PartII

1. In the absence of synchronization, why does this happen? What sequence of events is occurring that causes these errors to be raised?

Because there are two middlemen of the same type. Let me take the example of the Integer middleman. Without locking, when one of the integer middleman finds that the condition is met, in the meanwhile, the other integer middleman may take the head element away, causing the queue to be empty, or the head element to be a string type.

The sequence of events could be :

1.m1 and m2 find the condition is met at the same time

2.m1 takes the head element

3.When m2 wants to fetch the element, the queue becomes empty or the head element becomes a string

2. Even though the block isn't synchronized, and even though the queue is a regular LinkedList that doesn't support concurrency protection, why is there never a situation where the MAX_QUEUE_SIZE constraint is violated?

Because there is only one producer process. For the production process, only one process is writing at a time, so there is no problem. Single-threaded operation does not require locking. Because thread safety problems occur when multiple threads read and write simultaneously.

3. Write code to fix these overflows. You may NOT change the DELAY timings to do so, which influences the rate at which data is produced or consumed on the queues. Explain why your modifications work.

We should add a synchronized lock to Integer Queue and String Queue to avoid this problem.

Therefore, I changed the code of MiddleMan.java, beginning from line 56

```
if (out.size() >= 10) {
   continue;
}
else {
   synchronized (out){
      out.offer(outObj);
      if (out.contains(null)) {
         System.out.println("why did this happen?");
      }
      outObj = null;
   }
}
```