Dining Philosophers Problem

One common problem is the dining philosopher's problem to synchronization. One simple solution is making chopsticks a semaphore to prevent the accessing of the shared memory location. The problem can still in a deadlock if all the philosophers are hungry and pick up only their left chopstick at the same time. One way to prevent this is by making an odd philosopher only pick the left and then the right chopstick and even philosophers pick up the right and then the left chopstick. Only have four philosophers at most at the table. They have the philosophers pick the chopstick at the same time if both are available.

They solution that I made was that are five philosophers that implemented runnable to have make them threads. Each of them has two shared chopsticks that are shared with the neighbors. The part of the code that philosophers pick up the chopstick is protected by synchronize. This allows only have one thread in that block of code at the time on the monitor objects called chopsticks. So, all the philosophers must wait till that thread has finished executing the pickup, eat, and putdown functions on that instance of the chopstick. To prevent the deadlock, I have implemented that odd philosophers only pick the left and then the right chopstick and even philosophers pick up the right and then the left chopstick. This prevent circle of wait if all the philosophers pick up the chopstick from the same side.