### 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。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

1. Java1.5引入了泛型，所有的集合接口和实现都大量地使用它。
2. 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。
3. 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。
4. 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。
5. 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。

一些其它的接口有Queue、Dequeue、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。

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。

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。

### 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、Iterater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、通过迭代器fail-fast属性，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）。

### 12、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。

Fall—fast迭代器抛出ConcurrentModificationException，

fall—safe迭代器从不抛出ConcurrentModificationException。

### 13、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。

### 14、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。

### 15、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。

### 16、hashCode()和equals()方法有何重要性

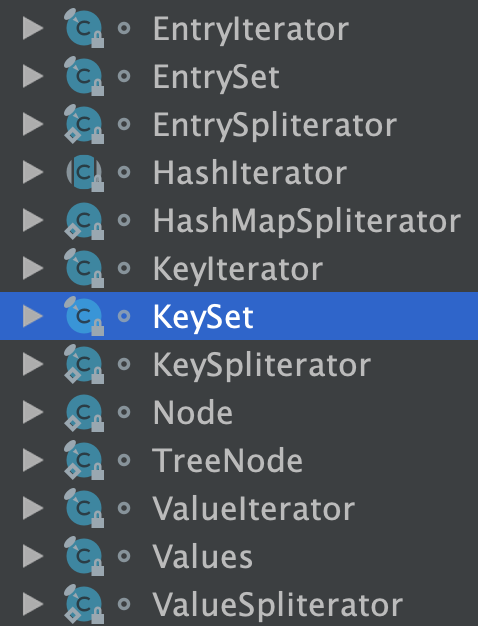
HashMap使用Key对象的hashCode()和equals()方法去决定key-value对的索引。

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。

### 17、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钟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。

### 18、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CocurrentHashMap。

### 19、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。

### 20、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。

### 21、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。

### 22、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。

### 23、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。

### 24、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。

### 25、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。

### 26、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。

### 27、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。

### 28、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。

### 29、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。

### 30、当一个集合被作为参数传递给一个函数时，如何才可以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。