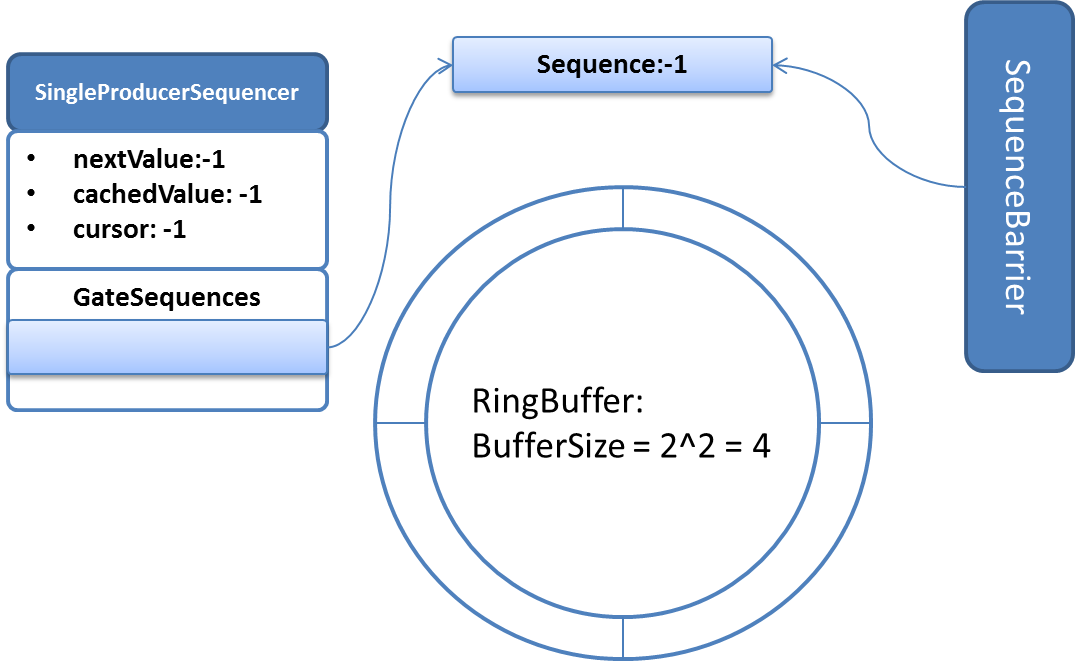
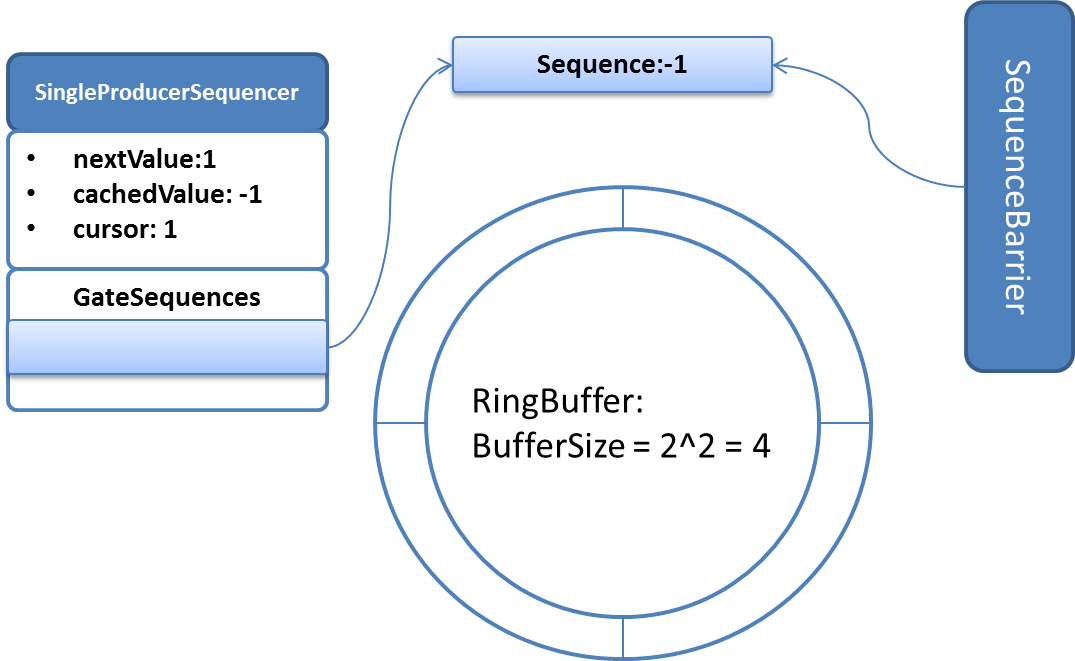
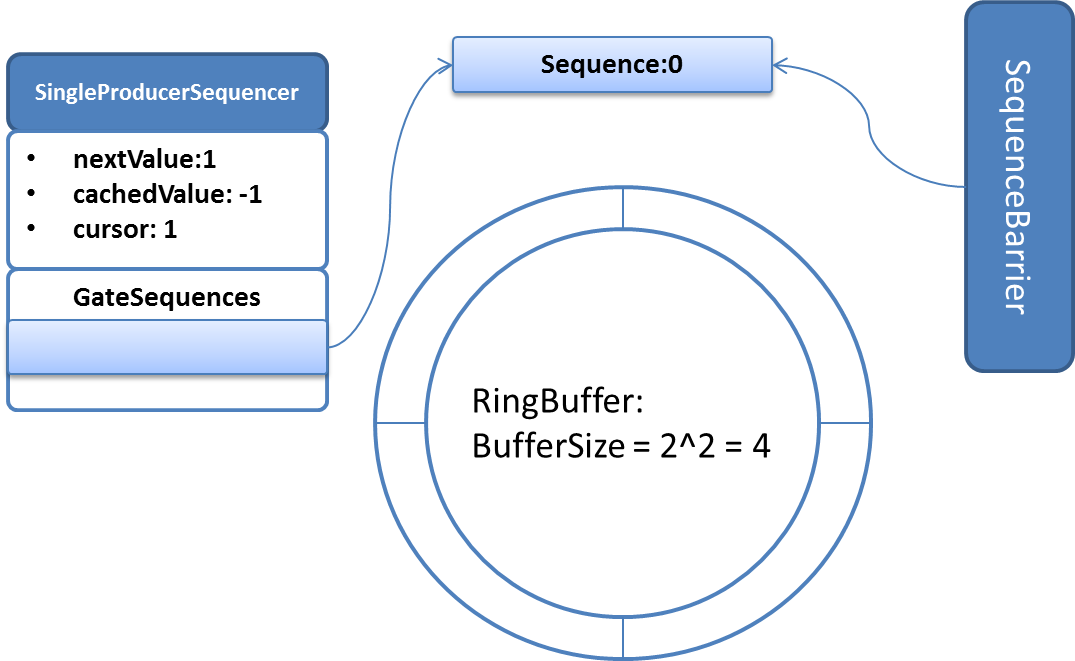
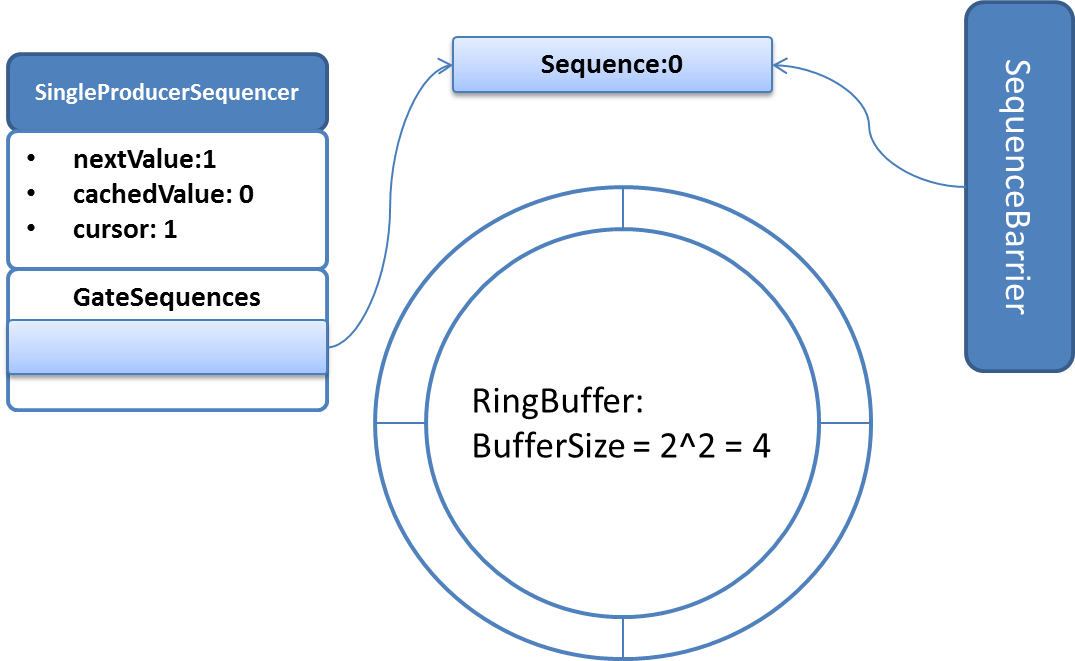
下面，我们针对SingleProducerSequencer画一个简单的工作流程：   
假设有如下RingBuffer和SingleProducerSequencer，以及对应的消费者辅助类SequenceBarrier，这里不画消费者，假设有不断通过SequenceBarrier消费的消费者。SingleProducerSequencer的gatingSequences数组内保存这一个指向某个Sequence的引用，同时这个Sequence也会被SequenceBarrier更新以表示消费者消费到哪里了。这里生产的Sequence还有消费的Sequence都是从零开始不断增长的，即使大于BufferSize，也可以通过sequence的值对BufferSize取模定位到RingBuffer上。   
假设SingleProducerSequencer这时生产两个Event，要放入RingBuffer。则假设先调用hasAvailableCapacity(2)判断下。代码流程是：   
wrapPoint = (nextValue + requiredCapacity) - bufferSize = (-1 + 2) - 4 = -3   
-3 < cachedValue所以不用检查gateSequences直接返回true。假设返回true，就开始填充，之后调用publish更新cursor，这样消费者调用isAvailable根据Cursor就可以判断，sequence:0和sequence:1可以消费了。   
   
假设这之后，消费者消费了一个Event，更新Sequence为0.   
   
之后，生产者要生产四个Event，调用hasAvailableCapacity(4)检查。代码流程是：   
wrapPoint = (nextValue + requiredCapacity) - bufferSize = (1 + 4) - 4 = 1   
1 > cachedValue所以要重新检查，这是最小的Sequence是0，但是1 > 仍然大于最小的Sequence，所以更新cachedValue，返回false。   
   
至此，展示了一个简单的生产过程，SingleProducerSequencer也就讲完啦。