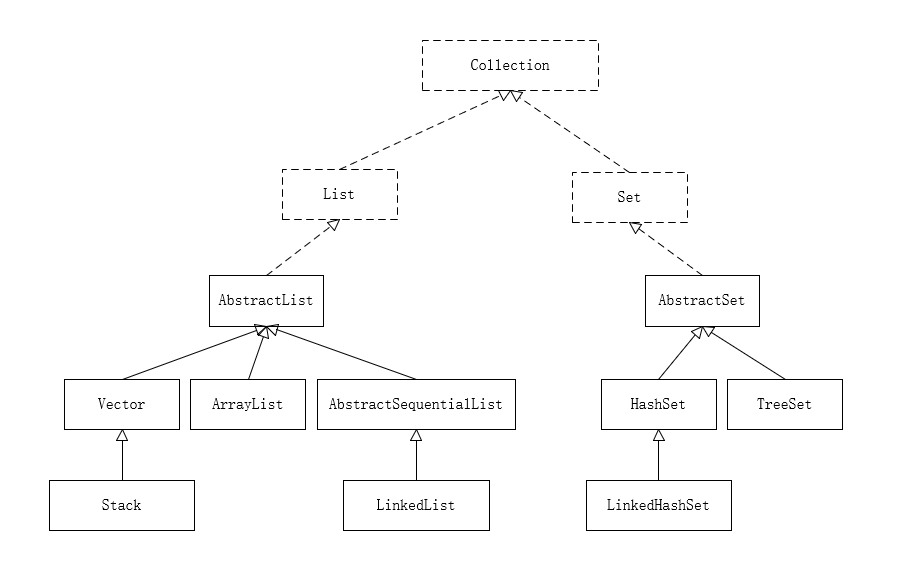
## 集合详解之 Collection + 面试题

先来看看集合的继承关系图，如下图所示：



enter image description here

其中：

* 外框为虚线的表示接口，边框为实线的表示类；
* 箭头为虚线的表示实现了接口，箭头为实线的表示继承了类。

为了方便理解，我隐藏了一些与本文内容无关的信息，隐藏的这些内容会在后面的章节中进行详细地介绍。

从图中可以看出，集合的根节点是 Collection，而 Collection 下又提供了两大常用集合，分别是：

* List：使用最多的有序集合，提供方便的新增、修改、删除的操作；
* Set：集合不允许有重复的元素，在许多需要保证元素唯一性的场景中使用。

下面我们分别对集合类进行详细地介绍。

### 集合使用

#### 1）Vector

Vector 是 Java 早期提供的线程安全的有序集合，如果不需要线程安全，不建议使用此集合，毕竟同步是有线程开销的。

使用示例代码：

Vector vector = new Vector();  
vector.add("dog");  
vector.add("cat");  
vector.remove("cat");  
System.out.println(vector);

程序执行结果：[dog]

#### 2）ArrayList

ArrayList 是最常见的非线程安全的有序集合，因为内部是数组存储的，所以随机访问效率很高，但非尾部的插入和删除性能较低，如果在中间插入元素，之后的所有元素都要后移。ArrayList 的使用与 Vector 类似。

#### 3）LinkedList

LinkedList 是使用双向链表数据结构实现的，因此增加和删除效率比较高，而随机访问效率较差。

LinkedList 除了包含以上两个类的操作方法之外，还新增了几个操作方法，如 offer() 、peek() 等，具体详情，请参考以下代码：

LinkedList linkedList = new LinkedList();  
// 添加元素  
linkedList.offer("bird");  
linkedList.push("cat");  
linkedList.push("dog");  
// 获取第一个元素  
System.out.println(linkedList.peek());  
// 获取第一个元素，并删除此元素  
System.out.println(linkedList.poll());  
System.out.println(linkedList);

程序的执行结果：

dog  
dog  
[cat, bird]

#### 4）HashSet

HashSet 是一个没有重复元素的集合。虽然它是 Set 集合的子类，实际却为 HashMap 的实例，相关源码如下：

public HashSet() {  
 map = new HashMap<>();  
}

因此 HashSet 是无序集合，没有办法保证元素的顺序性。

HashSet 默认容量为 16，每次扩充 0.75 倍，相关源码如下：

public HashSet(Collection<? extends E> c) {  
 map = new HashMap<>(Math.max((int) (c.size()/.75f) + 1, 16));  
 addAll(c);  
}

HashSet 的使用与 Vector 类似。

#### 5）TreeSet

TreeSet 集合实现了自动排序，也就是说 TreeSet 会把你插入数据进行自动排序。

示例代码如下：

TreeSet treeSet = new TreeSet();  
treeSet.add("dog");  
treeSet.add("camel");  
treeSet.add("cat");  
treeSet.add("ant");  
System.out.println(treeSet);

程序执行结果：[ant, camel, cat, dog]

可以看出，TreeSet 的使用与 Vector 类似，只是实现了自动排序。

#### 6）LinkedHashSet

LinkedHashSet 是按照元素的 hashCode 值来决定元素的存储位置，但同时又使用链表来维护元素的次序，这样使得它看起来像是按照插入顺序保存的。

LinkedHashSet 的使用与 Vector 类似。

### 集合与数组

集合和数组的转换可使用 toArray() 和 Arrays.asList() 来实现，请参考以下代码示例：

List<String> list = new ArrayList();  
list.add("cat");  
list.add("dog");  
// 集合转数组  
String[] arr = list.toArray(new String[list.size()]);  
// 数组转集合  
List<String> list2 = Arrays.asList(arr);

集合与数组的区别，可以参考[「数组和排序算法的应用 + 面试题」](https://gitbook.cn/gitchat/column/5d493b4dcb702a087ef935d9/topic/5d4d7ea069004b174ccfffef)的内容。

### 集合排序

在 Java 语言中排序提供了两种方式：Comparable 和 Comparator，它们的区别也是常见的面试题之一。下面我们彻底地来了解一下 Comparable 和 Comparator 的使用与区别。

#### 1）Comparable

Comparable 位于 java.lang 包下，是一个排序接口，也就是说如果一个类实现了 Comparable 接口，就意味着该类有了排序功能。

Comparable 接口只包含了一个函数，定义如下：

package java.lang;  
import java.util.\*;  
public interface Comparable {  
 public int compareTo(T o);  
}

**Comparable 使用示例** ，请参考以下代码：

class ComparableTest {  
 public static void main(String[] args) {  
 Dog[] dogs = new Dog[]{  
 new Dog("老旺财", 10),  
 new Dog("小旺财", 3),  
 new Dog("二旺财", 5),  
 };  
 // Comparable 排序  
 Arrays.sort(dogs);  
 for (Dog d : dogs) {  
 System.out.println(d.getName() + "：" + d.getAge());  
 }  
 }  
}  
class Dog implements Comparable<Dog> {  
 private String name;  
 private int age;  
 @Override  
 public int compareTo(Dog o) {  
 return age - o.age;  
 }  
 public Dog(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
 public String getName() {  
 return name;  
 }  
 public int getAge() {  
 return age;  
 }  
}

程序执行结果：

小旺财：3  
二旺财：5  
老旺财：10

如果 Dog 类未实现 Comparable 执行代码会报程序异常的信息，错误信息为：

Exception in thread “main” java.lang.ClassCastException: xxx cannot be cast to java.lang.Comparable

compareTo() 返回值有三种：

* e1.compareTo(e2) > 0 即 e1 > e2；
* e1.compareTo(e2) = 0 即 e1 = e2；
* e1.compareTo(e2) < 0 即 e1 < e2。

#### 2）Comparator

Comparator 是一个外部比较器，位于 java.util 包下，之所以说 Comparator 是一个外部比较器，是因为它无需在比较类中实现 Comparator 接口，而是要新创建一个比较器类来进行比较和排序。

Comparator 接口包含的主要方法为 compare()，定义如下：

public interface Comparator<T> {  
 int compare(T o1, T o2);  
}

**Comparator 使用示例** ，请参考以下代码：

class ComparatorTest {  
 public static void main(String[] args) {  
 Dog[] dogs = new Dog[]{  
 new Dog("老旺财", 10),  
 new Dog("小旺财", 3),  
 new Dog("二旺财", 5),  
 };  
 // Comparator 排序  
 Arrays.sort(dogs,new DogComparator());  
 for (Dog d : dogs) {  
 System.out.println(d.getName() + "：" + d.getAge());  
 }  
 }  
}  
class DogComparator implements Comparator<Dog> {  
 @Override  
 public int compare(Dog o1, Dog o2) {  
 return o1.getAge() - o2.getAge();  
 }  
}  
class Dog {  
 private String name;  
 private int age;  
 public Dog(String name, int age) {  
 this.name = name;  
 this.age = age;  
 }  
 public String getName() {  
 return name;  
 }  
 public int getAge() {  
 return age;  
 }  
}

程序执行结果：

小旺财：3  
二旺财：5  
老旺财：10

### 相关面试题

#### 1.List 和 Set 有什么区别？

答：区别分为以下几个方面：

* List 允许有多个 null 值，Set 只允许有一个 null 值；
* List 允许有重复元素，Set 不允许有重复元素；
* List 可以保证每个元素的存储顺序，Set 无法保证元素的存储顺序。

#### 2.哪种集合可以实现自动排序？

答：TreeSet 集合实现了元素的自动排序，也就是说无需任何操作，即可实现元素的自动排序功能。

#### 3.Vector 和 ArrayList 初始化大小和容量扩充有什么区别？

答：Vector 和 ArrayList 的默认容量都为 10，源码如下。

Vector 默认容量源码：

public Vector() {  
 this(10);  
}

ArrayList 默认容量源码：

private static final int DEFAULT\_CAPACITY = 10;

Vector 容量扩充默认增加 1 倍，源码如下：

private void grow(int minCapacity) {  
 // overflow-conscious code  
 int oldCapacity = elementData.length;  
 int newCapacity = oldCapacity + ((capacityIncrement > 0) ?  
 capacityIncrement : oldCapacity);  
 if (newCapacity - minCapacity < 0)  
 newCapacity = minCapacity;  
 if (newCapacity - MAX\_ARRAY\_SIZE > 0)  
 newCapacity = hugeCapacity(minCapacity);  
 elementData = Arrays.copyOf(elementData, newCapacity);  
}

其中 capacityIncrement 为初始化 Vector 指定的，默认情况为 0。

ArrayList 容量扩充默认增加大概 0.5 倍（oldCapacity + (oldCapacity >> 1)），源码如下（JDK 8）：

private void grow(int minCapacity) {  
 // overflow-conscious code  
 int oldCapacity = elementData.length;  
 int newCapacity = oldCapacity + (oldCapacity >> 1);  
 if (newCapacity - minCapacity < 0)  
 newCapacity = minCapacity;  
 if (newCapacity - MAX\_ARRAY\_SIZE > 0)  
 newCapacity = hugeCapacity(minCapacity);  
 // minCapacity is usually close to size, so this is a win:  
 elementData = Arrays.copyOf(elementData, newCapacity);  
}

#### 4.Vector、ArrayList、LinkedList 有什么区别？

答：这三者都是 List 的子类，因此功能比较相似，比如增加和删除操作、查找元素等，但在性能、线程安全等方面表现却又不相同，差异如下：

* Vector 是 Java 早期提供的动态数组，它使用 synchronized 来保证线程安全，如果非线程安全需要不建议使用，毕竟线程同步是有性能开销的；
* ArrayList 是最常用的动态数组，本身并不是线程安全的，因此性能要好很多，与 Vector 类似，它也是动态调整容量的，只不过 Vector 扩容时会增加 1 倍，而 ArrayList 会增加 50%；
* LinkedList 是双向链表集合，因此它不需要像上面两种那样调整容量，它也是非线程安全的集合。

#### 5.Vector、ArrayList、LinkedList 使用场景有什么区别？

答：Vector 和 ArrayList 的内部结构是以数组形式存储的，因此非常适合随机访问，但非尾部的删除或新增性能较差，比如我们在中间插入一个元素，就需要把后续的所有元素都进行移动。

LinkedList 插入和删除元素效率比较高，但随机访问性能会比以上两个动态数组慢。

#### 6.Collection 和 Collections 有什么区别？

答：Collection 和 Collections 的区别如下：

* Collection 是集合类的上级接口，继承它的主要有 List 和 Set；
* Collections 是针对集合类的一个帮助类，它提供了一些列的静态方法实现，如 Collections.sort() 排序、Collections.reverse() 逆序等。

#### 7.以下选项没有继承 Collection 接口的是？

A：List  
B：Set  
C：Map  
D：HashSet

答：C

#### 8.LinkedHashSet 如何保证有序和唯一性？

答：LinkedHashSet 底层数据结构由哈希表和链表组成，链表保证了元素的有序即存储和取出一致，哈希表保证了元素的唯一性。

#### 9.HashSet 是如何保证数据不可重复的？

答：HashSet 的底层其实就是 HashMap，只不过 HashSet 实现了 Set 接口并且把数据作为 K 值，而 V 值一直使用一个相同的虚值来保存，我们可以看到源码：

public boolean add(E e) {  
 return map.put(e, PRESENT)==null;// 调用 HashMap 的 put 方法,PRESENT 是一个至始至终都相同的虚值  
}

由于 HashMap 的 K 值本身就不允许重复，并且在 HashMap 中如果 K/V 相同时，会用新的 V 覆盖掉旧的 V，然后返回旧的 V，那么在 HashSet 中执行这一句话始终会返回一个 false，导致插入失败，这样就保证了数据的不可重复性。

#### 10.执行以下程序会输出什么结果？为什么？

Integer num = 10;  
Integer num2 = 5;  
System.out.println(num.compareTo(num2));

答：程序输出的结果是 1，因为 Integer 默认实现了 compareTo 方法，定义了自然排序规则，所以当 num 比 num2 大时会返回 1，Integer 相关源码如下：

public int compareTo(Integer anotherInteger) {  
 return compare(this.value, anotherInteger.value);  
}  
public static int compare(int x, int y) {  
 return (x < y) ? -1 : ((x == y) ? 0 : 1);  
}

#### 11.如何用程序实现后进先出的栈结构？

答：可以使用集合中的 Stack 实现，Stack 是标准的后进先出的栈结构，使用 Stack 中的 pop() 方法返回栈顶元素并删除该元素，示例代码如下。

Stack stack = new Stack();  
stack.push("a");  
stack.push("b");  
stack.push("c");  
for (int i = 0; i < 3; i++) {  
 // 移除并返回栈顶元素  
 System.out.print(stack.pop() + " ");  
}

程序执行结果：c b a

#### 12.LinkedList 中的 peek() 和 poll() 有什么区别？

答：peek() 方法返回第一个元素，但不删除当前元素，当元素不存在时返回 null；poll() 方法返回第一个元素并删除此元素，当元素不存在时返回 null。

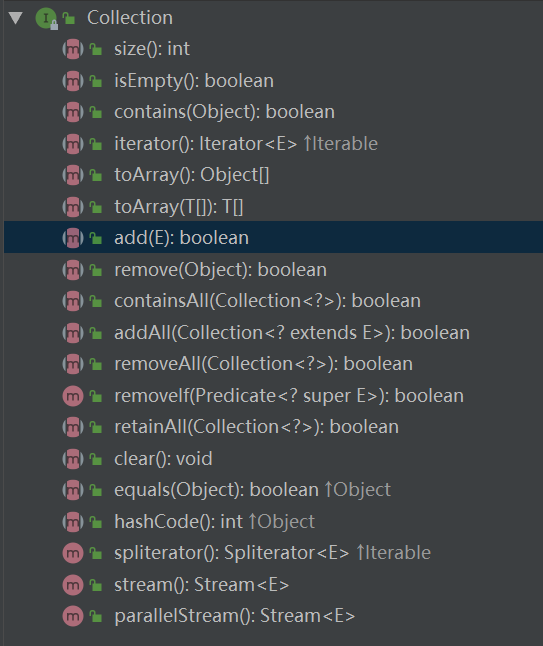
#### 13.Comparable 和 Comparator 有哪些区别？

答：Comparable 和 Comparator 的主要区别如下：

* Comparable 位于 java.lang 包下，而 Comparator 位于 java.util 包下；
* Comparable 在排序类的内部实现，而 Comparator 在排序类的外部实现；
* Comparable 需要重写 CompareTo() 方法，而 Comparator 需要重写 Compare() 方法；
* Comparator 在类的外部实现，更加灵活和方便。

### 总结

本文介绍的集合都实现自 Collection，因此它们都有同样的操作方法，如 add()、addAll()、remove() 等，Collection 接口的方法列表如下图：



当然部分集合也在原有方法上扩充了自己特有的方法，如 LinkedList 的 offer()、push() 等方法。本文也提供了数组和集合互转方法，List.toArray() 把集合转换为数组，Arrays.asList(array) 把数组转换为集合。最后介绍了 Comparable 和 Comparator 的使用和区别，Comparable 和 Comparator 是 Java 语言排序提供的两种排序方式，Comparable 位于 java.lang 包下，如果一个类实现了 Comparable 接口，就意味着该类有了排序功能；而 Comparator 位于 java.util 包下，是一个外部比较器，它无需在比较类中实现 Comparator 接口，而是要新创建一个比较器类来进行比较和排序。

[点击此处下载本文源码](https://github.com/vipstone/java-%20interview/tree/master/interview-code/src/main/java/com/interview)

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持