Java集合学习手册（9）：Java 集合对比

一、HashMap与HashTable的区别

HashMap和Hashtable的比较是Java面试中的常见问题，用来考验程序员是否能够正确使用集合类以及是否可以随机应变使用多种思路解决问题。HashMap的工作原理、ArrayList与Vector的比较以及这个问题是有关Java 集合框架的最经典的问题。

Hashtable是个过时的集合类，存在于Java API中很久了。在Java 4中被重写了，实现了Map接口，所以自此以后也成了Java集合框架中的一部分。Hashtable和HashMap在Java面试中相当容易被问到，甚至成为了集合框架面试题中最常被考的问题，所以在参加任何Java面试之前，都不要忘了准备这一题。

HashMap和Hashtable都实现了Map接口，但决定用哪一个之前先要弄清楚它们之间的分别。主要的区别有：

HashMap	HashTable
非线程安全（非线程同步）	线程安全（线程同步）
更适合于单线程	更适合于多线程
允许null值	不允许null值
迭代器Iterator是fail-fast迭代器	迭代器enumerator不是fail-fast的
初始容量为16	初始容量为11

- 两者最主要的区别在于Hashtable是线程安全，而HashMap则非线程安全

HashMap是非synchronized，而Hashtable是synchronized，这意味着Hashtable是线程安全的，多个线程可以共享一个Hashtable；而如果没有正确的同步的话，多个线程是不能共享HashMap的

由于Hashtable的实现方法里面都添加了synchronized关键字来确保线程同步，所以在单线程环境下它比HashMap要慢。如果你不需要同步，只需要单一线程，那么使用HashMap性能要好过Hashtable。仅在你需要完全的线程安全的时候使用Hashtable，而如果你使用Java 5或以上的话，请使用ConcurrentHashMap吧。

线程安全的实现原理：jvm有一个main memory，而每个线程有自己的working memory，一个线程对一个变量进行操作时，都要在自己的working memory里面建立一个copy，操作完之后再写入main memory。多个线程同时操作同一个变量，就可能会出现不可预知的结果。用synchronized的关键是建立一个镜像，这个镜像可以是要修改的变量也可以其他你认为合适的对象比如方法和类，然后通过给这个镜像加锁来实现线程安全，每个线程在获得这个锁之后，要执行完才会释放它得到的锁。这样就实现了所谓的线程安全。sychronized意味着在一次仅有一个线程能够更改Hashtable。就是说任何线程要更新Hashtable时要首先获得同步锁，其它线程要等到同步锁被释放之后才能再次获得同步锁更新Hashtable。

我们平时使用时若无特殊需求建议使用HashMap，在多线程环境下若使用HashMap需要使用Collections.synchronizedMap()方法来获取一个线程安全的集合（Collections.synchronizedMap()实现原理是Collections定义了一个SynchronizedMap的内部类，这个类实现了Map接口，在调用方法时使用synchronized来保证线程同步,当然了实际上操作的还是我们传入的HashMap实例，简单的说就是Collections.synchronizedMap()方法帮我们在操作HashMap时自动添加了synchronized来实现线程同步，类似的其它Collections.synchronizedXX方法也是类似原理）

- HashMap的迭代器Iterator是fail-fast迭代器，而Hashtable的enumerator迭代器不是fail-fast的。

当有其它线程改变了HashMap的结构（增加或者移除元素），将会抛出ConcurrentModificationException，但迭代器本身的remove()方法移除元素则不会抛出ConcurrentModificationException异常。但这并不是一个一定发生的行为，要看JVM。这条同样也是Enumeration和Iterator的区别。

Fail-safe和iterator迭代器相关。如果某个集合对象创建了Iterator或者ListIterator，然后其它的线程试图“结构上”更改集合对象，将会抛出ConcurrentModificationException异常。但其它线程可以通过set()方法更改集合对象是允许的，因为这并没有从“结构上”更改集合。但是假如已经从结构上进行了更改，再调用set()方法，将会抛出IllegalArgumentException异常。

结构上的更改指的是删除或者插入一个元素，这样会影响到map的结构。

- HashMap可以使用null作为key，而Hashtable则不允许null作为key

虽说HashMap支持null值作为key，不过建议还是尽量避免这样使用，因为一旦不小心使用了，若因此引发一些问题，排查起来很是费事。HashMap以null作为key时，总是存储在table数组的第一个节点上

- HashMap是对Map接口的实现，HashTable实现了Map接口和Dictionary抽象类
- HashMap的初始容量为16，Hashtable初始容量为11，两者的填充因子默认都是0.75
- HashMap扩容时是当前容量翻倍即:$capacity * 2$，Hashtable扩容时是容量翻倍+1即:

$capacity * 2 + 1$

- 两者计算hash的方法不同

Hashtable计算hash是直接使用key的hashcode对table数组的长度直接进行取模

```
int hash = key.hashCode();
int index = (hash & 0x7FFFFFFF) % tab.length;
```

HashMap计算hash对key的hashcode进行了二次hash，以获得更好的散列值，然后对table数组长度取摸

```
static final int hash(Object key) {
    int h;
    return (key == null) ? 0 : (h = key.hashCode()) ^ (h >>> 16);
}

static int indexFor(int h, int n) {
        return h & (n-1);
    }
```

二、HashSet与HashMap的区别

HashMap和HashSet的区别是Java面试中最常被问到的问题。如果没有涉及到Collection框架以及多线程的面试，可以说是不完整。而Collection框架的问题不涉及到HashSet和HashMap，也可以说是不完整。HashMap和HashSet都是collection框架的一部分，它们让我们能够使用对象的集合。collection框架有自己的接口和实现，主要分为Set接口，List接口和Queue接口。它们有各自的特点，Set的集合里不允许对象有重复的值，List允许有重复，它对集合中的对象进行索引，Queue的工作原理是FCFS算法(First Come, First Serve)。

首先让我们来看看什么是HashMap和HashSet，然后再来比较它们之间的分别。

2.1 什么是HashSet

HashSet实现了Set接口，它不允许集合中有重复的值，当我们提到HashSet时，第一件事情就是在将对象存储在HashSet之前，要先确保对象重写equals()和hashCode()方法，这样才能比较对象的值是否相等，以确保set中没有储存相等的对象。如果我们没有重写这两个方法，将会使用这个方法的默认实现。

public boolean add(Object o)方法用来在Set中添加元素，当元素值重复时则会立即返回false，如果成功添加的话会返回true。

HashSet不是key value结构，仅仅是存储不重复的元素，相当于简化版的HashMap，只是仅仅包含HashMap中的key而已。通过查看源码也证实了这一点，HashSet内部就是使用HashMap实现，只不过HashSet里面的HashMap所有的value都是同一个Object而已，因此HashSet也是非线程安全的，至于HashSet和Hashtable的区别，HashSet就是个简化的HashMap的。

下面是HashSet几个主要方法的实现，更具体的可以参考【Java学习手册：Java HashSet】

```
private transient HashMap<E,Object> map;
 private static final Object PRESENT = new Object();

 public HashSet() {
   map = new HashMap<E,Object>();
   }
public boolean contains(Object o) {
   return map.containsKey(o);
   }
public boolean add(E e) {
   return map.put(e, PRESENT)==null;
   }
public boolean add(E e) {
   return map.put(e, PRESENT)==null;
   }
public boolean remove(Object o) {
   return map.remove(o)==PRESENT;
   }


public void clear() {
   map.clear();
   }
```

2.2 什么是HashMap

HashMap实现了Map接口，Map接口对键值对进行映射。Map中不允许重复的键。Map接口有两个基本的实现，HashMap和TreeMap。TreeMap保存了对象的排列次序，而HashMap则不能。HashMap允许键和值为null。HashMap是非synchronized的，但collection框架提供方法能保证HashMap synchronized，这样多个线程同时访问HashMap时，能保证只有一个线程更改Map。

public Object put(Object Key,Object value)方法用来将元素添加到map中。

2.3 HashSet和HashMap的区别

HashMap	
HashMap实现了Map接口	HashSet实现了Set接口
HashMap储存键值对	HashSet仅仅存储对象
使用put()方法将元素放入map中	使用add()方法将元素放入set中
HashMap中使用键对象来计算hashcode值	HashSet使用成员对象来计算hashcode值
HashMap比较快，因为是使用唯一的键来获取对象	HashSet较HashMap来说比较慢

三、HashSet和TreeSet的区别

- Hashset 的底层是由hashTable实现的，add()，remove()，contains()方法的时间复杂度是O(1).可以放入null，但只能放入一个null。Treeset 底层是由红黑树实现的,add()，remove()，contains()方法的时间复杂度是O(logn)。不允许放入null值
- 如果需要在Treeset 中插入对象，需要实现Comparable 接口，为其指定比较策略：

```
public class TreeSet<E>
   extends AbstractSet<E>
   implements SortedSet<E>, Cloneable, java.io.Serializable
public class HashSet<E>
   extends AbstractSet<E>
   implements Set<E>, Cloneable, java.io.Serializable
```

其中SortedSet中组合了一个：Comparator<? super E> comparator();

HashSet是基于Hash算法实现的,其性能通常优于TreeSet,我们通常都应该使用HashSet,在我们需要排序的功能时,我门才使用TreeSet

四、ArrayList、LinkedList、Vector的底层实现和区别

4.1 ArrayList

ArrayList是一个可以处理变长数组的类型，这里不局限于“数”组，ArrayList是一个泛型类，可以存放任意类型的对象。顾名思义，ArrayList是一个数组列表，因此其内部是使用一个数组来存放对象的，因为Object是一切类型的父类，因而ArrayList内部是有一个Object类型的数组类存放对象。ArrayList类常用的方法有add()、clear()、get()、indexOf()、remove()、sort()、toArray()、toString()等等，同时ArrayList内部有一个私有类实现Iterator接口，因此可以使用iterator()方法得到ArrayList的迭代器，同时，还有一个私有类实现了ListIterator接口，因此ArrayList也可以调用listIterator()方法得到ListIterator迭代器。

由于ArrayList是依靠数组来存放对象的，只不过封装起来了而已，因此其一些查找方法的效率都是O(n)，跟普通的数组效率差不多，只不过这个ArrayList是一个可变”数组“，并且可以存放一切指定的对象。

另外，由于ArrayList的所有方法都是默认在单一线程下进行的，因此ArrayList不具有线程安全性。若想在多线程下使用，应该使用Colletions类中的静态方法synchronizedList()对ArrayList进行调用即可。

4.2 LinkedList

LinkedList可以看做为一个双向链表，所有的操作都可以认为是一个双向链表的操作，因为它实现了Deque接口和List接口。同样，LinkedList也是线程不安全的，如果在并发环境下使用它，同样用Colletions类中的静态方法synchronizedList()对LinkedList进行调用即可。

在LinkedList的内部实现中，并不是用普通的数组来存放数据的，而是使用结点来存放数据的，有一个指向链表头的结点first和一个指向链表尾的结点last。不同于ArrayList只能在数组末尾添加数据，LinkList可以很方便在链表头或者链表尾插入数据，或者在指定结点前后插入数据，还提供了取走链表头或链表尾的结点，或取走中间某个结点，还可以查询某个结点是否存在。add()方法默认在链表尾部插入数据。总之，LinkedList提供了大量方便的操作方法，并且它的插入或增加等方法的效率明显高于ArrayList类型，但是查询的效率要低一点，因为它是一个双向链表。

因此，LinkedList与ArrayList最大的区别是LinkedList更加灵活，并且部分方法的效率比ArrayList对应方法的效率要高很多，对于数据频繁出入的情况下，并且要求操作要足够灵活，建议使用LinkedList；对于数组变动不大，主要是用来查询的情况下，可以使用ArrayList。

4.3 Vector

Vector也是一个类似于ArrayList的可变长度的数组类型，它的内部也是使用数组来存放数据对象的。值得注意的是Vector与ArrayList唯一的区别是，Vector是线程安全的，即它的大部分方法都

包含有关键字synchronized，因此，若对于单一线程的应用来说，最好使用ArrayList代替Vector，因为这样效率会快很多（类似的情况有StringBuffer与StringBuilder）；而在多线程程序中，为了保证数据的同步和一致性，可以使用Vector代替ArrayList实现同样的功能。

五、数组(Array)和列表(ArrayList)的区别

- Array可以包含基本类型和对象类型，ArrayList只能包含对象类型。
- Array大小是固定的，ArrayList的大小是动态变化的。
- ArrayList提供了更多的方法和特性，比如：addAll()，removeAll()，iterator()等等。
- ArrayList可以存任何Object，如String等。

ArrayList与数组的区别主要就是由于动态增容的效率问题了。
对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。因此基本类型用Array，动态变化用ArrayList。