*/\*  
构建一个cache对象  
 1）存储结构：散列表  
 2）存储算法：FIFO  
 \*/*public class SimpleFIFOCache {  
 private int maxCap;  
 public SimpleFIFOCache(int maxCap){  
 this.maxCap = maxCap;  
 }  
 *//存储数据* private Map<Object,Object> map = new HashMap<>();  
 *//记录元素顺序* private Deque<Object> keyOrder = new LinkedList<>();  
 *//基于key获取val* public Object getObject(Object key){  
 Object val = map.get(key);  
 return val;  
 }  
 *//存储对象* public void putObject(Object key,Object val){  
 *//判断cache是否已满,如果满了则删除最先添加的元素* if(map.size()>=maxCap){  
 *//获取最先加入元素的key* Object firstKey = keyOrder.removeFirst();  
 *//从缓存中删除最先加入的元素* map.remove(firstKey);  
 }  
 *//添加新元素* map.put(key,val);  
 *//记录key对象(放在队尾)* keyOrder.addLast(key);  
 }  
 @Override  
 public String toString() {  
 return map.toString();  
 }  
}  
class TestSimpleFIFOCache{  
 public static void main(String[] args) {  
 SimpleFIFOCache cache = new SimpleFIFOCache(3);  
 cache.putObject("A",1);  
 cache.putObject("B",2);  
 cache.putObject("C",3);  
 cache.putObject("D",4);  
 System.*out*.println(cache);*//输出顺序BCD* }  
}