# Dining Philosophers Priority

In a previous lab we concluded we would use 3 logic-based methods to decide if philosophers should give up access to a chopstick (a semaphore). This time we propose that if all philosophers are left in a waiting state (i.e., that all philosophers currently hold one chopstick and are waiting on others), then we should force the philosopher with lowest priority to drop (release) their currently held chopstick and allow all others to attempt to retrieve the second chopstick.

We will do this with multiple steps.

1. We require a method to signal that the philosophers are currently in deadlock. A counter of some kind that increases when a philosopher occupies a single chopstick and decreases when the same philosopher acquires a second chopstick (note this is important since deadlock can only occur when every philosopher only has one chopstick, so we know we are in deadlock when the counter equals 5).
2. When a philosopher acquires their first chopstick, they are then entered into a priority queue.
3. The priority queue requires that the philosophers are compared in some way. To accomplish this, we will have to ensure that we use implements comparable and edit the compareTo method such that the id’s of the philosophers are used for comparison.
4. If the counter from step 1 reaches 5 deadlock has occurred and a notifyAll is called.
5. A condition should be on the acquire method, where if the current philosopher is equal to the head of the priority queue, then it should release its current (only) chopstick. This in turn should use a notifyAll to allow all other threads to acquire the free chopstick.