Dining Philosophers and Deadlocks

How to Introduce Deadlocks in the Dining Philosophers Solution 2 (Semaphore-Based)

In Solution 2, each chopstick is represented by a binary `Semaphore`, and each philosopher tries to acquire the left and right chopsticks before eating. By default, to avoid deadlock, the solution uses *asymmetric chopstick acquisition*: philosophers with even IDs acquire the left chopstick first, while those with odd IDs acquire the right chopstick first.

To **intentionally introduce deadlock**, you can make the following changes:

---

1. Make All Philosophers Acquire Chopsticks in the Same Order

Replace the asymmetric logic in the `pickup(int i)` method with a uniform acquisition order:

```java
public void pickup(int i) throws InterruptedException {
    int left = i;
    int right = (i + 1) % NUM_PHILOSOPHERS;

    // All philosophers now acquire left first, then right
    chopsticks[left].acquire();
    chopsticks[right].acquire();
}
```

This creates a circular wait:
Each philosopher holds the left chopstick and waits indefinitely for the right chopstick, which is held by their neighbor.

---

2. Remove or Minimize Thinking Delay

In the `main()` method, remove or shorten the thinking time before `pickup()` to make philosophers act simultaneously:

```java
```

```
Thread.sleep(0); // Or remove this line entirely
```

This increases the likelihood that all philosophers try to pick up chopsticks at the same time, triggering the deadlock condition.

Summary

These changes satisfy the **four Coffman conditions** for deadlock:

1. **Mutual exclusion** – Each chopstick (semaphore) can only be held by one philosopher at a time.
2. **Hold and wait** – Each philosopher holds one chopstick while waiting for another.
3. **No preemption** – A philosopher cannot forcibly take a chopstick from a neighbor.
4. **Circular wait** – A cycle of philosophers each waiting on the next.

With all four conditions met, deadlock becomes possible and likely.