Nimesh Silva

January 28, 2021

Dr. Bullumulle

CYB 5285

3.2 Assignment: Dining Philosophers

This program was written in Java. The aim is to create a procedure for philosophers to accomplish their goal of eating and dreaming without starving to death. The philosopher gets hungry after some time and wants to feed. The philosopher reaches for the forks on either side of him to do this. Once the philosopher succeeds in getting both people, he starts to feed. If he puts both forks down, his neighbor's philosopher has access to those forks. We need to lock it to simulate acquiring a fork so that no two philosophical threads obtain it at the same time. We use the synchronized keyword to obtain the fork object's internal monitor to accomplish this and prevent other threads from doing the same. This is performed in the process run(). For a moment, a philosopher thinks and then decides to eat. To explain the order in which actions occur, timestamps have also been applied to each operation. As generic Java objects, we model each of the forks and render as many of them as philosophers exist. Using the synchronized keyword, we pass each Philosopher his left and right forks that he tries to lock. Both philosophers, without triggering a deadlock, get their chance to think and eat. We analyzed the popular Dining Philosophers problem using threads in this paper. My code can be found here <https://github.com/nimeshsilva1997/JVM-Projects/tree/master/Java/Dining>.





