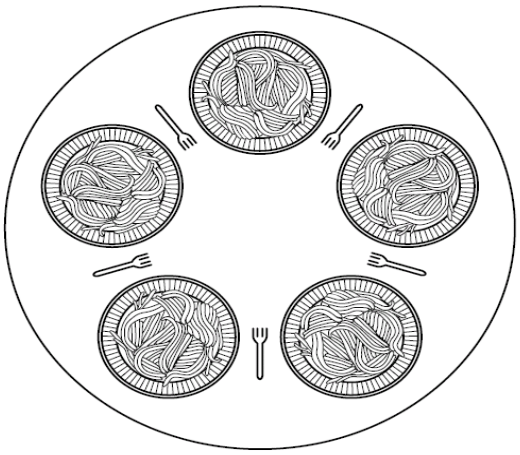
CS 3230 - Computer Architecture and Operating Systems
Spring 2017
Program #4
Issued 04/12/2017 - Due 04/30/2017

Consider the famous ***Dinning Philosophers problem***. The problem was first defined and solved by Dijkstra in 1972. **N** philosophers are seated at a round table, where each philosopher has a plate of spaghetti and there is a single fork between each two plates, as shown below. For more information about this problem, refer to chapter 6, section 6.6 in our text book.



A philosopher alternates between thinking and eating. In order to eat spaghetti, a philosopher must obtain both the left and the right forks, but releases them when thinking. If a fork is not available, a philosopher must wait for it to be released by his/her neighbor before being allowed to eat. Accordingly, at no time may two philosophers hold the same fork. That is, the main question is how can we enable all the philosophers to eat with the shared forks?

A philosopher is allowed to eat for no more than 1 second before releasing both forks to think again, so as not to starve the other philosophers. After a random number of milliseconds of thinking, a philosopher may try to eat again but has to re-acquire both forks. Therefore, a philosopher can be in one

of the following states:THINKING, WAIT_LEFT_FORK, WAIT_RIGHT_FORK, and EATING. Below is a fragment of a Philosopher class.

```
public class Philosopher extends Thread
{

        // each Philosopher is assigned an integer ID and two forks
        .
        .
        public void run()
           {
                while (! time to go home)
                {
                     System.out.println (ID + "THINKING");
                     // TODO: think for random number of milliseconds <=1 sec
                     // TODO: obtain the fork on the left
                    // TODO: obtain the fork on the right
                     System.out.println (ID + "EATING");
                     // TODO: eat for random number of milliseconds <= 1sec
                   // TODO: release the fork on the left
                  // TODO: release the fork on the right
                }
         }
        .
        .

}
```

In addition to the Philosopher class, you will need a Fork class for enforcing synchronized access to each fork.

Your main program must create **N** Fork objects and **N** Philosopher objects and assign to each Philosopher the correct pair of forks (modulo **N**), one shared with the Philosopher on the left and one with the Philosopher on the right. Create each Philosopher thread and start it running in your main procedure.

On termination, the program must report the following statistics:

- How many times each philosopher got to eat?
- How long (in milliseconds) each philosopher spent eating and thinking?

Such statistics can verify that no starvation occurs for any Philosopher.

REQUIREMENTS and SUBMISSION:

- You will optionally work in groups of two for this assignment. As such, you must split up the work appropriately. Indicate in the code comments who worked on each code block.
- Add a comment at the beginning of your code with group member names.
- When you are done, you should submit one copy for each into D2L DropBox.