Java 集合 容器

1. java 集合（容器）概述

(1) Collection (单列集合)

- Set: HashSet、TreeSet(无序)
- List: ArrayList、LinkedList、Vector(有序)

(2) Map (双列集合) K-V

- HashMap、HashTable、TreeMap

Java 集合， 也叫作容器，主要是由两大接口派生而来：一个是 Collection 接口，主要用于存放单一元素；另一个是 Map 接口，主要用于存放键值对。对于 Collection 接口，下面又有三个主要的子接口：List、Set 和 Queue。

2. List, Set, Queue, Map 区别?

- Set 代表无序的，元素不可重复的集合；
- List 代表有序的，元素可以重复的集合；
- Queue 代表先进先出（FIFO）的队列，存储的元素是有序的、可重复的。
- Map 代表具有映射关系（key-value）的集合；key 是无序的、不可重复的，value 是无序的、可重复的，每个键最多映射到一个值。

3. List 接口介绍

1. List 集合类中元素**有序**(即添加顺序和取出顺序一致)、且**可重复**
2. List 集合中的每个元素都有其对应的顺序索引，即支持索引

4. ArrayList 扩容机制

（1）ArrayList 中维护了一个 Object 类型的数组 elementData，transient Object[] elementData

（2）扩容机制：当创建 ArrayList 对象，如果使用无参构造器，初始 elementData 容量为 0，当第一次添加元素时，则扩容 elementData 容量为 10，如需再次扩容，则扩容 elementData 为 1.5 倍

（3）如果使用指定大小的构造器，则初始 elementData 容量为指定大小，如果需扩容，则直接扩容 elementData 为 1.5 倍

5. ArrayList 和 Vector 区别

	线程安全（同步）效率	扩容倍数
ArrayList	不安全，效率高	无参构造：第一次 10，第二次 1.5 倍；有参构造 1.5 倍
Vector	安全，效率不高	无参构造：默认 10，满后 2 倍扩容；有参构造 2 倍扩容

（1）线程安全方面：ArrayList 线程不安全，效率高；Vector 线程安全，效率不高；

（2）扩容方面：ArrayList 和 Vector 都会根据实际的需要动态的调整容量，只不过在 Vector 扩容每次会增加 1 倍，而 ArrayList 只会增加 50%。

6. ArrayList 和 LinkedList 区别

ArrayList 底层是一个动态数组，查改效率较高；

LinkedList 底层是一个双向链表，增删效率较高；

（1）数据结构实现：ArrayList 底层是一个动态数组，而 LinkedList 底层是一个双向链表。

（2）增删效率：LinkedList 要比 ArrayList 效率要高，因为 ArrayList 增删操作要影响数组内的其他数据的下标。

（3）改查效率：ArrayList 比 LinkedList 效率高, 因为 LinkedList 是线性的数据存储方式，所以需要移动指针从前往后依次查找。

7. Set 接口

- **无序(添加和取出的顺序不一致)，没有索引**
- **不允许重复**元素，所以最多包含一个 null

注意：取出的顺序的顺序虽然不是添加的顺序，但是他的固定.

8. HashSet

(1) 基本介绍

- HashSet 实现了 Set 接口
- HashSet 底层基于 HashMap 实现的
- 可以存放 null 值, 但是只能有一个
- 不能有重复对象

(2) HashSet 添加过程

HashSet 底层是 HashMap, HashMap 底层是(数组+ 链表+红黑树)

1. 首次扩容:

   先判断数组是否为空，若数组为空则进行第一次扩容（resize）；

2. 计算索引:

   通过 hash 算法，计算键值对在数组中的索引;

3. 插入数据:

   如果当前位置元素为空，则直接插入数据；

   如果当前位置元素非空，且 key 已存在，则直接覆盖其 value；

如果当前位置元素非空，且 key 不存在，则将数据链到链表末端；

若链表长度达到 8，则将链表转换成红黑树，并将数据插入树中；

4. 再次扩容

如果数组中元素个数（size）超过 threshold，则再次进行扩容操作。

9. HashMap 和 HashTable 区别

（1）HashMap 线程不安全，效率高；Hashtable 线程安全，效率低。

（2）HashMap 可以存储 null 的 key 和 value，但 null 作为键只能有一个，null 作为值可以有多个；Hashtable 不允许有 null 键和 null 值，否则会抛出 NullPointerException。

10. HashMap 和 HashSet 区别

HashSet 底层就是基于 HashMap 实现的，HashSet 调用 add()方法向 Set 中添加元素，其实就是调用 HashMap 的 put()方法向 map 中添加元素

11. HashMap 和 TreeMap 区别

相比于 HashMap 来说 TreeMap 主要多了对集合中的元素根据键排序的能力以及对集合内元素的搜索的能力。TreeMap 适合对一个有序的 key 集合进行遍历。

12. HashMap (HashSet)底层实现（HashMap 的 Put 方法）

HashMap 基于 Hash 算法实现的

1. 往 HashMap 中 put 元素时，根据 key 通过 hash 算法与与运算，计算当前对象的元素在数组中的下标;

2. 存储元素时，

- 如果当前下标位置元素为空，则直接插入数据；

- 如果当前下标位置元素非空，首先判断当前下标位置上的 node 类型，看是红黑树还是链表

  如果 key 已存在，则直接覆盖原始值 value；

  如果 key 不存在，则将数据添加到红黑树中或者链到链表末端；

3. 在添加元素的过程中，若链表长度达到 8，并且 table 的大小>=64，则将链表转换成红黑树，并将数据插入树中；

13 为什么要右移 16 位?

HashMap 方法 put 元素时，通过传入的值获取 hashcode 与 hashcode 的无符号右移 16 位，再做异或运算，为了减少 hash 碰撞，让输入值分布的更加均匀。

14. HashSet 如何检查重复?

不仅要比较 hash 值，同时还要结合 equals 方法比较。

当加入对象时，HashSet 会先计算对象的 hashcode 值来判断对象加入的位置，同时也会与其他加入的对象的 hash 值作比较，如果没有相符的 hash，HashSet 会假设对象没有重复出现。但是如果发现有相同 hash 值的对象，这时会调用 equals()方法来检查 hash 值 相等的对象是否真的相同。如果两者相同，HashSet 就放弃添加，不相同就添加到最后。

15. HashSet 的扩容机制

（1）HashSet 底层是 HashMap,第一次添加时，table 数组扩容到 16，临界值(threshold)是 16*加载因子， (loadFactor)是 0.75 = 12

（2）如果 table 数组使用到了临界值 12,就会扩容到 16* 2=32,新的临界值就是 32 * 0.75 =24,依次类推

（3）在 Java8 中,如果一条链表的元素个数到达 TREEIFY_THRESHOLD(默认是 8).并且 table 的大小>=MIN TREEIFY CAPACITY(默认 64),就会进行树化(红黑树)，否则仍然采用数组扩容机制

16. Iterator 迭代器

(1) Iterator 接口提供遍历任何 Collection 的接口。我们可以从一个 Collection 中使用迭代器方法来获取迭代器实例

(2) Iterator 对象称为迭代器,主要用于遍历 Collection 中的元素

(3) 所有实现了 Collection 接口的集合类都有一个 iterator()方法，用以返回一个实现了 Iterator 接口的对象，即可以返回一个迭代器。

```java
List<String> list = new ArrayList<>();
Iterator<String> it = list.iterator(); //得到一个集合的迭代器
while(it. hasNext()){ //判断是否还有下一个元素
  String obj = it. next(); //下移,将下移以后集合位置上的元素返回
  System. out. println(obj);
}
```

Java

注：Iterator 的特点是只能单向遍历，但是更加安全，因为它可以确保，在当前遍历的集合元素被更改的时候，就会抛出 ConcurrentModificationException 异常。

17. Iterator 和 ListIterator 区别?

（1）Iterator 可以遍历 Set 和 List 集合，而 ListIterator 只能遍历 List。

（2）Iterator 只能单向遍历，而 ListIterator 可以双向遍历（向前/后遍历）。

（3）ListIterator 从 Iterator 接口继承，然后添加了一些额外的功能，比如添加一个元素、替换一个元素、获取前面或后面元素的索引位置。

18. HashMap 查询时间复杂度?

（1）对于指定下标的查找，理想状态下是 O(1)

（2）通过给定值进行查找，单条链表查询时，时间复杂度为 O(n)；当链表长度大于 8 时，红黑树查询时间复杂度为 O(logN)，与二分查找类似。

19. LinkedList 和 ArrayList 的插入时间复杂度?

（1）LinkedList 仅仅在头尾插入或者删除元素的时候时间复杂度近似 O(1)，其他情况增删元素的时间复杂度都是 O(n)，因为 LinkedList 在插入或删除操作前，需要通过 for 循环找到该处索引的元素。

（2）ArrayList 添加元素时，直接在尾部添加，复杂度为 O(1)；指定位置添加元素，复杂度 O(n)。