Question 1:
In the absence of synchronization, thread A finds out the queue is not empty. If another thread removes whatever items are on the queue immediately, then in.peek() in thread A returns null. Calling .getClass() on null will throw NullPointerException. If another thread removes whatever items are on the queue after thread A called .getClass() successfully but before thread A called .remove(), then thread A calling .remove() on an empty queue will throw NoSuchElementException.

Question2:
MAX_QUEUE_SIZE constraint is never violated because there is only one producer thread scheduled by MonitorQueues. If there are more than 1 producer, then it is possible that multiple producers check at the same time and find that queue size is 99 and each of them try to put an object on the queue, causing the queue size to exceed 100.

Question 3:
While the concurrent collections in Java are designed for maintaining its own structure under multi-threaded access, it doesn't know about the queue size limitation implemented by the MiddleMan class. By simply synchronizing the "out" queue with a block containing both .size() and .offer() methods, the size checking and item addition must be done in one atomic instruction. This eliminates the possibility that multiple threads are checking the queue size and adding items at the same time thus the queue size will not go over the desired limit.