public class TestLRUCache {  
 public static void main(String[] args) {  
 SimpleLRUCache cache = new SimpleLRUCache(3);  
 cache.put("A",100);  
 cache.put("B",200);  
 cache.put("D",400);  
 cache.get("B");*//改变了访问顺序* cache.put("C",300);  
 System.*out*.println(cache);*//输出顺序：DBC* }  
}  
*/\*  
构建一个简单地LruCache对象  
 LinkedHashMap:  
 1）存储结构：链表+散列表  
 2）算法：FIFO，LRU  
 \*/*class SimpleLRUCache extends LinkedHashMap {  
 private int maxSize;*//最大容量* public SimpleLRUCache(int cap){  
 *//参数1：容量  
 //参数2：扩容因子  
 //参数3：记录添加顺序（默认），或者记录访问顺序（true）* super(cap,0.75f,true);  
 this.maxSize = cap;  
 }  
 *//此方法的返回值告诉map对象是否要移除元素  
 //此方法会在put()方法执行时调用* @Override  
 protected boolean removeEldestEntry(Map.Entry eldest) {  
 if(size()>maxSize) return true;  
 return false;  
 }  
}