


- [首页](#)
- [所有文章](#)
- [资讯](#)
- [Web](#)
- [架构](#)
- [基础技术](#)
- [书籍](#)
- [教程](#)
- [Java小组](#)
- [工具资源](#)

- 导航条 - 

## 40个Java集合面试问题和答案

2015/05/19 | 分类: [基础技术](#), [职业生涯](#) | [0 条评论](#) | 标签: [Java](#), [面试](#)

分享到:

124 译文出处: [Sanesee](#) 原文出处: [javacodegeeks](#)

### 1.Java集合框架是什么？说出一些集合框架的优点？



每种编程语言中都有集合，最初的Java版本包含几种集合类：Vector、Stack、HashTable和Array。随着集合的广泛使用，Java1.2提出了囊括所有集合接口、实现和算法的集合框架。在保证线程安全的情况下使用泛型和并发集合类，Java已经经历了很久。它还包括在Java并发包中，阻塞接口以及它们的实现。集合框架的部分优点如下：

- (1) 使用核心集合类降低开发成本，而非实现我们自己的集合类。
- (2) 随着使用经过严格测试的集合框架类，代码质量会得到提高。
- (3) 通过使用JDK附带的集合类，可以降低代码维护成本。
- (4) 复用性和可操作性。

### 2.集合框架中的泛型有什么优点？

Java1.5引入了泛型，所有的集合接口和实现都大量地使用它。泛型允许我们为集合提供一个可以容纳的对象类型，因此，如果你添加其它类型的任何元素，它会在编译时报错。这避免了在运行时出现ClassCastException，因为你将会在编译时得到报错信息。泛型也使得代码整洁，我们不需要使用显式转换和instanceOf操作符。它也给运行时带来好处，因为不会产生类型检查的字节码指令。

### 3.Java集合框架的基础接口有哪些？

Collection为集合层级的根接口。一个集合代表一组对象，这些对象即为它的元素。Java平台不提供这个接口任何直接的实现。

Set是一个不能包含重复元素的集合。这个接口对数学集合抽象进行建模，被用来代表集合，就如一副牌。

List是一个有序集合，可以包含重复元素。你可以通过它的索引来访问任何元素。List更像长度动态变换的数组。

Map是一个将key映射到value的对象。一个Map不能包含重复的key：每个key最多只能映射一个value。

一些其它的接口有Queue、Deque、SortedSet、SortedMap和ListIterator。

#### 4.为何Collection不从Cloneable和Serializable接口继承？

Collection接口指定一组对象，对象即为它的元素。如何维护这些元素由Collection的具体实现决定。例如，一些如List的Collection实现允许重复的元素，而其它的如Set就不允许。很多Collection实现有一个公有的clone方法。然而，把它放到集合的所有实现中也是没有意义的。这是因为Collection是一个抽象表现。重要的是实现。

当与具体实现打交道的时候，克隆或序列化的语义和含义才发挥作用。所以，具体实现应该决定如何对它进行克隆或序列化，或它是否可以被克隆或序列化。

在所有的实现中授权克隆和序列化，最终导致更少的灵活性和更多的限制。特定的实现应该决定它是否可以被克隆和序列化。

#### 5.为何Map接口不继承Collection接口？



尽管Map接口和它的实现也是集合框架的一部分，但Map不是集合，集合也不是Map。因此，Map继承Collection毫无意义，反之亦然。

如果Map继承Collection接口，那么元素去哪儿？Map包含key-value对，它提供抽取key或value列表集合的方法，但是它不适合“一组对象”规范。

#### 6.Iterator是什么？

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。迭代器取代了Java集合框架中的Enumeration。迭代器允许调用者在迭代过程中移除元素。

#### 7.Enumeration和Iterator接口的区别？

Enumeration的速度是Iterator的两倍，也使用更少的内存。Enumeration是非常基础的，也满足了基础的需要。但是，与Enumeration相比，Iterator更加安全，因为当一个集合正在被遍历的时候，它会阻止其它线程去修改集合。

迭代器取代了Java集合框架中的Enumeration。迭代器允许调用者从集合中移除元素，而Enumeration不能做到。为了使它的功能更加清晰，迭代器方法名已经经过改善。

#### 8.为何没有像Iterator.add()这样的方法，向集合中添加元素？

语义不明，已知的是，Iterator的协议不能确保迭代的次序。然而要注意，ListIterator没有提供一个add操作，它要确保迭代的顺序。

## 9.为何迭代器没有一个方法可以直接获取下一个元素，而不需要移动游标？

它可以在当前Iterator的顶层实现，但是它用得很少，如果将它加到接口中，每个继承都要去实现它，这没有意义。

## 10.Iterator和ListIterator之间有什么区别？

- (1) 我们可以使用Iterator来遍历Set和List集合，而ListIterator只能遍历List。
- (2) Iterator只可以向前遍历，而ListIterator可以双向遍历。
- (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。

## 11.遍历一个List有哪些不同的方式？

```
1 List<String> strList = new ArrayList<>();
2 //使用for-each循环
3 for(String obj : strList){
4     System.out.println(obj);
5 }
6 //using iterator
7 Iterator<String> it = strList.iterator();
8 while(it.hasNext()){
9     String obj = it.next();
10    System.out.println(obj);
11 }
```

使用迭代器更加线程安全，因为它可以确保，在当前遍历的集合元素被更改的时候，它会抛出ConcurrentModificationException。



## 12.通过迭代器fail-fast属性，你明白了什么？

每次我们尝试获取下一个元素的时候，Iterator fail-fast属性检查当前集合结构里的任何改动。如果发现任何改动，它抛出ConcurrentModificationException。Collection中所有Iterator的实现都是按fail-fast来设计的（ConcurrentHashMap和CopyOnWriteArrayList这类并发集合类除外）。

## 13.fail-fast与fail-safe有什么区别？

Iterator的fail-fast属性与当前的集合共同起作用，因此它不会受到集合中任何改动的影响。Java.util包中的所有集合类都被设计为fail-fast的，而java.util.concurrent中的集合类都为fail-safe的。Fail-fast迭代器抛出ConcurrentModificationException，而fail-safe迭代器从不抛出ConcurrentModificationException。

## 14.在迭代一个集合的时候，如何避免ConcurrentModificationException？

在遍历一个集合的时候，我们可以使用并发集合类来避免ConcurrentModificationException，比如使用CopyOnWriteArrayList，而不是ArrayList。

## 15.为何Iterator接口没有具体的实现？

Iterator接口定义了遍历集合的方法，但它的实现则是集合实现类的责任。每个能够返回用于遍历的Iterator的集合类都有它自己的Iterator实现内部类。

这就允许集合类去选择迭代器是fail-fast还是fail-safe的。比如，ArrayList迭代器是fail-fast的，而CopyOnWriteArrayList迭代器是fail-safe的。

## 16.UnsupportedOperationException是什么？

UnsupportedOperationException是用于表明操作不支持的异常。在JDK类中已被大量运用，在集合框架java.util.Collections.UnmodifiableCollection将会在所有add和remove操作中抛出这个异常。

## 17.在Java中，HashMap是如何工作的？

HashMap在Map.Entry静态内部类实现中存储key-value对。HashMap使用哈希算法，在put和get方法中，它使用hashCode()和equals()方法。当我们通过传递key-value对调用put方法的时候，HashMap使用Key hashCode()和哈希算法来找出存储key-value对的索引。Entry存储在LinkedList中，所以如果存在entry，它使用equals()方法来检查传递的key是否已经存在，如果存在，它会覆盖value，如果不存在，它会创建一个新的entry然后保存。当我们通过传递key调用get方法时，它再次使用hashCode()来找到数组中的索引，然后使用equals()方法找出正确的Entry，然后返回它的值。下面的图片解释了详细内容。

其它关于HashMap比较重要的问题是容量、负荷系数和阈值调整。HashMap默认的初始容量是32，负荷系数是0.75。阈值是为负荷系数乘以容量，无论何时我们尝试添加一个entry，如果map的大小比阈值大的时候，HashMap会对map的内容进行重新哈希，且使用更大的容量。容量总是2的幂，所以如果你知道你需要存储大量的key-value对，比如缓存从数据库里面拉取的数据，使用正确的容量和负荷系数对HashMap进行初始化是个不错的做法。

## 18.hashCode()和equals()方法有何重要性？

HashMap使用Key对象的hashCode()和equals()方法去决定key-value对的索引。当我们试着从HashMap中获取值的时候，这些方法也会被用到。如果这些方法没有被正确地实现，在这种情况下，两个不同Key也许会产生相同的hashCode()和equals()输出，HashMap将会认为它们是相同的，然后覆盖它们，而非把它们存储到不同的地方。同样的，所有不允许存储重复数据的集合类都使用hashCode()和equals()去查找重复，所以正确实现它们非常重要。equals()和hashCode()的实现应该遵循以下规则：

- (1) 如果o1.equals(o2)，那么o1.hashCode() == o2.hashCode()总是为true的。
- (2) 如果o1.hashCode() == o2.hashCode()，并不意味着o1.equals(o2)会为true。

## 19.我们能否使用任何类作为Map的key？

我们可以使用任何类作为Map的key，然而在使用它们之前，需要考虑以下几点：

- (1) 如果类重写了equals()方法，它也应该重写hashCode()方法。
- (2) 类的所有实例需要遵循与equals()和hashCode()相关的规则。请参考之前提到的这些规则。
- (3) 如果一个类没有使用equals()，你不应该在hashCode()中使用它。
- (4) 用户自定义key类的最佳实践是使之为不可变的，这样，hashCode()值可以被缓存起来，拥有更好的性能。不可变的类也可以确保hashCode()和equals()在未来不会改变，这样就会解决与可变相关的问题了。

比如，我有一个类MyKey，在HashMap中使用它。

```
1 //传递给MyKey的name参数被用于equals()和hashCode()中
2 MyKey key = new MyKey('Pankaj'); //assume hashCode=1234
3 myHashMap.put(key, 'Value');
4 // 以下的代码会改变key的hashCode()和equals()值
5 key.setName('Amit'); //assume new hashCode=7890
6 //下面会返回null，因为HashMap会尝试查找存储同样索引的key，而key已被改变了，匹配失败，返回null
7 myHashMap.get(new MyKey('Pankaj'));
```

那就是为何String和Integer被作为HashMap的key大量使用。

## 20.Map接口提供了哪些不同的集合视图？

Map接口提供三个集合视图：

(1) Set keyset(): 返回map中包含的所有key的一个Set视图。集合是受map支持的，map的变化会在集合中反映出来，反之亦然。当一个迭代器正在遍历一个集合时，若map被修改了（除迭代器自身的移除操作以外），迭代器的结果会变为未定义。集合支持通过Iterator的Remove、Set.remove、removeAll、retainAll和clear操作进行元素移除，从map中移除对应的映射。它不支持add和addAll操作。

(2) Collection values(): 返回一个map中包含的所有value的一个Collection视图。这个collection受map支持的，map的变化会在collection中反映出来，反之亦然。当一个迭代器正在遍历一个collection时，若map被修改了（除迭代器自身的移除操作以外），迭代器的结果会变为未定义。集合支持通过Iterator的Remove、Set.remove、removeAll、retainAll和clear操作进行元素移除，从map中移除对应的映射。它不支持add和addAll操作。

(3) Set<Map.Entry<K,V>> entrySet(): 返回一个map中包含的所有映射的一个集合视图。这个集合受map支持的，map的变化会在collection中反映出来，反之亦然。当一个迭代器正在遍历一个集合时，若map被修改了（除迭代器自身的移除操作，以及对迭代器返回的entry进行setValue外），迭代器的结果会变为未定义。集合支持通过Iterator的Remove、Set.remove、removeAll、retainAll和clear操作进行元素移除，从map中移除对应的映射。它不支持add和addAll操作。

## 21.HashMap和HashTable有何不同？

(1) HashMap允许key和value为null，而HashTable不允许。

(2) HashTable是同步的，而HashMap不是。所以HashMap适合单线程环境，HashTable适合多线程环境。

(3) 在Java1.4中引入了LinkedHashMap，HashMap的一个子类，假如你想要遍历顺序，你很容易从HashMap转向LinkedHashMap，但是HashTable不是这样的，它的顺序是不可预知的。

(4) HashMap提供对key的Set进行遍历，因此它是fail-fast的，但HashTable提供对key的Enumeration进行遍历，它不支持fail-fast。

(5) HashTable被认为是个遗留的类，如果你寻求在迭代的时候修改Map，你应该使用ConcurrentHashMap。

## 22.如何决定选用HashMap还是TreeMap？

对于在Map中插入、删除和定位元素这类操作，HashMap是最好的选择。然而，假如你需要对一个有序的key集合进行遍历，TreeMap是更好的选择。基于你的collection的大小，也许向HashMap中

添加元素会更快，将map换为TreeMap进行有序key的遍历。

## 23.ArrayList和Vector有何异同点？

ArrayList和Vector在很多时候都很类似。

- (1) 两者都是基于索引的，内部由一个数组支持。
- (2) 两者维护插入的顺序，我们可以根据插入顺序来获取元素。
- (3) ArrayList和Vector的迭代器实现都是fail-fast的。
- (4) ArrayList和Vector两者允许null值，也可以使用索引值对元素进行随机访问。

以下是ArrayList和Vector的不同点。

- (1) Vector是同步的，而ArrayList不是。然而，如果你寻求在迭代的时候对列表进行改变，你应该使用CopyOnWriteArrayList。
- (2) ArrayList比Vector快，它因为有同步，不会过载。
- (3) ArrayList更加通用，因为我们可以使用Collections工具类轻易地获取同步列表和只读列表。

## 24.Array和ArrayList有何区别？什么时候更适合用Array？

Array可以容纳基本类型和对象，而ArrayList只能容纳对象。

Array是指定大小的，而ArrayList大小是固定的。

Array没有提供ArrayList那么多功能，比如addAll、removeAll和iterator等。尽管ArrayList明显是更好的选择，但也有些时候Array比较好用。

- (1) 如果列表的大小已经指定，大部分情况下是存储和遍历它们。
- (2) 对于遍历基本数据类型，尽管Collections使用自动装箱来减轻编码任务，在指定大小的基本类型的列表上工作也会变得很慢。
- (3) 如果你要使用多维数组，使用[][]比List<List<>>更容易。

## 25.ArrayList和LinkedList有何区别？

ArrayList和LinkedList两者都实现了List接口，但是它们之间有些不同。

- (1) ArrayList是由Array所支持的基于一个索引的数据结构，所以它提供对元素的随机访问，复杂度为O(1)，但LinkedList存储一系列的节点数据，每个节点都与前一个和下一个节点相连接。所以，尽管有使用索引获取元素的方法，内部实现是从起始点开始遍历，遍历到索引的节点然后返回元素，时间复杂度为O(n)，比ArrayList要慢。
- (2) 与ArrayList相比，在LinkedList中插入、添加和删除一个元素会更快，因为在一个元素被插入到中间的时候，不会涉及改变数组的大小，或更新索引。
- (3) LinkedList比ArrayList消耗更多的内存，因为LinkedList中的每个节点存储了前后节点的引用。



## 26.哪些集合类提供对元素的随机访问？

ArrayList、HashMap、TreeMap和HashTable类提供对元素的随机访问。

## 27.EnumSet是什么？

java.util.EnumSet是使用枚举类型的集合实现。当集合创建时，枚举集合中的所有元素必须来自单个指定的枚举类型，可以是显示的或隐式的。EnumSet是不同步的，不允许值为null的元素。它也提供了一些有用的方法，比如copyOf(Collection c)、of(E first,E...rest)和complementOf(EnumSet s)。

## 28.哪些集合类是线程安全的？

Vector、HashTable、Properties和Stack是同步类，所以它们是线程安全的，可以在多线程环境下使用。Java1.5并发API包括一些集合类，允许迭代时修改，因为它们都工作在集合的克隆上，所以它们在多线程环境中是安全的。

## 29.并发集合类是什么？

Java1.5并发包（java.util.concurrent）包含线程安全集合类，允许在迭代时修改集合。迭代器被设计为fail-fast的，会抛出ConcurrentModificationException。一部分类为：CopyOnWriteArrayList、ConcurrentHashMap、CopyOnWriteArraySet。

## 30.BlockingQueue是什么？

Java.util.concurrent.BlockingQueue是一个队列，在进行检索或移除一个元素的时候，它会等待队列变为非空；当在添加一个元素时，它会等待队列中的可用空间。BlockingQueue接口是Java集合框架的一部分，主要用于实现生产者-消费者模式。我们不需要担心等待生产者有可用的空间，或消费者有可用的对象，因为它都在BlockingQueue的实现类中被处理了。Java提供了集中BlockingQueue的实现，比如ArrayBlockingQueue、LinkedBlockingQueue、PriorityBlockingQueue、SynchronousQueue等。

## 31.队列和栈是什么，列出它们的区别？

栈和队列两者都被用来预存储数据。java.util.Queue是一个接口，它的实现类在Java并发包中。队列允许先进先出（FIFO）检索元素，但并非总是这样。Deque接口允许从两端检索元素。

栈与队列很相似，但它允许对元素进行后进先出（LIFO）进行检索。

Stack是一个扩展自Vector的类，而Queue是一个接口。

## 32.Collections类是什么？

Java.util.Collections是一个工具类仅包含静态方法，它们操作或返回集合。它包含操作集合的多态算法，返回一个由指定集合支持的新集合和其它一些内容。这个类包含集合框架算法的方法，比如折半搜索、排序、混编和逆序等。

## 33.Comparable和Comparator接口是什么？

如果我们想使用Array或Collection的排序方法时，需要在自定义类里实现Java提供Comparable接口。Comparable接口有compareTo(T OBJ)方法，它被排序方法所使用。我们应该重写这个方法，如果“this”对象比传递的对象参数更小、相等或更大时，它返回一个负整数、0或正整数。但是，在

大多数实际情况下，我们想根据不同参数进行排序。比如，作为一个CEO，我想对雇员基于薪资进行排序，一个HR想基于年龄对他们进行排序。这就是我们需要使用Comparator接口的情景，因为Comparable.compareTo(Object o)方法实现只能基于一个字段进行排序，我们不能根据对象排序的需要选择字段。Comparator接口的compare(Object o1, Object o2)方法的实现需要传递两个对象参数，若第一个参数比第二个小，返回负整数；若第一个等于第二个，返回0；若第一个比第二个大，返回正整数。

### 34.Comparable和Comparator接口有何区别？

Comparable和Comparator接口被用来对对象集合或者数组进行排序。Comparable接口被用来提供对象的自然排序，我们可以使用它来提供基于单个逻辑的排序。

Comparator接口被用来提供不同的排序算法，我们可以选择需要使用的Comparator来对给定的对象集合进行排序。

### 35.我们如何对一组对象进行排序？

如果我们需要对一个对象数组进行排序，我们可以使用Arrays.sort()方法。如果我们需要排序一个对象列表，我们可以使用Collection.sort()方法。两个类都有用于自然排序（使用Comparable）或基于标准的排序（使用Comparator）的重载方法sort()。Collections内部使用数组排序方法，所有它们两者都有相同的性能，只是Collections需要花时间将列表转换为数组。

### 36.当一个集合被作为参数传递给一个函数时，如何才能确保函数不能修改它？

在作为参数传递之前，我们可以使用Collections.unmodifiableCollection(Collection c)方法创建一个只读集合，这将确保改变集合的任何操作都会抛出UnsupportedOperationException。

### 37.我们如何从给定集合那里创建一个synchronized的集合？

我们可以使用Collections.synchronizedCollection(Collection c)根据指定集合来获取一个synchronized（线程安全的）集合。

### 38.集合框架里实现的通用算法有哪些？

Java集合框架提供常用的算法实现，比如排序和搜索。Collections类包含这些方法实现。大部分算法是操作List的，但一部分对所有类型的集合都是可用的。部分算法有排序、搜索、混编、最大最小值。

### 39.大写的O是什么？举几个例子？

大写的O描述的是，就数据结构中的一系列元素而言，一个算法的性能。Collection类就是实际的数据结构，我们通常基于时间、内存和性能，使用大写的O来选择集合实现。比如：例子1：ArrayList的get(index i)是一个常量时间操作，它不依赖list中元素的数量。所以它的性能是O(1)。例子2：一个对于数组或列表的线性搜索的性能是O(n)，因为我们需要遍历所有的元素来查找需要的元素。

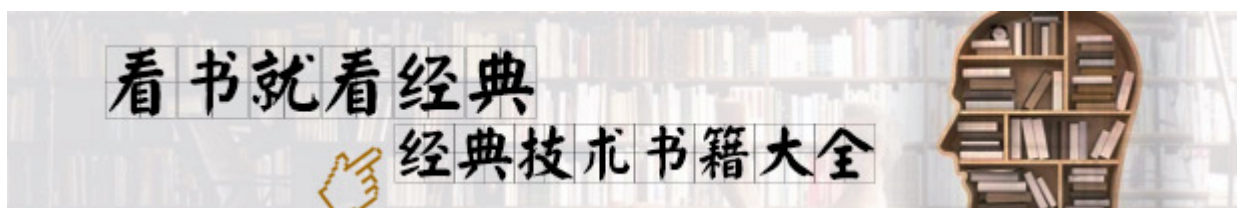
### 40.与Java集合框架相关的有哪些最好的实践？

(1) 根据需求选择正确的集合类型。比如，如果指定了大小，我们会选用Array而非ArrayList。如果我们想根据插入顺序遍历一个Map，我们需要使用TreeMap。如果我们不想重复，我们应该使用Set。



- (2) 一些集合类允许指定初始容量，所以如果我们能够估计到存储元素的数量，我们可以使用它，就避免了重新哈希或大小调整。
- (3) 基于接口编程，而非基于实现编程，它允许我们后来轻易地改变实现。
- (4) 总是使用类型安全的泛型，避免在运行时出现ClassCastException。
- (5) 使用JDK提供的不可变类作为Map的key，可以避免自己实现hashCode()和equals()。
- (6) 尽可能使用Collections工具类，或者获取只读、同步或空的集合，而非编写自己的实现。它将会提供代码重用性，它有着更好的稳定性和可维护性。

124



## 相关文章

- [115个Java面试题和答案——终极列表（上）](#)
- [115个Java面试题和答案——终极列表（下）](#)
- [Java 引用类型简述](#)
- [JAVA 动态代理](#)
- [跟上 Java 8 — 你忽略了的新特性](#)
- [关于烂代码的那些事（下）](#)
- [说说 JAVA 代理模式](#)
- [关于烂代码的那些事（中）](#)
- [关于烂代码的那些事（上）](#)
- [关于 Java 你不知道的十件事](#)



## 发表评论

Comment form

Name\*

姓名

邮箱\*

请填写邮箱

网站 (请以 http://开头)

请填写网站地址

评论内容\*