Kiet Nguyen, #027353519

Thai Tran, #026819986

Report #2

The program design is composed of four files: DiningPhilosophers, DiningServer, DiningServerImpl, and Philosopher. A ReentrantLock is created, and the methods for DiningServerImpl (provided by the DiningServer interface) are implemented first. The takeForks(int philNumber) method is used to allow a philosopher (thread) to lock a fork, indicating that the fork is in their possession. Once a fork has been locked, it cannot be locked by another philosopher until it is released.

The Philosopher.java class has three variables: "lFork" and "rFork" (instances of DiningServerImpl), and "philosopherNumber", which makes it easier to keep track of each philosopher without using 0 as their number. The class also includes a function called "run()" that outlines the philosopher's sequence of actions: first, the philosopher tries to pick up and lock the left fork (if it is available), then they pick up and lock the right fork (if it is available), they eat, and finally, they unlock both forks.

In DiningPhilosophers.java, we instantiated five philosophers and five forks. We assigned each philosopher to their left and right forks, following the convention in the Dining-Philosopher problem. However, we modified the implementation for the fifth (last) philosopher such that they would pick up their right fork first instead of their left, unlike the other four philosophers. This prevents a deadlock case of circular wait. Once the philosophers and their forks are created and linked, we create a thread for each philosopher and start their execution.

The work distributed was equal, as both team members collaborated through online messaging and together in the library on the project. Kiet Nguyen worked on creating the Philosophers in the DiningPhilosophers file, along with debugging the program. Thai Tran implemented the Philosophers file and the DiningServerImpl file to get the program running.

Video recording link: https://youtu.be/Ly3xwTYlxNY