设计 RandomPool 结构

【题目】

设计一种结构，在该结构中有如下三个功能：

- insert(key)：将某个 key 加入到该结构，做到不重复加入。
- delete(key)：将原本在结构中的某个 key 移除。
- getRandom()：等概率随机返回结构中的任何一个 key。

【要求】

Insert、delete 和 getRandom 方法的时间复杂度都是 $O(1)$。

【难度】

尉 ★★☆☆

【解答】

这种结构假设叫 Pool，具体实现如下：

1．包含两个哈希表 keyIndexMap 和 indexKeyMap。

2．keyIndexMap 用来记录 key 到 index 的对应关系。

3．indexKeyMap 用来记录 index 到 key 的对应关系。

4．包含一个整数 size，用来记录目前 Pool 的大小，初始时 size 为 0。

5．执行 insert(newKey)操作时，将(newKey,size)放入 keyIndexMap，将(size,newKey)放入 indexKeyMap，然后把 size 加 1，即每次执行 insert 操作之后 size 自增。

6．执行 delete（deleteKey）操作时(关键步骤)，假设 Pool 最新加入的 key 记为 lastKey，lastKey 对应的 index 信息记为 lastIndex。要删除的 key 为 deleteKey，对应的 index 信息记为 deleteIndex。那么先把 lastKey 的 index 信息换成 deleteKey，即在 keyIndexMap 中把记录(lastKey,lastIndex)变为(lastKey,deleteIndex)，并在 indexKeyMap 中把记录(deleteIndex, deleteKey)变为(deleteIndex,lastKey)。然后在 keyIndexMap 中删除记录(deleteKey, deleteIndex)，并在 indexKeyMap 中把记录(lastIndex,lastKey)删除。最后 size 减 1。这么做相当于把 lastKey 放到了 deleteKey 的位置上，保证记录的 index 还是连续的。

7. 进行 getRandom 操作时，根据当前的 size 随机得到一个 index，步骤 6 可保证 index 在范围[0~size-1]上，都对应着有效的 key，然后把 index 对应的 key 返回即可。

具体请参看如下代码中的 Pool 类。

```
public class Pool<K> {
        private HashMap<K, Integer> keyIndexMap;
        private HashMap<Integer, K> indexKeyMap;
        private int size;

        public Pool() {
                this.keyIndexMap = new HashMap<K, Integer>();
                this.indexKeyMap = new HashMap<Integer, K>();
                this.size = 0;
        }

        public void insert(K key) {
                if (!this.keyIndexMap.containsKey(key)) {
                        this.keyIndexMap.put(key, this.size);
                        this.indexKeyMap.put(this.size++, key);
                }
        }

        public void delete(K key) {
                if (this.keyIndexMap.containsKey(key)) {
                        int deleteIndex = this.keyIndexMap.get(key);
                        int lastIndex = --this.size;
                        K lastKey = this.indexKeyMap.get(lastIndex);
                        this.keyIndexMap.put(lastKey, deleteIndex);
                        this.indexKeyMap.put(deleteIndex, lastKey);
                        this.keyIndexMap.remove(key);
                        this.indexKeyMap.remove(lastIndex);
                }
        }

        public K getRandom() {
                if (this.size == 0) {
                        return null;
                }
                int randomIndex = (int) (Math.random() * this.size);
                return this.indexKeyMap.get(randomIndex);
        }

}
```