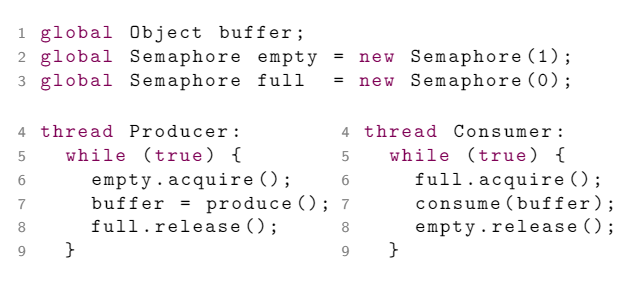
RACE CONDITION – A timing dependent error involving shared state.  
Mutual Exclusion Problem (MEP) – Mutex, Absence of deadlock, Free from starvation.

STARVATION - a process is perpetually denied necessary resources to process its work.

SEMAPHORES – An abstract data type with atomic expressions: aquire (wait) and release (signal)

- and has two data fields: permissions (int > -1) and a list of processes.

All a MUTEX is, is a BINARY SEMAPHORE (int = 0 || 1)

=============================================EXAMPLES=============================================

| global Semaphore mutex = new Semaphore (1); |

| thread P: { | thread Q: { |

| // non-critical section | // non-critical section |

| mutex.aquire(); | mutex.aquire(); |

| // critical section | // critical section |

| mutex.release(); | mutex.release(); |

| // non-critical section | // non-critical section |

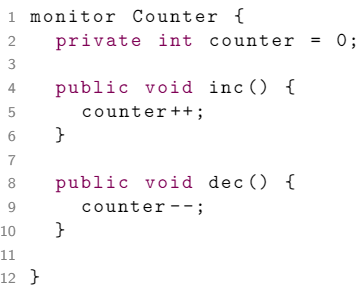
| } | } |

==================================================================================================

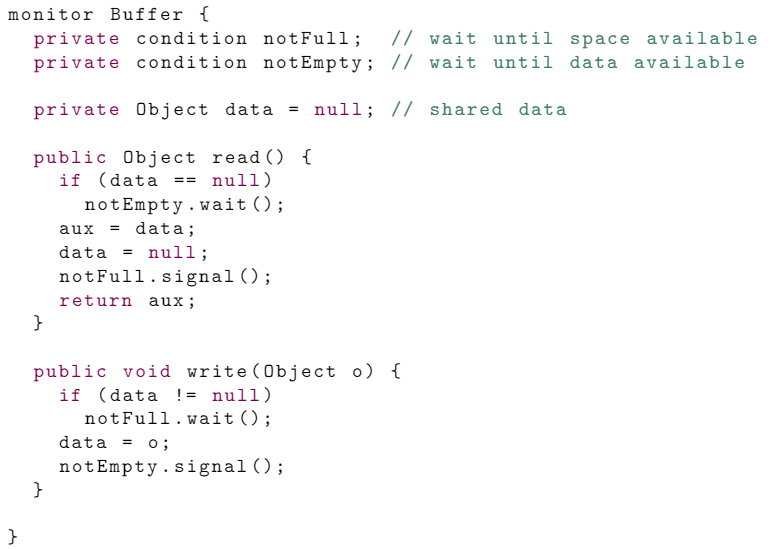
MONITORS – Combines ADTs and mutual exclusion. It comes equipped with a LOCK or MUTEX.

Ex. Counter has inc() and dec(). No two threads can simultaneously modify counter.

CONDITION VARIABLES come with monitors. Include: wait(), signal(), and empty()

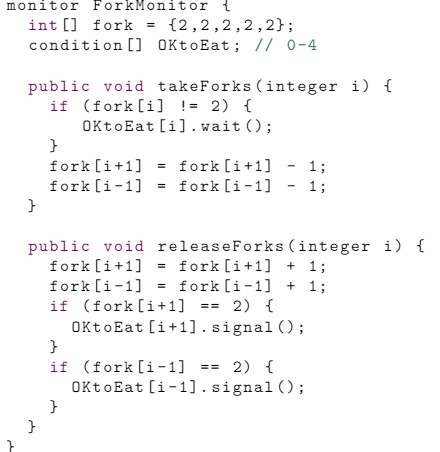
=========================================MONITOR-EXAMPLES=========================================

🡨 Simple Example

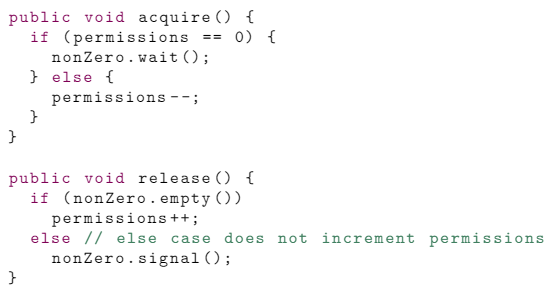


Buffer

Example🡪

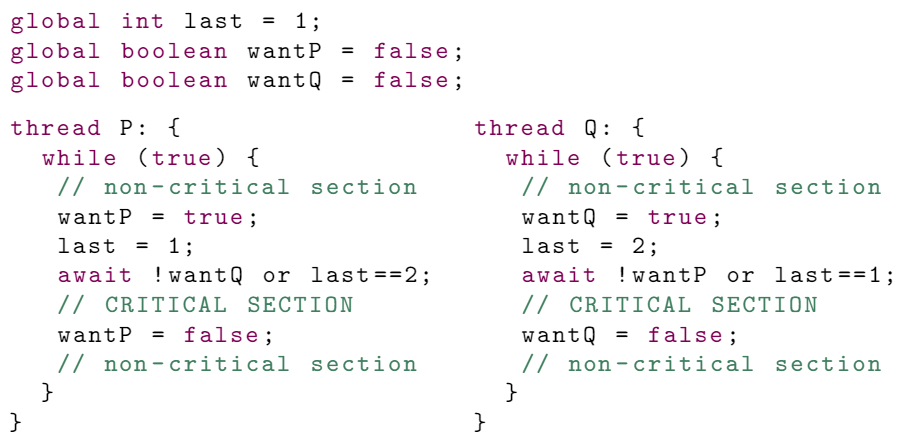


🡨 Dining Philosophers



🡨 Fair

Semaphore

==============================================EXTRAS==============================================

🡨 Peterson’s Algorithm (Solves MEP)