链滴

Java 基础：HashMap 中 putAll 方法的疑惑

作者：HuixiaZhang
原文链接：https://ld246.com/article/1539250951365
来源网站：链滴

最近回顾了下HashMap的源码（JDK1.7），当读到putAll方法时，发现了之前写的TODO标记，当由于时间匆忙没来得及深究，现在回顾到了就再仔细思考了下

```
 @Override public void putAll(Mapextends K, ? extends V> m) { int numKeysToBeAdded = m.size(); if (numKeysToBeAdded == 0) return; // TODO 这里的numKeysToBeAdded是不是应该要this.size+m.size()呢？ // TODO 这里确实有点问题，下面的for循环中put操作可能会导致再次resize，怪怎么没人提出这个问题呢？
    if (numKeysToBeAdded > threshold) { // +1是为了补上被强转为int而抹去的小数部分
        int targetCapacity = (int)(numKeysToBeAdded / loadFactor + 1); if (targetCapacity > MAXIMUM_CAPACITY)
            targetCapacity = MAXIMUM_CAPACITY; int newCapacity = table.length; while (newCapacity < targetCapacity)
            newCapacity <<= 1; if (newCapacity > table.length)
            resize(newCapacity);
    } for (Map.Entryextends K, ? extends V> e : m.entrySet())
        put(e.getKey(), e.getValue());
            }
```

如注释中所示 numKeysToBeAdded > threshold 就是想提前判断Map是否需要扩容，如果需的话则直接一步到位，从而防止循环中的put操作引起多次扩容，以次来减小 resize 方法带来的性能销。

但是：我们看方法的第一行，int numKeysToBeAdded = m.size()；如果要实现扩容一步到位的话，里的 numKeysToBeAdded 不应该是当前Map的size加m的size之和吗？this.size + m.size() > threhold

就扩容才能保证m中所有元素添加到当前HashMap后只触发一次resize 。

测试代码如下，直接debug HashMap的putAll方法，我们可以看到整个putAll是进行了两次resie

```
Map map = new HashMap(4);
Map m = new HashMap(8);
map.put("a", "haha");
map.put("b", "haha");
map.put("c", "haha");
m.put("1", "a");
m.put("2", "a");
m.put("3", "a");
m.put("4", "a");
map.putAll(m);
```

JDK1.8的HashMap已经实现已经做了很大的修改，但是当我切换到1.8 debug时还是resize了两，为什么呢？仔细看下面的注释（当时看源码的时候直接把这段注释忽略了，汗），JDK的大神们给了如下的解释，显然他们也知道这个问题，但是主要考虑到m和当前的HashMap中可能存在重复的ke，这样的话就可能造成HashMap浪费了比较大的空间（画外音：HashMap默认加载因子为0.75的设初衷不就是采取了空间换时间的思想嘛??）

```
/* * Expand the map if the map if the number of mappings to be added
 * is greater than or equal to threshold.  This is conservative; the
 * obvious condition is (m.size() + size) >= threshold, but this
```

```
* condition could result in a map with twice the appropriate capacity,
* if the keys to be added overlap with the keys already in this map.
* By using the conservative calculation, we subject ourself
* to at most one extra resize. */
```

在HashMap中 size 肯定会小于或等于 threshold ，所以putAll时当 m.size() > threshold 进行扩容HashMap的容量增加至少1倍，则因为存在 m.size() > size 所以就算 m.size() + size > threshold(一次扩容后) 只要再做一次扩容就可以满足HashMap的规则了。

更全的学习注释可以参考：https://github.com/hiccup234/misc/blob/master/src/main/java/top/iccup/jdk/container/mycontainer/MyHashMap7.java