Set集合、Map集合

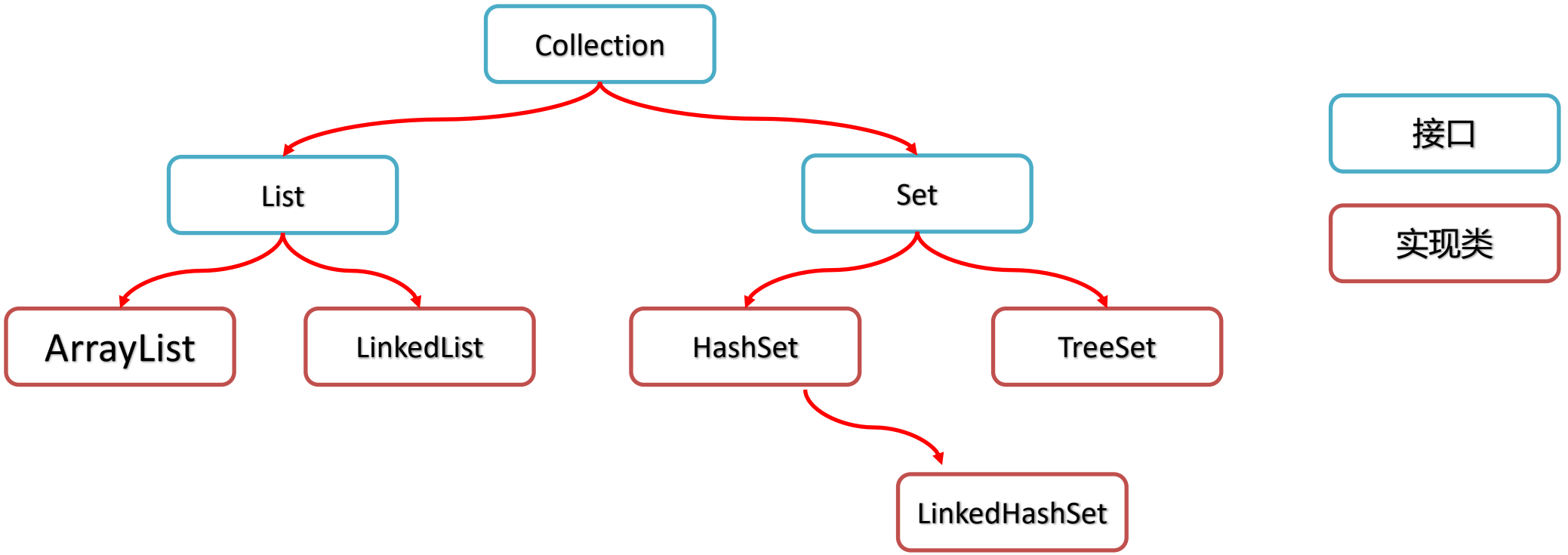
今天同学们需要学会什么

Set系列集合的特点	集合工具类Collections	综合案例	Map集合体系	集合的嵌套
Set系列集合的特点和底层原理	快速的对集合进行元素的添加，排序等操作	把Collection家族的集合应用起来解决一些问题	Map体系的集合能解决什么问题，有哪些体系，各自的特点是什么样的	开发中集合中的元素可能又是一种集合形式，这种方式很常见，需要认识，并学会对其进行处理。

目录

Contents

Collection集合体系



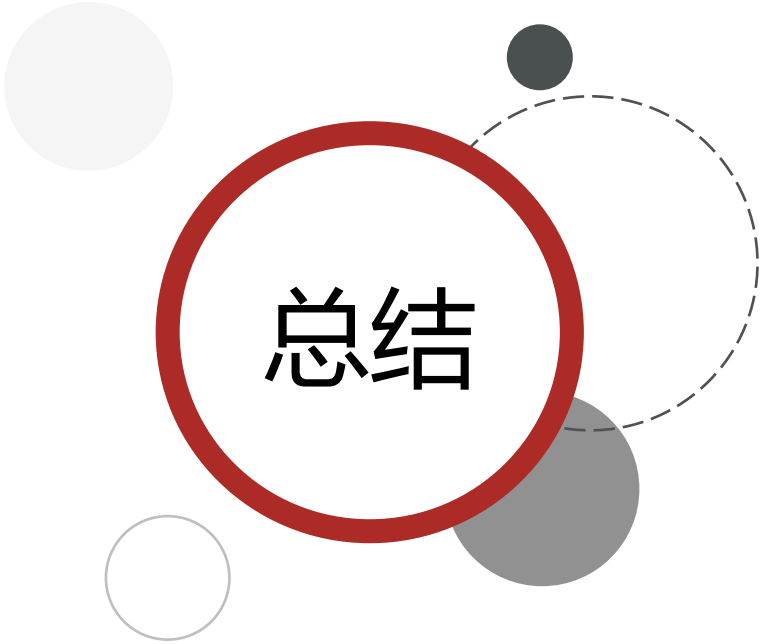
Set系列集合特点

- 无序：存取顺序不一致
- 不重复：可以去除重复
- 无索引：没有带索引的方法，所以不能使用普通for循环遍历，也不能通过索引来获取元素。

Set集合实现类特点

- HashSet：无序、不重复、无索引。
- LinkedHashSet：**有序**、不重复、无索引。
- TreeSet：**排序**、不重复、无索引。

Set集合的功能上基本上与Collection的API一致。



1. Set系列集合的特点。
   - **无序、不重复、无索引。**
2. Set集合的实现类特点。
   - **HashSet无序、不重复、无索引。**
   - **LinkedHashSet 有序、不重复、无索引。**
   - **TreeSet 可排序、不重复、无索引。**

目录

Contents

HashSet底层原理

- HashSet集合底层采取**哈希表**存储的数据。
- 哈希表是一种对于增删改查数据性能都较好的结构。

哈希表的组成

- JDK8之前的，底层使用**数组+链表**组成
- JDK8开始后，底层采用**数组+链表+红黑树**组成。

在了解哈希表之前需要先理解哈希值的概念

哈希值

- 是JDK根据对象的**地址**，按照某种规则算出来的int类型的**数值。**

Object类的API

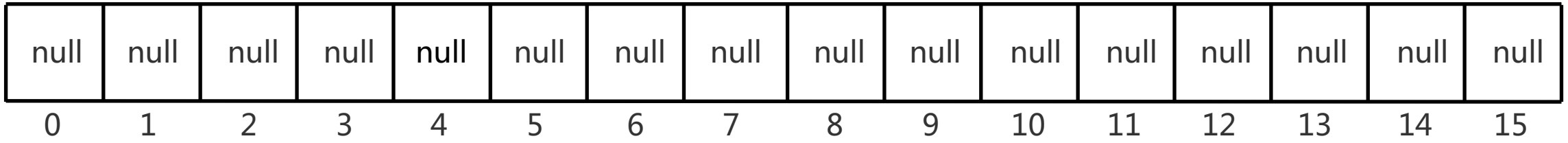
- public int hashCode()：返回对象的哈希值

对象的哈希值特点

- 同一个对象多次调用hashCode()方法返回的哈希值是相同的
- 默认情况下，不同对象的哈希值是不同的。

HashSet1.7版本原理解析：数组 + 链表 +（结合哈希算法）

table[]



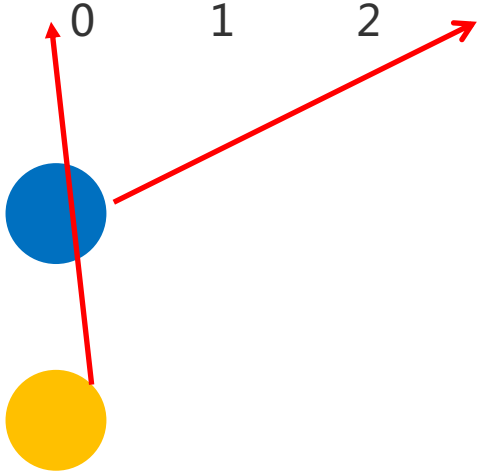

```
Set<String> sets = new HashSet<>();
```

① 创建一个默认长度16的数组，数组名table

HashSet1.7版本原理解析：数组 + 链表 +（结合哈希算法）

table[]

null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

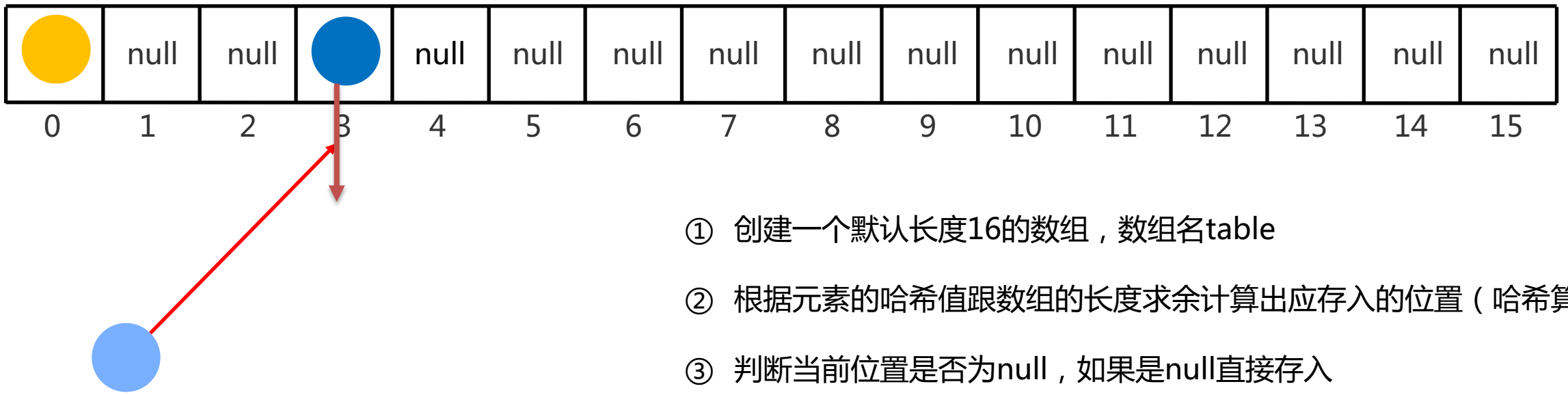


① 创建一个默认长度16的数组，数组名table

② 根据元素的**哈希值**跟**数组的长度求余**计算出应存入的位置（**哈希算法**）

③ 判断当前位置是否为null，如果是null直接存入

HashSet1.7版本原理解析：数组 + 链表 +（结合哈希算法）



① 创建一个默认长度16的数组，数组名table

② 根据元素的哈希值跟数组的长度求余计算出应存入的位置（哈希算法）

③ 判断当前位置是否为null，如果是null直接存入

④ 如果位置不为null，表示有元素，则调用equals方法比较

⑤ 如果一样，则不存，如果不一样，则存入数组

- JDK 7新元素占老元素位置，指向老元素
- JDK 8中新元素挂在老元素下面

HashSet1.7版本原理解析：数组 + 链表 +（结合哈希算法）

① 创建一个默认长度16的数组，数组名table

② 根据元素的哈希值跟数组的长度求余计算出应存入的位置（哈希算法）

③ 判断当前位置是否为null，如果是null直接存入

④ 如果位置不为null，表示有元素，则调用equals方法比较

⑤ 如果一样，则不存，如果不一样，则存入数组，

- JDK 7新元素占老元素位置，指向老元素
- JDK 8中新元素挂在老元素下面

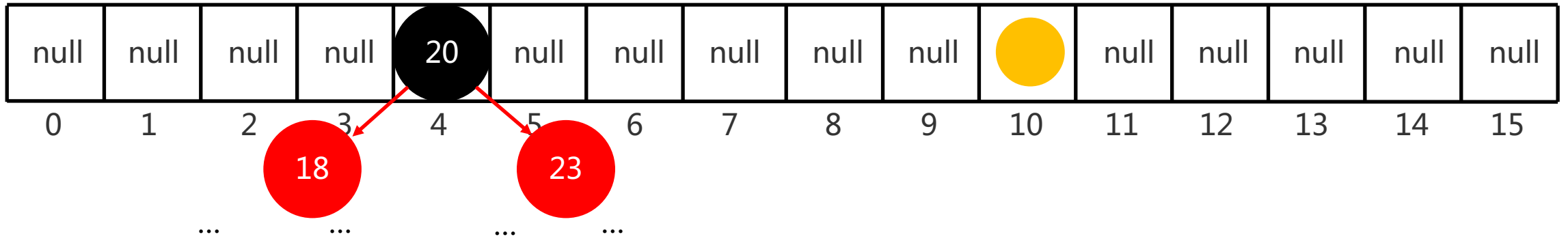
结论：哈希表是一种对于增删改查数据性能都较好的结构。

JDK1.8版本开始HashSet原理解析

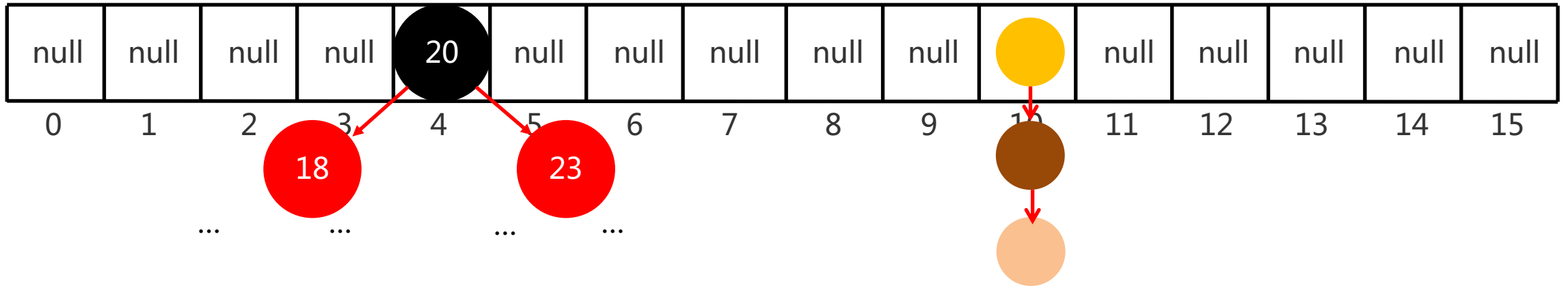
- 底层结构：哈希表（数组、链表、红黑树的结合体）
- 当挂在元素下面的数据过多时，查询性能降低，从JDK8开始后，当链表长度超过**8**的时候，自动转换为红黑树。

HashSet1.8版本原理解析

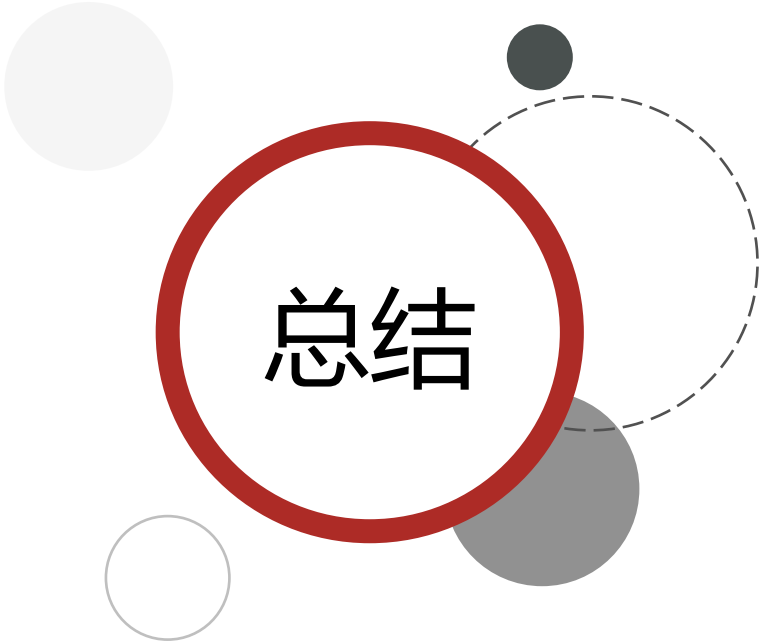
HashSet1.8版本原理解析



HashSet1.8版本原理解析

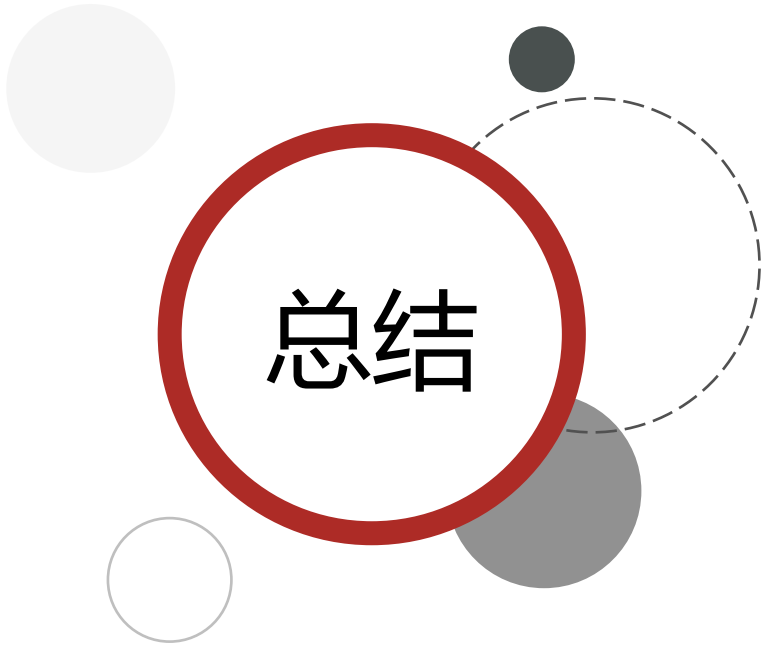


结论：JDK8开始后，哈希表对于红黑树的引入进一步提高了操作数据的性能。



1. Set集合的底层原理是什么样的

- **JDK8之前的，哈希表：底层使用数组+链表组成**
- **JDK8开始后，哈希表：底层采用数组+链表+红黑树组成。**



2.. 哈希表的详细流程

① 创建一个默认长度16，默认加载因为0.75的数组，数组名table

② 根据元素的哈希值跟数组的长度计算出应存入的位置

③ 判断当前位置是否为null，如果是null直接存入，如果位置不为null，表示有元素， 则调用equals方法比较属性值，如果一样，则不存，如果不一样，则存入数组。

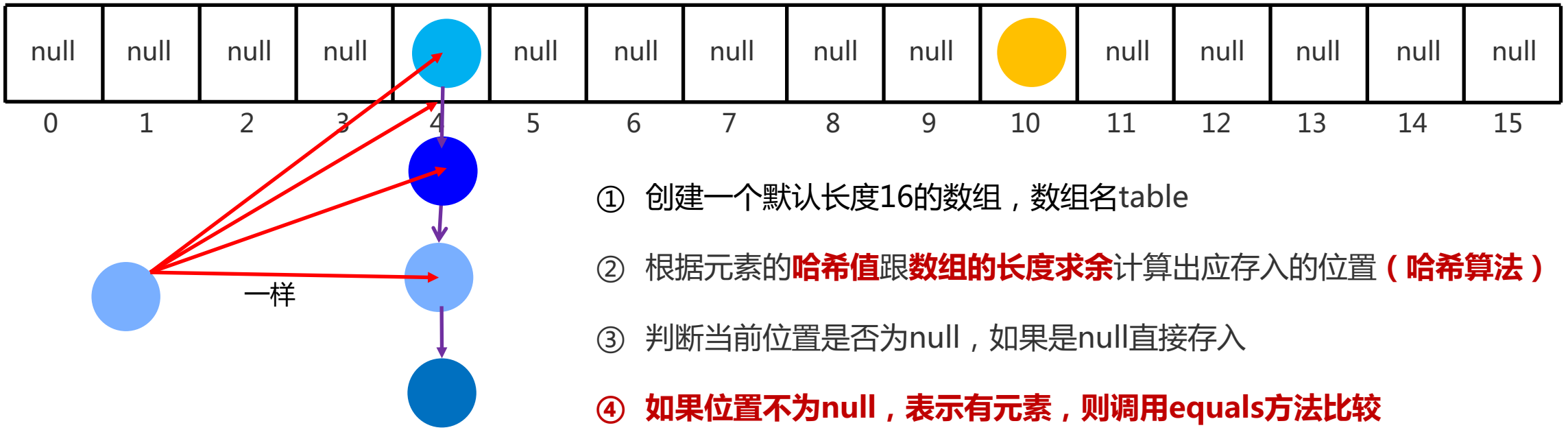
④ 当数组存满到16*0.75=12时，就自动扩容，每次扩容原先的两倍

目录

Contents

HashSet去重复原理解析

Segment[]



① 创建一个默认长度16的数组，数组名table

② 根据元素的**哈希值**跟**数组的长度求余**计算出应存入的位置（**哈希算法**）

③ 判断当前位置是否为null，如果是null直接存入

④ 如果位置不为null，表示有元素，则调用equals方法比较

⑤ 如果一样，则不存，如果不一样，则存入数组，

结论：如果希望Set集合认为2个内容一样的对象是重复的，
必须重写对象的hashCode()和equals()方法

案例 Set集合去重复

需求：

创建一个存储学生对象的集合，存储多个学生对象，使用程序实现在控制台遍历该集合，要求：学生对象的成员变量值相同，我们就认为是同一个对象

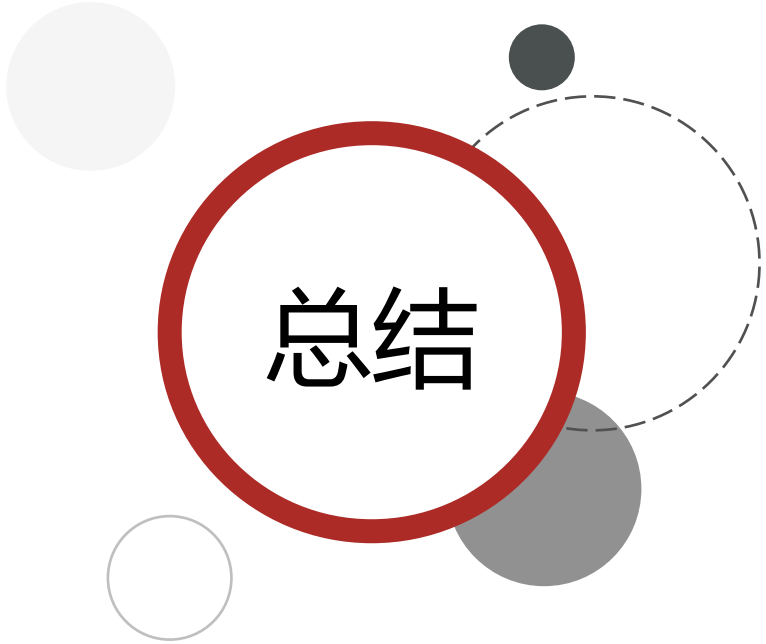
分析

① 定义学生类，创建HashSet集合对象, 创建学生对象

② 把学生添加到集合

③ 在学生类中重写两个方法，hashCode()和equals()，自动生成即可

④ 遍历集合(增强for)



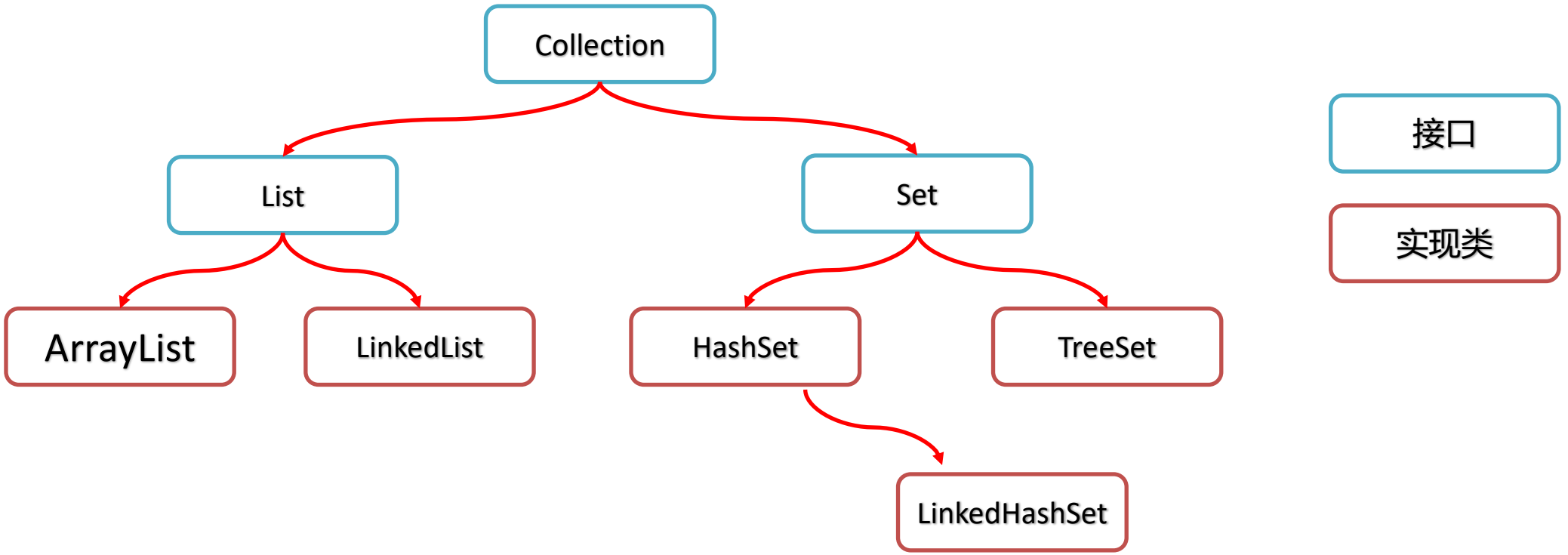
1. 如果希望Set集合认为2个内容相同的对象是重复的应该怎么办？

- **重写对象的hashCode和equals方法。**

目录

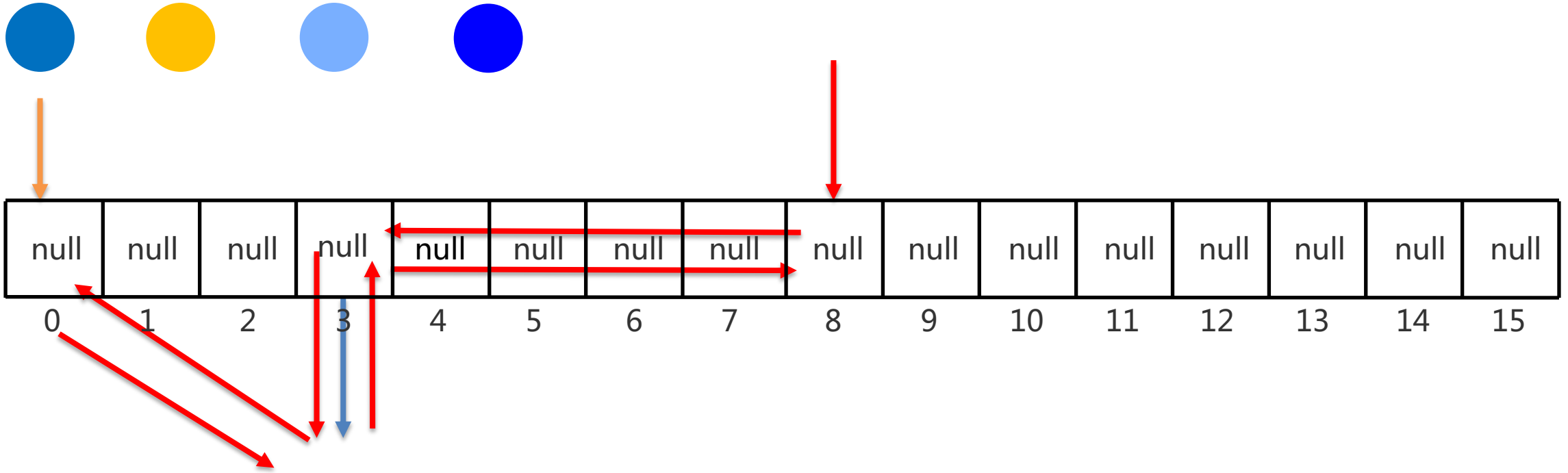
Contents

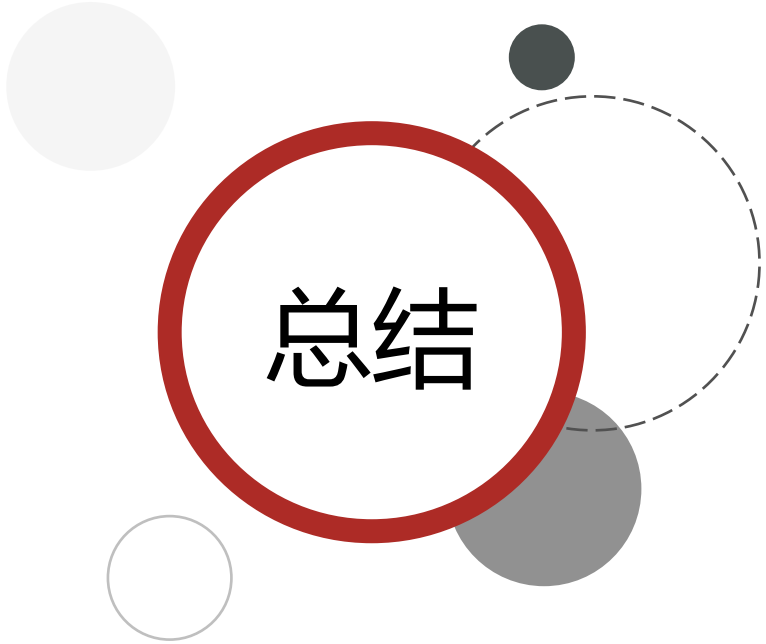
Collection集合体系



LinkedHashSet集合概述和特点

- **有序**、不重复、无索引。
- 这里的有序指的是保证存储和取出的元素顺序一致
- **原理**：底层数据结构是依然哈希表，只是每个元素又额外的多了一个双链表的机制记录存储的顺序。





1. LinkedHashSet集合的特点和原理是怎么样的？
   - **有序、不重复、无索引**
   - **底层基于哈希表，使用双链表记录添加顺序。**

目录

Contents

TreeSet集合概述和特点

- 不重复、无索引、可排序
- 可排序：按照元素的大小默认升序（有小到大）排序。
- TreeSet集合底层是基于红黑树的数据结构实现排序的，增删改查性能都较好。
- **注意：TreeSet集合是一定要排序的，可以将元素按照指定的规则进行排序。**

TreeSet集合默认的规则

- 对于数值类型：Integer，Double，官方默认按照大小进行升序排序。
- 对于字符串类型：默认按照首字符的编号升序排序。
- 对于自定义类型如Student对象，TreeSet无法直接排序。

结论：想要使用TreeSet存储自定义类型，需要制定排序规则

自定义排序规则

- TreeSet集合存储对象的的时候有2种方式可以设计自定义比较规则

方式一

- 让自定义的类（如学生类）**实现Comparable接口**重写里面的**compareTo**方法**来定制比较规则。**

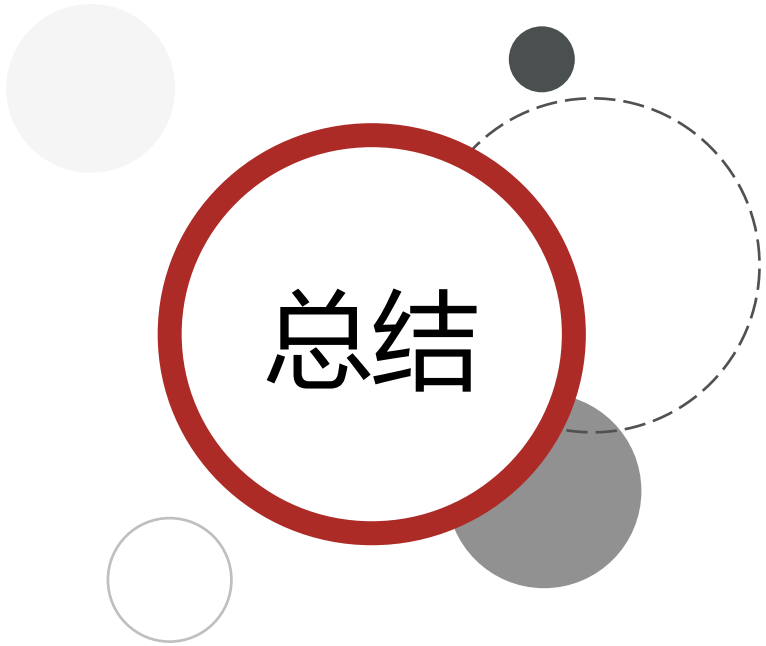
方式二

- **TreeSet集合有参数构造器，可以设置Comparator接口对应的比较器对象，来定制比较规则。**

两种方式中，关于返回值的规则：

- 如果认为第一个元素大于第二个元素返回正整数即可。
- 如果认为第一个元素小于第二个元素返回负整数即可。
- 如果认为第一个元素等于第二个元素返回0即可，此时Treeset集合只会保留一个元素，认为两者重复。

注意：如果TreeSet集合存储的对象有实现比较规则，集合也自带比较器，默认使用集合自带的比较器排序。



1. TreeSet集合的特点是怎么样的？
   - **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. TreeSet集合自定义排序规则有几种方式
   - **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。**

练习 TreeSet集合进行对象排序

需求：键盘录入3个学生信息(姓名,语文成绩,数学成绩,英语成绩),按照总分从高到低输出到控制台

分析

① 定义学生类

② 创建TreeSet集合对象，通过比较器排序进行排序

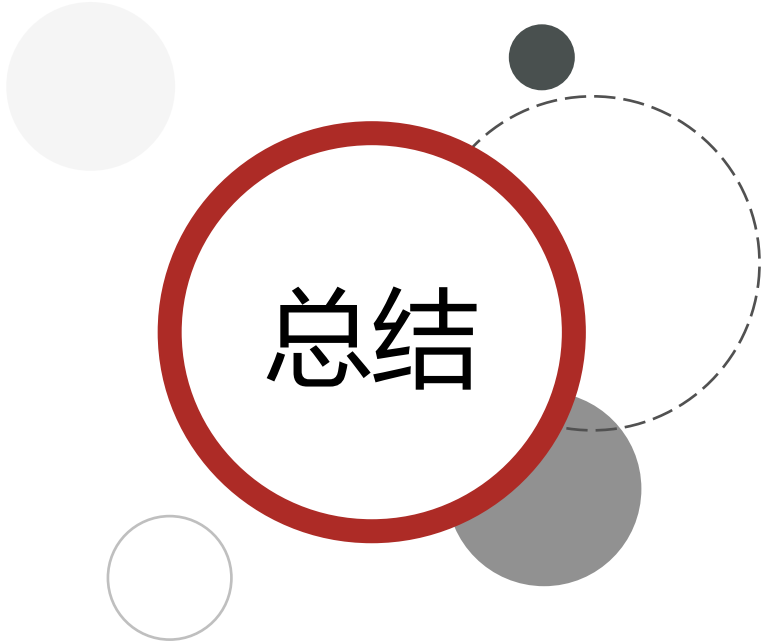
③ 创建学生对象

④ 把学生对象添加到集合

⑤ 遍历集合

目录

Contents



1. 如果希望元素可以重复，又有索引，索引查询要快？
   - **用ArrayList集合，基于数组的。（用的最多）**
2. 如果希望元素可以重复，又有索引，增删首尾操作快？
   - **用LinkedList集合，基于链表的。**
3. 如果希望增删改查都快，但是元素不重复、无序、无索引。
   - **用HashSet集合，基于哈希表的。**
4. 如果希望增删改查都快，但是元素不重复、有序、无索引。
   - **用LinkedHashSet集合，基于哈希表和双链表。**
5. 如果要对对象进行排序。
   - **用TreeSet集合，基于红黑树。后续也可以用List集合实现排序。**

目录

Contents

可变参数

- 可变参数用在形参中可以接收多个数据。
- 可变参数的格式：数据类型...参数名称

可变参数的作用

- 接收参数非常灵活，方便。可以不接收参数，可以接收1个或者多个参数，也可以接收一个数组
- **可变参数在方法内部本质上就是一个数组。**

可变参数的注意事项：

- 1.一个形参列表中可变参数只能有一个
- 2.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集合工具类

- java.utils.Collections:是集合工具类
- 作用：Collections并不属于集合，是用来操作集合的工具类。

Collections常用的API

方法名称	说明
`public static <T> boolean addAll(Collection<? super T> c, T... elements)`	给集合对象批量添加元素
`public static void shuffle(List<?> list)`	打乱List集合元素的顺序

Collections排序相关API

- 使用范围：只能对于List集合的排序。

排序方式1：

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

注意：本方式不可以直接对自定义类型的List集合排序，除非自定义类型实现了比较规则Comparable接口。

排序方式2：

方法名称	说明
`public static <T> void sort(List<T> list,Comparator<? super T> c)`	将集合中元素按照指定规则排序

1. 为何Collections的API只能针对于List集合排序。

方法名称
`public static <T> void sort(List<T> list)`
`public static <T> void sort(List<T> list , Comparator<? super T> c)`

目录

Contents

案例 斗地主游戏

需求：

在启动游戏房间的时候，应该提前准备好54张牌，完成洗牌、发牌、牌排序、逻辑。

分析：

① ：当系统启动的同时需要准备好数据的时候，就可以用静态代码块了。

② ：洗牌就是打乱牌的顺序。

③ ：定义三个玩家、依次发出51张牌

④ ：给玩家的牌进行排序(拓展)

⑤ ：输出每个玩家的牌数据。

目录

Contents

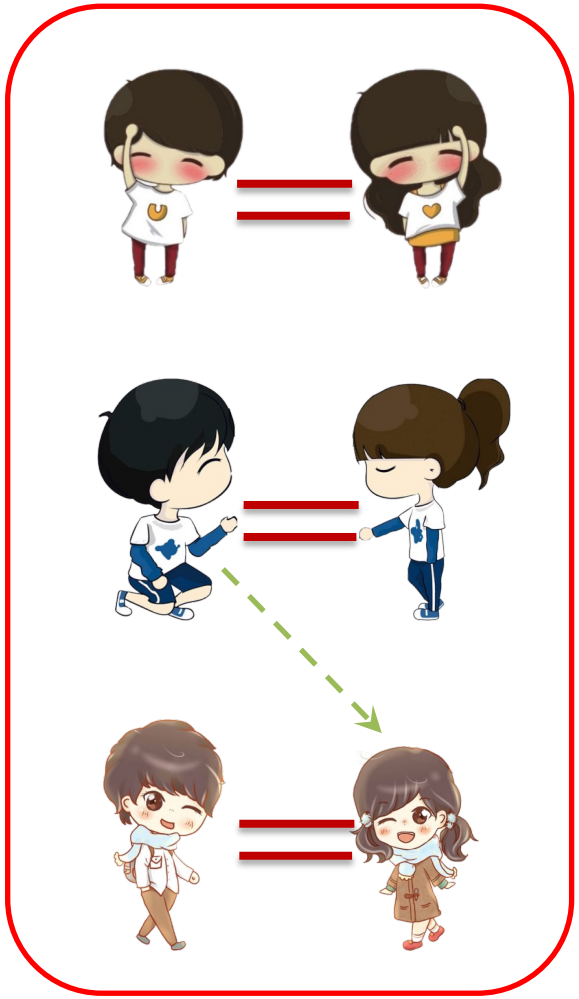
Map集合概述和使用

- Map集合是一种双列集合，每个元素包含两个数据。
- Map集合的每个元素的格式：key=value(键值对元素)。
- Map集合也被称为“**键值对集合**”。

Map集合整体格式：

- Collection集合的格式: [元素1,元素2,元素3..]
- Map集合的完整格式：{key1=value1 , key2=value2 , key3=value3 , ...}

Map集合的键



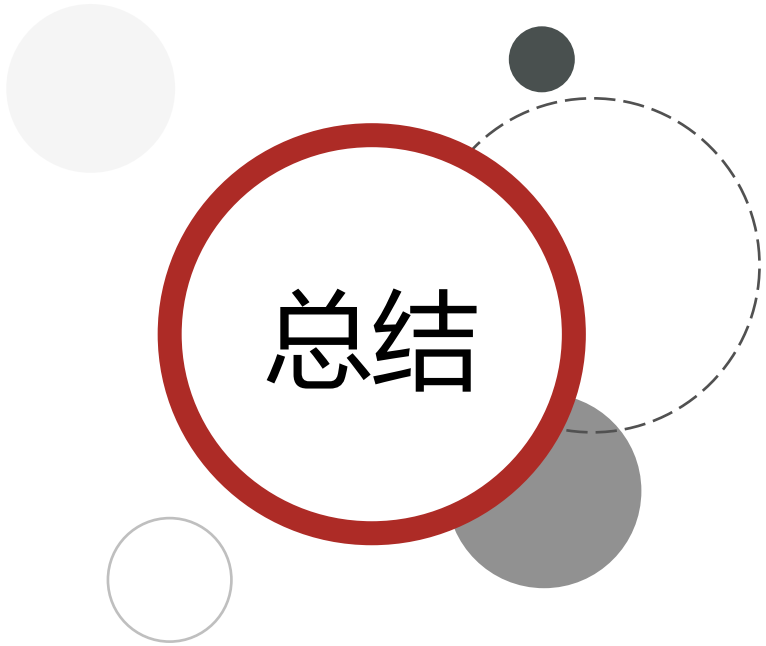
Map集合的值

Map集合的使用场景之一：购物车系统

分析

- 购物车提供的四个商品和购买的数量在后台需要容器存储。
- 每个商品对象都一一对应一个购买数量。
- 把商品对象看成是Map集合的建，购买数量看成Map集合的值。

{商品1=2，商品2=3 ， 商品3 = 2 ，商品4= 3}



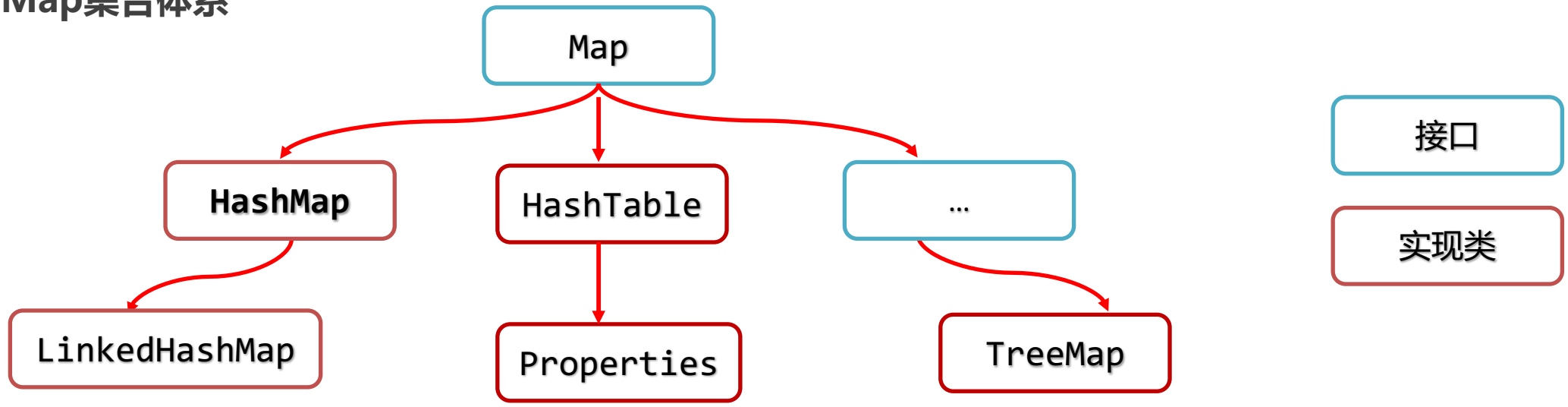
1. Map集合是什么？使用场景是什么样的？

- **Map集合是键值对集合**
- **Map集合非常适合做类购物车这样的业务场景。**

目录

Contents

Map集合体系



说明

- **使用最多的Map集合是HashMap。**
- 重点掌握HashMap , LinkedHashMap , TreeMap。其他的后续理解。

Map集合的键
无序、不重复的

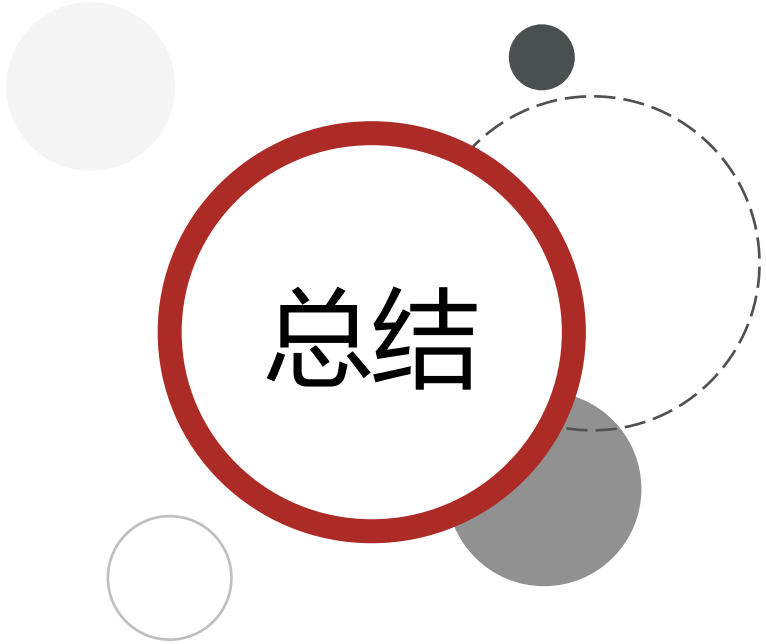
Map集合的值
值不做要求
可以重复

Map集合体系特点

- Map集合的特点都是由键决定的。
- Map集合的键是无序,不重复的，无索引的，值不做要求（可以重复）。
- Map集合后面重复的键对应的值会覆盖前面重复键的值。
- Map集合的键值对都可以为null。

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。（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。
- TreeMap：元素按照建是**排序**，不重复，无索引的，值不做要求。



1. Map集合的特点

- HashMap:元素按照键是无序，不重复，无索引，值不做要求。（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。
- TreeMap：元素按照建是排序，不重复，无索引的，值不做要求。

目录

Contents

Map集合

- Map是双列集合的祖宗接口，它的功能是全部双列集合都可以继承使用的。

Map API如下:

方法名称	说明
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()	集合的长度，也就是集合中键值对的个数

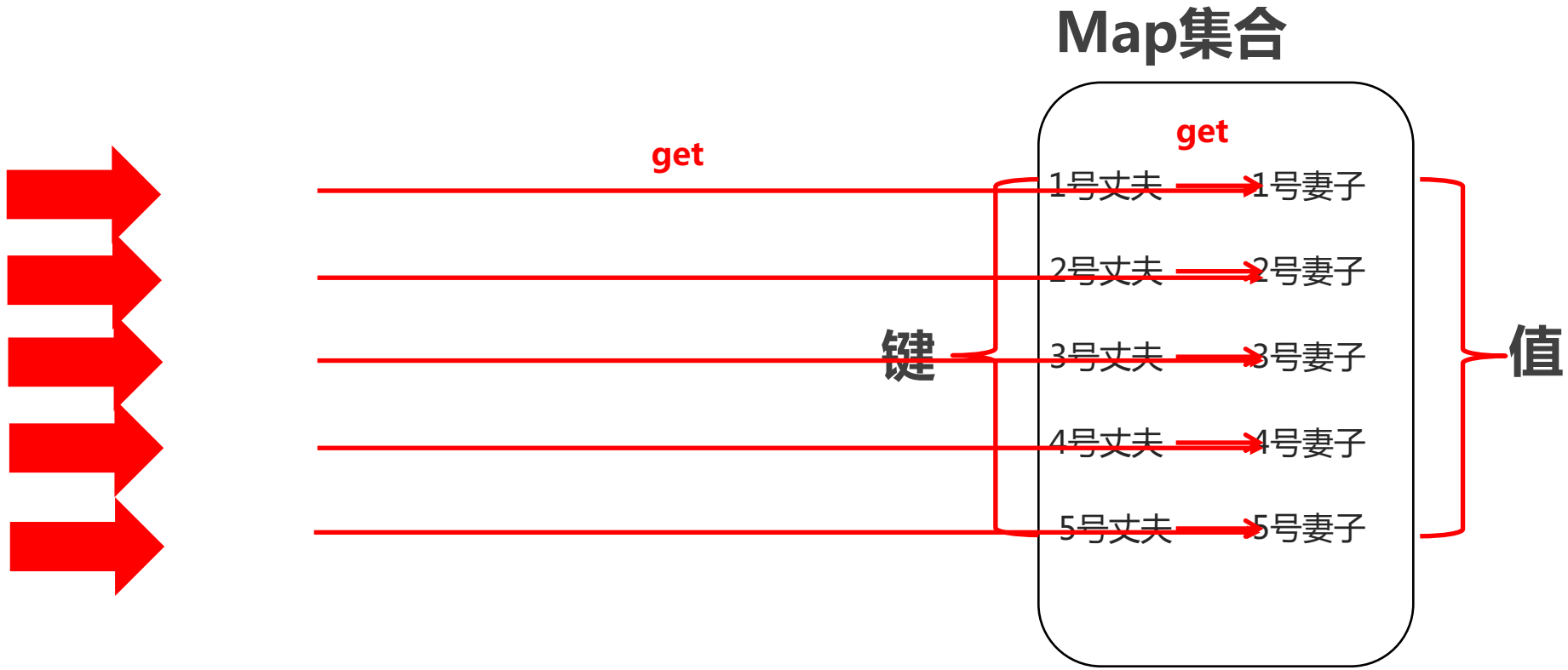
目录

Contents

Map集合的遍历方式有：3种。

- 方式一：键找值的方式遍历：先获取Map集合全部的键，再根据遍历键找值。
- 方式二：键值对的方式遍历，把“键值对“看成一个整体，难度较大。
- 方式三：JDK 1.8开始之后的新技术：Lambda表达式。

遍历Map集合方式一：键找值流程



Map集合的遍历方式一：键找值

- 先获取Map集合的全部键的Set集合。
- 遍历键的Set集合，然后通过键提取对应值。

键找值涉及到的API:

方法名称	说明
Set<K> keySet()	获取所有键的集合
V get(Object key)	根据键获取值

目录

Contents

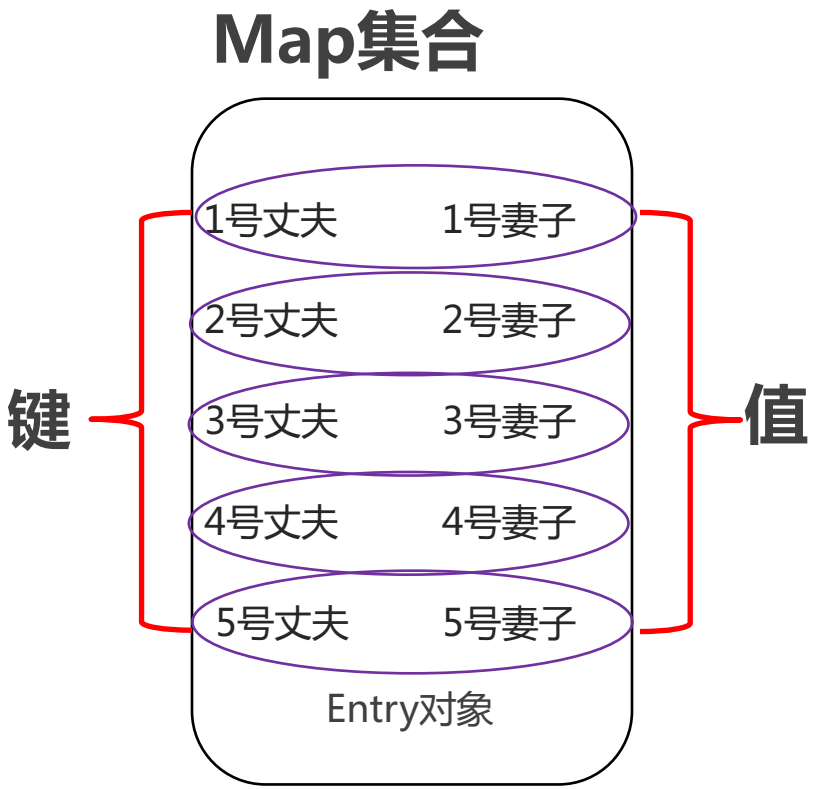
Map集合的遍历方式二：键值对

- 先把Map集合转换成Set集合，Set集合中每个元素都是键值对实体类型了。
- 遍历Set集合，然后提取键以及提取值。

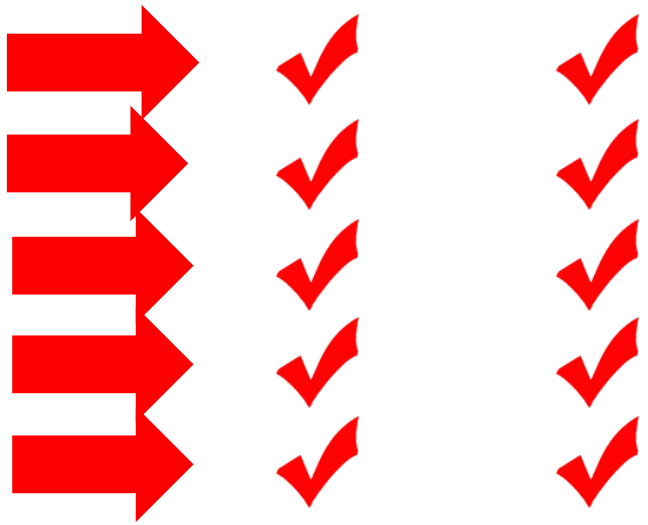
键值对涉及到的API:

方法名称	说明
Set<Map.Entry<K,V>> entrySet()	获取所有键值对对象的集合
K getKey()	获得键
V getValue()	获取值

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



目录

Contents

Map集合的遍历方式三：Lambda

- 得益于JDK 8开始的新技术Lambda表达式，提供了一种更简单、更直接的遍历集合的方式。

Map结合Lambda遍历的API

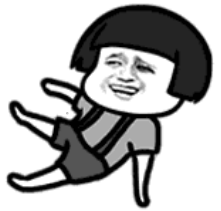
方法名称	说明
`default void forEach(BiConsumer<? super K, ? super V> action)`	结合lambda遍历Map集合

流程

maps = {huawei=1000, 手表=10, 生活用品=10, iphoneX=100}

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

就问你服不服



案例

Map集合案例-统计投票人数

需求

某个班级80名学生，现在需要组成秋游活动，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。

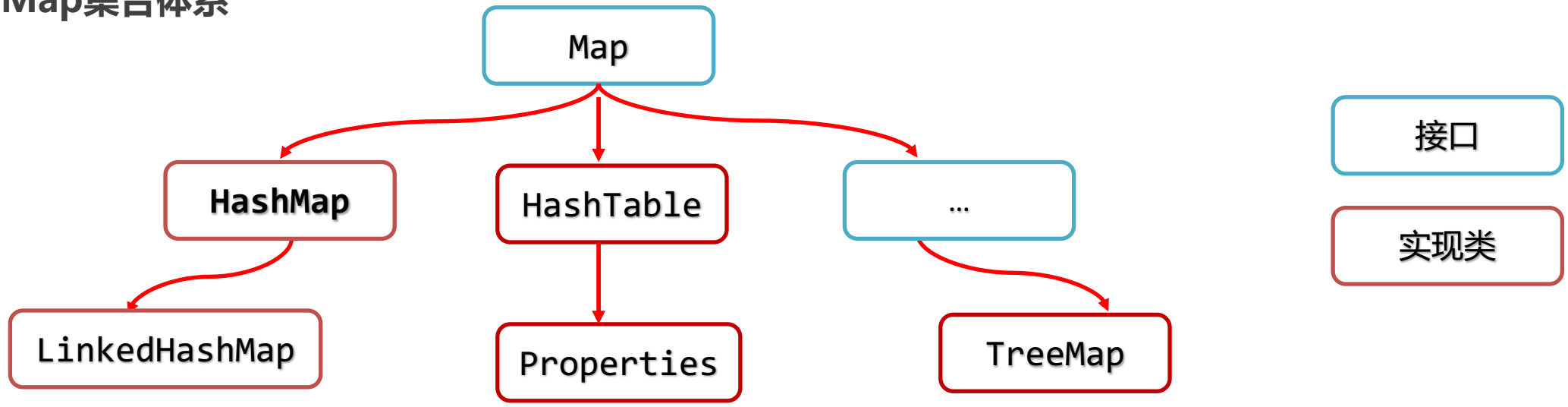
分析

- 将80个学生选择的数据拿到程序中去。
- 定义Map集合用于存储最终统计的结果。
- 遍历80个学生选择的数据，看Map集合中是否存在，不存在存入“数据=1 “，存在则其对应值+1,

目录

Contents

Map集合体系



说明

- **使用最多的Map集合是HashMap。**
- 重点掌握HashMap , LinkedHashMap , TreeMap。其他的后续理解。

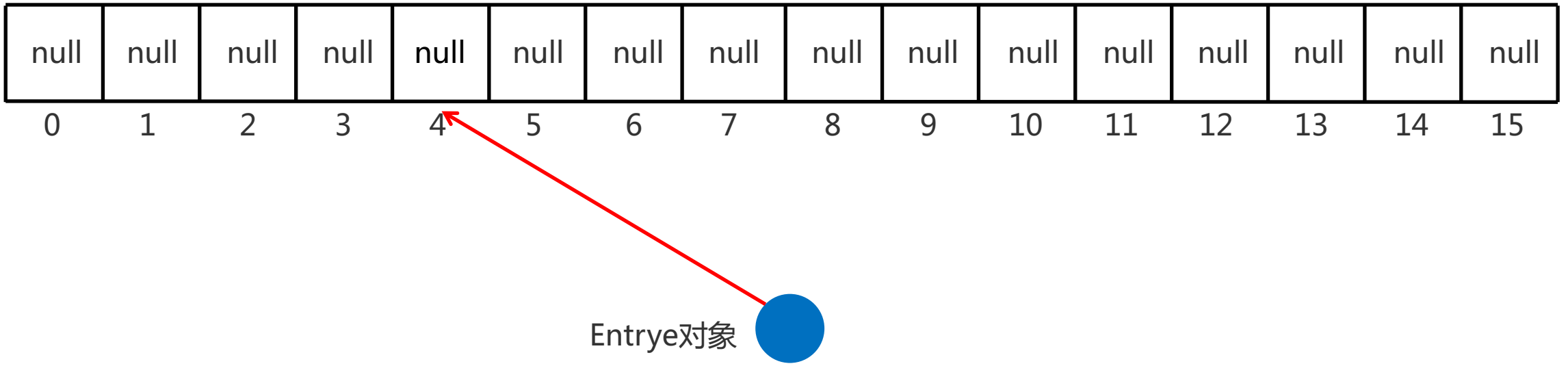
HashMap的特点

- HashMap是Map里面的一个实现类。特点都是由键决定的：无序、不重复、无索引
- 没有额外需要学习的特有方法，直接使用Map里面的方法就可以了。
- HashMap跟HashSet底层原理是一模一样的，都是哈希表结构，只是HashMap的每个元素包含两个值而已。

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

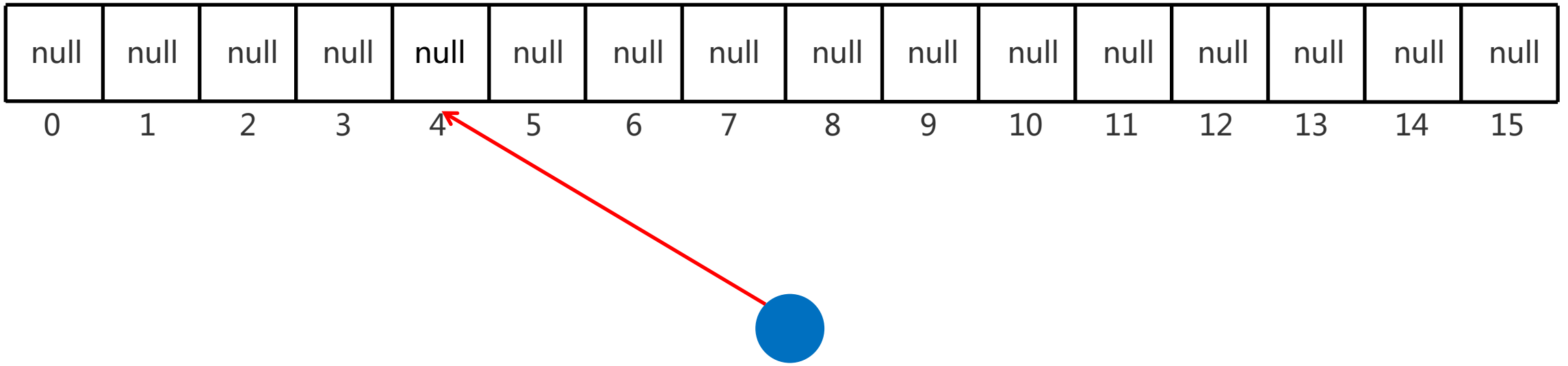
```
public HashSet() {
    map = new HashMap<>();
}
```

HashMap的添加规则

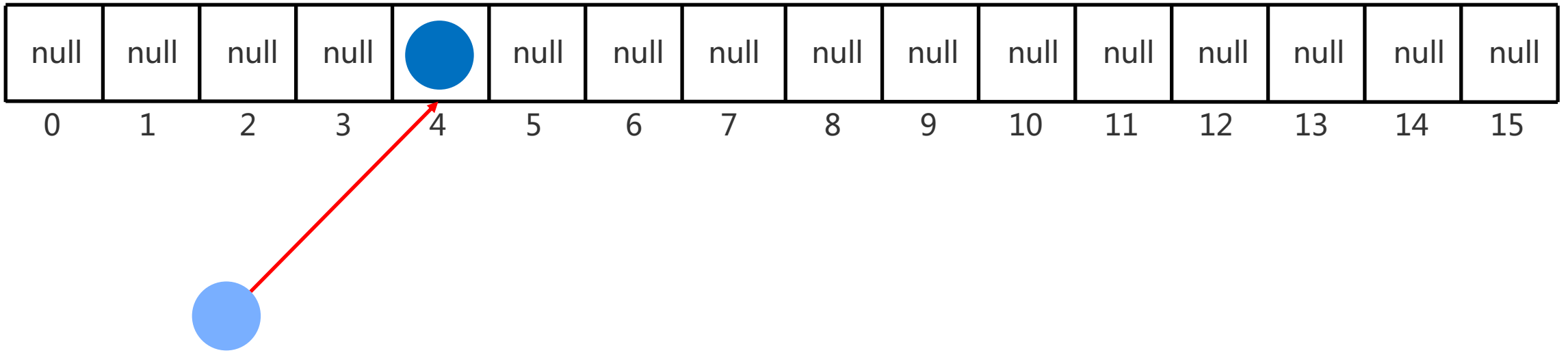



```
map.put(" 键"," 值);
```

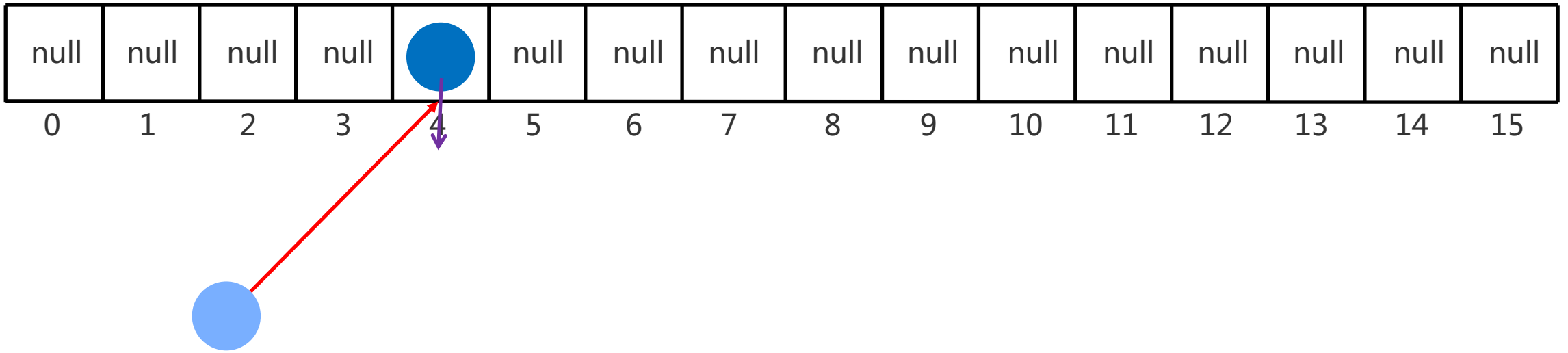
HashMap的添加规则



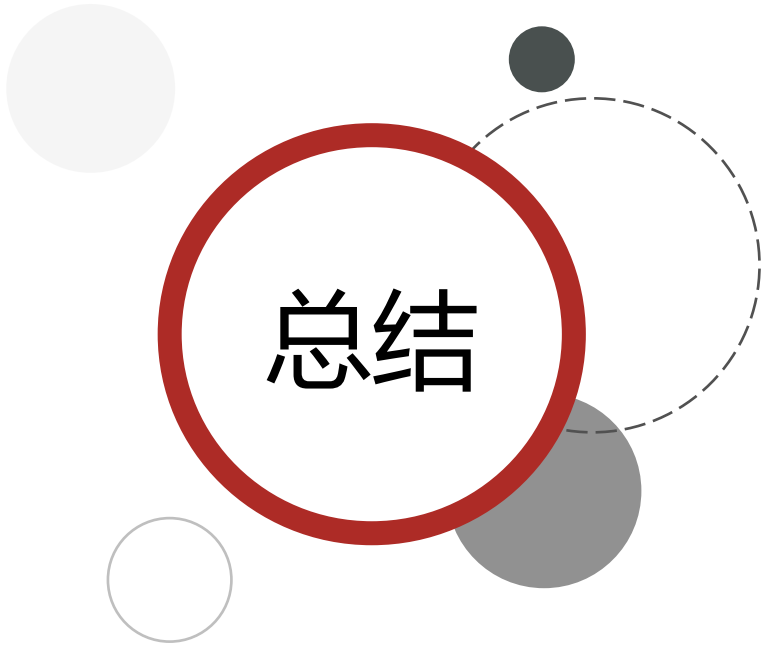
HashMap的添加规则



HashMap的添加规则



HashMap的添加规则



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。
- 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。
- 基于哈希表。增删改查的性能都较好。

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

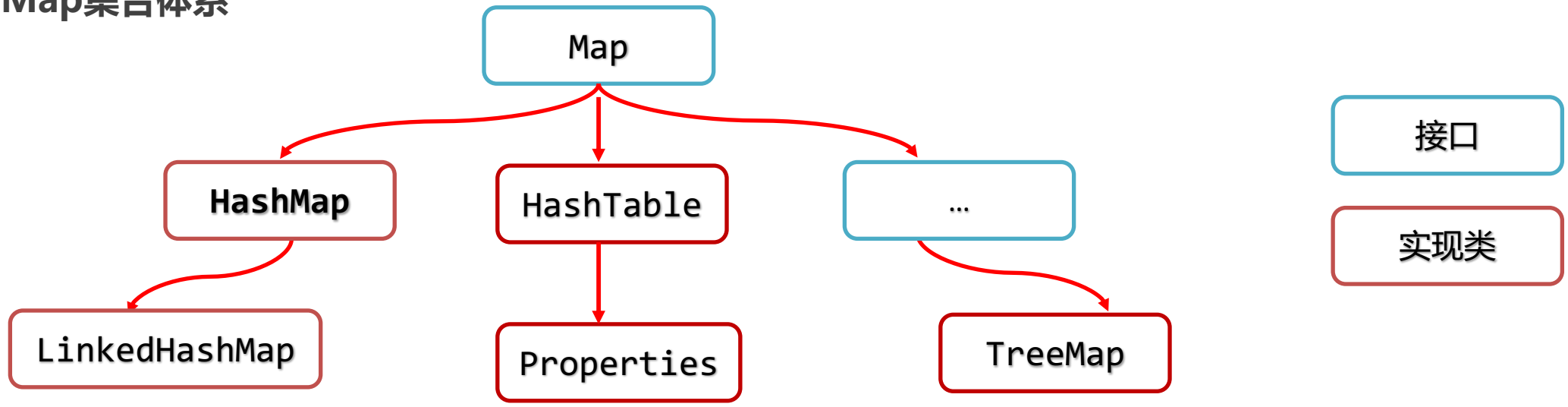
思路：

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

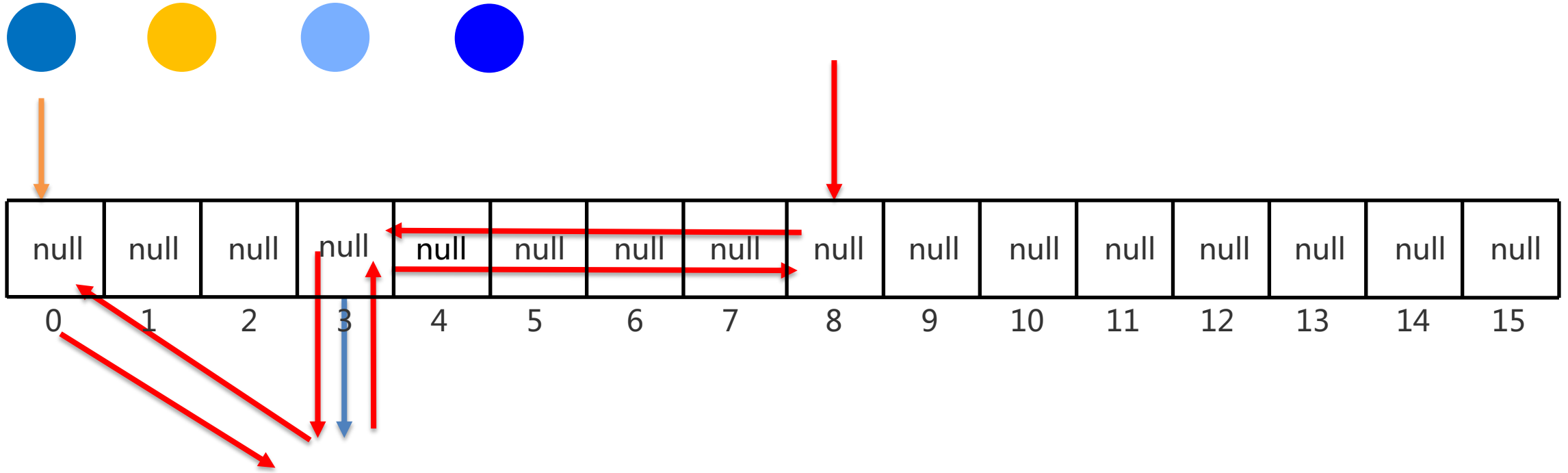


说明

- **使用最多的Map集合是HashMap。**
- 重点掌握HashMap , LinkedHashMap , TreeMap。其他的后续理解。

LinkedHashMap集合概述和特点

- **由键决定：有序**、不重复、无索引。
- 这里的有序指的是保证存储和取出的元素顺序一致
- **原理**：底层数据结构是依然哈希表，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存储的顺序。



目录

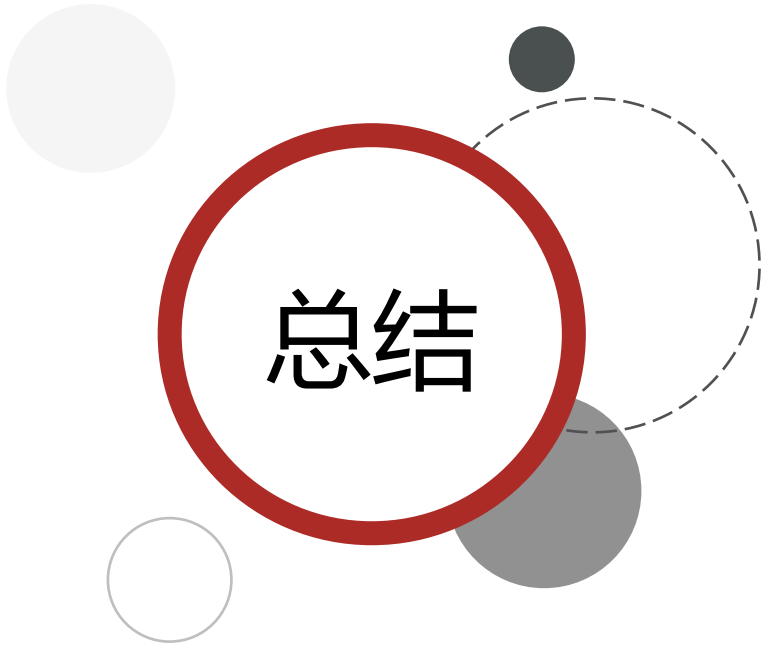
Contents

TreeMap集合概述和特点

- 由键决定特性：不重复、无索引、可排序
- 可排序：按照键数据的大小默认升序（有小到大）排序。**只能对键排序。**
- **注意：TreeMap集合是一定要排序的，可以默认排序，也可以将键按照指定的规则进行排序**
- TreeMap跟TreeSet一样底层原理是一样的。

TreeMap集合自定义排序规则有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。



1. TreeMap集合的特点是怎么样的？
   - **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集合自定义排序规则有几种方式
   - **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。**

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：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

案例：TreeMap练习

需求：字符串“aababcabcdabcde”

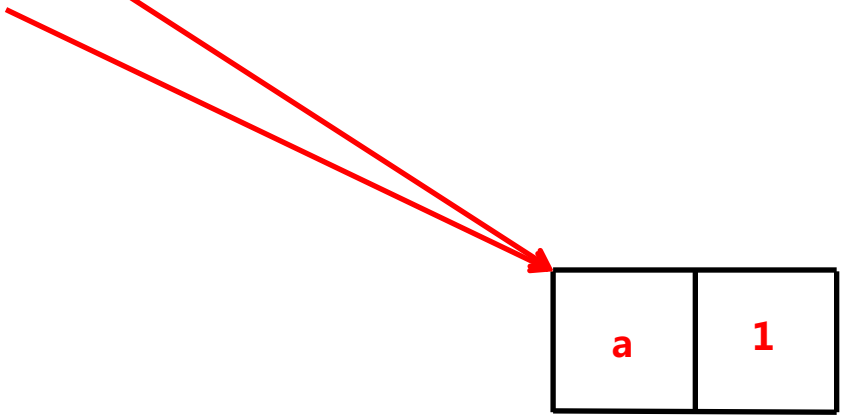
请统计字符串中每一个字符出现的次数，并按照以下格式输出

输出结果：

a(5)b(4)c(3)d(2)e(1)

案例：TreeMap练习

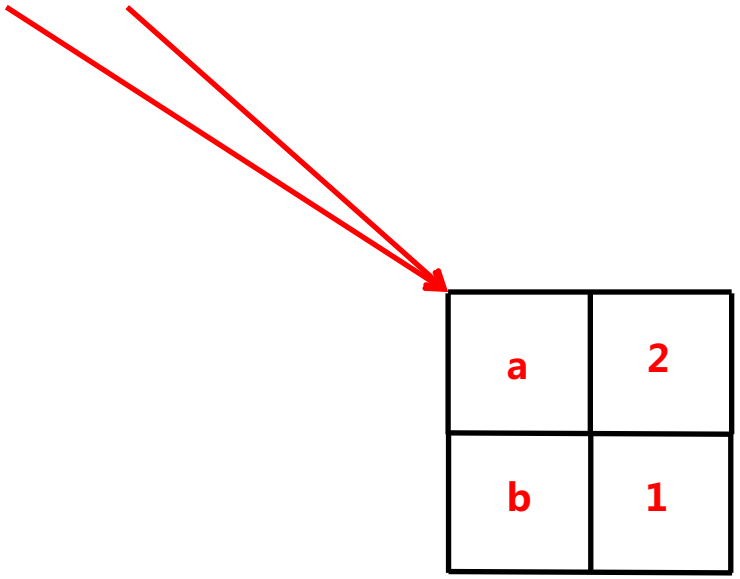
a a b a b c a b c d a b c d e



- Map集合中，键存字符，值存出现的次数
- 遍历字符串，得到每一个字符
- 到集合中看是否包含这个字符
- 如果不包含，表示是第一次出现
- 如果包含，表示不是第一次出现

案例：TreeMap练习

a a b a b c a b c d a b c d e



- Map集合中，键存字符，值存出现的次数
- 遍历字符串，得到每一个字符
- 到集合中看是否包含这个字符
- 如果不包含，表示是第一次出现
- 如果包含，表示不是第一次出现

目录

Contents

案例

Map集合案例-统计投票人数(集合的嵌套)

需求

- 某个班级多名学生，现在需要组成秋游活动，班长提供了四个景点依次是（A、B、C、D）,每个学生可以选择多个景点，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，需要记住每个学生选择的情况。
- 定义Map集合用于存储最终统计的结果。

传智教育旗下高端IT教育品牌