Múltiplas Variáveis de Condição

Sistemas Distribuídos





```
class Warehouse {
 private Lock 1 = new ReentrantLock();
 private Map<String, Item> stock = new HashMap<>();
 private class Item {
  int q = 0;
  Condition isEmpty = 1.newCondition();
 }
 public void supply (String item, int quantity);
 public void consume (String[] items)
   throws InterruptedException;
```





Método Warehouse.supply

```
void supply (String item, int quantity) {
    Item it = this.get(item);
    it.q += quantity;
}
```

Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {
   for (String item : items) {
     Item it = this.get(item);
     it.q -= 1;
   }
}
```





- Cliente Greedy
 - Método Warehouse.supply





- Cliente Greedy
 - Método Warehouse.supply

```
public void supply (String item, int quantity) {
    l.lock();
    Item it = get(item);
    it.q += quantity;
    it.isEmpty.signalAll();
    l.unlock();
}
```





- Cliente Greedy
 - Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {
    l.lock();
    for (String item : items) {
        Item it = this.get(item);
        while (it.q == 0) {
            it.isEmpty.await();
        }
        it.q -= 1;
    }
    l.unlock();
}
```





- Cliente Greedy
 - Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {
    l.lock();
    for (String item : items) {
        Item it = this.get(item);
        while (it.q == 0) {
            it.isEmpty.await();
        }
        it.q -= 1;
    }
    l.unlock();
}
```







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
    l.lock();
    consume_('item2');
    consume_('item3');
    consume_('item4');
    l.unlock();
}
```

```
T2

Cliente 2 {
    l.lock();
    consume_('item2');
    l.unlock();
}
```

```
T3

Cliente 3 {
    l.lock();
    supply_('item3',5);
    l.unlock();
}

Cliente 3 {
    l.lock();
    supply_('item2',5);
    l.unlock();
}
```

```
Lock: em espera p/ lock: Wait-Set:
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
  l.lock();
  consume_('item2');
  consume_('item3');
  consume_('item4');
  l.unlock();
```

```
Cliente 2 {
  l.lock();
  consume_('item2');
  l.unlock();
```

 T_2

```
T_3
Cliente 3 {
  l.lock();
   supply_('item3',5);
  l.unlock();
Cliente 3 {
  l.lock();
   supply_('item2',5);
   l.unlock();
```

Lock: TI em espera p/ lock: T2,T3 Wait-Set:

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
```

Cliente 2: consume (['item2'])

Cliente 3: supply ('item3', 5)

Cliente 3: supply ('item2', 5)

```
\mathsf{T}_{\mathsf{I}}
                            while(item.q == 0)
                               item.isEmpty.await();
Cliente 1 {
                            item.q -= 1;
   l.lock();
   consume_('item2');
   consume_('item3');
                                     l.unlock();
   consume_('item4');
   l.unlock();
```

T_3

```
Cliente 3 {
  l.lock();
   supply_('item3',5);
   l.unlock();
Cliente 3 {
  l.lock();
   supply_('item2',5);
   l.unlock();
```

Lock: TI em espera p/ lock: T2,T3

Wait-Set:

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
```

Cliente 3: supply ('item3', 5)

Cliente 3: supply ('item2', 5)

```
T_
Cliente 1 {
    l.lock();
    consume_('item2');
    consume_('item3');
    consume_('item4');
    l.unlock();
...
while(item.q == 0)
    item.isEmpty.await();
    item.q -= 1;
...
}
...
}
```

T₃

```
Cliente 3 {
    l.lock();
    supply_('item3',5);
    l.unlock();
}

Cliente 3 {
    l.lock();
    supply_('item2',5);
    l.unlock();
}
```

```
Lock: T1 em espera p/ lock: T2,T3
```

Wait-Set:

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
```

Cliente 3: supply ('item3', 5)

Cliente 3: supply ('item2', 5)

```
while(item.q == 0)
Cliente 1 {
                           item.isEmpty.await();
   l.lock();
                         item.q -= 1;
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

T_3

```
Cliente 3 {
  l.lock();
   supply_('item3',5);
  l.unlock();
Cliente 3 {
  l.lock();
   supply_('item2',5);
   l.unlock();
```

Lock: em espera p/ lock: T2,T3

Wait-Set: Tl(item3)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
    l.lock();
    consume_('item2');
    consume_('item3');
    consume_('item4');
    l.unlock();
}
```

```
T2

Cliente 2 {
    l.lock();
    consume_('item2');
    l.unlock();
}
```

```
while(item.q == 0)
   item.isEmpty.await();
item.q == 1;
...

l.unlock();
}

Cliente 3 {
   l.lock();
   supply_('item2',5);
   l.unlock();
}
```

```
Lock: T2 em espera p/ lock: T3
Wait-Set: T1(item3)
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_2
Cliente 2 {
   l.lock();
   consume_('item2');
   l.unlock();
```

```
while(item.q == 0)
  item.isEmpty.await();
item.q -= 1;
                          5);
         l.unlock();
     Cliente 3 {
        l.lock();
         supply_('item2',5);
        l.unlock();
```

Lock:	em espera p∕ lock: ⊤3
Wait-Set: TI (it	em3),T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                       T_3
Cliente 1 {
                              Cliente 2 {
                                                            Cliente 3 {
                                                               l.lock();
                                 l.lock();
   l.lock();
   consume_('item2');
                                 consume_('item2');
                                                                supply_('item3',5);
                                 l.unlock();
   consume_('item3');
                                                                l.unlock();
   consume_('item4');
   l.unlock();
                        Embora o Cliente 1 esteja à espera do
                                                               iente 3 {
                                                               l.lock();
                       'item3', este já consumiu o 'item2'. Desta
                                                                supply_('item2',5);
                       forma, o Cliente 2 ficará bloqueado até
                                                                l.unlock();
                           ao reabastecimento do 'item2'.
```

Lock: em espera p/ lock: T3

Mai4 Ca4	T 1/	(:	Т .	(: ₄ 2)	
Wait-Set:	11(item 3), I Z((itemz))

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_2
Cliente 2 {
   l.lock();
   consume_('item2');
   l.unlock();
```

```
T_3
Cliente 3 {
  l.lock();
   supply_('item3',5);
   l.unlock();
}
Cliente 3 {
  l.lock();
   supply_('item2',5);
   l.unlock();
```

Lock: T3 em espera p/ lock: Wait-Set: T1(item3),T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_3
item.q += quantity;
                           Cliente 3 {
item.isEmpty.signalAll();
                              l.lock();
                              supply_('item3',5);
l.unlock();
                              l.unlock();
                           }
                           Cliente 3 {
                              l.lock();
                              supply_('item2',5);
                              l.unlock();
```

Lock: T3 em espera p/ lock:

Wait-Set: T1(item3),T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
item.q += quantity;
                           Cliente 3 {
item.isEmpty.signalAll();
                              l.lock();
                              supply_('item3',5);
l.unlock();
                              l.unlock();
                           }
                           Cliente 3 {
                              l.lock();
```

```
em espera p/ lock:
```

Wait-Set: T1(item3),T2(item2)

Lock: T3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5



supply_('item2',5);

l.unlock();

 T_3





```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T3

Cliente 3 {
    l.lock();
    supply_('item3',5);
    l.unlock();
}

Cliente 3 {
    l.lock();
    supply_('item2',5);
    l.unlock();
}
```

Lock: TI em espera p/ lock: T3

Wait-Set: T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	4	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_2
Cliente 2 {
   l.lock();
   consume_('item2');
   l.unlock();
```

```
T_3
Cliente 3 {
  l.lock();
   supply_('item3',5);
   l.unlock();
}
Cliente 3 {
  l.lock();
   supply_('item2',5);
   l.unlock();
```

Lock: T∣ em espera p/ lock: T3 Wait-Set: T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	4	4







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_2
                                        T_3
Cliente 2 {
                              Cliente 3 {
   l.lock();
                                l.lock();
                                 supply_('item3',5);
   consume_('item2');
                                 l.unlock();
   l.unlock();
  item.q += quantity;
                              Cliente 3 {
  item.isEmpty.signalAll();
                                l.lock();
                                 supply_('item2',5);
                                 l.unlock();
                              }
```

```
Lock: T3
            em espera p/ lock:
```

Wait-Set: T2(item2)	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	4	4







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
    l.lock();
    consume_('item2');
    consume_('item3');
    consume_('item4');
    l.unlock();
}
```

Wait-Set: T2(item2)

```
T_2
                                        T_3
Cliente 2 {
                              Cliente 3 {
                                l.lock();
   l.lock();
   consume_('item2');
                                 supply_('item3',5);
   l.unlock();
                                 l.unlock();
  item.q += quantity;
                              Cliente 3 {
  item.isEmpty.signalAll();
                                l.lock();
                                 supply_('item2',5);
                                 l.unlock();
                              }
```

```
Lock: T3 em espera p/ lock:
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	4	4







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                     while(item.q == 0)
                                                        item.isEmpty.await();
Cliente 1 {
                             Cliente 2 {
                                                      item.q -= 1;
                                l.lock();
   l.lock();
                                                                                5);
                                consume_('item2');
   consume_('item2');
   consume_('item3');
                                                               l.unlock();
                                l.unlock();
   consume_('item4');
                                                           }
   l.unlock();
                                                           Cliente 3 {
                                                              l.lock();
                                                              supply_('item2',5);
                                                              l.unlock();
```

```
Lock: T2 em espera p/ lock:
Wait-Set:
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	4	4

}





Cliente Cooperativo

}

Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {
    1.lock();
    int i = 0;
    while (i < items.length) {</pre>
     Item it = this.get(items[i]);
     i++;
     while (it.q == 0) {
       it.isEmpty.await();
      i = 0;
    for (String item : items) {
     this.get(item).q -= 1;
    l.unlock();
```





- Cliente Cooperativo
 - Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {
    1.lock();
    int i = 0;
    while (i < items.length) {</pre>
      Item it = this.get(items[i]);
      i++;
                                               preconsume_
      while (it.q == 0) {
                                               Operador que regista a
       it.isEmpty.await();
                                               intenção de consumir um item.
        = 0;
    for (String item : items)
      this.get(item).q -= 1;
                                        consume_
                                        Abstração do operador consumir,
    1.unlock();
                                        para um único item.
                                                                INESCTEC
}
```



```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                    T_3
Cliente 1 {
                             Cliente 2 {
                                                          Cliente 3 {
   l.lock();
                                l.lock();
                                                             l.lock();
                                                             supply_('item3',5);
   preconsume_('item2');
                                preconsume_('item2');
   preconsume_('item3');
                                consume_('item2');
                                                             l.unlock();
   preconsume_('item4');
                                l.unlock();
                                                          }
   consume_('item2');
                                                          Cliente 3 {
   consume_('item3');
                                                             l.lock();
   consume_('item4');
                                                              supply_('item2',5);
   l.unlock();
                                                              l.unlock();
                                                          }
```

Lock: em espera p/ lock:
Wait-Set: registo:

	String	'item1'	'item2'	'item3'	'item4'
I	tem.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                          T_3
Cliente 1 {
                        Cliente 2 {
                                                 Cliente 3 {
                                                   l.lock();
  l.lock();
                          l.lock();
                                                   supply_('item3',5);
  preconsume_('item2');
                          preconsume_('item2');
                          consume_('item2');
  preconsume_('item3');
                                                   l.unlock();
  consume_('item2');
                                                 Cliente 3 {
  consume_('item3');
                                                   l.lock();
  consume_('item4');
                                                    supply_('item2',5);
  l.unlock();
                                                    l.unlock();
```

```
Lock: TI
                em espera p/ lock: T2,T3
Wait-Set:
                 registo: TI = 0
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                             Item it = this.get('item2');
                             i++;
Cliente 1 {
                                                           Cliente 3 {
                             while (it.q == 0) {
                                                              l.lock();
   l.lock();
                              it.isEmpty.await();
                                                              supply_('item3',5);
   preconsume_('item2');
                               i = 0;
   preconsume_('item3');
                                                              l.unlock();
   preconsume_('item4');
   consume_('item2');
                                                           Cliente 3 {
   consume_('item3');
                                                              l.lock();
   consume_('item4');
                                                               supply_('item2',5);
   l.unlock();
                                                              l.unlock();
```

```
Lock: TI
                 em espera p/ lock: T2,T3
Wait-Set:
                 registo: TI = I
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                             Item it = this.get('item2');
                              <u>i</u>++;
Cliente 1 {
                                                            Cliente 3 {
                             while (it.q == 0) {
                                                               l.lock();
   l.lock();
                               it.isEmpty.await();
                                                                supply_('item3',5);
   preconsume_('item2');
                               i = 0;
   preconsume_('item3');
                                                                l.unlock();
   preconsume_('item4');
   consume_('item2');
                                                             Cliente 3 {
                                                               1.lock();
   consume_('item3');
   consume_('item4');
                                                                supply_('item2',5);
   l.unlock();
                                                                l.unlock();
```

```
Lock: TI em espera p/lock: T2,T3

Wait-Set: registo: TI = I
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                             Item it = this.get('item3');
Cliente 1 {
                                                           Cliente 3 {
                             i++;
                                                              l.lock();
   l.lock();
                             while (it.q == 0) {
   preconsume_('item2');
                                                              supply_('item3',5);
                              it.isEmpty.await();
   preconsume_('item3');
                                                              l.unlock();
                               i = 0;
   preconsume_('item4');
   consume_('item2');
                                                           Cliente 3 {
   consume_('item3');
                                                              l.lock();
                                                               supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                              l.unlock();
```

```
Lock: TI em espera p/lock: T2,T3

Wait-Set: registo: TI = 2
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                             Item it = this.get('item3');
Cliente 1 {
                                                            Cliente 3 {
                              <u>i</u>++;
                                                               l.lock();
   l.lock();
                             while (it.q == 0) {
   preconsume_('item2');
                                                                supply_('item3',5);
                               it.isEmpty.await();
   preconsume_('item3');
                                                                l.unlock();
                               i = 0:
   preconsume_('item4');
   consume_('item2');
                                                            Cliente 3 {
   consume_('item3');
                                                               l.lock();
                                                                supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                                l.unlock();
```

Lock: T1 em espera p/ lock: T2,T3

Wait-Set: registo: T1 = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                             Item it = this.get('item3');
Cliente 1 {
                                                            Cliente 3 {
                              <u>i</u>++;
   l.lock();
                                                               l.lock();
                             while (it.q == 0) {
   preconsume_('item2');
                                                                supply_('item3',5);
                               it.isEmpty.await();
   preconsume_('item3');
                                                                l.unlock();
                               i = 0;
   preconsume_('item4');
   consume_('item2');
                                                             Cliente 3 {
   consume_('item3');
                                                               l.lock();
                                                                supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                                l.unlock();
```

Lock: em espera p/ lock: T2,T3

Wait-Set: Tl(item3) registo: Tl = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                      Item it = this.get('item2');
Cliente 1 {
                             Cliente 2 {
                                                      i++;
                                                      while (it.q == 0) {
   l.lock();
                                l.lock();
                                                        it.isEmpty.await();
   preconsume_('item2');
                                preconsume_('item2'
                                                        i = 0:
   preconsume_('item3');
                                consume_('item2');
   preconsume_('item4');
                                l.unlock();
   consume_('item2');
   consume_('item3');
                                                              l.lock();
   consume_('item4');
                                                              supply_('item2',5);
   l.unlock();
                                                              l.unlock();
```

```
Lock: T2
                 em espera p/ lock: T3
Wait-Set: TI(item3) registo: TI = 2; T2 = I
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                       Item it = this.get('item2');
Cliente 1 {
                             Cliente 2 {
                                                        <u>i++;</u>
                                                       while (it.q == 0) {
   l.lock();
                                 l.lock();
                                                         it.isEmpty.await();
   preconsume_('item2');
                                 preconsume_('item2'
                                                          i = 0:
   preconsume_('item3');
                                 consume_('item2');
   preconsume_('item4');
                                 l.unlock();
   consume_('item2');
   consume_('item3');
                                                               l.lock();
   consume_('item4');
                                                                supply_('item2',5);
   l.unlock();
                                                               l.unlock();
```

```
Lock: T2
           em espera p/ lock: T3
```

Wait-Set: TI(item3) registo: TI = 2;T2 = I

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                     T_3
Cliente 1 {
                             Cliente 2 {
                                                       this.get('item2').q -= 1;
   l.lock();
                                l.lock();
   preconsume_('item2');
                                preconsume_('fitem2')
   preconsume_('item3');
                                consume_('item2');
                                                              l.unlock();
   preconsume_('item4');
                                l.unlock();
   consume_('item2');
                                                           Cliente 3 {
                                                             1.lock();
   consume_('item3');
                                                              supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                              l.unlock();
```

Lock: em espera p/ lock: T3

Wait-Set: TI(item3) registo: TI = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
Cliente 1 {
   l.lock();
   preconsume_('item2');
   preconsume_('item3');
   preconsume_('item4');
   consume_('item2');
   consume_('item3');
   consume_('item4');
   l.unlock();
```

```
T_3
item.q += quantity;
                           Cliente 3 {
item.isEmpty.signalAll();
                              l.lock();
                              supply_('item3',5);
consume_('item2');
                              l.unlock();
l.unlock();
                           }
                           Cliente 3 {
                              1.lock();
                              supply_('item2',5);
                              l.unlock();
```

```
Lock: T3
                em espera p/ lock: Tl
Wait-Set:
                    registo: TI = 2
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                     T_3
Cliente 1 {
                             Cliente 2 {
                                                           Cliente 3 {
                                                              l.lock();
   l.lock();
                                                              supply_('item3',5);
   preconsume_('item2');
                               item.q += quantity;
   preconsume_('item3');
                                                              Truntlock();
                               item.isEmpty.signalAll();
   preconsume_('item4');
   consume_('item2');
                                                           Cliente 3 {
   consume_('item3');
                                                              l.lock();
   consume_('item4');
                                                              supply_('item2',5);
   l.unlock();
                                                              l.unlock();
                                                           }
```

```
em espera p/ lock:
    Lock: T3
Wait-Set: TI(item3) registo: TI = 2
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_3
                              Item it = this.get('item3');
Cliente 1 {
                                                             Cliente 3 {
                              <u>i</u>++;
   l.lock();
                                                               l.lock();
                              while (it.q == 0) {
   preconsume_('item2');
                                                                supply_('item3',5);
                               it.isEmpty.await();
   preconsume_('item3');
                                                                l.unlock();
                                i = 0;
   preconsume_('item4');
                                                             Cliente 3 {
   consume_('item2');
   consume_('item3');
                                                                l.lock();
                                                                supply_('item2',5);
   consume_('item4');
                                                                l.unlock();
   l.unlock();
                                                             }
```

```
Lock: TI em espera p/lock:

Wait-Set: registo: TI = 0
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                    T_3
Cliente 1 {
                            Cliente 2 {
                                                          Cliente 3 {
   l.lock();
                                l.lock();
                                                            l.lock();
                                                             supply_('item3',5);
   preconsume_('item2');
                                preconsume_('item2');
                                consume_('item2');
   preconsume_('item3');
                                                             l.unlock();
   preconsume_('item4');
                                l.unlock();
   consume_('item2');
                                                          Cliente 3 {
   consume_('item3');
                                                             l.lock();
   consume_('item4');
                                                             supply_('item2',5);
   l.unlock();
                                                             l.unlock();
                                                          }
```

```
Lock: TI
                em espera p/ lock:
Wait-Set:
                 registo: TI = I
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                    T_3
Cliente 1 {
                            Cliente 2 {
                                                          Cliente 3 {
   l.lock();
                               l.lock();
                                                            l.lock();
                                                             supply_('item3',5);
   preconsume_('item2');
                               preconsume_('item2');
   preconsume_('item3');
                                consume_('item2');
                                                             l.unlock();
   preconsume_('item4');
                                l.unlock();
   consume_('item2');
                                                          Cliente 3 {
   consume_('item3');
                                                             l.lock();
   consume_('item4');
                                                             supply_('item2',5);
   l.unlock();
                                                             l.unlock();
                                                          }
```

```
Lock: Tl em espera p/lock:

Wait-Set: registo: Tl = 2
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                    T_3
Cliente 1 {
                            Cliente 2 {
                                                          Cliente 3 {
   l.lock();
                                l.lock();
                                                             l.lock();
                                                             supply_('item3',5);
   preconsume_('item2');
                                preconsume_('item2');
   preconsume_('item3');
                                consume_('item2');
                                                             l.unlock();
   preconsume_('item4');
                                l.unlock();
   consume_('item2');
                                                          Cliente 3 {
   consume_('item3');
                                                             l.lock();
                                                             supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                             l.unlock();
                                                          }
```

```
Lock: TI
                em espera p/ lock:
Wait-Set:
                 registo: TI = 3
```

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                        T_3
Cliente 1 {
                              Cliente 2 {
                                                              Cliente 3 {
                                  l.lock():
   l.lock();
                                                                 l.lock();
   preconsume_('item2')
                           this.get('item2').q -= 1;
   preconsume_('it<del>em3</del>→)
                                                                 l.unlock();
   preconsume_('/item4')
                                                              }
   consume_('item2');
                                                              Cliente 3 {
   consume_('item3');
                                                                 l.lock();
   consume_('item4');
                                                                 l.unlock();
   l.unlock();
                                                              }
```

```
supply_('item3',5);
supply_('item2',5);
```

Lock: T∣	em espera p/ lock:
Wait-Set:	registo: TI = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	5	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T_2
                                                                     T_3
Cliente 1 {
                             Cliente 2 {
                                                           Cliente 3 {
   l.lock();
                                l.lock();
                                                              l.lock();
                                preconsume_('item2');
                                                              supply_('item3',5);
   preconsume_('item2');
   preconsume_('item3')[
                                                              l.unlock();
   preconsume_('item4+)
                                                           }
                          this.get('item3').q -= 1;
   consume_('item2');
                                                           Cliente 3 {
   consume_('item3');
                                                              l.lock();
                                                              supply_('item2',5);
   consume_('item4');
   l.unlock();
                                                              l.unlock();
                                                           }
```

Lock: TI em espera p/ lock:

registo: TI = 3

Wait-Set:

	String	'item1'	'item2'	'item3'	'item4'
:	Item.quantity	10	4	4	5







```
Cliente 1: consume (['item2', 'item3', 'item4'])
Cliente 2: consume (['item2'])
Cliente 3: supply ('item3', 5)
Cliente 3: supply ('item2', 5)
```

```
T3

Cliente 3 {
    l.lock();
    supply_('item3',5);
    l.unlock();
}

Cliente 3 {
    l.lock();
    supply_('item2',5);
    l.unlock();
}
```

Lock:	em espera p/ lock:
Wait-Set:	registo: TI = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	4	4





Sugestões

- Tornar o stock de cada produto limitado
- Suportar o acesso concorrente ao armazém
- Adicionar os métodos de registar e remover produtos do armazém



