Solution:

Here's a Solution

One solution of this problem is to use semaphores. The semaphores which will be used here are:

mutex, a **binary semaphore** which is used to acquire and release the lock, it’s act like a gate which let one thread to enter.(initialized to value 1)

* empty: a **counting semaphore** whose initial value is the number of slots in the buffer, since initially all slots are empty. (Initialized to value n)
* full, a **counting semaphore** whose (initial value is 0) .

At any instant, the current value of empty represents the number of empty slots in the buffer and full represents the number of occupied slots in the buffer

Consumer class:

We have here cook shelf(shared variable between producer and consumer) variable which is initially null(free) So in the constructor we send our consumer to initial the list. Function get Serve meals ,we put synchronized block to protect our shared variable and make a while loop if it’s empty print that the shelf is empty and there isn’t pizza and use wait() to cook shelf which mean that wait until the producer to produce another pizza Then wait for (5seconds) that indicate the number of seconds the servant take to serve the customer Then notify(wakeup) the producer to produce another pizza For customers

Text

Description automatically generated

The application class:

That is our main class that create the threads that will handle the tasks for cooker and servant(producer and consumer)

Text

Description automatically generated