[Java集合类: Set、List、Map、Queue使用场景梳理](https://www.cnblogs.com/LittleHann/p/3690187.html)（全）[**http://www.cnblogs.com/LittleHann/p/3690187.html**](http://www.cnblogs.com/LittleHann/p/3690187.html)

[**http://wiki.jikexueyuan.com/project/java-collection/linkedhashmap.html**](http://wiki.jikexueyuan.com/project/java-collection/linkedhashmap.html)

**Collection 和 Collections的区别。**

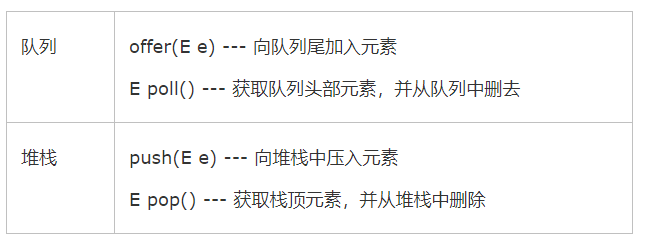
Collections是个java.util下的类，它包含有各种有关集合操作的静态方法。Collections是一个集合框架的帮助类，里面包含一些对集合的排序，搜索以及序列化的操作。最根本的是Collections 是一个包装类。它包含有各种有关集合操作的静态多态方法。此类不能实例化，就像一个工具类，服务于Java的Collection框架

Collection 是一个集合接口。它提供了对集合对象进行基本操作的通用接口方法

# 队列和栈

队列（queue），先进先出

堆栈（stack），后进先出



# 堆和栈

Java堆和栈的区别

java中堆和栈的区别自然是面试中的常见问题，下面几点就是其具体的区别

 1、各司其职

最主要的区别就是栈内存用来存储局部变量和方法调用。

而堆内存用来存储Java中的对象。无论是成员变量，局部变量，还是类变量，它们指向的对象都存储在堆内存中。

 2、独有还是共享

栈内存归属于单个线程，每个线程都会有一个栈内存，其存储的变量只能在其所属线程中可见，即栈内存可以理解成线程的私有内存。而堆内存中的对象对所有线程可见。堆内存中的对象可以被所有线程访问。

 3、异常错误

如果栈内存没有可用的空间存储方法调用和局部变量，JVM会抛出java.lang.StackOverFlowError。

而如果是堆内存没有可用的空间存储生成的对象，JVM会抛出java.lang.OutOfMemoryError。

4、空间大小

栈的内存要远远小于堆内存，如果你使用递归的话，那么你的栈很快就会充满。如果递归没有及时跳出，很可能发生StackOverFlowError问题。

你可以通过-Xss选项设置栈内存的大小。-Xms选项可以设置堆的开始时的大小，-Xmx选项可以设置堆的最大值。

# 哈希表原理

哈希算法也称为散列算法，是将数据依特定算法直接指定到一个地址上

当集合要添加新的元素时，先调用这个元素的hashCode方法，就一下子能定位到它应该放置的物理位置上。 如果这个位置上没有元素，它就可以直接存储在这个位置上，不用再进行任何比较了；如果这个位置上已经有元素了， 就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址。

Java对于eqauls方法和hashCode方法是这样规定的：

1、如果两个对象相同，那么它们的hashCode值一定要相同；

2、如果两个对象的hashCode相同，它们并不一定相同

# Set

1.不允许重复对象

2.允许添加空值，只能添加一个

3.无序容器，

4.Set 接口最流行的几个实现类是 HashSet、LinkedHashSet 以及 TreeSet。最流行的是基于 HashMap 实现的 HashSet；TreeSet 还实现了 SortedSet 接口，因此 TreeSet 是一个根据其 compare() 和 compareTo() 的定义进行排序的有序容器。

## HashSet

线程不安全，存取速度快

底层实现是一个HashMap（保存数据），实现Set接口

默认初始容量为16（为何是16，见下方对HashMap的描述）

加载因子为0.75：即当 元素个数 超过 容量长度的0.75倍 时，进行扩容

扩容增量：原容量的 1 倍

如 HashSet的容量为16，一次扩容后是容量为32

不能保证元素的排列顺序，顺序有可能发生变化

集合元素可以是null,但只能放入一个null

注意，如果要把一个对象放入HashSet中，重写该对象对应类的equals方法，也应该重写其hashCode()方法。其规则是如果两个对象通过equals方法比较返回true时，其 hashCode也应该相同。另外，对象中用作equals比较标准的属性，都应该用来计算 hashCode的值。

## TreeSet是 SortedSet接口的实现类

TreeSet类型是J2SE中唯一可实现自动排序的类型

TreeSet是SortedSet接口的唯一实现类，TreeSet可以确保集合元素处于排序状态。TreeSet支持两种排序方式，自然排序 和定制排序。

自然排序为默认的排序方式根据集合元素的大小，以升序排列。向 TreeSet中加入的应该是同一个类的对象。TreeSet判断两个对象不相等的方式是两个对象通过equals方法返回false，或者通过CompareTo方法比较没有返回0。

定制排序，应该使用Comparator接口，实现 int compare(To1,To2)方法

## LinkedHashSet

LinkedHashSet集合同样是根据元素的hashCode值来决定元素的存储位置，但是它同时使用链表维护元素的次序。这样使得元素看起 来像是以插入顺 序保存的，也就是说，当遍历该集合时候，LinkedHashSet将会以元素的添加顺序访问集合的元素。

LinkedHashSet在迭代访问Set中的全部元素时，性能比HashSet好，但是插入时性能稍微逊色于HashSet。

# List

1.允许重复的对象

2.可以添加空值，并且可以添加多个

3.有序的容器，保持插入顺序，输出顺序 就是插入顺序

4.常用的实现类有 ArrayList、LinkedList 和 Vector。ArrayList 最为流行，它提供了使用索引的随意访问，而 LinkedList 则对于经常需要从 List 中添加或删除元素的场合更为合适。

## ArrayList

元素有序，可重复

ArrayList 是一个数组队列，相当于 动态数组。与Java中的数组相比，它的容量能动态增长

ArrayList中的操作不是线程安全的！所以，建议在单线程中才使用ArrayList，而在多线程中可以选择Vector或者CopyOnWriteArrayList

数组的结构，读不加锁，但是写的时候加锁，并复制一份新数据，再添加新数据的时候，会有两个对象在内存中，在remove时，会将数据对象数据设置为null，帮助GC。数据的一致性，就是在最终会保证数据的一致性，但是不能实时保证数据的一致性。

在jdk1.6中，初始化容量值为10，无最大值，扩容为原来的1.5倍。

在jdk1.7中，初始化容量值为0，添加数据时为10，最大值为2的32次方-8，

add操作可以理解为直接将数组的内容置位，remove操作可以理解为删除index为0的节点，并将后面元素移到0处

## LinkedList

元素有序，可重复，不是线程安全的

LinkedList包含两个重要的成员：header 和 size。

header是双向链表的表头，它是双向链表节点所对应的类Entry的实例。Entry中包含成员变量： previous, next, element。其中，previous是该节点的上一个节点，next是该节点的下一个节点，element是该节点所包含的值。

查找某个位置的节点

当我们调用get(int location)时，首先会比较“location”和“双向链表长度的1/2”；若前者大，则从链表头开始往后查找，直到location位置；否则，从链表末尾开始先前查找，直到location位置

　　size是双向链表中节点的个数。

LinkedList 是一个继承于AbstractSequentialList的双向链表。它也可以被当作堆栈、队列或双端队列进行操作。

## Vector

底层是数组实现的和ArrayList一样

初始默认长度都为10

Vector的方法都是同步的(Synchronized),是线程安全的，所以速度慢

当Vector中的元素超过它的初始大小时,Vector会将它的容量翻倍

# Map

1.map的每个entry由键值对组成，键对象必须唯一，值可重复

2.允许有多个空值，但只能有一个空值键

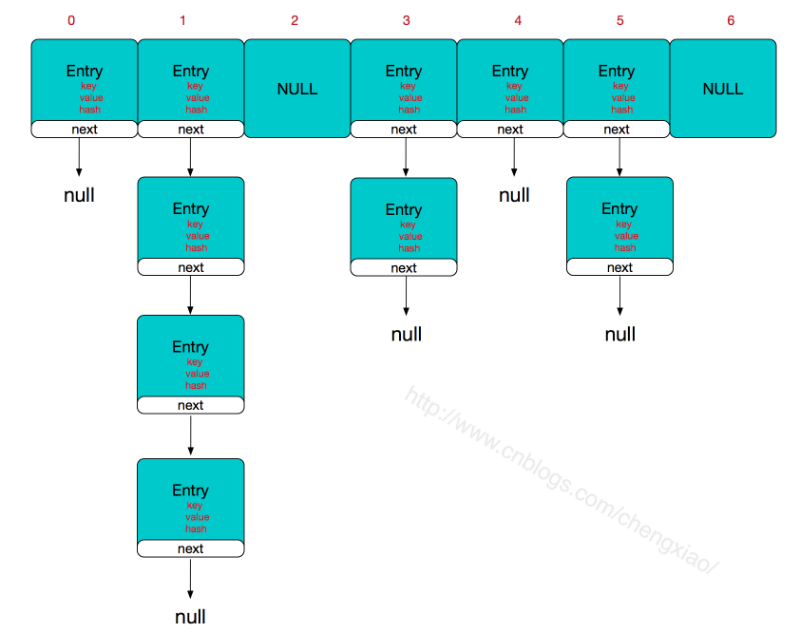
3.HashMap是无序的，TreeMap默认是升序 和LinkedHashMap是记录插入顺序

4.Map 接口最流行的几个实现类是 HashMap、LinkedHashMap、Hashtable 和 TreeMap。（HashMap、TreeMap最常用）

## HashMap

不是线程安全的

HashMap的主干是一个Entry数组。Entry是HashMap的基本组成单元，每一个Entry包含一个key-value键值对



HashMap由数组+链表组成的，数组是HashMap的主体，链表则是主要为了解决哈希冲突而存在的，如果定位到的数组位置不含链表（当前entry的next指向null）,那么对于查找，添加等操作很快，仅需一次寻址即可；如果定位到的数组包含链表，对于添加操作，其时间复杂度为O(n)，首先遍历链表，存在即覆盖，否则新增；对于查找操作来讲，仍需遍历链表，然后通过key对象的equals方法逐一比对查找。所以，性能考虑，HashMap中的链表出现越少，性能才会越好。

### 扩容机制

当发生哈希冲突并且size大于阈值的时候，需要进行数组扩容，扩容时，需要新建一个长度为之前数组2倍的新的数组，然后将当前的Entry数组中的元素全部传输过去，扩容后的新数组长度为之前的2倍，所以扩容相对来说是个耗资源的操作。

## Put操作

put 元素的时候，先根据 key 的 hashCode 重新计算 hash 值，根据 hash 值得到这个元素在数组中的位置（即下标），如果数组该位置上已经存放有其他元素了，那么在这个位置上的元素将以链表的形式存放，***新加入的放在链头，最先加入的放在链尾***。如果数组该位置上没有元素，就直接将该元素放到此数组中的该位置上

## Get操作

需要根据key的hashcode算出元素在数组中的下标,之后遍历Entry对象链表,直到找到元素为止。

### 字段说明

Table（entry 数组长度为2的32次幂）

Size hashmap保存键值对数量

Threshold hashmap容量阈值=容量x加载因子

Loadfactor 加载因子

Modcount 修改次数

默认的加载因子 0.75

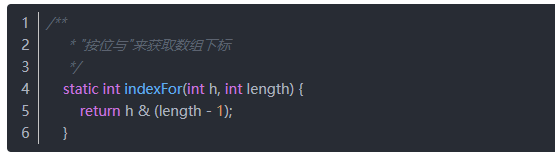
初始化容量 16

|  |
| --- |
| Put操作过程  1. 判断是否为null，如果为null，则放在table[0]的位置 2. 计算hash值，是根据key的hashcode计算，key.hashcode()。 3. 计算索引位置index，可以根据除留余数法 4. 遍历链表，key是否存在，如果存在则用新的值替代，   判断hashcode是否相等，  判断==或者equals   1. 修改modcount++ 2. 键值不存在，添加并返回   判断键值是否够用，如果够用，不用扩容否则扩容为2的n次方 |

### 工作原理：

将键值对传递给put()方法时，它调用键对象的hashCode()方法来计算hashcode，让后找到bucket位置来储存Entry对象。当获取对象时，通过键对象的equals()方法找到正确的键值对，然后返回值对象。HashMap使用链表来解决碰撞问题，当发生碰撞了，对象将会储存在链表的下一个节点中。 HashMap在每个链表节点中储存键值对对象。

hashmap始终将自己的桶保持在2的n次方，这是为什么？

indexFor这个方法解释了这个问题

计算机里面位运算是基本运算，位运算的效率是远远高于取余%运算的

## Hashtable

效率低 ,内部实现和HashMap类似，线程安全的（其实是直接在方法上增加了synchronized关键字，比较古老，落后，低效的同步方式），还有就是它的key、value都不为null，**但如在Hashtable中有类似put(null,null)的操作，编译同样可以通过，因为key和value都是Object类型，但运行时会抛出NullPointerException异常，这是JDK的规范规定的。**

默认加载因子也是 0.75，HashTable在不指定容量的情况下的默认容量为11，而HashMap为16，Hashtable不要求底层数组的容量一定要为2的整数次幂，而HashMap则要求一定为2的整数次幂。因为HashTable是直接使用除留余数法定位地址。且Hashtable计算hash值，直接用key的hashCode()。

Hashtable扩容时，将容量变为原来的2倍加1，而HashMap扩容时，将容量变为原来的2倍。

## LinkedHashMap

不是线程安全的，LinkedHashMap 是 HashMap 的一个子类，它保留插入的顺序。

LinkedHashMap 采用的 hash 算法和 HashMap 相同，但是它重新定义了数组中保存的元素 Entry，该 Entry 除了保存当前对象的引用外，还保存了其上一个元素 before 和下一个元素 after 的引用，从而在哈希表的基础上又构成了***双向链接列表***

根据链表中元素的顺序可以分为：按插入顺序的链表，和按访问顺序(调用 get 方法)的链表。默认是按插入顺序排序，如果指定按访问顺序排序，那么调用get方法后，会将这次访问的元素移至链表尾部，不断访问可以形成按访问顺序排序的链表。

## TreeMap

有序的，非线程安全的 ，红黑树实现的。

TreeMap 默认排序规则：按照key的字典顺序来排序（升序）

当然，也可以自定义排序规则：要实现Comparator接口。

HashMap存取元素的时间复杂度是O(1)的常量级，而TreeMap对元素的操作复杂度为O(log n)。虽然在操作性能方面，TreeMap不占优势，但是因为它使用红黑树（平衡二叉查找树）实现，所以它内部的元素都是排好序的。当需要查找的元素是排好序的，TreeMap的优势就体现出来了。

TreeMap包含了tree和Map的性质，它所包含的的Entry类应该有以下6个成员变量：

左子结点引用、右子节点引用、父节点引用、key、value、color。

