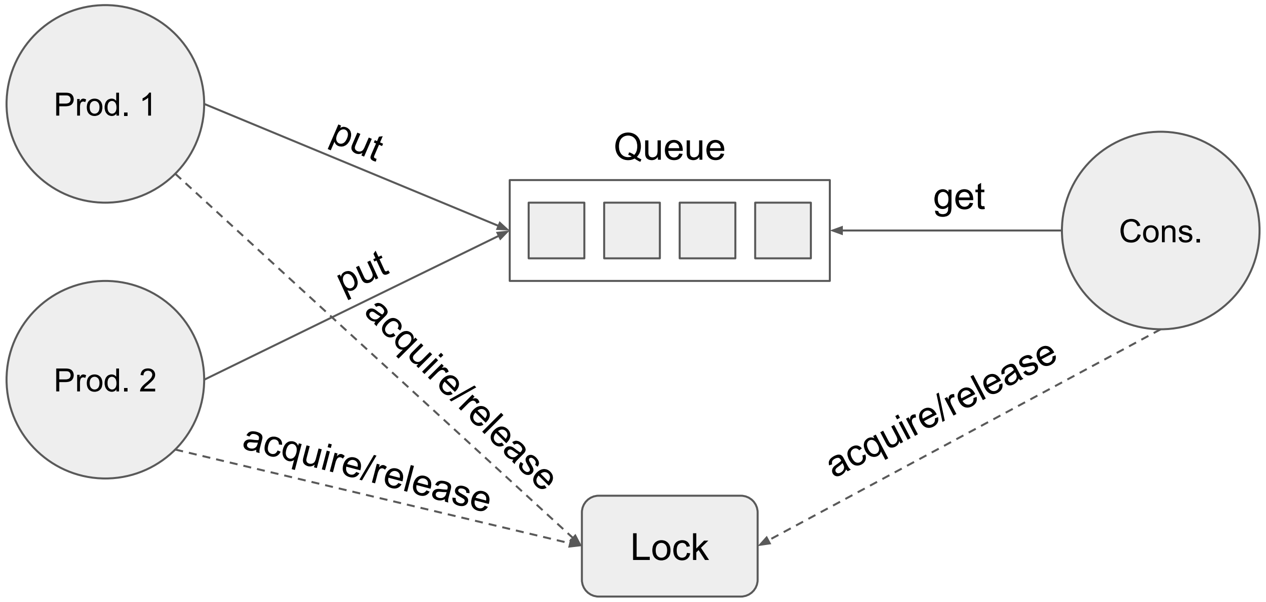
Exercise 2

Distributed Software Systems — Prof. Paolo Ciancarini
Università di Bologna
A.Y. 2023/2024
Riccardo Scotti (0001060133)

Scenario


Prod. 1
Prod. 2
put
put
Queue
get
Cons.
acquire/release
acquire/release
acquire/release
Lock

Producers behavior

1. produce data and JSON serialize it
2. acquire lock (to ensure mutual access to the queue)
3. put data inside the queue
4. release lock

Consumer behavior

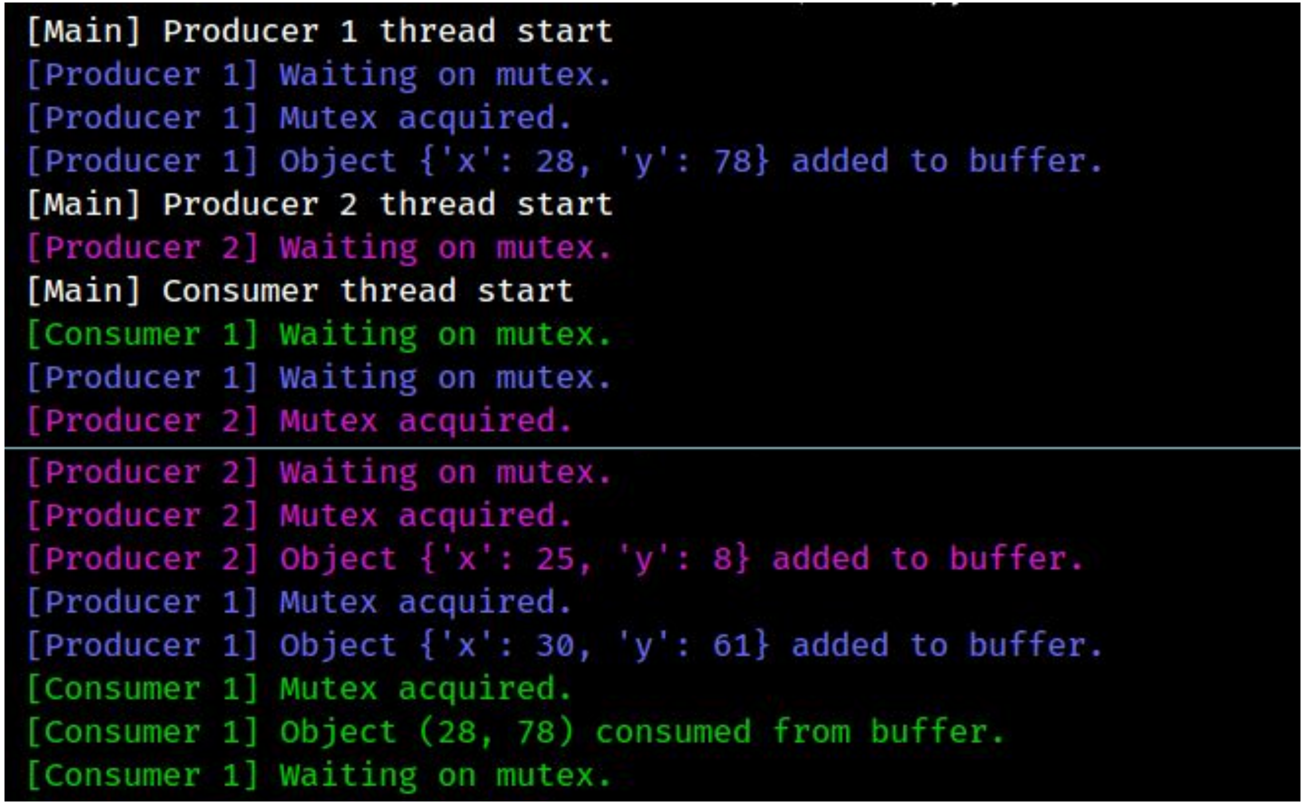
1. acquire lock
2. if queue is not empty
   a. get one element from queue
   b. deserialize it and print it as a formatted string
   c. release lock
3. if queue is empty, release lock immediately

Possible issues

- there is only **one** lock; there is no mechanism to ensure a fair balancing in the access to the resource between the two producers, or between producers and consumer;
- the consumer unnecessarily acquires the lock when the queue is empty.

Both of these problems can be solved by using a more sophisticated mechanism for synchronization, such as a monitor.

Concurrent behavior



```
[Main] Producer 1 thread start
[Producer 1] Waiting on mutex.
[Producer 1] Mutex acquired.
[Producer 1] Object {'x': 28, 'y': 78} added to buffer.
[Main] Producer 2 thread start
[Producer 2] Waiting on mutex.
[Main] Consumer thread start
[Consumer 1] Waiting on mutex.
[Producer 1] Waiting on mutex.
[Producer 2] Mutex acquired.
[Producer 2] Waiting on mutex.
[Producer 2] Mutex acquired.
[Producer 2] Object {'x': 25, 'y': 8} added to buffer.
[Producer 1] Mutex acquired.
[Producer 1] Object {'x': 30, 'y': 61} added to buffer.
[Consumer 1] Mutex acquired.
[Consumer 1] Object (28, 78) consumed from buffer.
[Consumer 1] Waiting on mutex.
```

The three threads exhibit a concurrent behavior, in the fact that they are executing at the same time (in reality they are interleaved, as we can see from the outputs on the shell).

Observations

- when the lock is released, the access to the resource is not granted in a *first come first served* fashion; rather, it seems that there is no criterion whatsoever behind the choice;
- as said before, by using a mechanism such as a monitor, we could specify a particular scheduling policy which may, for example, ensure that a consumer can try to gain access to the queue only when it is not empty, or that a single producer cannot access the queue two times in a row.