<https://mp.weixin.qq.com/s?__biz=MzAwOTE3NDY5OA==&mid=2647911615&idx=4&sn=44a48f3d9815c5546be77f8109e5287c&chksm=8344e77ab4336e6ce7d818f405f70e5695cdbf2bc1932643f8e8e360410526da61a16027060d&scene=21#wechat_redirect>

# 十个经典Java 集合面试题

## 1.Java的HashMap是如何工作的？

HashMap是一个针对数据结构的键值，每个键都会有相应的值，关键是识别这样的值。HashMap 基于 hashing 原理，我们通过 put ()和 get ()方法储存和获取对象。当我们将键值对传递给 put ()方法时，它调用键对象的 hashCode ()方法来计算 hashcode，让后找到 bucket 位置来储存值对象。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 LinkedList 来解决碰撞问题，当发生碰撞了，对象将会储存在 LinkedList 的下一个节点中。HashMap 在每个 LinkedList 节点中储存键值对对象。

## 2.什么是快速失败的故障安全迭代器？

快速失败的Java迭代器可能会引发ConcurrentModifcationException在底层集合迭代过程中被修改。故障安全作为发生在实例中的一个副本迭代是不会抛出任何异常的。

快速失败的故障安全范例定义了当遭遇故障时系统是如何反应的。例如，用于失败的快速迭代器ArrayList和用于故障安全的迭代器ConcurrentHashMap。

## 3.Java BlockingQueue是什么？

Java BlockingQueue是一个并发集合util包的一部分。BlockingQueue队列是一种支持操作，它等待元素变得可用时来检索，同样等待空间可用时来存储元素。

## 4.什么时候使用ConcurrentHashMap？

在问题2中我们看到ConcurrentHashMap被作为故障安全迭代器的一个实例，它允许完整的并发检索和更新。当有大量的并发更新时，ConcurrentHashMap此时可以被使用。

这非常类似于Hashtable，但ConcurrentHashMap不锁定整个表来提供并发，所以从这点上ConcurrentHashMap的性能似乎更好一些。所以当有大量更新时ConcurrentHashMap应该被使用。

## 5.哪一个List实现了最快插入？

LinkedList和ArrayList是另个不同变量列表的实现。ArrayList的优势在于动态的增长数组，非常适合初始时总长度未知的情况下使用。LinkedList的优势在于在中间位置插入和删除操作，速度是最快的。

LinkedList实现了List接口，允许null元素。此外LinkedList提供额外的get，remove，insert方法在LinkedList的首部或尾部。这些操作使LinkedList可被用作堆栈（stack），队列（queue）或双向队列（deque）。

ArrayList实现了可变大小的数组。它允许所有元素，包括null。每个ArrayList实例都有一个容量（Capacity），即用于存储元素的数组的大小。这个容量可随着不断添加新元素而自动增加，但是增长算法并没有定义。当需要插入大量元素时，在插入前可以调用ensureCapacity方法来增加ArrayList的容量以提高插入效率。

## 6.Iterator和ListIterator的区别

●ListIterator有add()方法，可以向List中添加对象，而Iterator不能。

●ListIterator和Iterator都有hasNext()和next()方法，可以实现顺序向后遍历，但是ListIterator有hasPrevious()和previous()方法，可以实现逆向（顺序向前）遍历。Iterator就不可以。

●ListIterator可以定位当前的索引位置，nextIndex()和previousIndex()可以实现。Iterator没有此功能。

●都可实现删除对象，但是ListIterator可以实现对象的修改，set()方法可以实现。Iterator仅能遍历，不能修改。

## 7.什么是CopyOnWriteArrayList，它与ArrayList有何不同？

CopyOnWriteArrayList是ArrayList的一个线程安全的变体，其中所有可变操作（add、set等等）都是通过对底层数组进行一次新的复制来实现的。相比较于ArrayList它的写操作要慢一些，因为它需要实例的快照。

CopyOnWriteArrayList中写操作需要大面积复制数组，所以性能肯定很差，但是读操作因为操作的对象和写操作不是同一个对象，读之间也不需要加锁，读和写之间的同步处理只是在写完后通过一个简单的'='将引用指向新的数组对象上来，这个几乎不需要时间，这样读操作就很快很安全，适合在多线程里使用，绝对不会发生ConcurrentModificationException ，因此CopyOnWriteArrayList适合使用在读操作远远大于写操作的场景里，比如缓存。

## 8.迭代器和枚举之间的区别

**如果面试官问这个问题，那么他的意图一定是让你区分Iterator不同于Enumeration的两个方面：**

●Iterator允许移除从底层集合的元素。

●Iterator的方法名是标准化的。

## 9.Hashmap如何同步?

**当我们需要一个同步的HashMap时，有两种选择：**

●使用Collections.synchronizedMap（..）来同步HashMap。

●使用ConcurrentHashMap的

这两个选项之间的首选是使用ConcurrentHashMap，这是因为我们不需要锁定整个对象，以及通过ConcurrentHashMap分区地图来获得锁。

## 10.IdentityHashMap和HashMap的区别

IdentityHashMap是Map接口的实现。不同于HashMap的，这里采用参考平等。

●在HashMap中如果两个元素是相等的，则key1.equals(key2)

●在IdentityHashMap中如果两个元素是相等的，则**key1 == key2**