记)

\* 唐僧

\* 孙悟空

\* 第三个ArrayList集合的元素：(水浒传)

\* 武松

\* 鲁智深

\*/

public class HashMapIncludeArrayListTest {

public static void main(String[] args) {

//创建集合对象

HashMap<String,ArrayList<String>> hm = new HashMap<String,ArrayList<String>>();

//创建元素1

ArrayList<String> sgyy = new ArrayList<String>();

sgyy.add("诸葛亮");

sgyy.add("赵云");

//把元素添加到hm中

hm.put("三国演义", sgyy);

//创建元素2

ArrayList<String> xyj = new ArrayList<String>();

xyj.add("唐僧");

xyj.add("孙悟空");

//把元素添加到hm中

hm.put("西游记", xyj);

//创建元素3

ArrayList<String> shz = new ArrayList<String>();

shz.add("武松");

shz.add("鲁智深");

//把元素添加到hm中

hm.put("水浒传", shz);

//遍历集合

Set<String> set = hm.keySet();

for(String key : set) {

System.out.println(key);

ArrayList<String> value = hm.get(key);

for(String s : value) {

System.out.println("\t"+s);

}

}

}

}