1. LinkedHashMap和HashMap最大的区别是可以保证遍历时，输入和输出的顺序是一样的。

其实现原理是添加了LinkedHashMap.Entry类型的head和last，其中LinkedHashMap.Entry继承了HashMap.Node类，并且添加了LinkedHashMap.Entry类型的before 和after引用构成双向链表。

1. LinkedHashMap可以通过构造函数（**public** LinkedHashMap(**int** initialCapacity, **float** loadFactor, **boolean** accessOrder)）设定最近最少使用算法。如果某一节点被访问，他就会被移到链表的最后面。

**void** afterNodeAccess(Node<K,V> e) { *// move node to last* LinkedHashMap.Entry<K,V> last;

**if** (**accessOrder** && (last = **tail**) != e) {  
 LinkedHashMap.Entry<K,V> p =  
 (LinkedHashMap.Entry<K,V>)e, b = p.**before**, a = p.**after**;  
 p.**after** = **null**;

**//双向链表拉链过程**  
 **if** (b == **null**)  
 **head** = a;  
 **else** b.**after** = a;  
 **if** (a != **null**)  
 a.**before** = b;  
 **else** last = b;  
 **if** (last == **null**)  
 **head** = p;  
 **else** {  
 p.**before** = last;  
 last.**after** = p;  
 }  
 **tail** = p;  
 ++**modCount**;  
 }  
}