实现了LRU缓存功能，统计最近哪些学生团队使用系统的情况，作为学生的考核因素之一。

①LRU缓存----用LinkedHashMap实现

LinkedHashMap是一种有序的集合类，它可以按照插入顺序或者访问顺序进行存储元素，通过构造器的boolean accessOrder来控制，true表示按照访问顺序进行存储。那么它是如何实现LRU缓存的呢？

通过构造器，传入初始容量，负载因子，还有true的accessOrder，，此时在插入元素的时候，LinkedHashMap没有自身的putVal函数，调用的是HashMap的putVal，但是它重写了newNode方法，内部调用了linkNodeLast，将新节点放到表尾。然后插入完成之后，调用afterNodeAccess回调方法，当accessOrder为true的时候，说明刚插入的节点为刚访问过的节点，放在表尾。再调用afterNodeInsertion判断是否应该以LRU方式移除元素，需要三个条件

①evict为true，表示非构造器引起的插入集合，就是不是初始化的意思。

②first不为null，说明表不为空

③removeEldestEntry返回true，但是此方法在LinkedHashMap中是默认返回false的，因此得进行重写

LRUCache类继承LinkedHashMap类，成员变量有个缓存容量大小的capacity，构造器初始化capacity的构造器，内部调用super的指定初始容量，负载因子和accessOrder为true的构造器。重写removeEldestEntry方法，返回 size（） > capacity.

缺点：高并发状态下容易出现线程不安全的问题。

实现了过期时间缓存的功能---用于实现，有些请求可能会短时间内重复请求多次，例如，公司的一些基本信息等变化不大的数据，可能在5分钟要反复请求多次，此时如果都是去数据库查询，那么会影响查询效率，采用一个List存储数据库的特定请求的结果，还需要实现一个过期时间缓存，将时间间隔内的请求存入缓存队列中，然后一旦到了过期时间，就从list中删除指定请求的缓存。

使用DelayQueue实现，支持延时输出元素的无界阻塞队列，队列使用PriorityQueue实现，队列中的元素都得实现Delayed接口，创建元素时，指定缓存的时间，过了缓存时间就从队列中取出。一旦能从队列中取出元素，说明该元素的缓存时间已经到了，就可以真正存储请求的list中删除对应的元素。

让请求的结果，例如公司的基本信息的查询结果定义为一个类DelayResult，实现Delayed接口，成员变量有Company对象和缓存时间、实现compareTo方法和getDelay方法，getDelay方法返回缓存时间与当前时间的差值的毫秒值，然后compareTo通过比较两个元素的getDelay的值进行排序，时间越小，优先级越高，即越靠近堆顶。然后调用循环调用队列的poll方法，内部使用peek得到堆顶元素first，如果first不为null，且first.getDelay大于0，说明还没到时间，返回null，否则说明到时间了，返回poll元素的值，并将公司信息从list中移除。