Recitation 8

Zeinab Khalifa

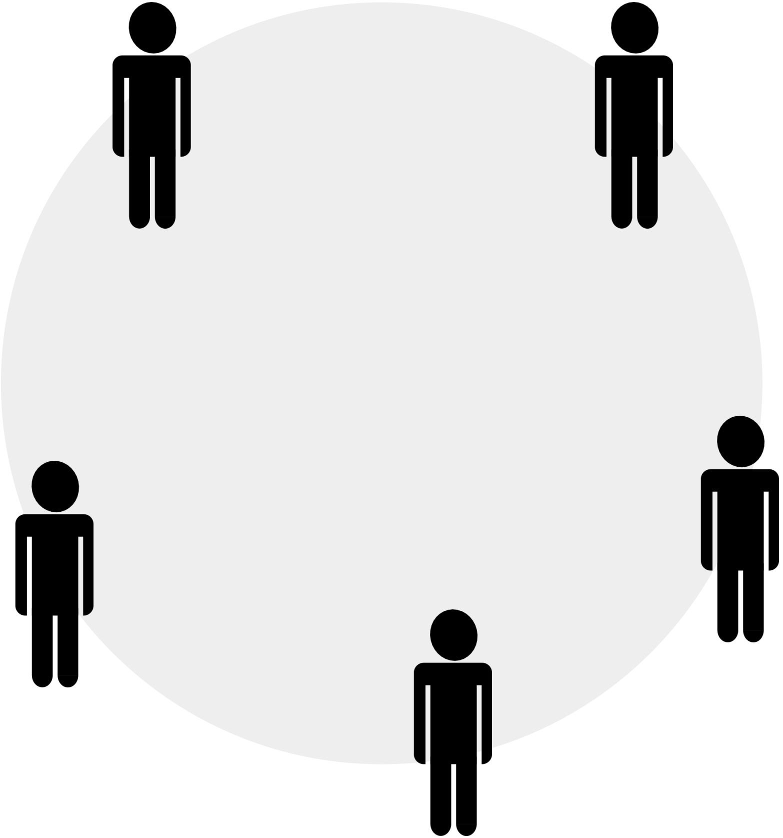
October 22nd, 2020

Carnegie Mellon University Qatar

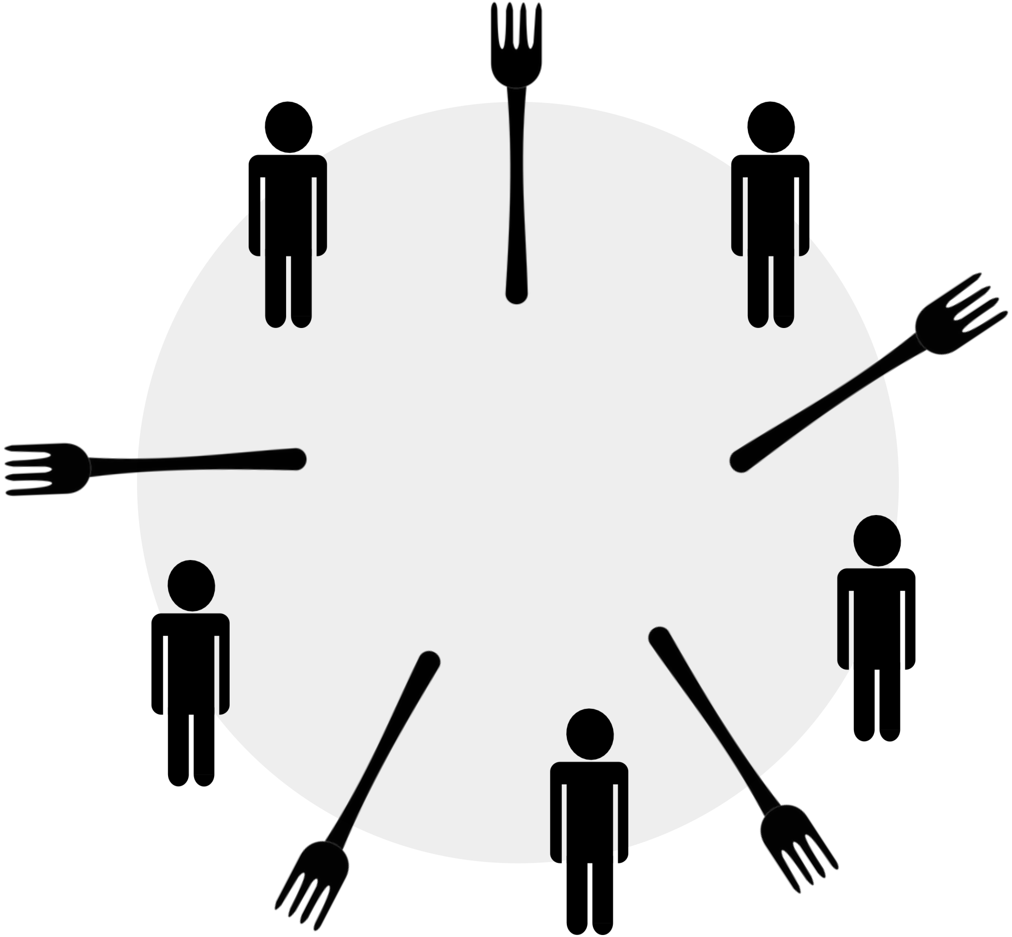
Announcements

- PS4 will be released on October 26th, 2020 and due on November 4th, 2020
- P2 is due on October 28th, 2020.

Dining Philosophers

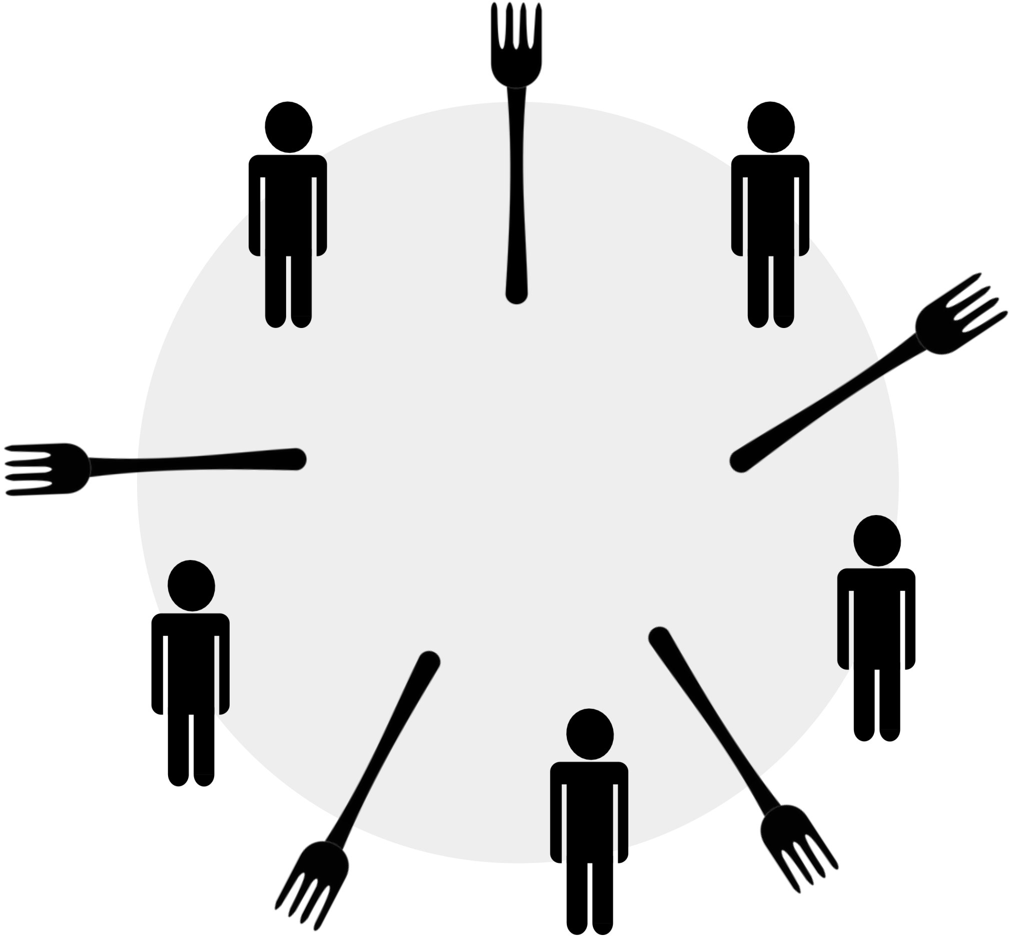


Dining Philosophers



Dining Philosophers

- *Actions: Thinking and Eating*
- *Each P needs a pair of forks*
- *When a P is done eating, he is back to thinking and puts back his forks*



Dining Philosophers

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Step 2: think until the right chopstick is available; when it is, pick up;

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Step 2: think until the right chopstick is available; when it is, pick up;

Step 3: when both chopsticks are held, eat for a fixed amount of time;

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Step 2: think until the right chopstick is available; when it is, pick up;

Step 3: when both chopsticks are held, eat for a fixed amount of time;

Step 4: then, put the right chopstick down;

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Step 2: think until the right chopstick is available; when it is, pick up;

Step 3: when both chopsticks are held, eat for a fixed amount of time;

Step 4: then, put the right chopstick down;

Step 5: then, put the left chopstick down;

Dining Philosophers

Step 1: think until the left chopstick is available; when it is, pick up;

Step 2: think until the right chopstick is available; when it is, pick up;

Step 3: when both chopsticks are held, eat for a fixed amount of time;

Step 4: then, put the right chopstick down;

Step 5: then, put the left chopstick down;

Step 6: repeat from the beginning.

Dining Philosophers

A concurrent system with a need for synchronization, should ensure

Correctness **Efficiency** **Fairness**

Dining Philosophers

A concurrent system with a need for synchronization, should ensure

Correctness

No two philosophers should be using the same chopsticks at the same time.

Efficiency

Fairness

Dining Philosophers

A concurrent system with a need for synchronization, should ensure

Correctness

No two philosophers should be using the same chopsticks at the same time.

Efficiency

Philosophers do not wait too long to pick-up chopsticks when they want to eat.

Fairness

Dining Philosophers

A concurrent system with a need for synchronization, should ensure

Correctness

No two philosophers should be using the same chopsticks at the same time.

Efficiency

Philosophers do not wait too long to pick-up chopsticks when they want to eat.

Fairness

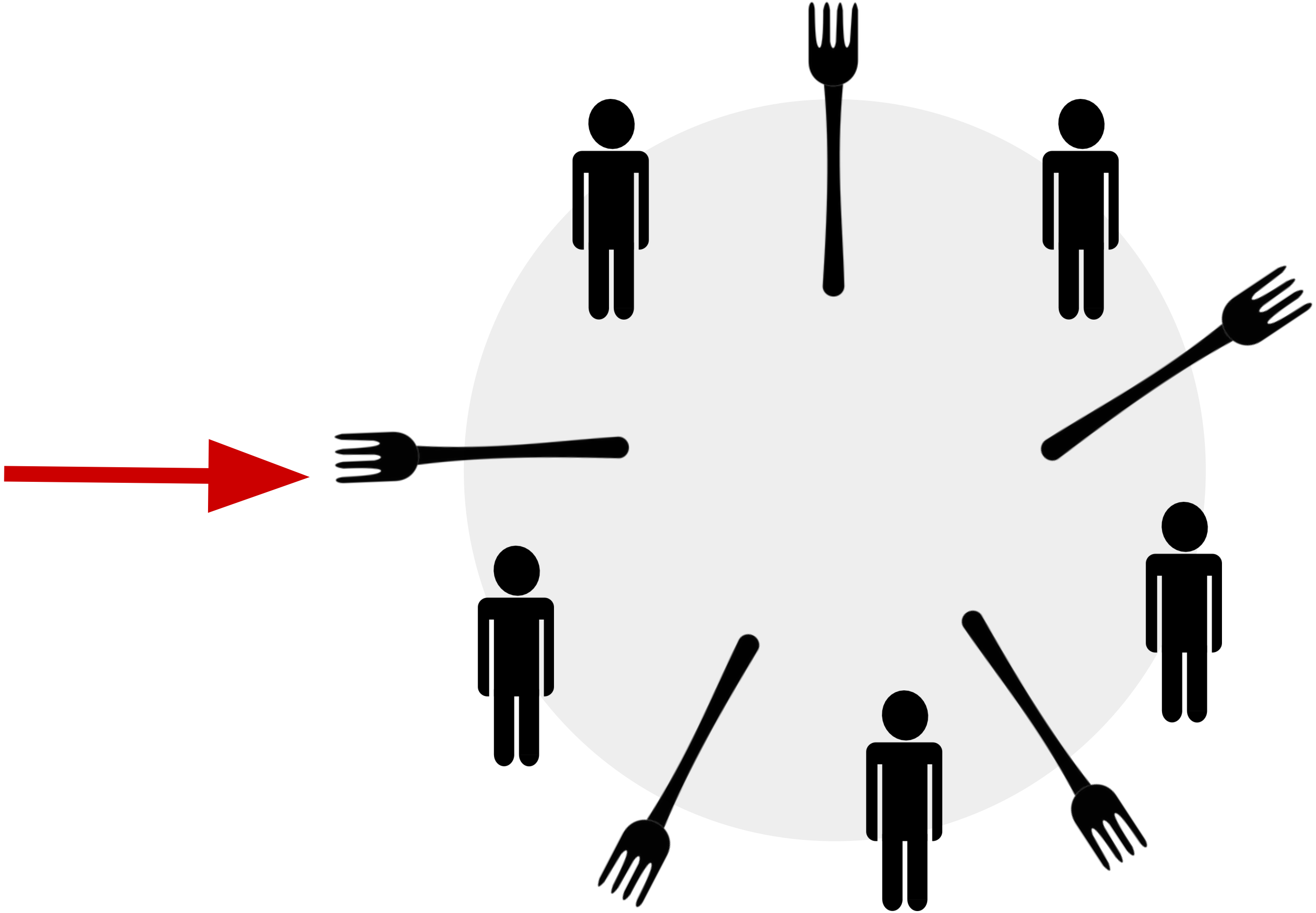
No philosopher should be unable to pick up chopsticks forever and starve

Pseudocode

```
while(true) {
    // Initially, thinking about life, universe, and everything
    think();
    // Take a break from thinking, hungry now
    pick_up_left_fork();
    pick_up_right_fork();
    eat();
    put_down_right_fork();
    put_down_left_fork();

    // Not hungry anymore. Back to thinking!
}
```

What’s wrong with the previous code?



Lock on Objects!

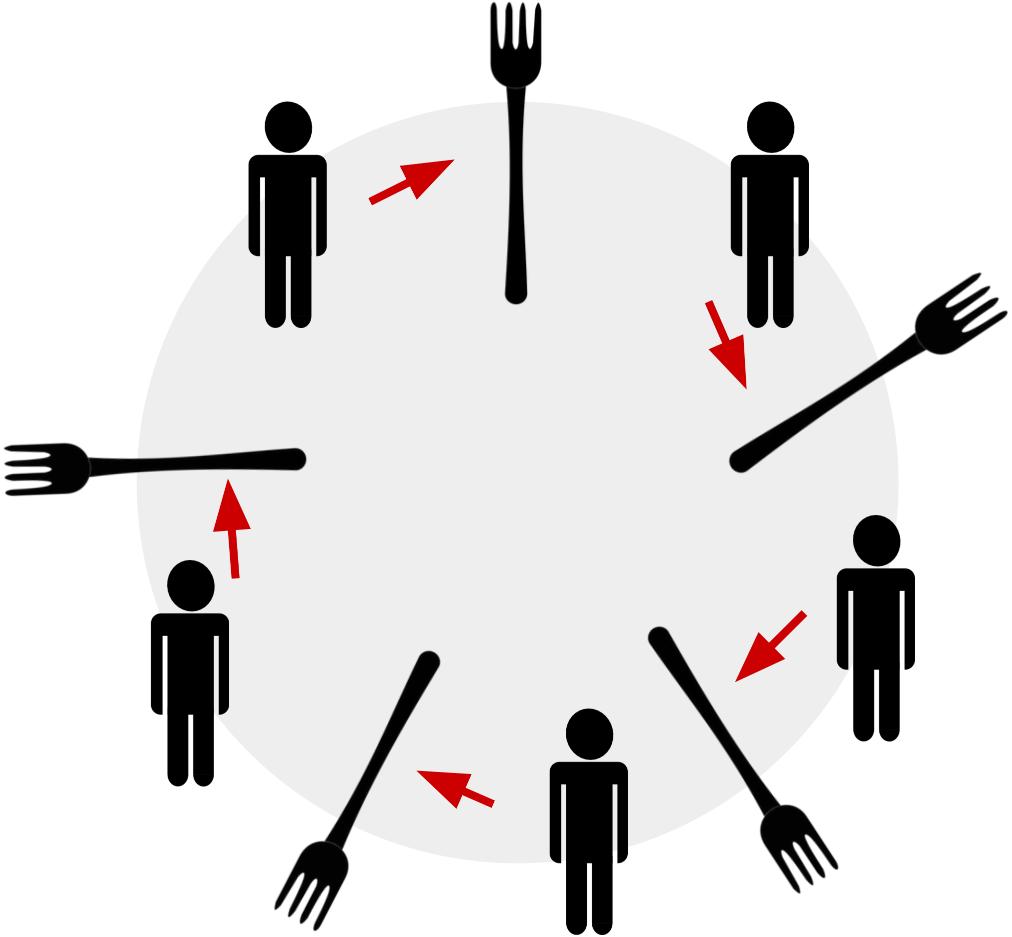
Still problematic!

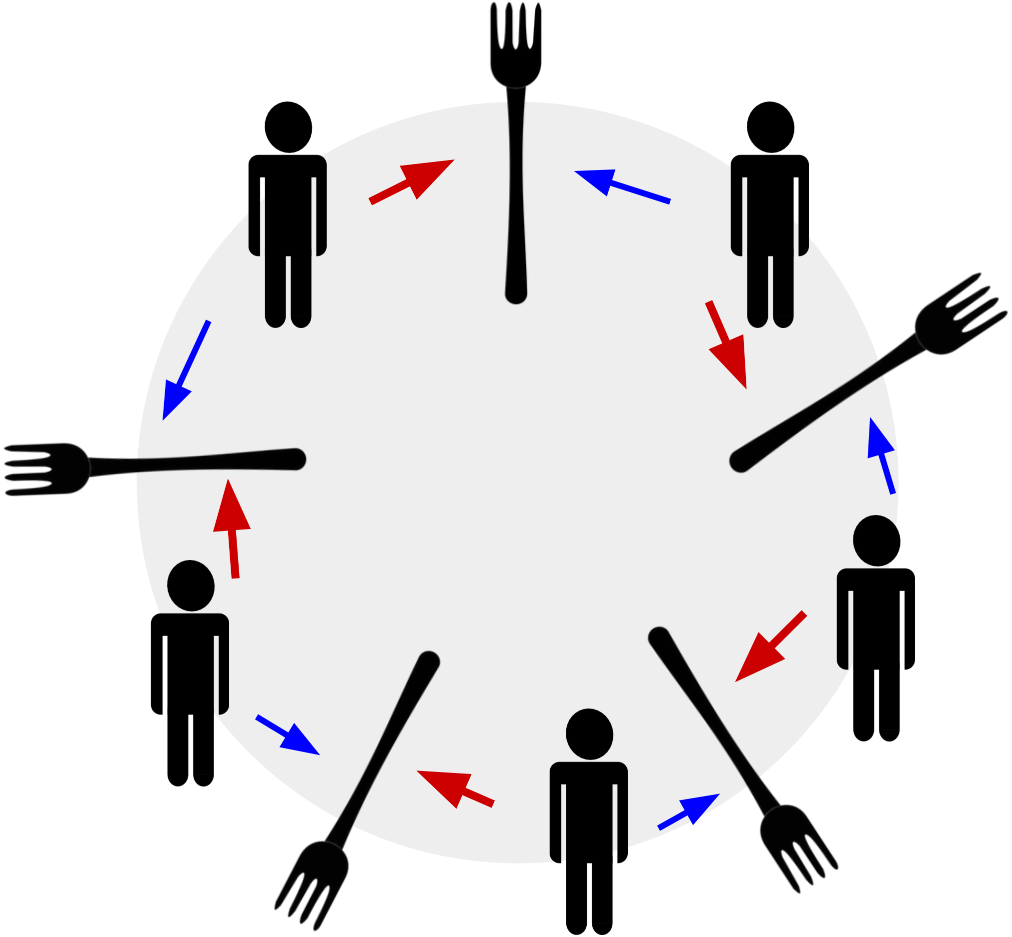
Detecting Deadlocks using the Terminal

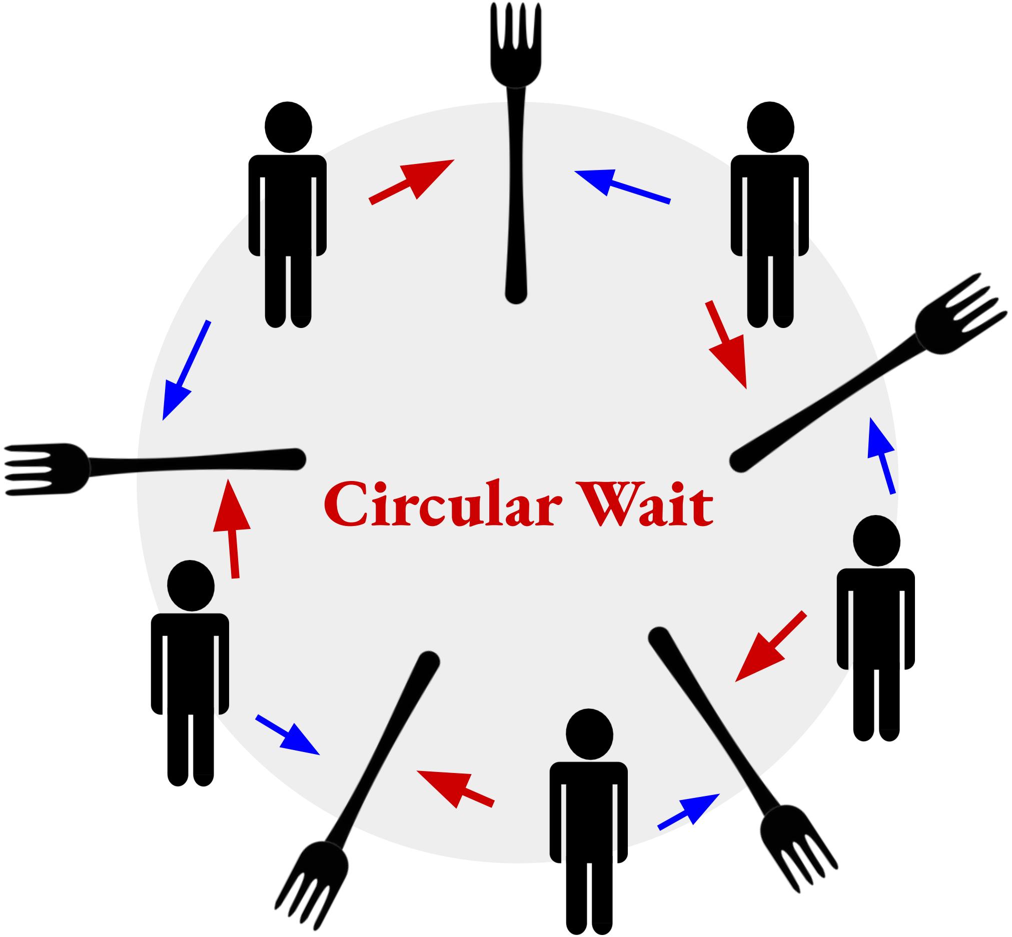
java -classpath . DiningPhilosophers (in ubuntu)

Jps -l -m (lists the running)

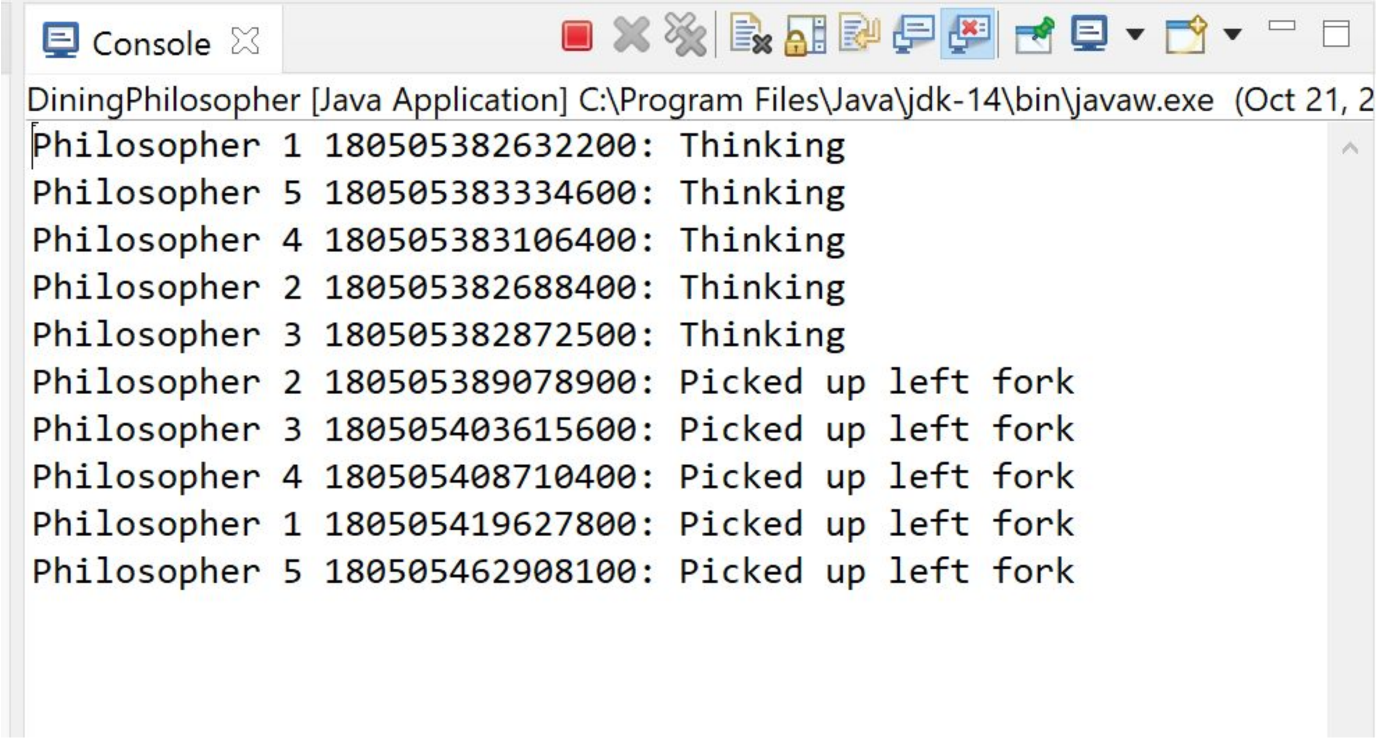
Jstack <process_number>







Circular Wait



DiningPhilosopher [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Oct 21, 2

```
Philosopher 1 180505382632200: Thinking
Philosopher 5 180505383334600: Thinking
Philosopher 4 180505383106400: Thinking
Philosopher 2 180505382688400: Thinking
Philosopher 3 180505382872500: Thinking
Philosopher 2 180505389078900: Picked up left fork
Philosopher 3 180505403615600: Picked up left fork
Philosopher 4 180505408710400: Picked up left fork
Philosopher 1 180505419627800: Picked up left fork
Philosopher 5 180505462908100: Picked up left fork
```

Dining Philosophers

A concurrent system with a need for synchronization, should ensure

Correctness

No two philosophers should be using the same chopsticks at the same time.

Efficiency

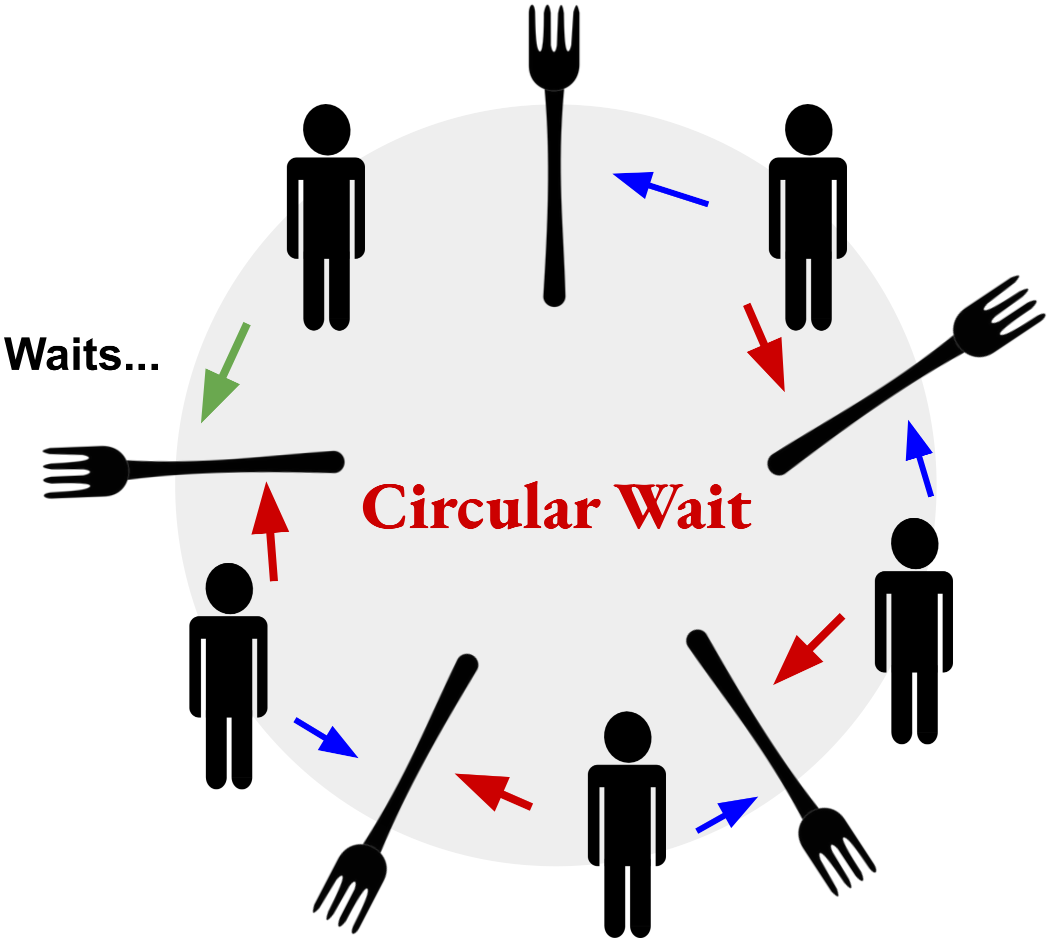
Philosophers do not wait too long to pick-up chopsticks when they want to eat.

❌

Fairness

No philosopher should be unable to pick up chopsticks forever and starve

How can we break the cycle?



Circular Wait

Waits...

Only 4 philosophers at a time...

Assume we now have an additional waiter who allows only 4 philosophers at the table at any given time. The waiter will only allow another philosopher to join once there are <4 philosophers at the table. Is there a desirable property of concurrent systems that is still violated? If so, give an example of when it is violated.