集合进阶（二）

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

可变参数

- 就是一种特殊形参，定义在方法、构造器的形参列表里，格式是：数据类型...参数名称；

可变参数的特点和好处

- 特点：可以不传数据给它；可以传一个或者同时传多个数据给它；也可以传一个数组给它。
- 好处：常常用来灵活的接收数据。

可变参数的注意事项：

- **可变参数在方法内部就是一个数组。**
- 一个形参列表中可变参数只能有一个
- 可变参数必须放在形参列表的最后面

假如需要定义一个方法求和，该方法可以灵活的完成如下需求：

- 计算1个数据的和。
- 计算2个数据的和。
- 计算3个数据的和。
- 计算n个数据的和，甚至可以支持不接收参数进行调用。

目录

Contents

Collections

- 是一个用来操作集合的工具类

Collections提供的常用静态方法

方法名称	说明
public static <T> boolean **addAll(Collection<? super T> c, T... elements)**	给集合批量添加元素
public static void **shuffle(List<?> list)**	打乱List集合中的元素顺序
public static <T> void **sort(List<T> list)**	对List集合中的元素进行升序排序
public static <T> void **sort(List<T> list，Comparator<? super T> c)**	对List集合中元素，按照比较器对象指定的规则进行排序

Collections只能支持对List集合进行排序

排序方式1：

方法名称	说明
public static <T> void sort(List<T> list)	对List集合中元素按照默认规则排序

注意：本方法可以直接对自定义类型的List集合排序，但自定义类型必须实现了Comparable接口，指定了比较规则才可以。

排序方式2：

方法名称	说明
public static <T> void sort(List<T> list，Comparator<? super T> c)	对List集合中元素，按照比较器对象指定的规则进行排序

目录

Contents

案例 斗地主游戏

分析业务需求

- 总共有54张牌
- 点数: "3","4","5","6","7","8","9","10","J","Q","K","A","2“
- 花色: "♠", "♥", "♣", "♦“
- 大小王: "　", "　“
- 斗地主：发出51张牌，剩下3张做为底牌。

分析实现

- 在启动游戏房间的时候，应该提前准备好54张牌
- 接着，需要完成洗牌、发牌、对牌排序、看牌

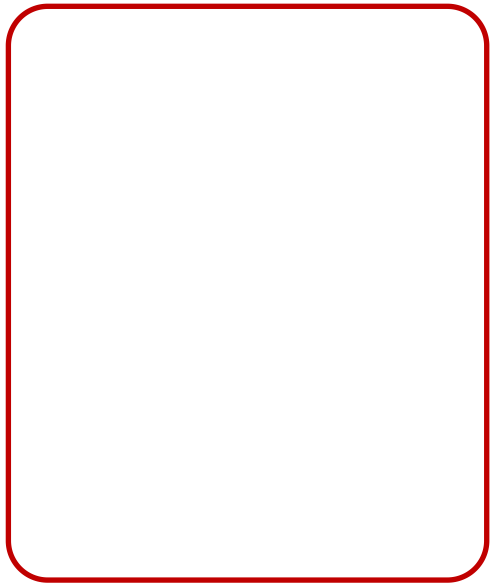
目录

Contents

认识Map集合

Collection

单列集合

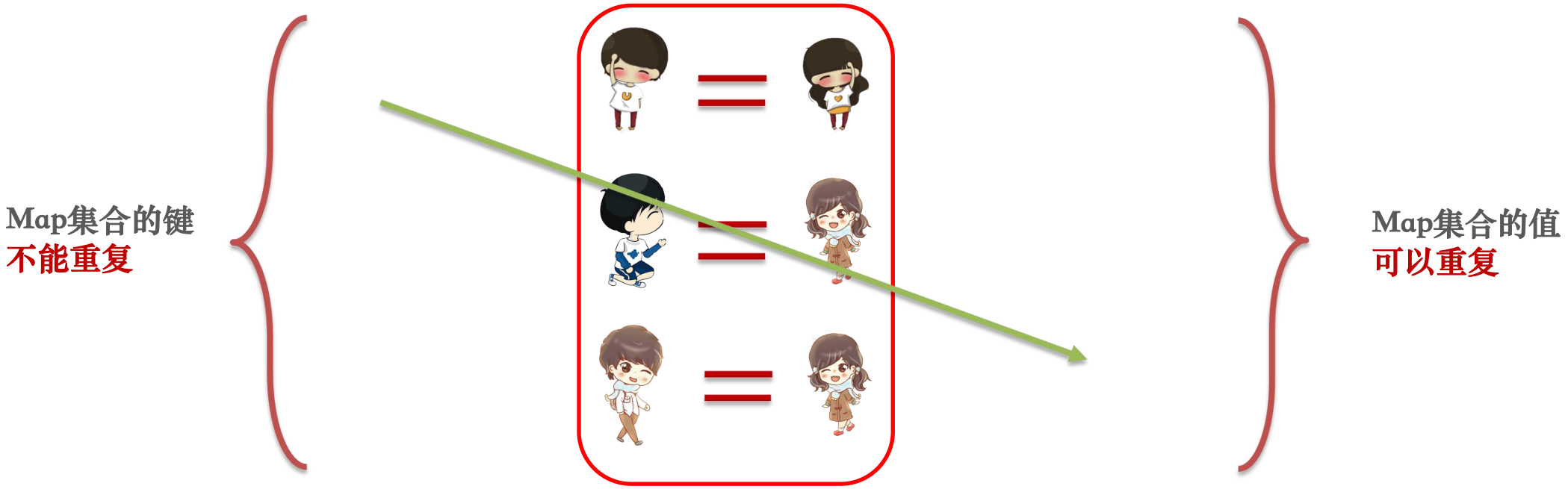


Map

双列集合

认识Map集合

- Map集合称为双列集合，格式：{key1=value1 , key2=value2 , key3=value3 , ...}， 一次需要存一对数据做为一个元素.
- Map集合的每个元素“key=value”称为一个键值对/键值对对象/一个Entry对象，Map集合也被叫做“键值对集合”
- Map集合的所有键是不允许重复的，但值可以重复，键和值是一一对应的，每一个键只能找到自己对应的值

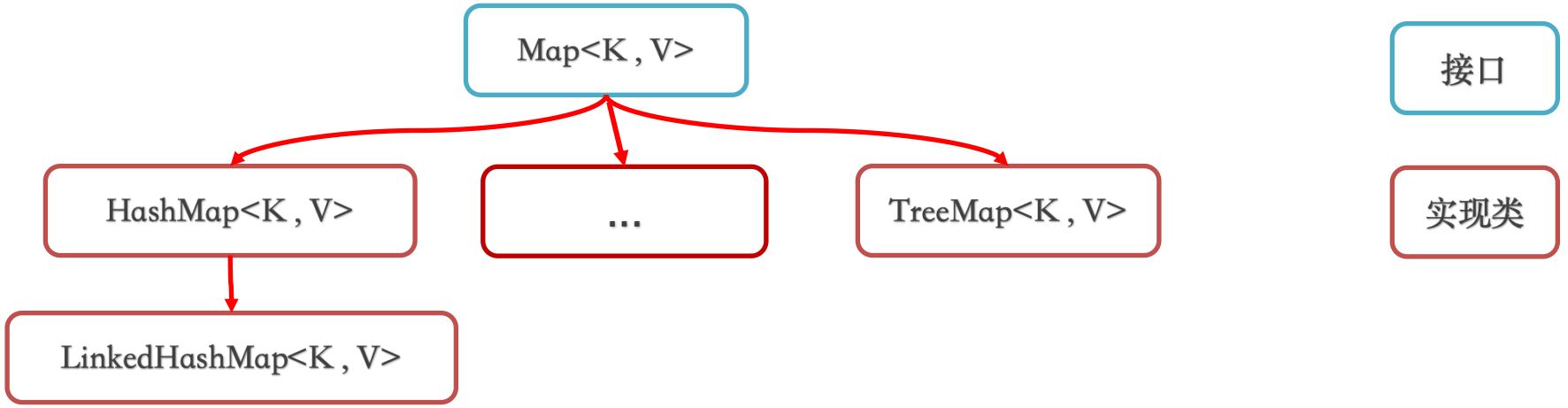


Map集合在什么业务场景下使用

{商品1=2，商品2=3， 商品3 = 2 ，商品4= 3}

需要存储一一对应的数据时，就可以考虑使用Map集合来做

Map集合体系



Map集合体系的特点

注意：Map系列集合的特点都是由键决定的，值只是一个附属品，值是不做要求的

- HashMap（由键决定特点）：无序、不重复、无索引；　（用的最多）
- LinkedHashMap （由键决定特点）:由键决定的特点：有序、不重复、无索引。
- TreeMap （由键决定特点）:按照大小默认升序排序、不重复、无索引。

总结

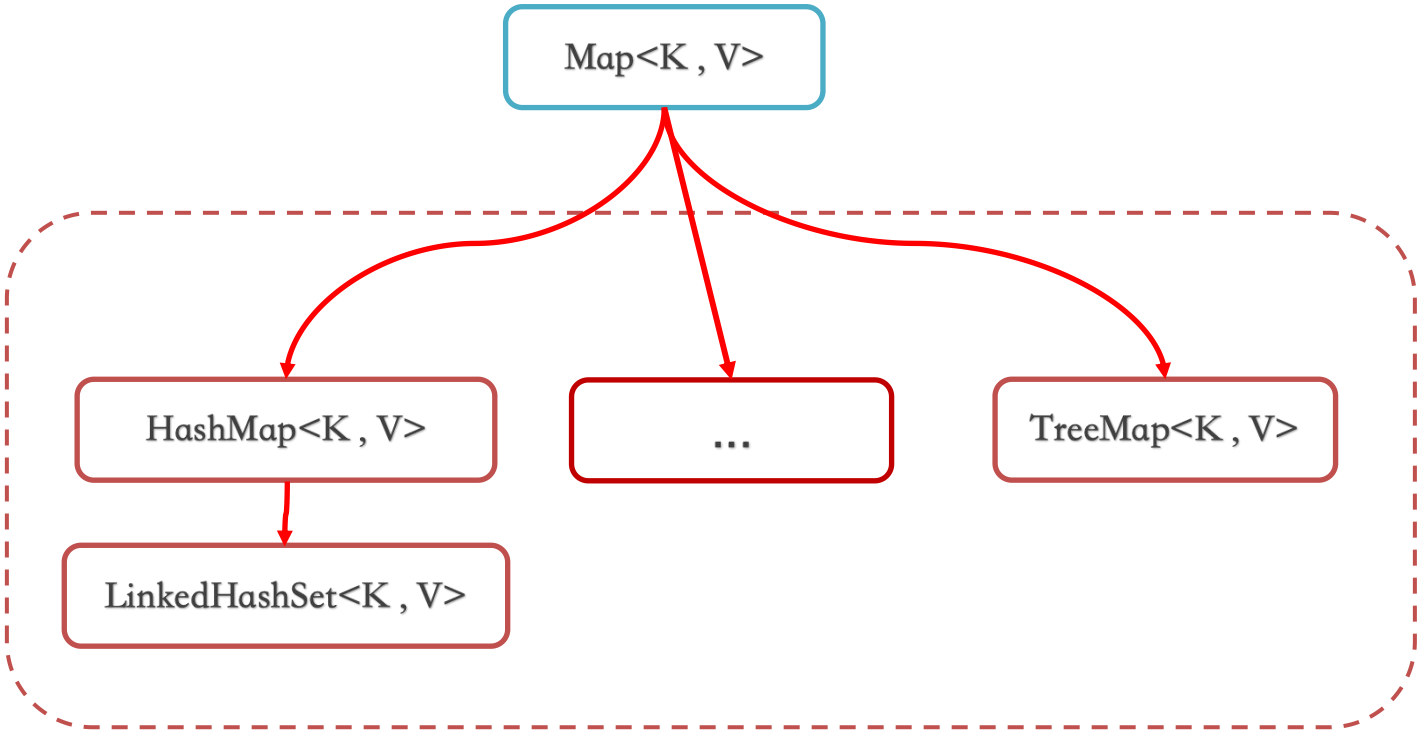
1. Map集合是什么？什么时候可以考虑使用Map集合？
   - Map集合是键值对集合
   - 需要存储一一对应的数据时，就可以考虑使用Map集合来做
2. Map集合的实现类有哪些？各自的特点是？
   - HashMap: 元素按照键是无序，不重复，无索引，值不做要求。
   - LinkedHashMap: 元素按照键是有序，不重复，无索引，值不做要求。
   - TreeMap：元素按照建是排序，不重复，无索引的，值不做要求。

目录

Contents

为什么要先学习Map的常用方法？

- Map是双列集合的祖宗，它的功能是全部双列集合都可以继承过来使用的。



为什么要先学习Map的常用方法？

- Map是双列集合的祖宗，它的功能是全部双列集合都可以继承过来使用的。

Map的常用方法如下：

方法名称	说明
public V put(K key,V value)	添加元素
public int size()	获取集合的大小
public void clear()	清空集合
public boolean isEmpty()	判断集合是否为空，为空返回true，反之
public V get(Object key)	根据键获取对应值
public V remove(Object key)	根据键删除整个元素
public boolean containsKey(Object key)	判断是否包含某个键
public boolean containsValue(Object value)	判断是否包含某个值
public Set<K> keySet()	获取全部键的集合
public Collection<V> values()	获取Map集合的全部值

目录

Contents

Map集合的遍历方式

键找值
01

先获取Map集合全部的键，

再通过遍历键来找值

键值对
02

把“键值对“看成一个整体

进行遍历（难度较大）

Lambda
03

JDK 1.8开始之后的新技

术（非常的简单）

Map集合的遍历方式一

键找值
01

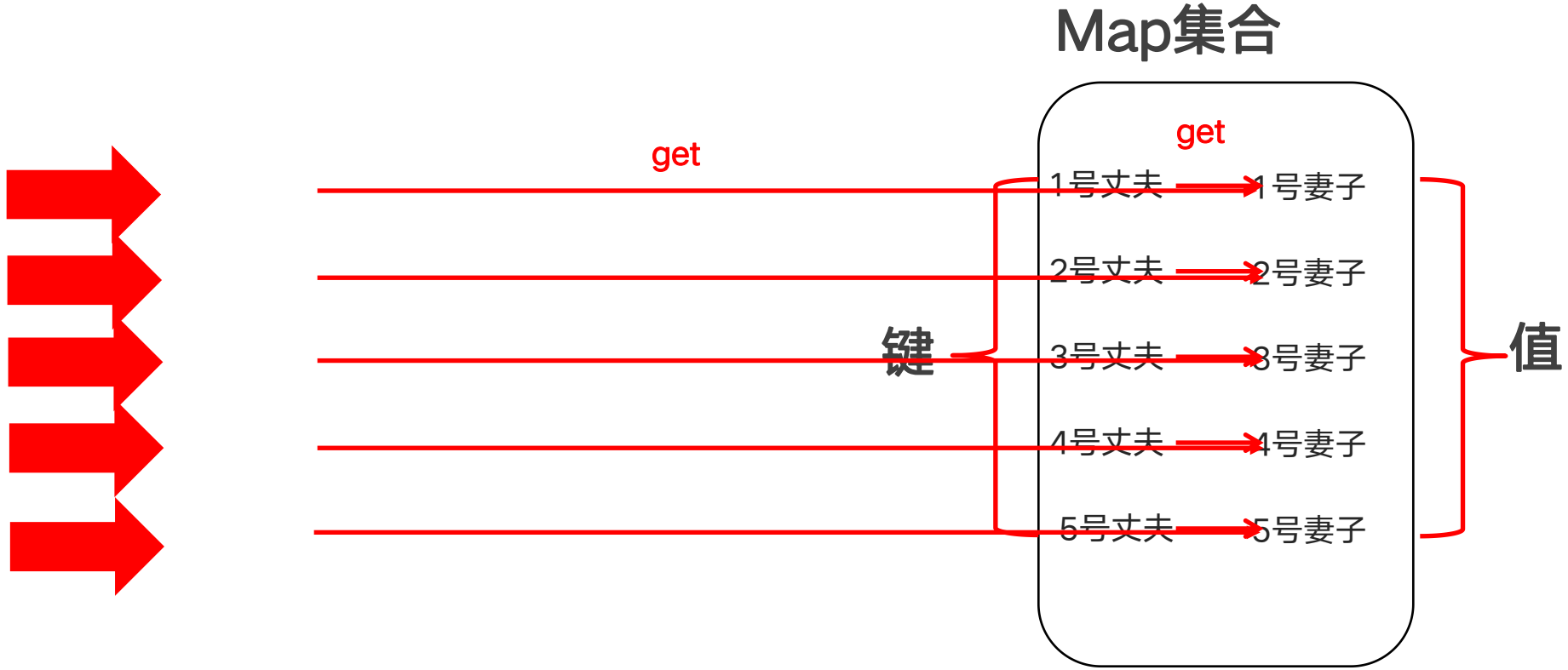
先获取Map集合全部的键，

再通过遍历键来找值

需要用到Map的如下方法：

方法名称	说明
public Set<K> keySet()	获取所有键的集合
public V get(Object key)	根据键获取其对应的值

遍历Map集合方式一：键找值流程



Map集合的遍历方式

键找值
01

先获取Map集合全部的键，

再通过遍历键来找值

键值对
02

把“键值对“看成一个整体

进行遍历（难度较大）

Lambda
03

JDK 1.8开始之后的新技

术（非常的简单）

Map集合的遍历方式二：键值对

键值对
02

把“键值对“看成一个整体

进行遍历（难度较大）

```
Map<String, Double> map = new HashMap<>();
map.put("蜘蛛精", 169.8);
map.put("紫霞", 165.8);
map.put("至尊宝", 169.5);
map.put("牛魔王", 183.6);
System.out.println(map);
// map = {蜘蛛精=169.8, 牛魔王=183.6, 至尊宝=169.5, 紫霞=165.8}
```

```
for (元素类型 变量: 集合){
    ...
}

for (元素类型 kv: map){
    ...
}
```

元素类型无法直接确定！

Map集合的遍历方式二：键值对

```
Map<String, Double> map = new HashMap<>();
map.put("蜘蛛精", 169.8);
map.put("紫霞", 165.8);
map.put("至尊宝", 169.5);
map.put("牛魔王", 183.6);
System.out.println(map);
// map = {蜘蛛精=169.8, 牛魔王=183.6, 至尊宝=169.5, 紫霞=165.8}
```

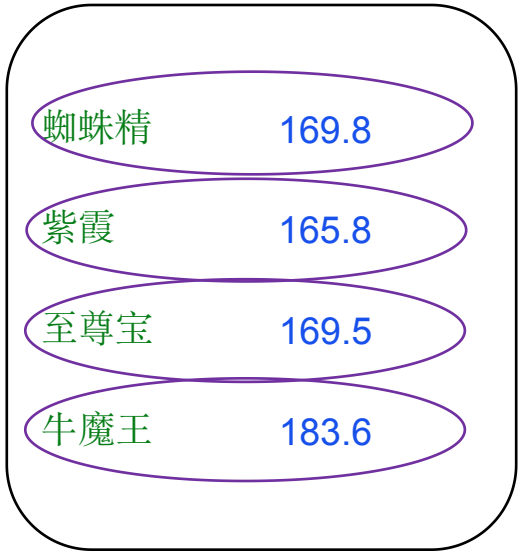
Map提供的方法	说明
Set<Map.Entry<K, V>> entrySet()	获取所有“键值对”的集合

Map.Entry提供的方法	说明
K getKey()	获取键
V getValue()	获取值

```
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);
}
```

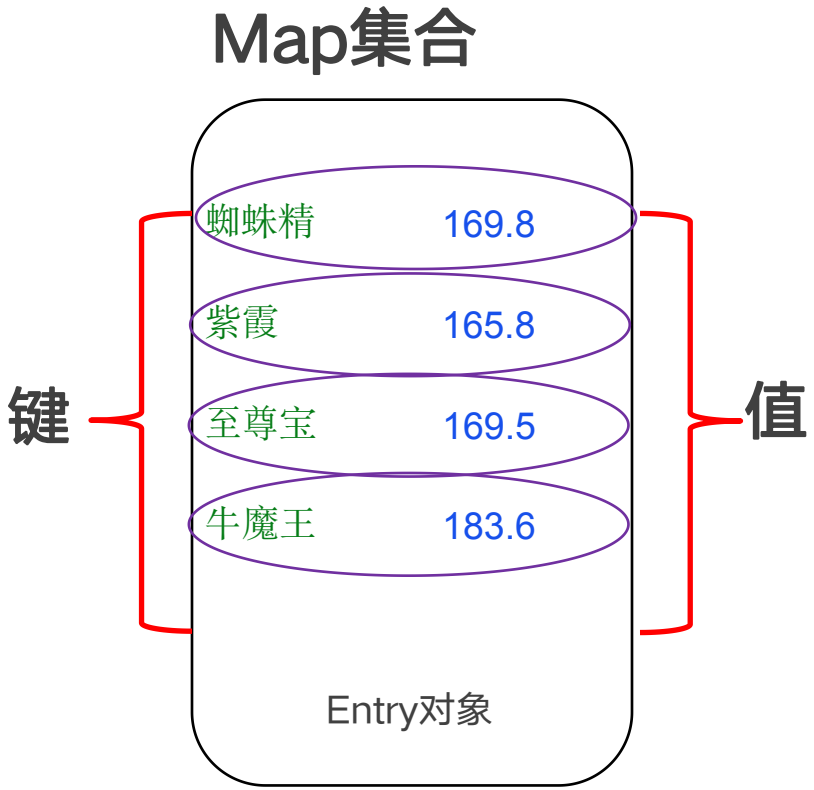
Set< **Map.Entry<String, Double>** >



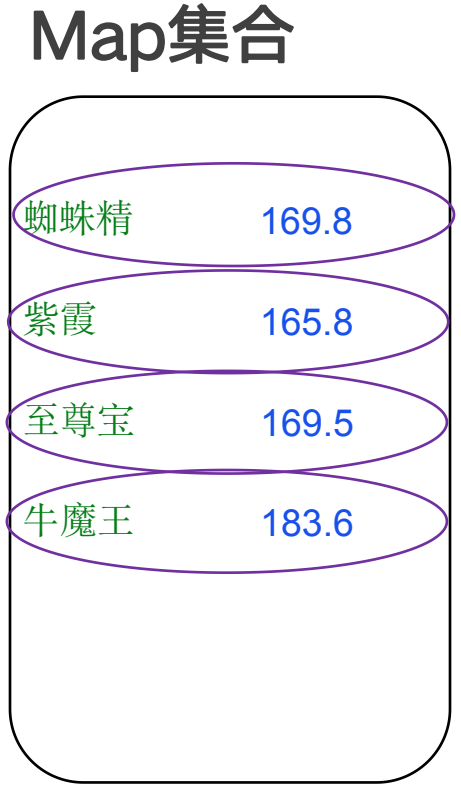
Entry对象

Map.Entry<String, Double>

遍历Map集合方式二：键值对流程



遍历Map集合方式二：键值对流程



Map集合的遍历方式

键找值
01

先获取Map集合全部的键，

再通过遍历键来找值

键值对
02

把“键值对“看成一个整体

进行遍历（难度较大）

Lambda
03

JDK 1.8开始之后的新技

术（非常的简单）

Map集合的遍历方式三：Lambda

- 需要用到Map的如下方法

方法名称	说明
default void forEach(BiConsumer<? super K, ? super V> action)	结合lambda遍历Map集合

流程

map = {蜘蛛精=169.8, 紫霞=165.8, 至尊宝=169.5, 牛魔王=183.6}

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

案例 Map集合的案例-统计投票人数

Map集合的常用方法

Map集合的遍历方式

案例

Map集合的案例-统计投票人数

需求

- 某个班级80名学生，现在需要组织秋游活动，班长提供了四个景点依次是（A、B、C、D）,每个学生只能选择一个景点，请统计出最终哪个景点想去的人数最多。

分析

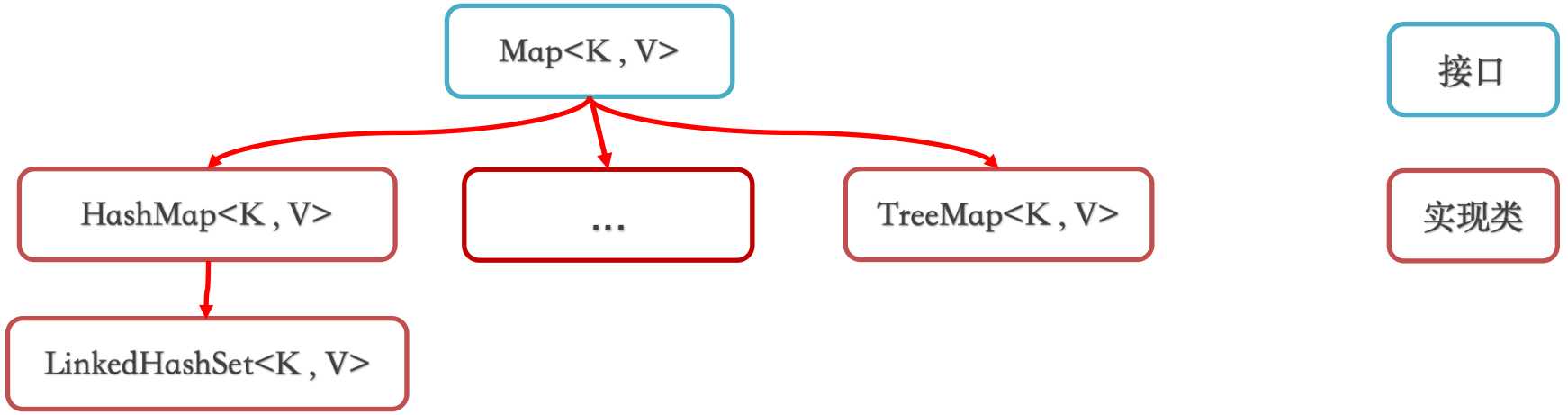
- 将80个学生选择的数据拿到程序中去，[A, A, B , A, B, C, D, ...]
- 准备一个Map集合用于存储统计的结果，Map<String，Integer>，键是景点，值代表投票数量。
- 遍历80个学生选择的景点，每遍历一个景点，就看Map集合中是否存在该景点，不存在存入“景点=1“，存在则其对应值+1,

需要存储一一对应的数据时，就可以考虑使用Map集合来做

目录

Contents

Map集合体系



Map集合体系的特点

- HashMap（由键决定特点）：无序、不重复、无索引；　（用的最多）
- LinkedHashMap （由键决定特点）:由键决定的特点：有序、不重复、无索引。
- TreeMap （由键决定特点）:按照大小默认升序排序、不重复、无索引。

HashMap集合的底层原理

- HashMap跟HashSet的底层原理是一模一样的，都是基于哈希表实现的。

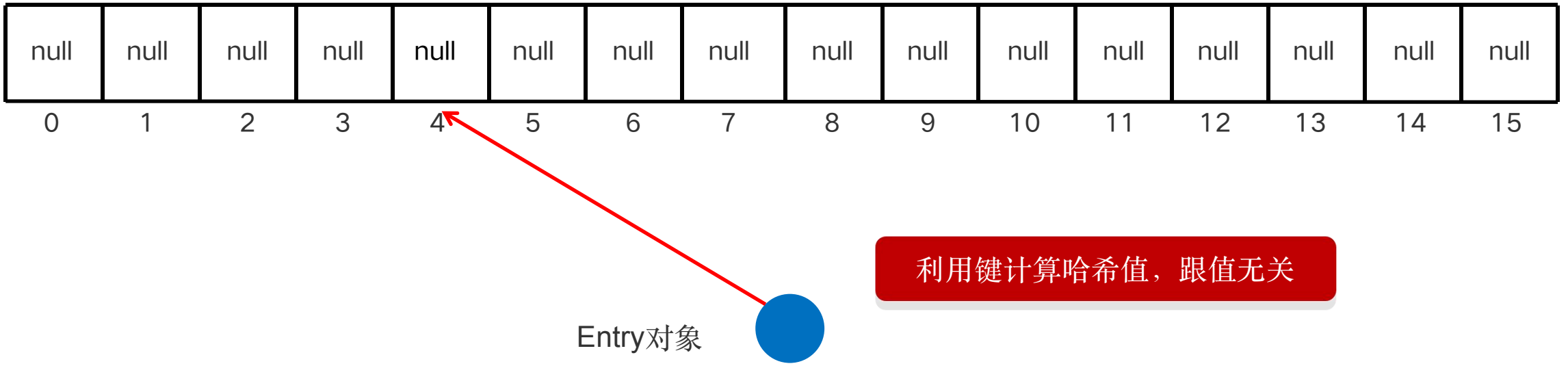
实际上：原来学的Set系列集合的底层就是基于Map实现的，只是Set集合中的元素只要键数据，不要值数据而已。

```
public HashSet() {
    map = new HashMap<>();
}
```

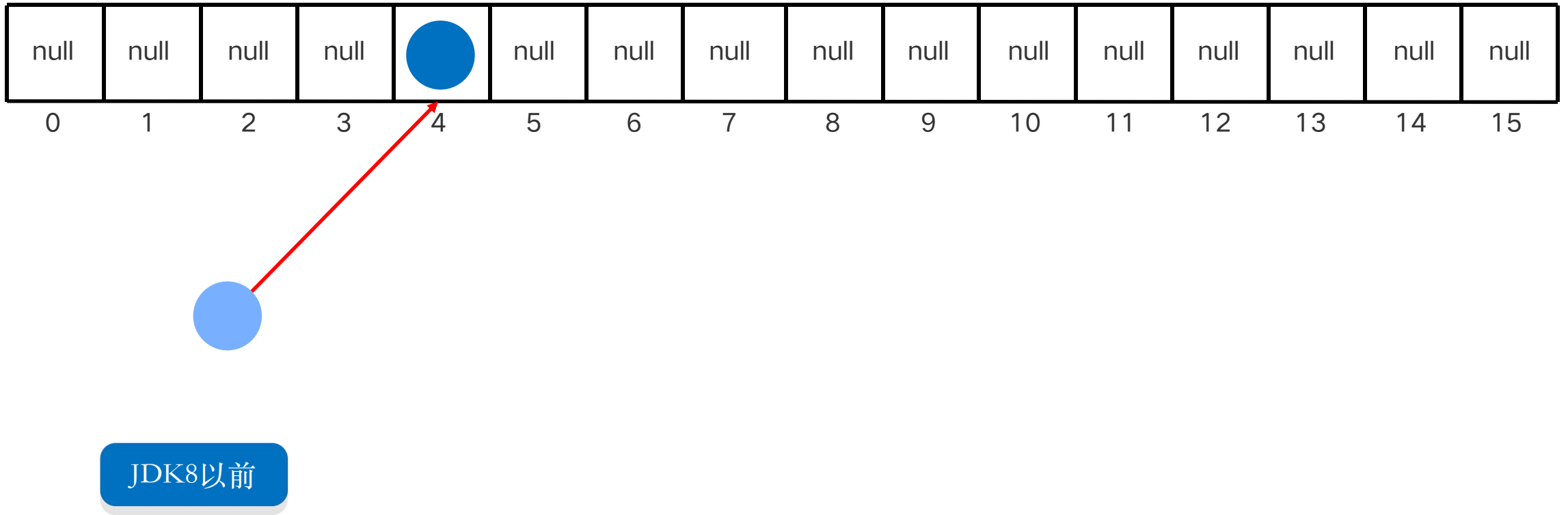
哈希表

- JDK8之前，哈希表 = **数组+链表**
- JDK8开始，哈希表 = **数组+链表+红黑树**
- 哈希表是一种增删改查数据，性能都较好的数据结构。

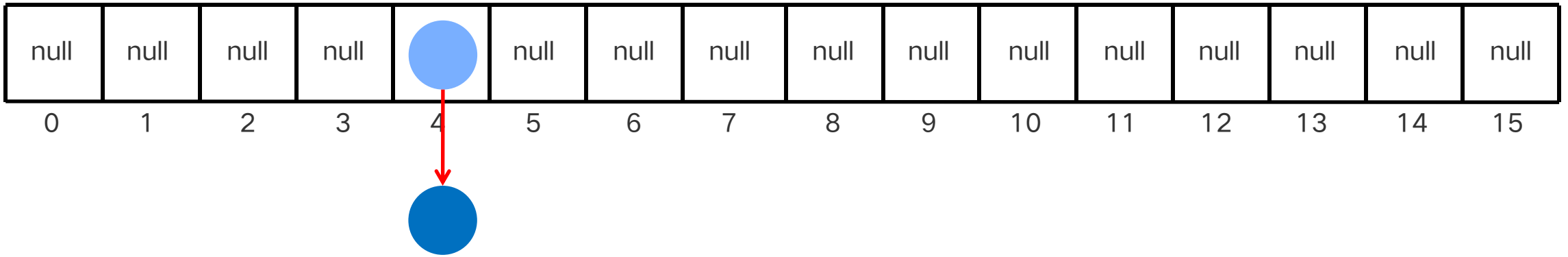
HashMap 的底层原理



HashMap 的底层原理

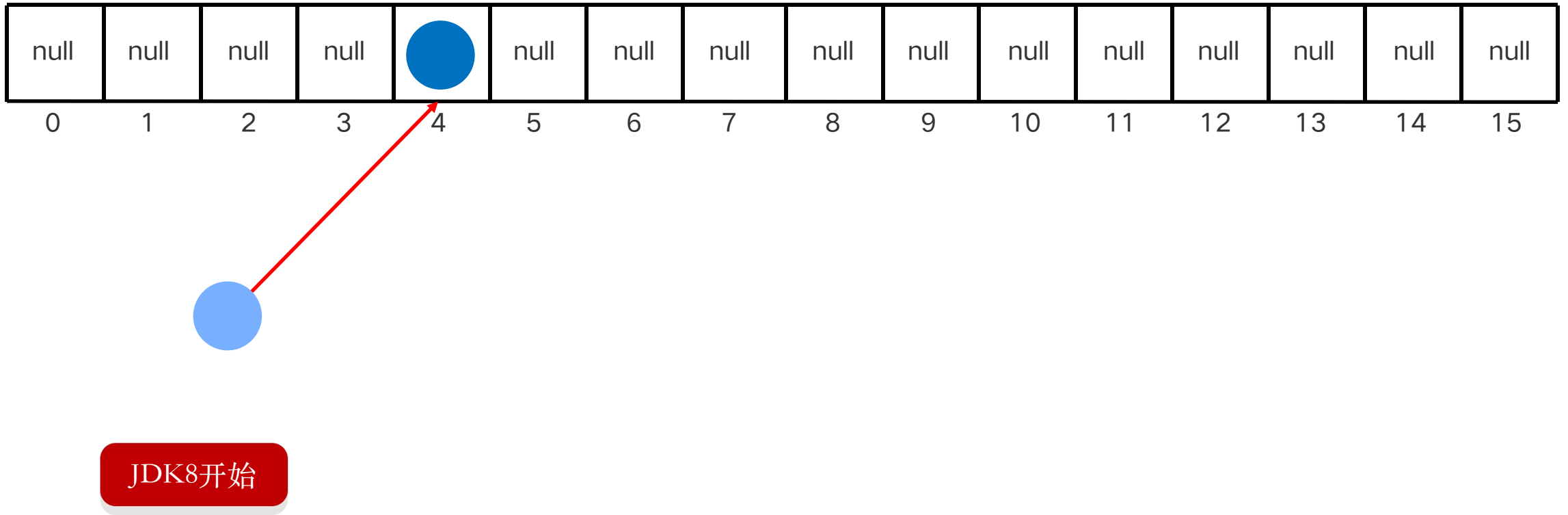


HashMap 的底层原理

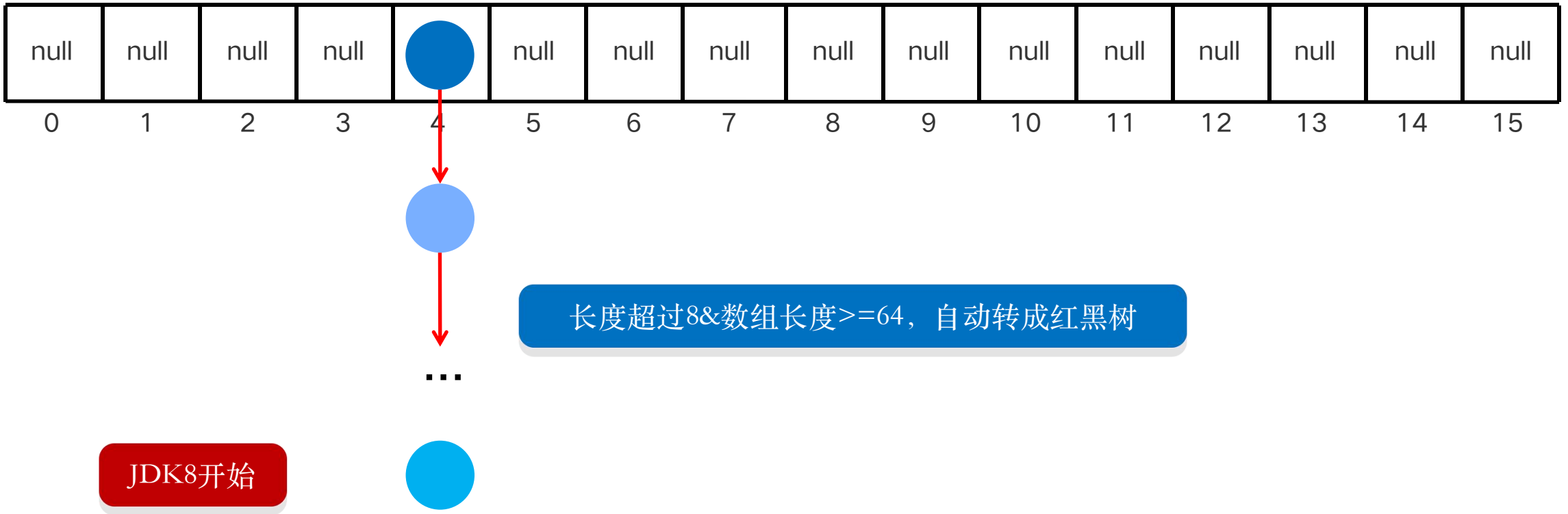


JDK8以前

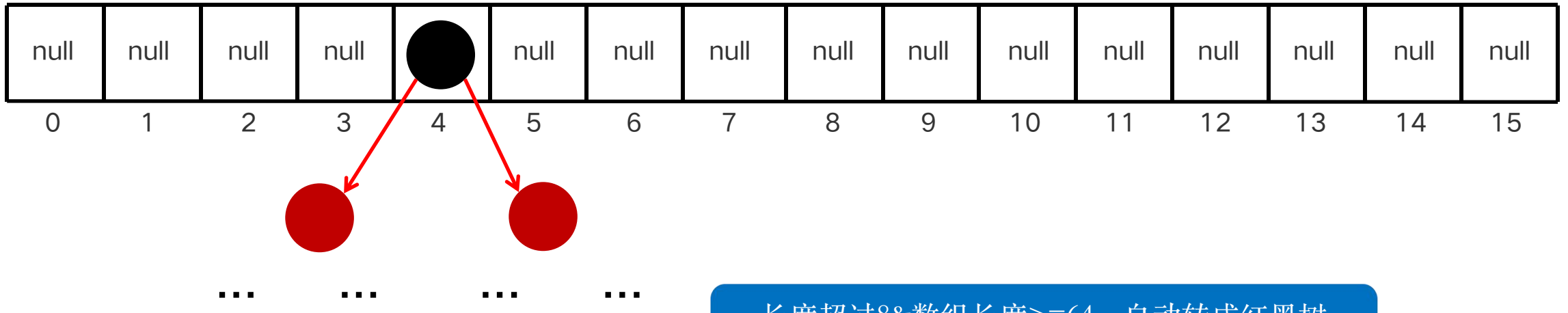
HashMap 的底层原理



HashMap 的底层原理



HashMap 的底层原理

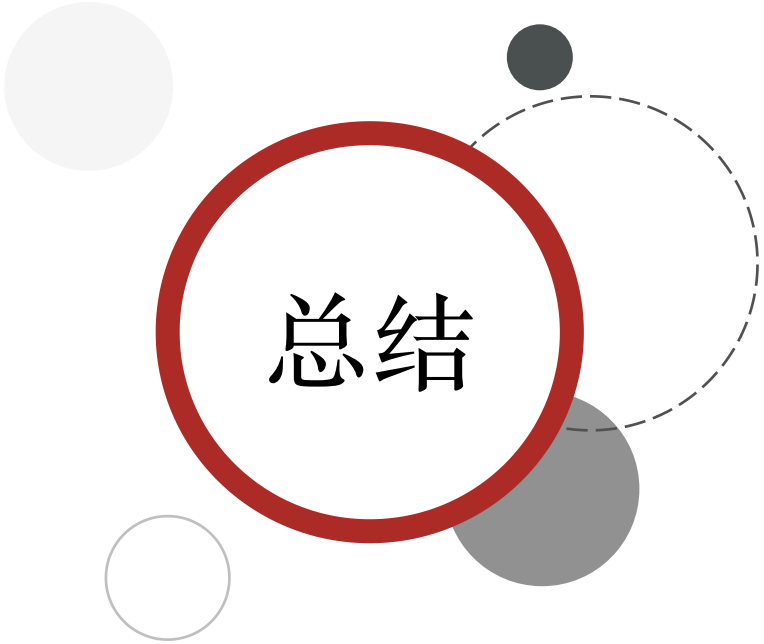


长度超过8&数组长度>=64，自动转成红黑树

JDK8开始

HashMap底层是基于哈希表实现的

- HashMap集合是一种增删改查数据，性能都较好的集合
- 但是它是无序，不能重复，没有索引支持的（由键决定特点）
- HashMap的键依赖hashCode方法和equals方法保证**键的唯一**
- 如果键存储的是自定义类型的对象，可以通过重写hashCode和equals方法，这样可以保证多个对象内容一样时，HashMap集合就能认为是重复的。



1、HashMap的特点和底层原理？

- **由键决定：无序、不重复、无索引。HashMap底层是哈希表结构的。**
- **基于哈希表。增删改查的性能都较好。**

2、HashMap如何实现键的唯一性的？

- **依赖hashCode方法和equals方法保证键的唯一。**
- **如果键要存储的是自定义对象，需要重写hashCode和equals方法。**

案例：HashMap集合存储自定义对象并遍历

需求：创建一个HashMap集合，键是学生对象(Student)，值是籍贯(String)。存储三个键值对元素，并遍历

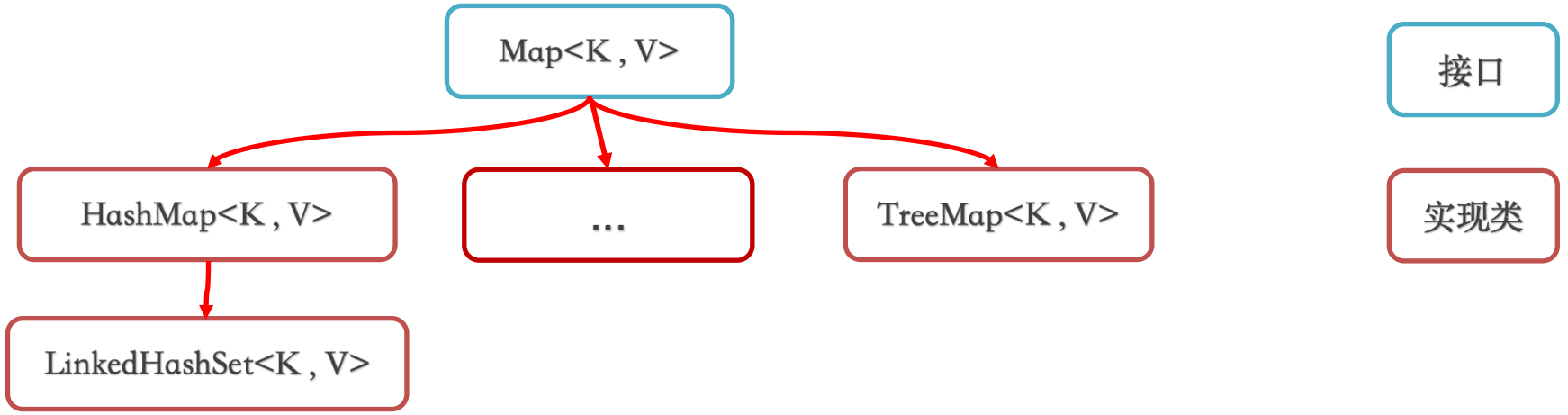
思路：

① 定义学生类
② 创建HashMap集合对象
③ 创建学生对象
④ 把学生添加到集合
⑤ 遍历集合

目录

Contents

Map集合体系



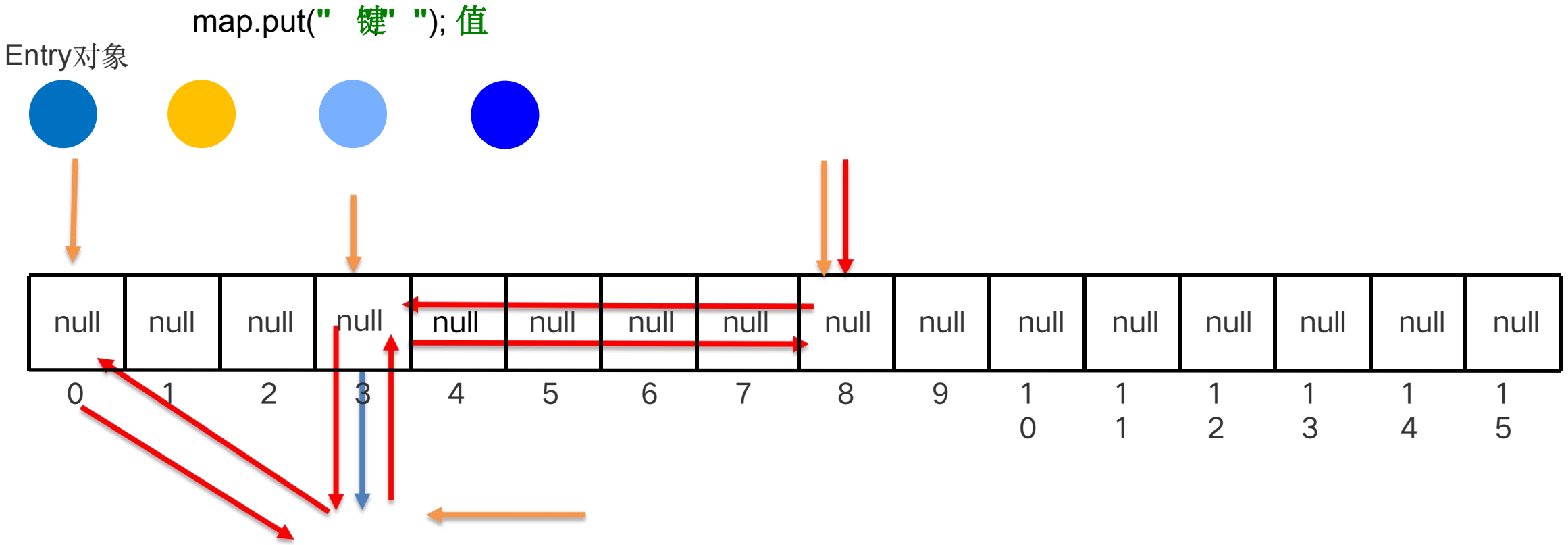
Map集合体系的特点

- HashMap（由键决定特点）：无序、不重复、无索引；　（用的最多）
- LinkedHashMap （由键决定特点）：有序、不重复、无索引。
- TreeMap （由键决定特点）:按照键的大小默认升序排序、不重复、无索引。

LinkedHashMap集合的原理

- 底层数据结构依然是基于哈希表实现的，只是每个键值对元素又额外的多了一个双链表的机制记录元素顺序(保证有序)。

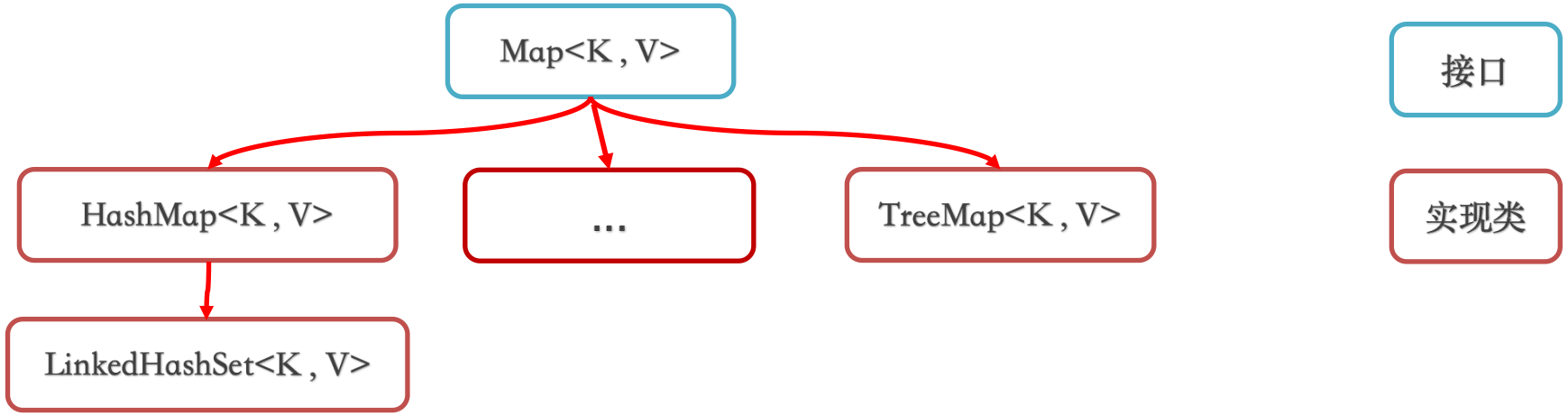
实际上：原来学习的LinkedHashSet集合的底层原理就是LinkedHashMap。



目录

Contents

Map集合体系



Map集合体系的特点

- HashMap（由键决定特点）: 无序、不重复、无索引； （用的最多）
- LinkedHashMap （由键决定特点）:由键决定的特点：有序、不重复、无索引。
- TreeMap （由键决定特点）:按照键的大小默认升序排序、不重复、无索引。

TreeMap

- 特点：不重复、无索引、可排序(按照键的大小默认升序排序，**只能对键排序**)
- 原理：TreeMap跟TreeSet集合的底层原理是一样的，都是基于红黑树实现的排序。

TreeMap集合同样也支持两种方式来指定排序规则

- 让类实现Comparable接口，重写比较规则。
- TreeMap集合有一个有参数构造器，支持创建Comparator比较器对象，以便用来指定比较规则。

总结

1. TreeMap集合的特点、原理是怎么样的？
   - **根据键可排序、不重复、无索引**
   - **底层基于红黑树实现排序，增删改查性能较好**
2. TreeMap集合对自定义类型的对象排序，有几种方式指定排序规则？
   - **2种。**
   - **类实现Comparable接口，重写比较规则。**
   - **集合自定义Comparator比较器对象，重写比较规则。**

案例：TreeMap练习

需求：创建一个TreeMap集合，键是学生对象(Student)，值是籍贯(String)。
学生属性姓名和年龄，按照年龄进行排序并遍历。

思路：

① 定义学生类
② 创建TreeMap集合对象
③ 创建学生对象
④ 把学生添加到集合
⑤ 遍历集合

目录

Contents

集合的嵌套

指的是集合中的元素又是一个集合

Map集合实现类特点

- HashMap:元素按照键是无序，不重复，无索引，值不做要求，基于哈希表（与Map体系一致）
- LinkedHashMap:元素按照键是**有序**，不重复，无索引，值不做要求，基于哈希表
- TreeMap：元素只能按照键**排序**，不重复，无索引的，值不做要求，可以做排序

案例

Map集合案例-省和市

需求

- 要求在程序中记住如下省份和其对应的城市信息，记录成功后，要求可以查询出湖北省的城市信息

江苏省 = 南京市，扬州市，苏州市，无锡市，常州市

湖北省 = 武汉市，孝感市，十堰市，宜昌市，鄂州市

河北省 = 石家庄市，唐山市，邢台市，保定市，张家口市

分析

- 定义一个Map集合，键用表示省份名称，值表示城市名称，注意：城市会有多个。
- 根据“湖北省”这个键获取对应的值展示即可。

目录

Contents

JDK 8开始最大的改变：

Lambda

Stream

什么是Stream?

- 也叫Stream流，是Jdk8开始新增的一套API (java.util.stream.*)，可以用于操作集合或者数组的数据。
- 优势： Stream流大量的结合了Lambda的语法风格来编程，提供了一种更加强大，更加简单的方式操作集合或者数组中的数据，代码更简洁，可读性更好。

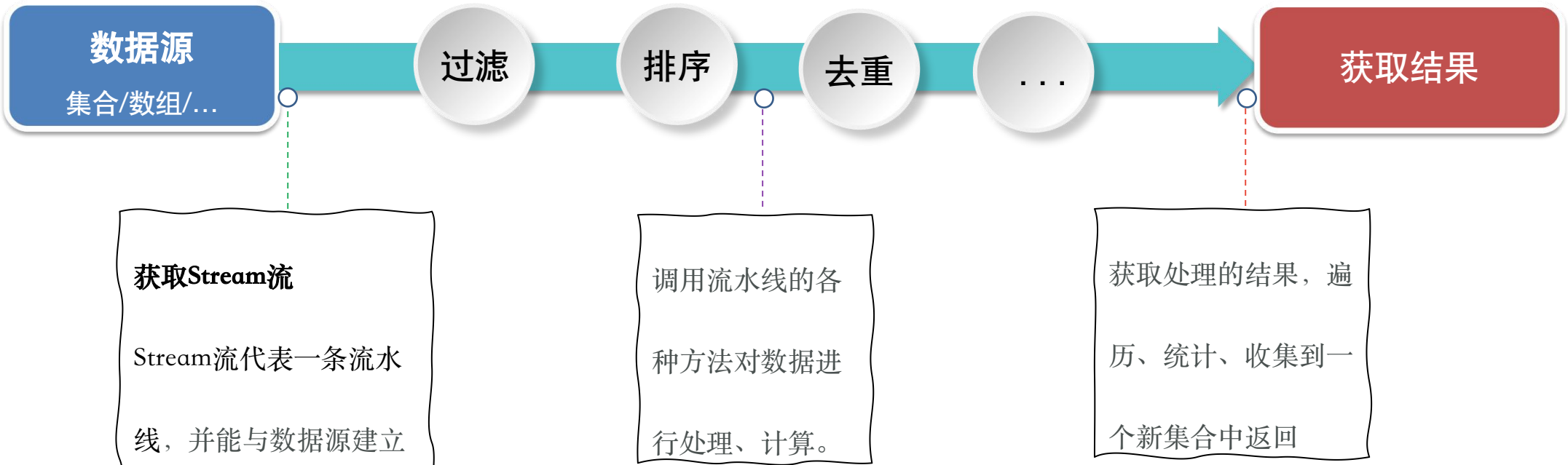
案例 体验Stream流

需求：

```
List<String> list = new ArrayList<>();
list.add("张无忌");
list.add("周芷若");
list.add("赵敏");
list.add("张强");
list.add("张三丰");
```

- 把集合中所有以“张”开头，且是3个字的元素存储到一个新的集合。

Stream流的使用步骤



总结

1、Stream是什么？有什么作用？ 结合了什么技术？

- **简化集合、数组操作的API。结合了Lambda表达式。**

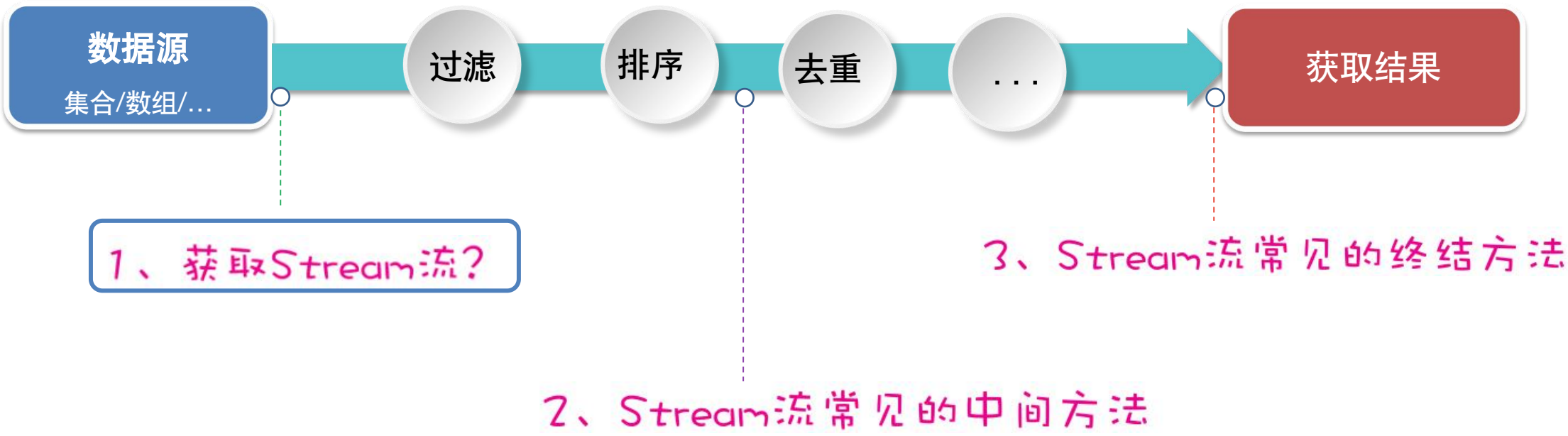
2、说说Stream流处理数据的步骤是什么？

- **先得到集合或者数组的Stream流。**
- **然后调用Stream流的方法对数据进行处理。**
- **获取处理的结果。**

目录

Contents

Stream流的常用方法



1、获取Stream流？

- 获取 集合 的Stream流，

public interface **Stream<T>** …{ }

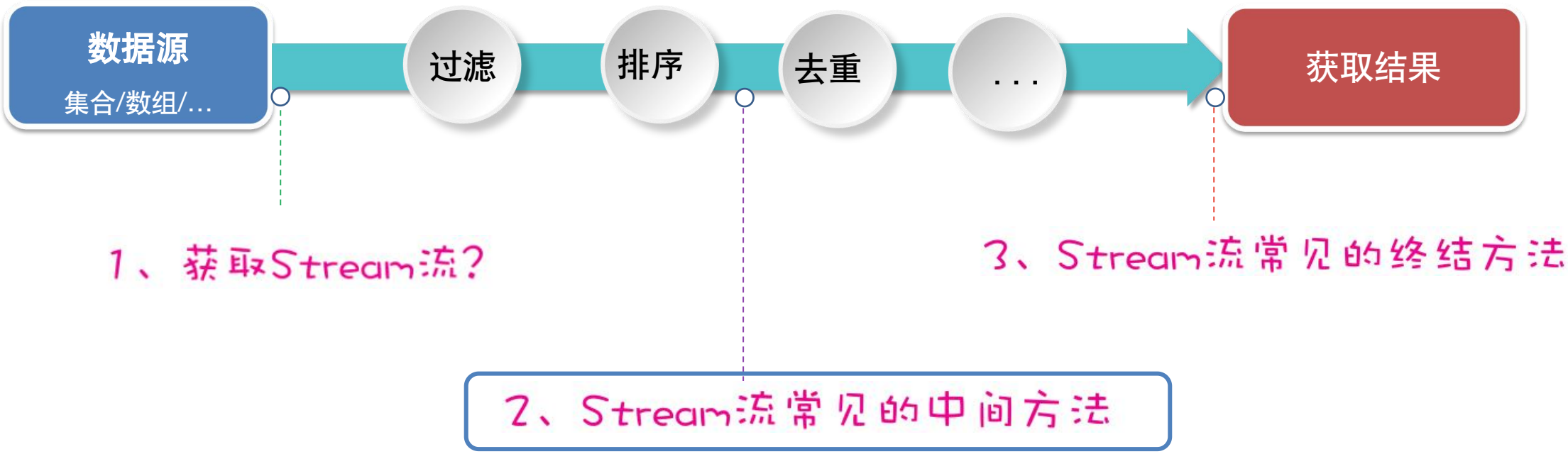
	说明
default **Stream<E> stream()**	获取当前集合对象的Stream流

- 获取 数组 的Stream流

Arrays类提供的如下 方法	说明
public static <T> Stream<T> **stream(T[] array)**	获取当前数组的Stream流

Stream类提供的如下 方法	说明
public static<T> Stream<T> **of(T... values)**	获取当前接收数据的Stream流

Stream流的常用方法



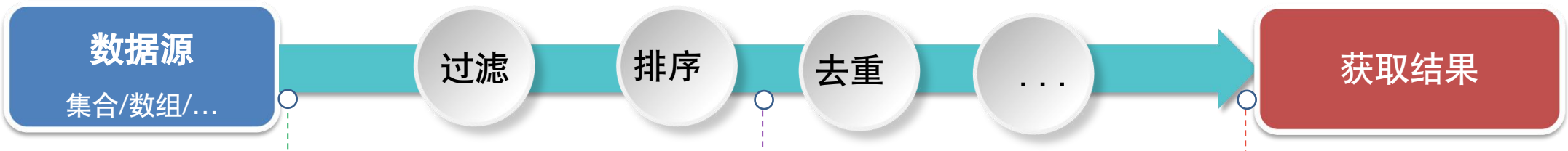
- 中间方法指的是调用完成后会返回新的Stream流，可以继续使用(支持链式编程)。

2、Stream流常见的中间方法

- 中间方法指的是调用完成后会返回新的Stream流，可以继续使用(支持链式编程)。

Stream提供的常用中间方法	说明
Stream<T> **filter**(Predicate<? super T> predicate)	用于对流中的数据进行过滤。
Stream<T> **sorted**()	对元素进行升序排序
Stream<T> **sorted**(Comparator<? super T> comparator)	按照指定规则排序
Stream<T> **limit**(long maxSize)	获取前几个元素
Stream<T> **skip**(long n)	跳过前几个元素
Stream<T> **distinct**()	去除流中重复的元素。
<R> Stream<R> **map**(Function<? super T,? extends R> mapper)	对元素进行加工，并返回对应的新流
static <T> Stream<T> **concat**(Stream a, Stream b)	合并a和b两个流为一个流

Stream流的常用方法



1、获取Stream流？

2、Stream流常见的中间方法

3、Stream流常见的终结方法

- 终结方法指的是调用完成后，不会返回新Stream了，没法继续使用流了。

3、Stream流常见的终结方法

- 终结方法指的是调用完成后，不会返回新Stream了，没法继续使用流了。

Stream提供的常用终结方法	说明
void forEach(Consumer action)	对此流运算后的元素执行遍历
long count()	统计此流运算后的元素个数
Optional<T> max(Comparator<? super T> comparator)	获取此流运算后的最大值元素
Optional<T> min(Comparator<? super T> comparator)	获取此流运算后的最小值元素

3、Stream流常见的终结方法

- **收集Stream流**：就是把Stream流操作后的结果转回到集合或者数组中去返回。
- Stream流：方便操作集合/数组的**手段**；　集合/数组：才是开发中的**目的**。

Stream提供的常用终结方法	说明
R collect(Collector collector)	把流处理后的结果收集到一个指定的集合中去
Object[] toArray()	把流处理后的结果收集到一个数组中去

Collectors工具类提供了具体的收集方式	说明
public static <T> Collector toList()	把元素收集到List集合中
public static <T> Collector toSet()	把元素收集到Set集合中
public static Collector toMap(Function keyMapper , Function valueMapper)	把元素收集到Map集合中

传智教育旗下高端IT教育品牌