Dining Philosophers Problem with Arbitrator Solition

Arbitrator Solition:

The idea of this solution is to introduce a third party that monitors the usage of resources. In this case, we would properly call it a waiter. The principles are as follows:

When a unit of work wishes to access a resource, it asks the arbitrator for permission first. That is, when a philosopher wishes to eat, she first asks the waiter.

1. The arbitrator gives one permission at a time.

2. Only the unit that has the permission can access shared resources.

3. Releasing a resource does not need permissions.

There are two major setbacks for this solution. First of all, a new central entity is introduced, which would require additional resources. Also, it results in **reduced parallelism**: if a philosopher is eating and one of their neighbors requests the chopsticks, all other philosophers must wait until this request has been fulfilled even if their chopsticks are still on the table.

Solition:

- We take necessary parts from the teacher’s code
- Then we implement the Arbitrator Solition to the code
- And in the main code we chance table function to display our method