Java容器可分为两大类：

- Collection
- - List
  - - **ArrayList**
    - LinkedList
    - Vector(了解，已过时)
  -
  - Set
  - - TreeSet
    - **HashSet**
    - - LinkedHashSet
- Map
- - **HashMap**
  - - LinkedHashMap
  - TreeMap
  - ConcurrentHashMap
  - Hashtable(了解，，已过时)

着重标出的那些就是我们**用得最多**的容器。

其实，我也不知道要怎么总结好，因为之前写每一篇的时候都总结过了。现在又把他们重新罗列出来好像有点水，所以，我决定去回答一些Java容器的面试题！

当然了，我的答案未必就是正确的。如果有错误的地方大家多多包含，希望不吝在评论区留言指正~~

一、ArrayList和Vector的区别

共同点：

- 这两个类都实现了List接口，它们都是**有序**的集合(存储有序)，**底层是数组。**我们可以按位置索引号取出某个元素，**允许元素重复和为null。**

区别：

- **同步性：**
- - ArrayList是非同步的
  - Vector是同步的
  - 即便需要同步的时候，我们可以使用Collections工具类来构建出同步的ArrayList而不用Vector
- **扩容大小：**
- - Vector增长原来的一倍，ArrayList增长原来的0.5倍

二、HashMap和Hashtable的区别

共同点：

- 从存储结构和实现来讲基本上都是相同的，都是实现Map接口~

区别：

- **同步性：**
- - HashMap是非同步的
  - Hashtable是同步的
  - 需要同步的时候，我们往往不使用，而使用ConcurrentHashMapConcurrentHashMap基于JDK1.8源码剖析
- **是否允许为null：**
- - HashMap允许为null
  - Hashtable不允许为null
- **contains方法**
- - 这知识点是在牛客网刷到的，没想到这种题还会有(我不太喜欢)....
  - Hashtable有contains方法
  - HashMap把Hashtable的contains方法去掉了，改成了containsValue和containsKey
- **继承不同：**
- - HashMap

    extends AbstractMap
  - public class Hashtable

    extends Dictionary

三、List和Map的区别

共同点：

- 都是Java常用的容器，都是接口(ps：写出来感觉好像和没写一样.....)

不同点：

- **存储结构不同**：
- - List是存储单列的集合
  - Map存储的是key-value键值对的集合
- **元素是否可重复**：
- - List允许元素重复
  - Map不允许key重复
- **是否有序**：
- - List集合是有序的(存储有序)
  - Map集合是无序的(存储无序)

四、Set里的元素是不能重复的，那么用什么方法来区分重复与否呢? 是用==还是equals()?

我们知道Set集合实际**大都使用的是Map集合的put方法来添加元素。**

以HashSet为例，HashSet里的元素不能重复，在源码(HashMap)是这样体现的：

```
    // 1. 如果key 相等
    if (p.hash == hash &&
        ((k = p.key) == key || (key != null && key.equals(k))))
        e = p;
    // 2. 修改对应的value
       if (e != null) { // existing mapping for key
            V oldValue = e.value;
            if (!onlyIfAbsent || oldValue == null)
                e.value = value;
            afterNodeAccess(e);
            return oldValue;
       }
```

添加元素的时候，如果key(也对应的Set集合的元素)相等，那么则修改value值。而在Set集合中，value值仅仅是一个Object对象罢了(**该对象对Set本身而言是无用的**)。

也就是说：Set集合如果添加的元素相同时，**是根本没有插入的(仅修改了一个无用的value值)**！从源码(HashMap)中也看出来，**==和equals()方法都有使用**！

五、Collection和Collections的区别

1. Collection是集合的上级**接口**，继承它的有Set和List接口
2. Collections是集合的**工具类**，提供了一系列的静态方法对集合的搜索、查找、同步等操作

六、说出ArrayList,LinkedList的存储性能和特性

ArrayList的底层是数组，LinkedList的底层是双向链表。

- ArrayList它支持以角标位置进行索引出对应的元素(随机访问)，而LinkedList则需要遍历整个链表来获取对应的元素。因此**一般来说ArrayList的访问速度是要比LinkedList要快的**
- ArrayList由于是数组，对于删除和修改而言消耗是比较大(复制和移动数组实现)，LinkedList是双向链表删除和修改只需要修改对应的指针即可，消耗是很小的。因此**一般来说LinkedList的增删速度是要比ArrayList要快的**

6.1扩展：

ArrayList的增删**未必**就是比LinkedList要慢。

- 如果增删都是在**末尾**来操作【每次调用的都是remove()和add()】，此时ArrayList就不需要移动和复制数组来进行操作了。如果数据量有百万级的时，**速度是会比LinkedList要快的。**(我测试过)

- 如果**删除操作**的位置是在**中间。**由于LinkedList的消耗主要是在遍历上，ArrayList的消耗主要是在移动和复制上(底层调用的是arraycopy()方法，是native方法)。
- - LinkedList的遍历速度是要慢于ArrayList的复制移动速度的
  - 如果数据量有百万级的时，**还是ArrayList要快。**(我测试过)

七、Enumeration和Iterator接口的区别

这个我在前面的文章中也没有详细去讲它们，只是大概知道的是：Iterator替代了Enumeration，Enumeration是一个旧的迭代器了。

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。**

- 我们在做练习的时候，迭代时会不会经常出错，抛出ConcurrentModificationException异常，说我们在遍历的时候还在修改元素。
- 这其实就是fail-fast机制~具体可参考博文：https://blog.csdn.net/panweiwei1994/article/details/77051261

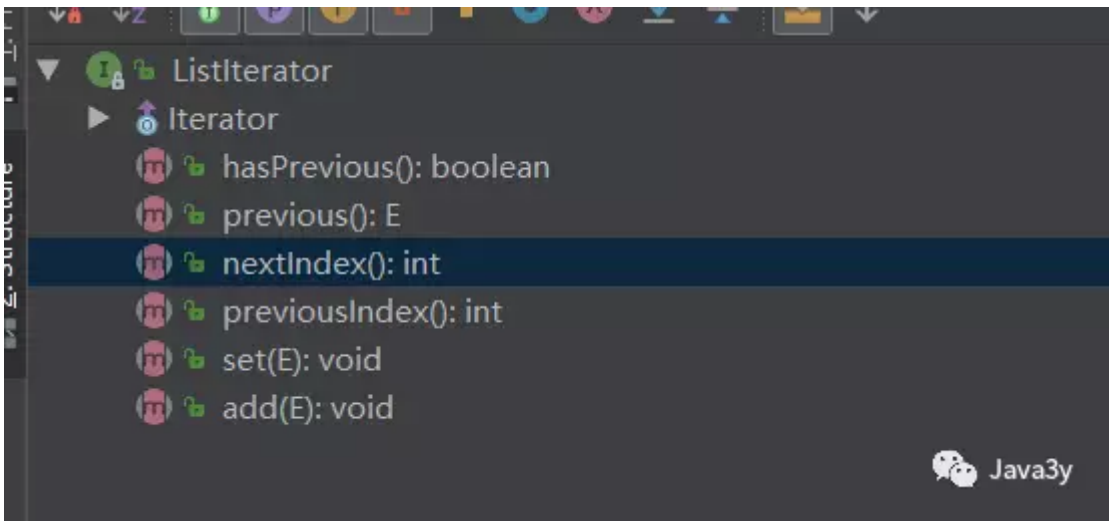
区别有三点：

- Iterator的方法名比Enumeration更科学
- Iterator有fail-fast机制，比Enumeration更安全
- Iterator能够删除元素，Enumeration并不能删除元素

八、ListIterator有什么特点

- ListIterator**继承了**Iterator接口，它用于**遍历List集合的元素。**
- ListIterator可以实现**双向遍历,添加元素，设置元素**

看一下源码的方法就知道了：



九、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

十、Java中HashMap的key值要是为类对象则该类需要满足什么条件?

需要同时重写该类的hashCode()方法和它的equals()方法。

- 从源码可以得知，在插入元素的时候是**先算出该对象的hashCode**。如果hashcode相等话的。那么表明该对象是存储在同一个位置上的。
- 如果调用equals()方法，**两个key相同**，则**替换元素**
- 如果调用equals()方法，**两个key不相同**，则说明该**hashCode仅仅是碰巧相同**，此时是散列冲突，将新增的元素放在桶子上

一般来说，我们会认为：**只要两个对象的成员变量的值是相等的，那么我们就认为这两个对象是相等的**！因为，Object底层比较的是两个对象的地址，而对我们开发来说这样的意义并不大~这也就为什么我们要重写`equals()`方法

重写了equals()方法，就要重写hashCode()的方法。因为**equals()认定了这两个对象相同**，而**同一个对象调用hashCode()方法时**，是应该返回相同的值的！

十一、与Java集合框架相关的有哪些最好的实践

1. **根据需要**确定集合的类型。如果是单列的集合，我们考虑用Collection下的子接口ArrayList和Set。如果是映射，我们就考虑使用Map~
2. 确定完我们的集合类型，我们接下来**确定使用该集合类型下的哪个子类**~我认为可以简单分成几个步骤：
3.
   - 去找Tree红黑树类型的(JDK1.8)
   - 去找Linked双向列表结构的
   - 去找线程安全的集合类使用
   - 是否需要同步
   - 迭代时是否需要有序(插入顺序有序)
   - 是否需要排序(自然顺序或者手动排序)
4. 估算存放集合的数据量有多大，无论是List还是Map，它们实现动态增长，都是有性能消耗的。在初始集合的时候给出一个**合理的容量**会减少动态增长时的消耗~
5. **使用泛型**，避免在运行时出现ClassCastException
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

十二、ArrayList集合加入1万条数据，应该怎么提高效率

ArrayList的默认初始容量为10，要插入大量数据的时候需要不断扩容，而扩容是非常影响性能的。因此，现在明确了10万条数据了，我们可以**直接在初始化的时候就设置ArrayList的容量**！

这样就可以提高效率了~