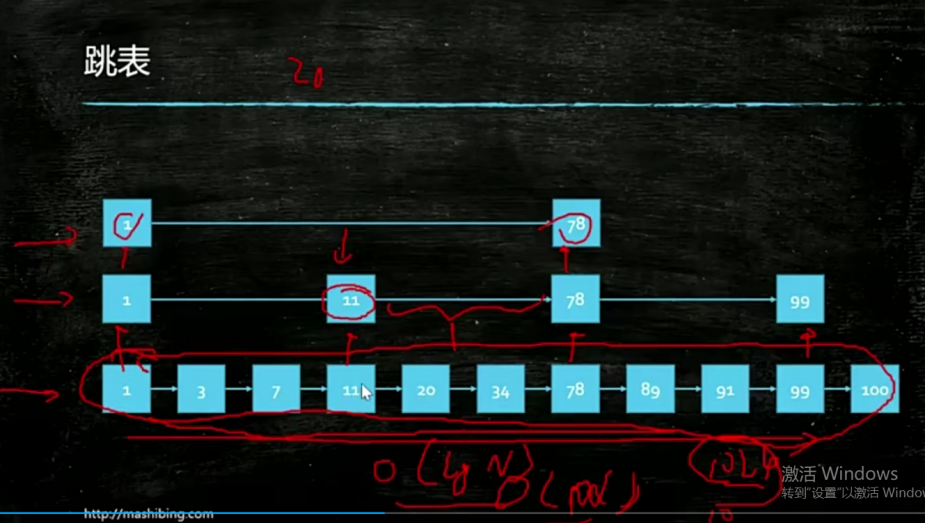


1. Map

* HashTable：方法都加synchronized，有缺陷，现在基本不用。
* HashMap：都不加synchronized。
* synchronizedMap：可以通过Collection.synchronizedMap()把HashMap变为同步，其内部还是使用synchronized，效率也不是特别高，唯一可能的是锁细化了，可能效率提高一点。
* ConcurrentHashMap：支持多线程，使用lock实现。
* ConcurrentSkipListMap：有排序，支持多线程，使用lock实现。



写操作，由于很多判断条件（链表、红黑树），未必比HashTable、synchronizedMap高。

读操作效率很高（数据结构优化）。

1. List

* Vector：方法都加synchronized，有缺陷，现在基本不用。
* ArrayList：方法都不加synchronized。
* SynchronizedList：可以通过Collection.synchronizedList()把ArrayList变为同步
* CopyOnWriteList：写时加锁，读不加锁。适合读很多，写比较少情况。

1. Queue（为了支持高并发）

常用方法：offer（有返回值）、add（会抛异常）、peek（取出不删除）、poll（取出删除）

（1）常用Queue

* ConcurrentLinkedQueue：同步的，lock实现
* LinkedBlockingQueue：阻塞队列，链表实现。多了两个方法：put（添加，如果队列满了，则线程阻 塞等待）、take（取出，如果队列为空，则线程阻塞等待）。天生实现生产者、消费者模型。可以用于MQ。
* ArrayBlockingQueue：阻塞队列，数组实现。有put、take方法。

（2）排序Queue

* PriorityQueue：优先队列，最小堆实现，会继续排序。
* DelayQueue：按照时间进行优先级排列。有put、take方法。可用于按时间进行任务调度。

（3）阻塞等待Queue（transfer：放进去不能立刻离开，需要等待消费）

* SynchronousQueue：容量为0，需要有消费者去消费生产者（调用put操作会阻塞，等待消费者take消费），不能调用add因为容量为0。可用于线程通信。（单个线程等待消费）
* LinkedTransferQueue：有transfer方法（往队列添加数据后，需要阻塞等待别人取走）。可用于等待是否被处理，关注请求是否被处理。（多个线程等待消费）