Hashtable和HashMap的区别

- Hashtable是一个线性安全的Map实现，但是HashMap是线性不安全的实现，所以hashMap比Hashtable性能高一点；但如果有多个线性访问同一个Map对象时，使用Hashtable实现类会更好。
- Hashtable不允许使用null作为key和value,如果试图把null值方剂Hashtable中，将会引发NullPointerException异常；但Hashtable可以使用null作为key或value.
- 由于HashMap里的key不能重复，所以HashMap里最多只有一个key-value对的key我null,但可以有无数个key-value对的value为null.
- Hashtable和HashMap不能保证其中key-value对的顺序，判断两个key相等的标准是，两个key通过equals()方法不叫返回true,两个key的hashCode值也相等。 **Hashtable传入null值**

```
Hashtable h1 = new Hashtable();
h1.put(null,null);
System.out.println(h1);
```

会报如下错误：

```
"C:\Program Files\Java\jdk1.8.0_171\bin\java" ...
Exception in thread "main" java.lang.NullPointerException
    at java.util.Hashtable.put(Hashtable.java:460)
    at list.FixedSizeList.main(FixedSizeList.java:27)
```

HashMap传入null值

```
HashMap h2 = new HashMap();
h2.put(null,null);
h2.put(null,null);
h2.put("aa",null);
System.out.println(h2);
```

输出结果如下图：

```
"C:\Program Files\Java\jdk1.8.0_171\bin\java" ...
{null=null, aa=null}
```

注：
一般不建议使用Hashtable实现类,即使需要创建线程安全的Map实现类，也无须使用Hashtable实现类，可以通过Collections工具类吧HashMap变成线程安全的。

各Map实现类的性能分析

1.对于Map的常用实现类而言，虽然HashMap和Hashtable的实现机制几乎一样，但是由于Hashtable是一个古老的，线程安全的集合，因此HashMap通常比Hashtable要快。

2.TreeMap通常比HashMap,Hashtable要慢（尤其在插入，删除key-value对时更慢），因为TreeMap底层采用红黑树来管理key-value对（红黑树的每个节点就是一个key-value）。
3.使用TreeMap的一个好处就是：TreeMap中的key-value对总是处于有序状态，无须专门进行排序操作。当TreeMap被填充之后，就可以调用keySet()取得key组成的Set,然后使用toArray()方法生成key数组，接下来使用Arrays的binarySearch()方法在已排序的数组中快速的查询对象。
4.对于一般的应用场景，程序应该多考虑使用hashMap,因为HashMap正是为了快速查询设计的（HashMap底层其实也是采用数组来存储key-value对），但如果程序需要一个总是排好序的Map时，则考虑使用TreeMap. 5.LinkedHashMap比HashMap慢一点，因为它需要维护链表来保持Map中的key-value时的添加顺序。IdentityHashMap性能没有特别出色之处，因为他采用与HashMap基本相似的实现，只是它使用==而不是equals()方法来判断元素相等。EnumMap的性能最好，但它只能使用同一个枚举值作为key.