1. For this situation a factory would be better. The main reason for this is because each function is distinct and pulls updates on their own (even if they happen at similar intervals). A singleton would let each function stay independent as they would all be waiting for the next available update. Also with a factory if one statistic fails to load or exceeds the expected bounds and returns errors, it will not affect your other functions. The simples way to set this up is to define a port number to each function and have that function pull from the server on the specific port. Server-side the server would know that a connection on a given port means a specific piece of data to return.
2. A flyweight would be better since all of the flights need to be available. A proxy would only allocate the data when needed but since we would need to store flights that are months in advanced a flyweight would be more effective and memory efficient.
3. The visitor pattern is the way to go in this situation. Iterators don’t actually care what the underlining object is when it grabs the next one so it wouldn’t be able to elevate its status or make changes based on that. The visitor can actually look at this next item make the proper modifications. Now that said in many implementations of these two patters both are often use. So in this case u could use the iterator to actually get the next socket and use the visitor after to make modifications if needed.
4. Double checked locking has problems with platform independence so that seems to be out of both the reader and the writing. Now for the writer, one would want a lock pattern as so the data is secure and only 1 entity who has proper access can edit the stock. Now for the reader, one should implement no described pattern as that would slow the system down and lock a few people out after so many people are trying to read at once. Which we defiantly do not want. With a secure write pattern and an endless read access pattern this should work will for the stock market system.