Mert Ilgınoğlu, 26774

CS 307 Homework 2 Report

At first, I have tried to implement take_forks() and put_forks() functions outside of run. But this gave me buggy results. Then I chose to do them in run part.

In the first part, we create a random integer between 1-10 with random module of Java. Then we sleep them according to that number. This corresponds to philosophers coming to the table in random times. Then we use putplate to indicate they have come to the table and follow with our barrier code. In barrier, we release all the semaphores except current philosopher's, then acquire the current philosopher's semaphore 4 times (4 times because the other rest of the philosophers will release this current philosopher's semaphore once) . This enables us to create a barrier when we have 5 philosophers.

When all is at the table, we start dining. At first philosophers think for 0-10 seconds. Then we come to the take_forks() part in the lecture slides. Here, we use mutex since we are going to access state array, which is a shared variable. Then we put philosopher to hungry and test if he can eat, with test() function. If he can, we up his semaphore in test() function and release mutex and down its semaphore. (if he couldn't eat it means that we couldn't up his semaphore in test. Hence philosopher will be staying there until it can eat) If it could eat, then we move onto put_forks() part. Here, we put forks and use test on its neighbors, to see if they are hungry currently. If so, then we are going to up their semaphore in test, hence they will able to continue from and start eating. This will go in a loop forever.