Elijah Smith
COP4520 HW3
P2 – Atmospheric Temperature Reading Module

This simulation takes a coarse-grained approach to ensuring the list of hottest and coldest temperatures is thread safe between the 8 modules. The only access point is through a synchronized method offerTemp, which locks the list entirely and attempts to insert the provided temperature at the appropriate location (if there is one). Using this approach, we guarantee mutual exclusion between all 8 temperature sensors and will not run into any unexpected behavior. Each temperature is guaranteed to get their chance to offer their temperature. In this simulation this may not happen in perfect order (each sensor offers for one minute, then the next, etc.) depending on how Java decides to choose the next SensorThread to acquire the lock, but this is irrelevant in the context where each thread would be submitting one number every minute and the order in which they offer temperatures is completely inconsequential.

While a coarse-grained approach may cause bottlenecking in situations where lots of threads hope to acquire the same lock and spend time holding it, the context of this problem means that it is a perfectly serviceable and efficient choice. With each sensor submitting one reading (a very quick operation) every minute before waiting for the next reading, there will never be contention that may cause bottlenecking.

When experimenting, there are again no clear parameters to tweak and evaluate efficiency. Each execution takes around 5ms to complete the simulation, which doesn't actually wait any time between readings.