实战23： 基于JDK的LRU算法实现

1. LRU算法

缓存淘汰算法--LRU算法LRU（Least recently used，最近最少使用）算法

根据数据的历史访问记录来进行淘汰数据，其核心思想是"如果数据最近被访问过，那么将来被访问的几率也更高"

再Java中可以非常简单的实现LRU算法，主要利用的是LinkedHashMap容器

1.1 LRU算法实现

inkedHashMap底层就是用的HashMap加双链表实现的，而且本身已经实现了按照访问顺序的存储。此外，LinkedHashMap中本身就实现了一个方法removeEldestEntry用于判断是否需要移除最不常读取的数，方法默认是直接返回false，不会移除元素

因此沃恩只需要重写这个方法，可以实现当缓存满之后，就移除最不常用的数据

```java
public class LruCache<K, V> extends LinkedHashMap<K, V> {
    private int size;

    public LruCache(int size) {
        super(size, 0.75f, true);
        this.size = size;
    }

    @Override
    protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
        // 当元素个数，超过指定的大小时，淘汰最老的数据
        return size() > size;

    }

    public static void main(String[] args) {
        LruCache<String, Integer> cache = new LruCache<>(4);
        for (int i = 0; i < 6; i++) {
            cache.put("key_" + i, i);
            System.out.println(cache);
        }

        System.out.println(cache.size);
    }
}
```