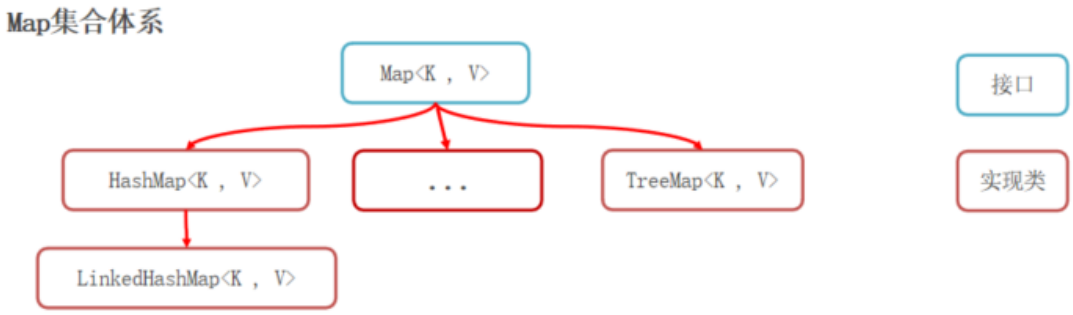
Map

一、`Map`：双列集合

1.简介：

- `Map`是双列集合的祖宗，每个元素包含一个键值对，因此也称为键值对集合
- 格式：`{key1=value1,key2=value2,...}`
- `Map`的键不允许重复，但值允许重复
- 键和值一一对应，每个键只能找到自己对应的值

2.体系结构：



3.常用方法：

(1) `V put(K key,V value)` 添加键值对

(2) `int size()` 获取`Map`集合的大小

(3) `void clear()` 清空`Map`集合

(4) `boolean isEmpty()` 判断`Map`集合是否为空

(5) `V get(Object key)` 根据键获取对应的值

(6) `V remove(Object key)` 根据键删除对应的值

(7) `boolean containsKey(Object key)` 判断`Map`集合是否包含某个键

(8) `boolean containsValue(Object value)` 判断`Map`集合是否包含某个值

(9) `Set<K> keySet()` 获取全部键组成的`Set`集合

(10) `Collection<V> values()` 获取全部值组成的`Collection`集合

4.遍历方式

先准备一个`Map`集合：

```
Map<String,Double> map=new HashMap<>();
map.put("zsh",173.5);
map.put("zjl",179.3);
map.put("zxj",165.2);
map.put("ym",235.0);
//map = {ym=235.0, zjl=179.3, zxj=165.2, zsh=173.5}
```

`I`.键找值：先获取集合全部键，再通过遍历键来获取全部值

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

`II`.键值对：把键值对看作一个整体，进行遍历

```
Set<Map.Entry<String,Double>> entries=map.entrySet();
for (Map.Entry<String, Double> entry : entries) {
    String key=entry.getKey();
    double value=entry.getValue();
    System.out.println(key+"-->"+value);
}
```

`III`.`Lambda`表达式：（**最方便，推荐使用！**）

```
map.forEach((key,value) -> {
    System.out.println(key+"-->"+value);
});
```

二、`HashMap`：由键决定特点--无序、不重复、无索引

1.底层原理：

与`HashSet`底层原理一样，都是基于**哈希表**实现

实际上，之前学习的`Set`系列集合的底层都是基于`Map`实现，`Set`实际上就是特殊的`Map`，只不过少了键所对应的值而已

```
//HashSet的无参构造器
public HashSet(){
    map=new HashMap<>();
}
```

2.特点：

- `HashMap`的键依赖`hashCode()`和`equals()`方法来保证**键的唯一**
- 若键存储的是自定义类型的对象，则可以通过重写`hashCode()`和`equals()`方法来实现**不重复**原则

三、`LinkedHashMap`：由键决定特点--有序、不重复、无索引

底层原理：

与`LinkedHashSet`底层原理一样，都是基于**哈希表**和**双链表**实现

四、`TreeMap`：由键决定特点--按照大小默认升序排序、不重复、无索引

1.底层原理：

与`TreeSet`底层原理一样，都是基于**红黑树**实现的排序

2.排序规则：

同样也支持2种方式来指定排序规则：

- 让自定义的类实现`Comparable`接口，重写排序规则
- `TreeMap`有一个有参构造器，支持创建`Comparator`比较器对象，用于指定排序规则