Parallel Programming
Exercise Session 4

Spring 2024

Schedule

Post-Discussion Ex. 3	25’
Pipelining Recap	15’
Break	
Pre-Discussion Ex. 4	10’
Quiz	10’

Post-Discussion Exercise 3

Counter

Let’s count number of times a given event occurs

```
public interface Counter {
  public void increment();
  public int value();
}
```

```
// background threads
for (int i = 0; i < numIterations; i++) {
  // perform some work

  counter.increment();
}

// progress thread
while (isWorking) {
  System.out.println(counter.value());
}
```

10 iterations each

Counter

0

Thread 1

0

Thread 2

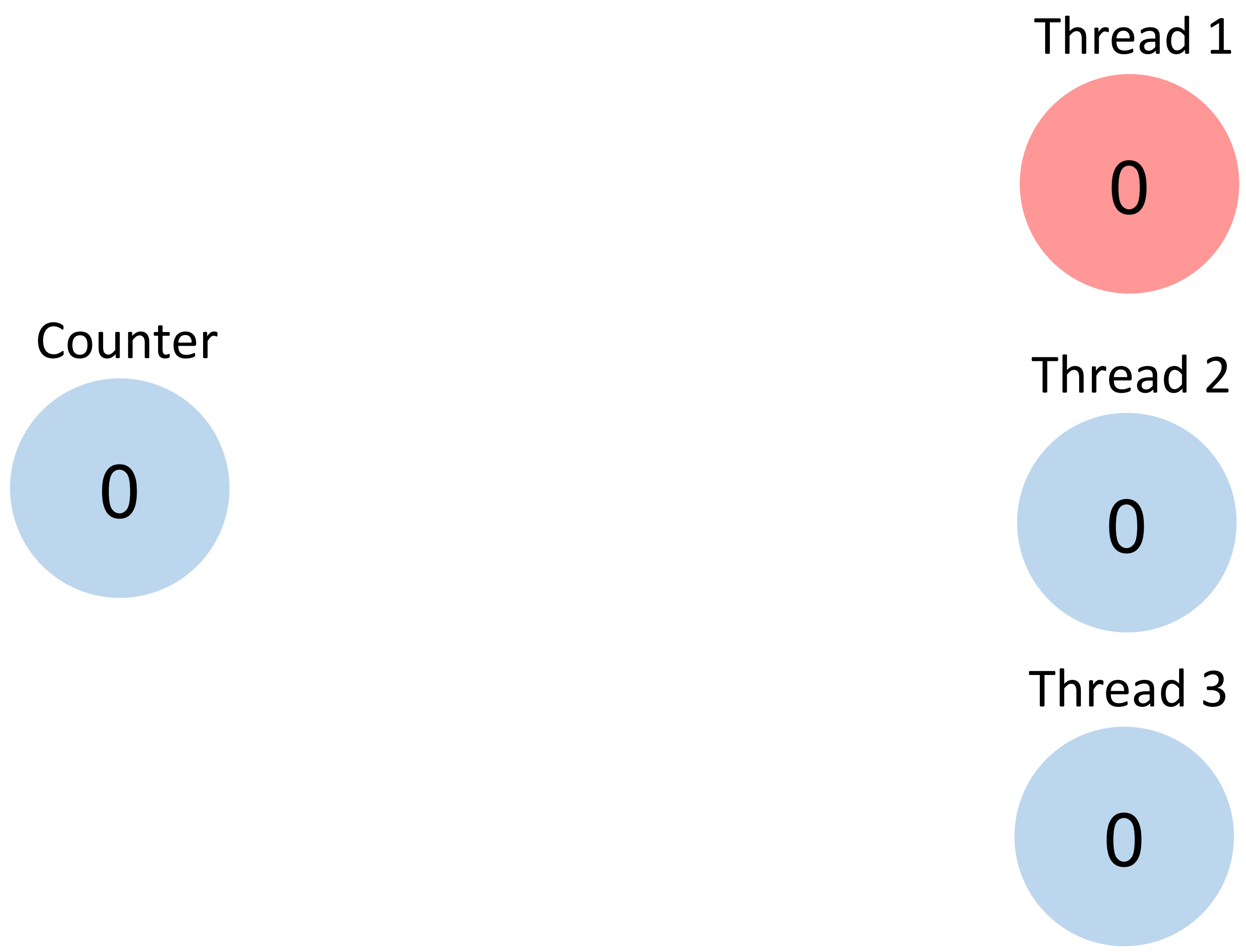
0

Thread 3

0

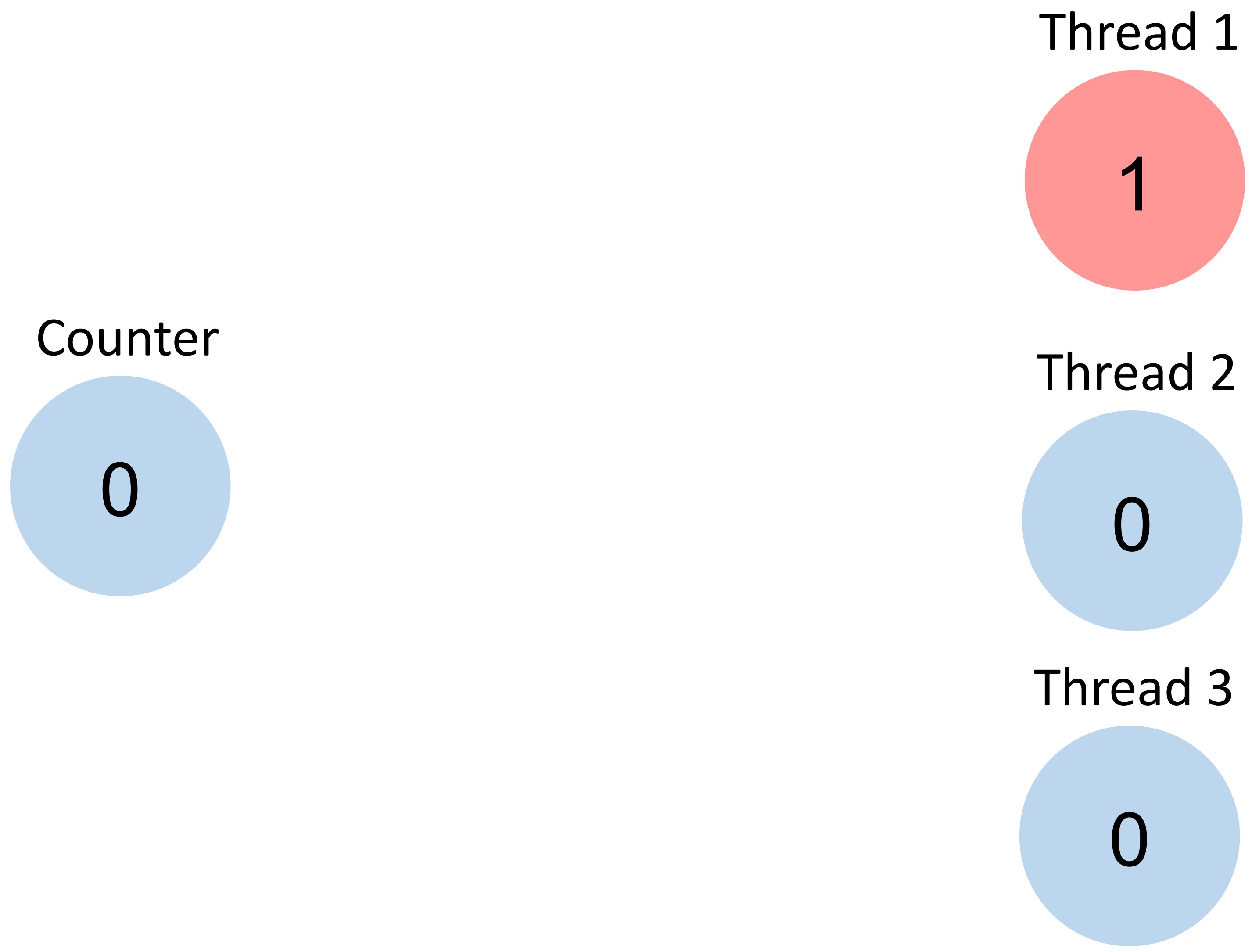
value of the
shared Counter

number of times
`increment()` is called



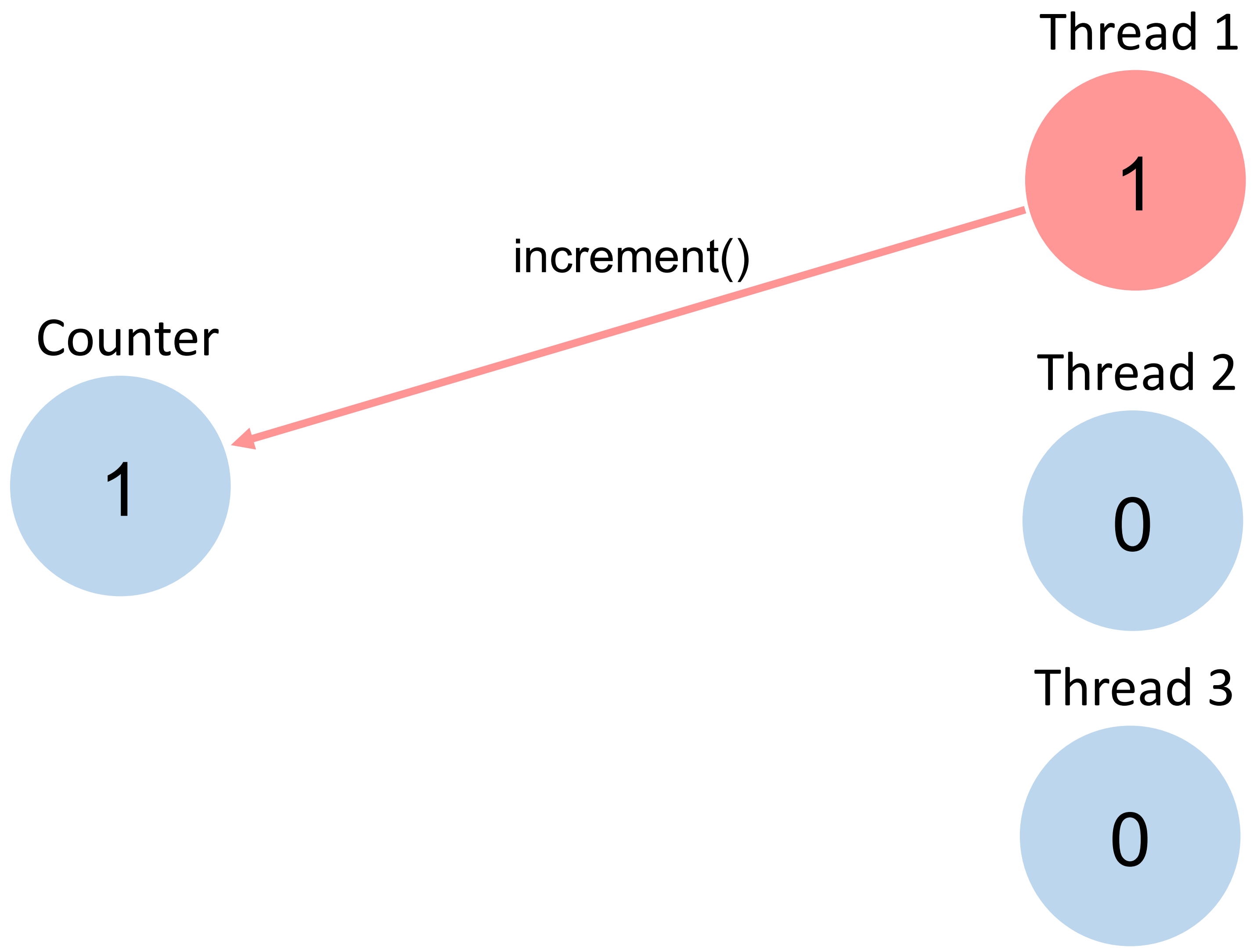
value of the
shared Counter

number of times
`increment()` is called



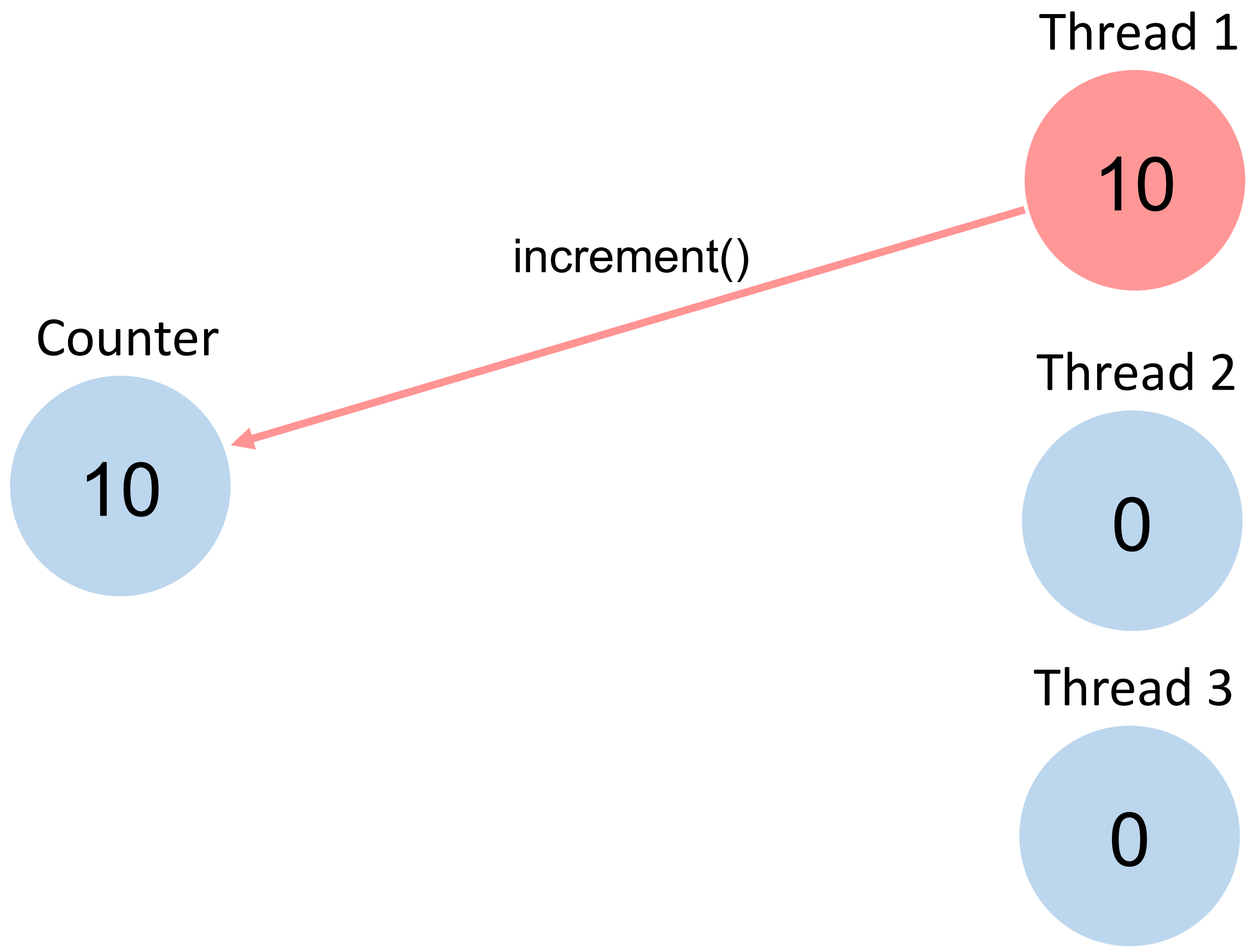
value of the shared Counter

number of times `increment()` is called



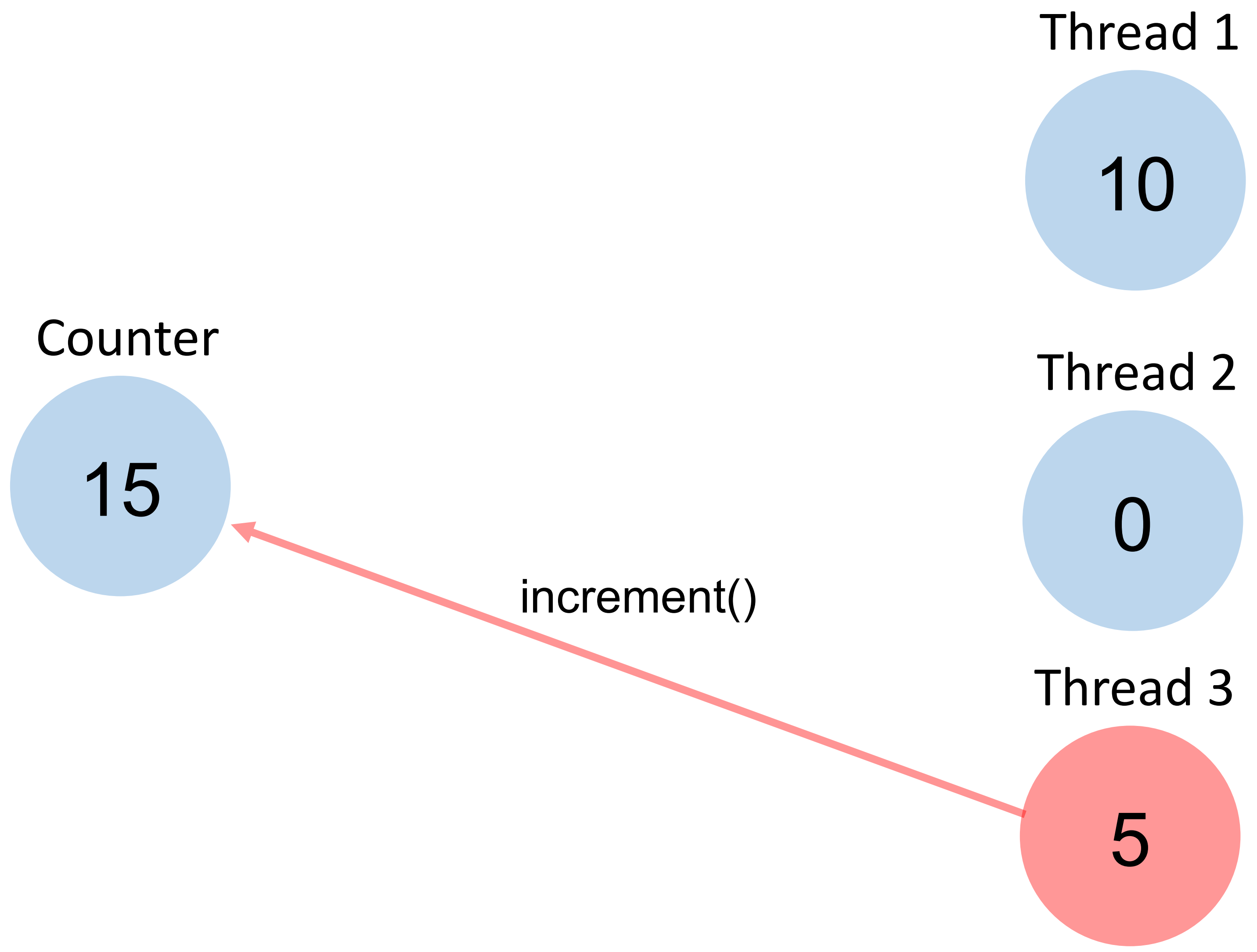
value of the shared Counter

number of times `increment()` is called



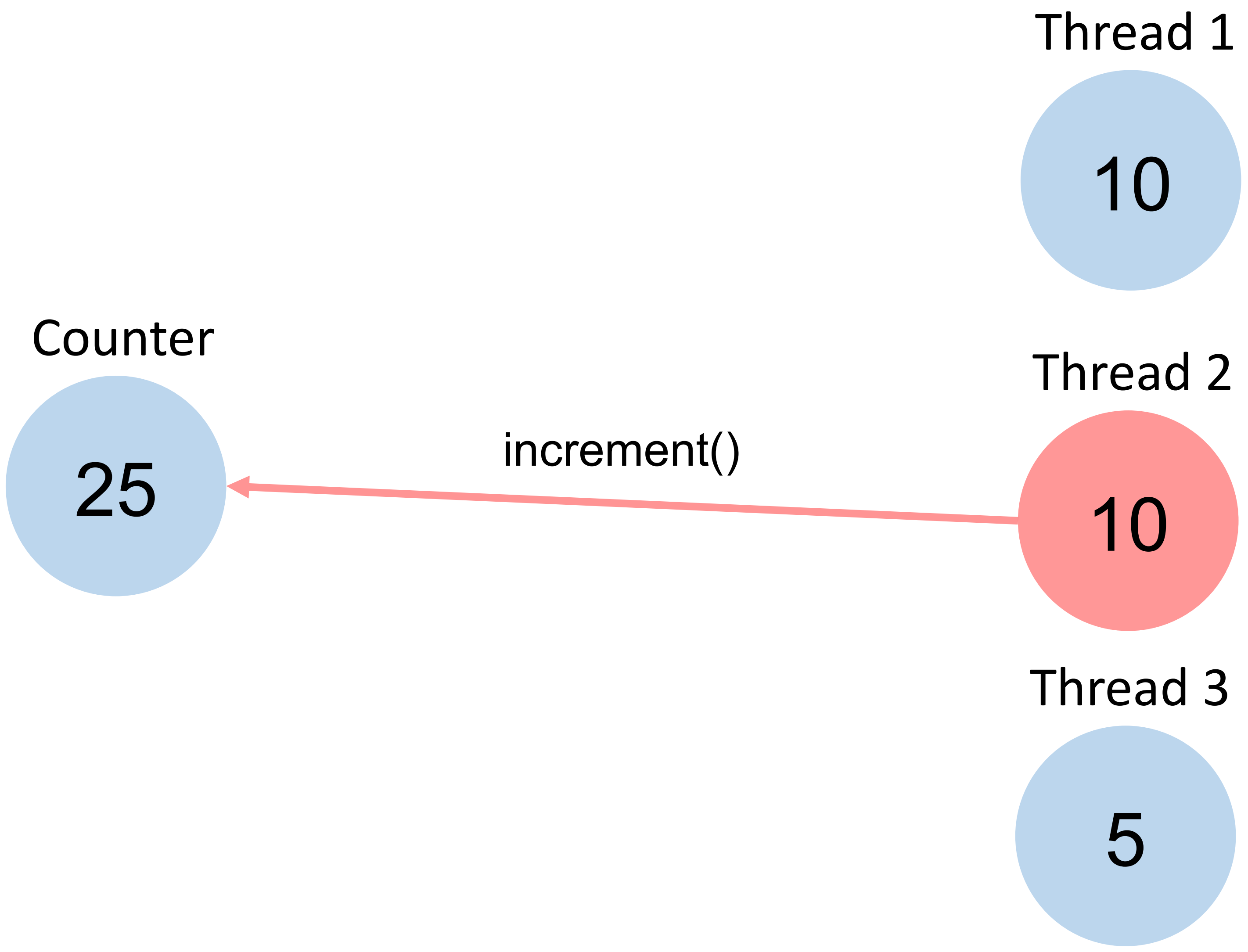
value of the
shared Counter

number of times
`increment()` is called



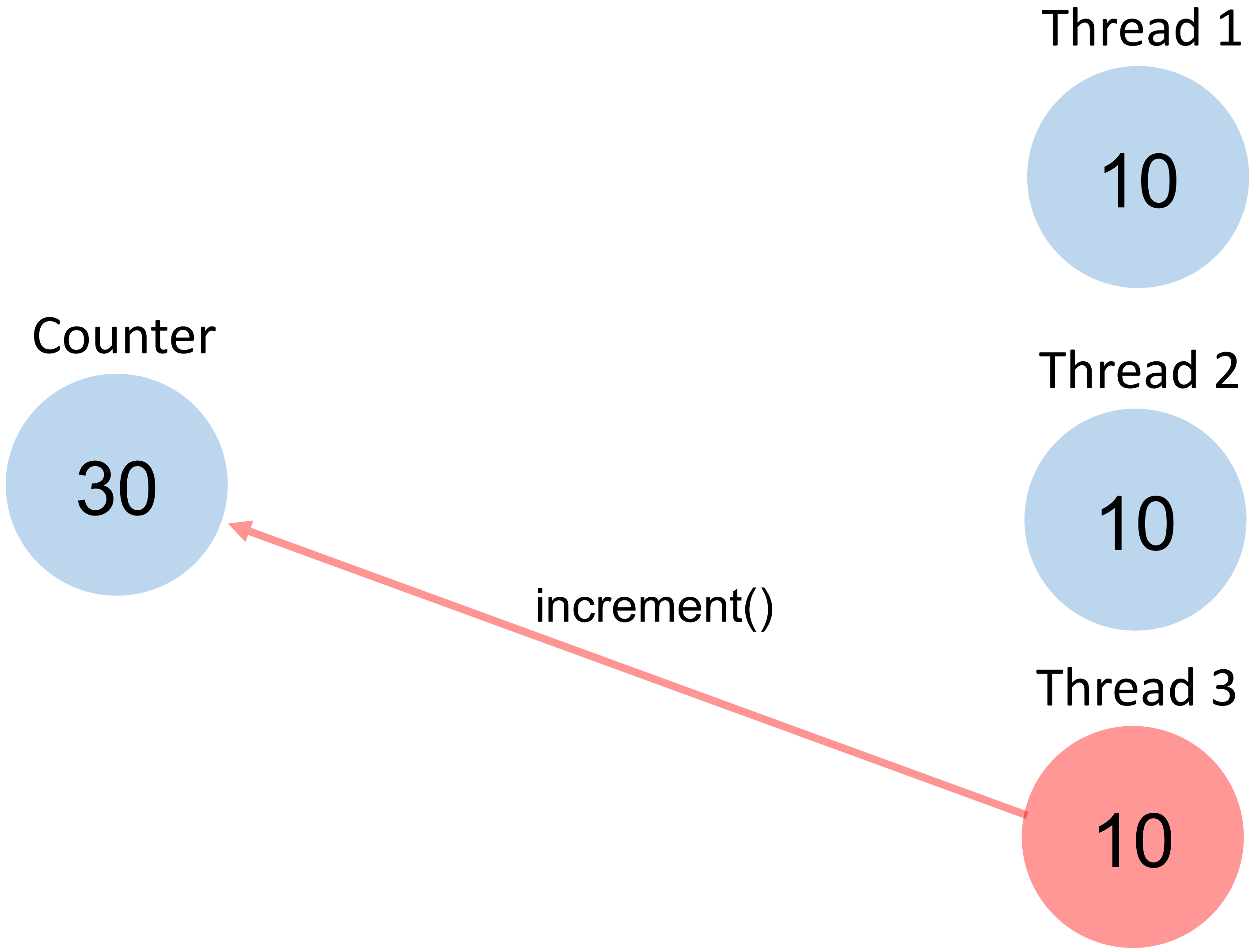
value of the shared Counter

number of times `increment()` is called



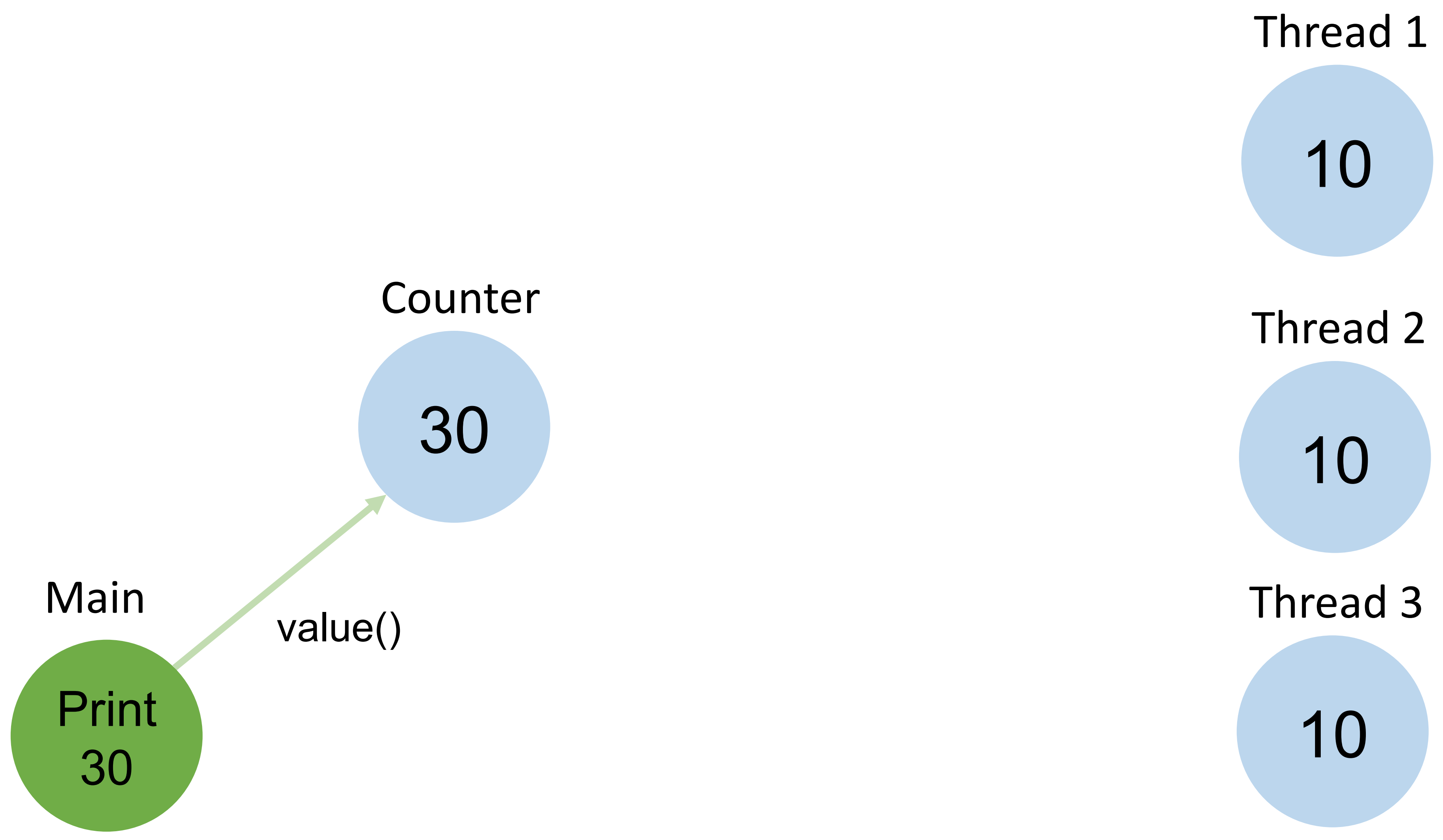
value of the
shared Counter

number of times
`increment()` is called



value of the shared Counter

number of times `increment()` is called



read the Counter value

value of the shared Counter

number of times `increment()` is called

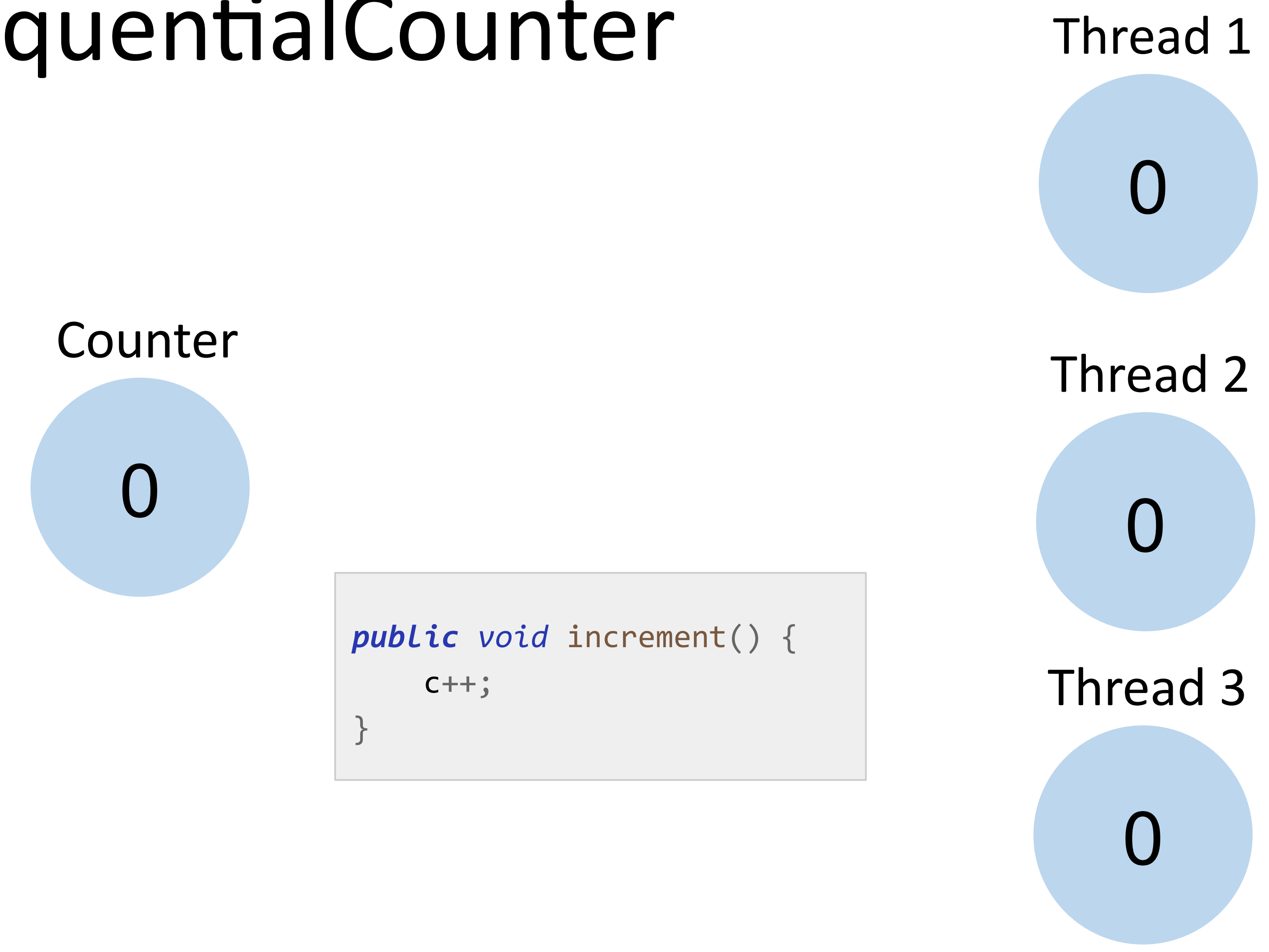
Task A: SequentialCounter

```
public class SequentialCounter implements Counter {


    public void increment() {
        ??
    }

    public int value() {
        ??
    }
}
```

Task A: SequentialCounter

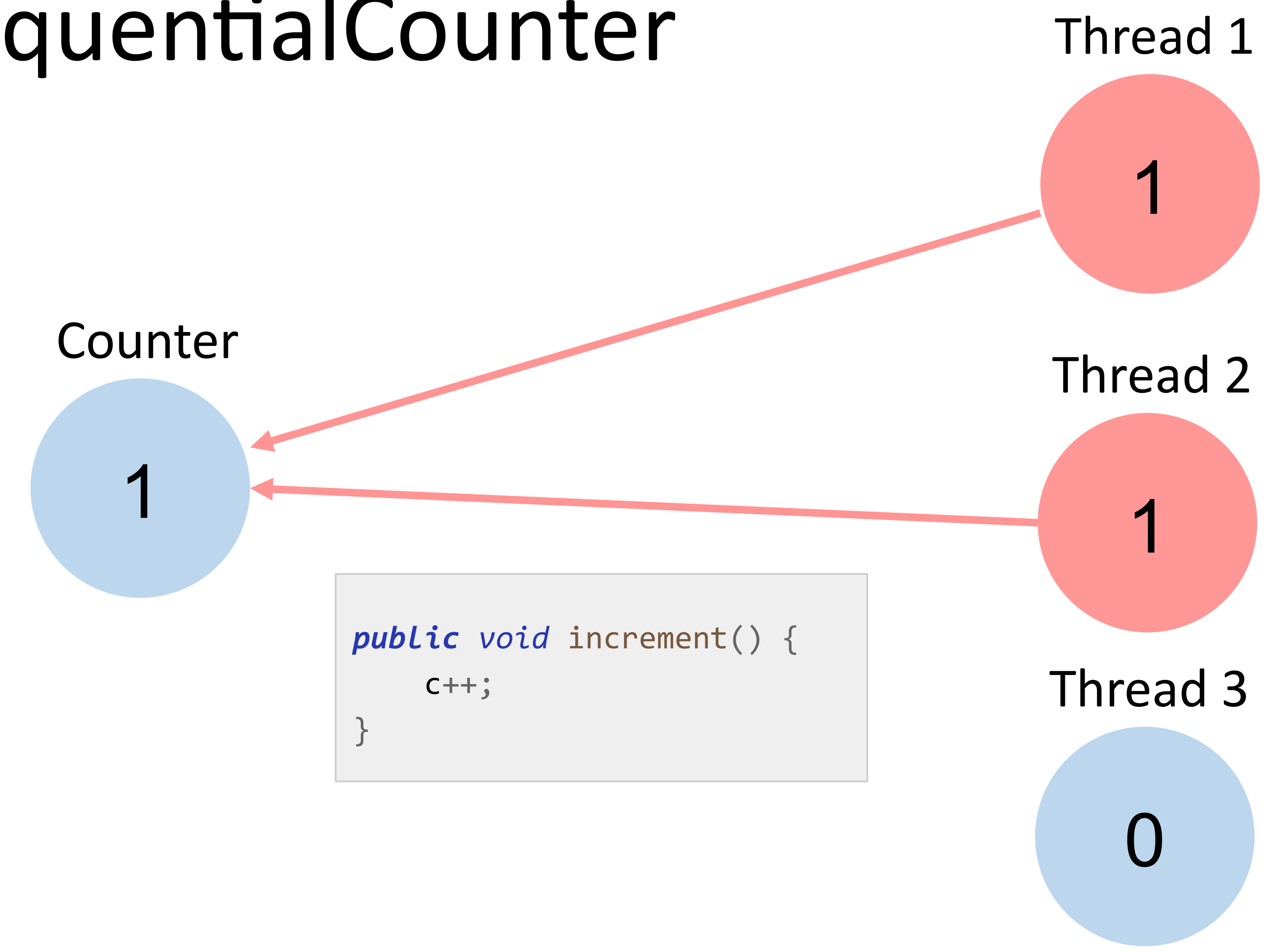
```java
public class SequentialCounter implements Counter {
    private int c = 0;

    public void increment() {
        c++;
    }

    public int value() {
        return c;
    }
}
```

Task A: SequentialCounter



```
public void increment() {
    c++;
}
```

Task A: SequentialCounter

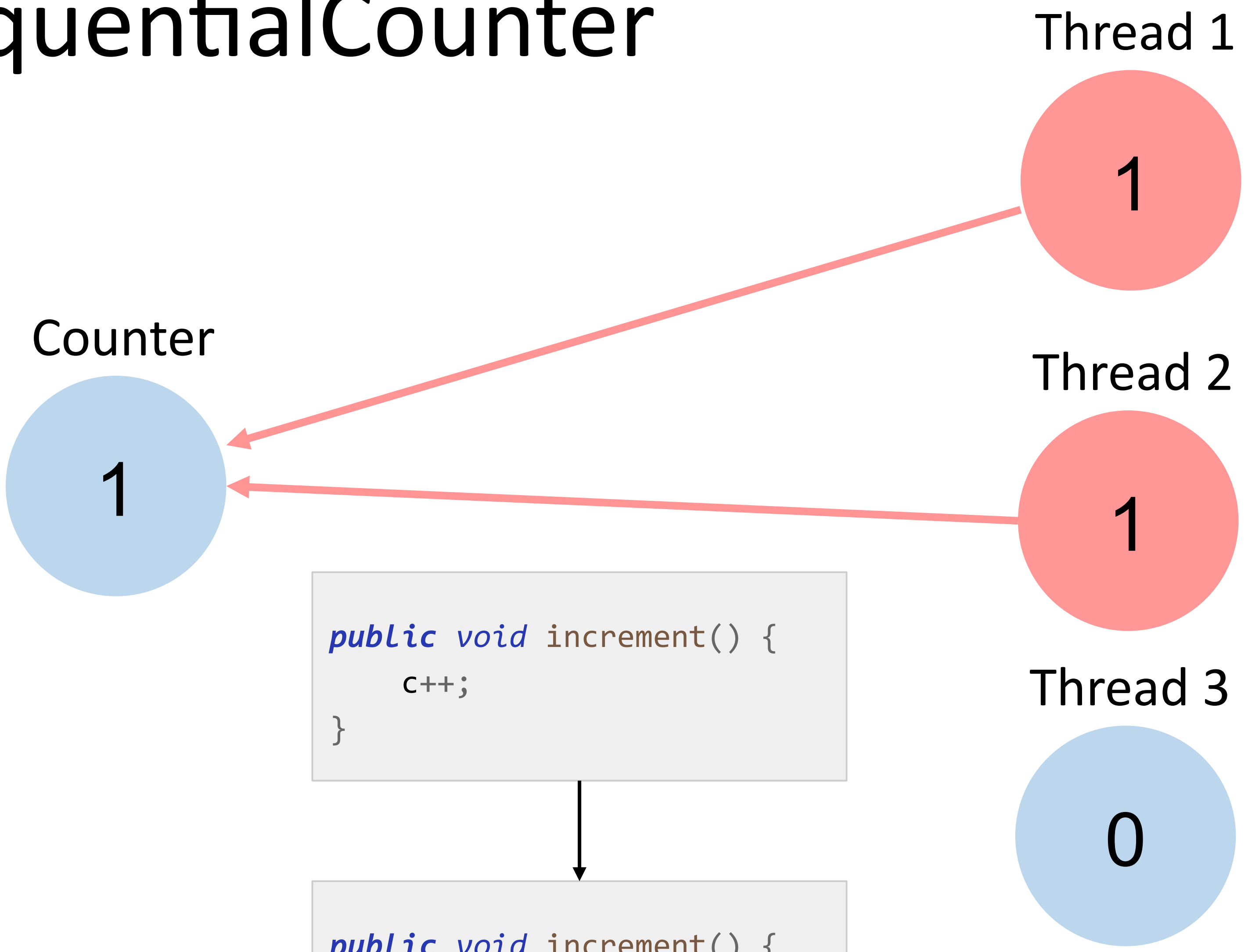


conflicting access!

How is this possible?

```java
public void increment() {
    c++;
}
```

Task A: SequentialCounter



conflicting access!

How is this possible?

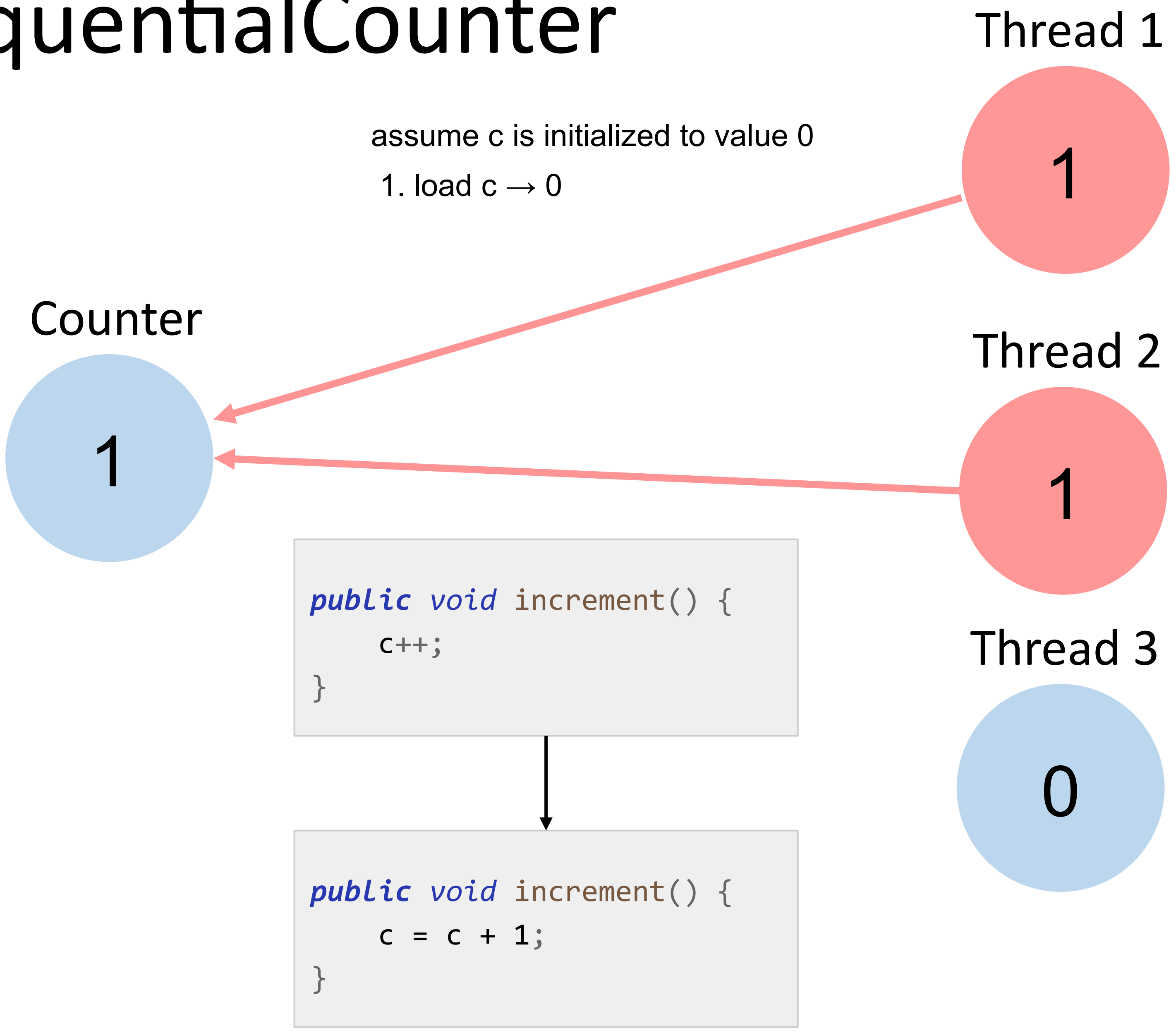
```
public void increment() {
    c++;
}
```

```
public void increment() {
    c = c + 1;
}
```

Task A: SequentialCounter

conflicting access!

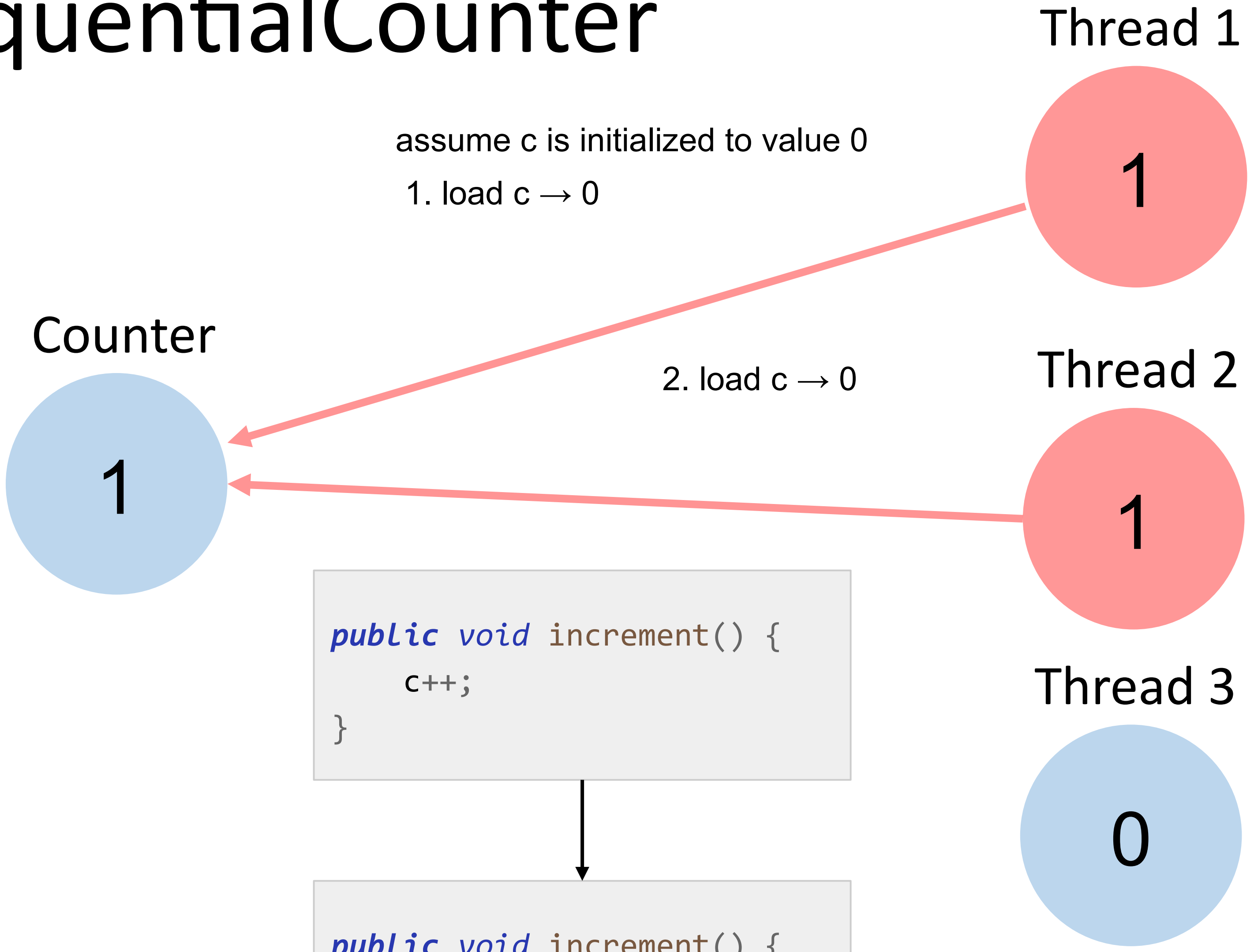
How is this possible?



```
public void increment() {
    c++;
}
```

```
public void increment() {
    c = c + 1;
}
```

Task A: SequentialCounter



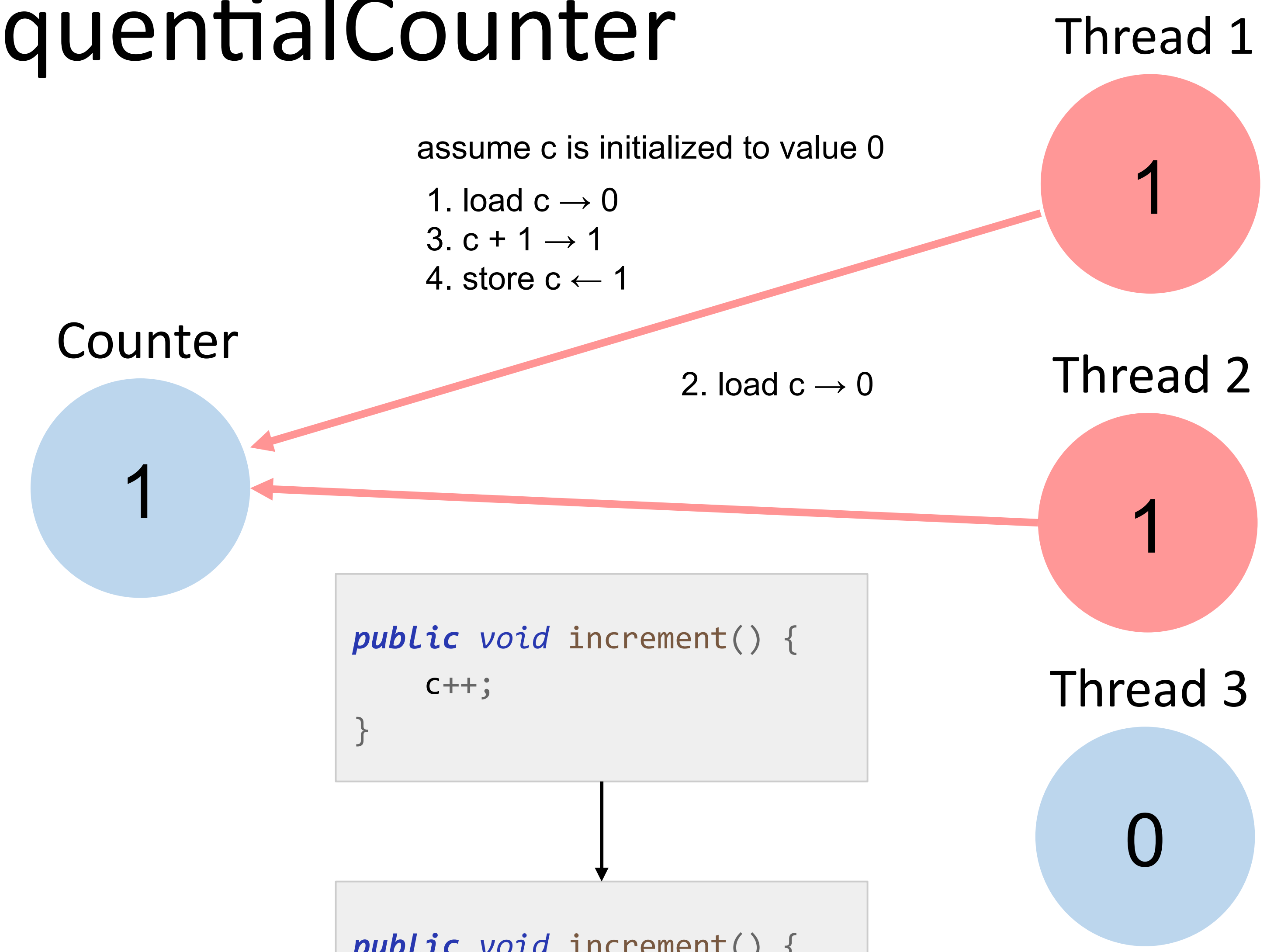
conflicting access!

How is this possible?

```java
public void increment() {
    c++;
}
```

```java
public void increment() {
    c = c + 1;
}
```

Task A: SequentialCounter



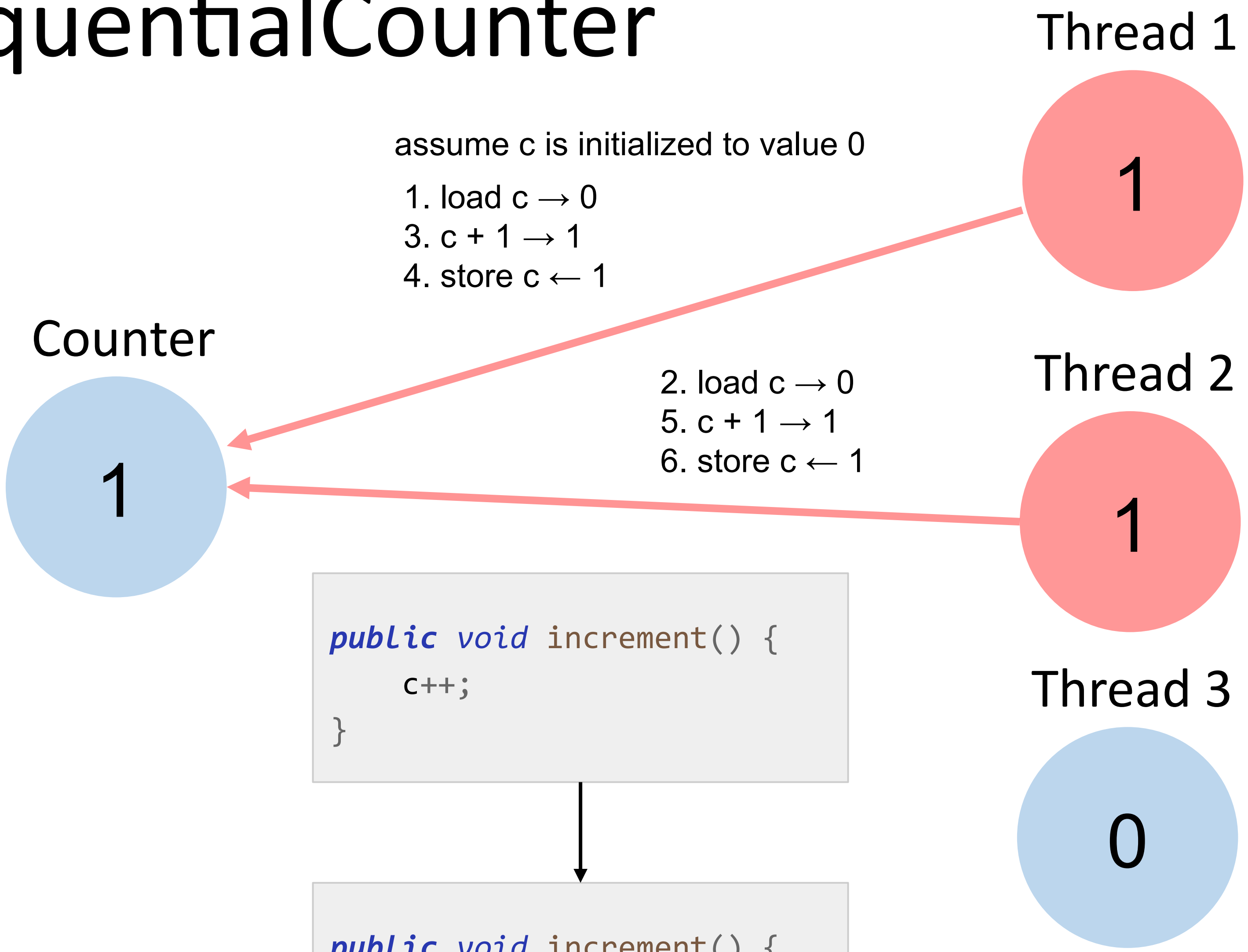
conflicting access!

How is this possible?

```
public void increment() {
    c++;
}
```

```
public void increment() {
    c = c + 1;
}
```

Task A: SequentialCounter



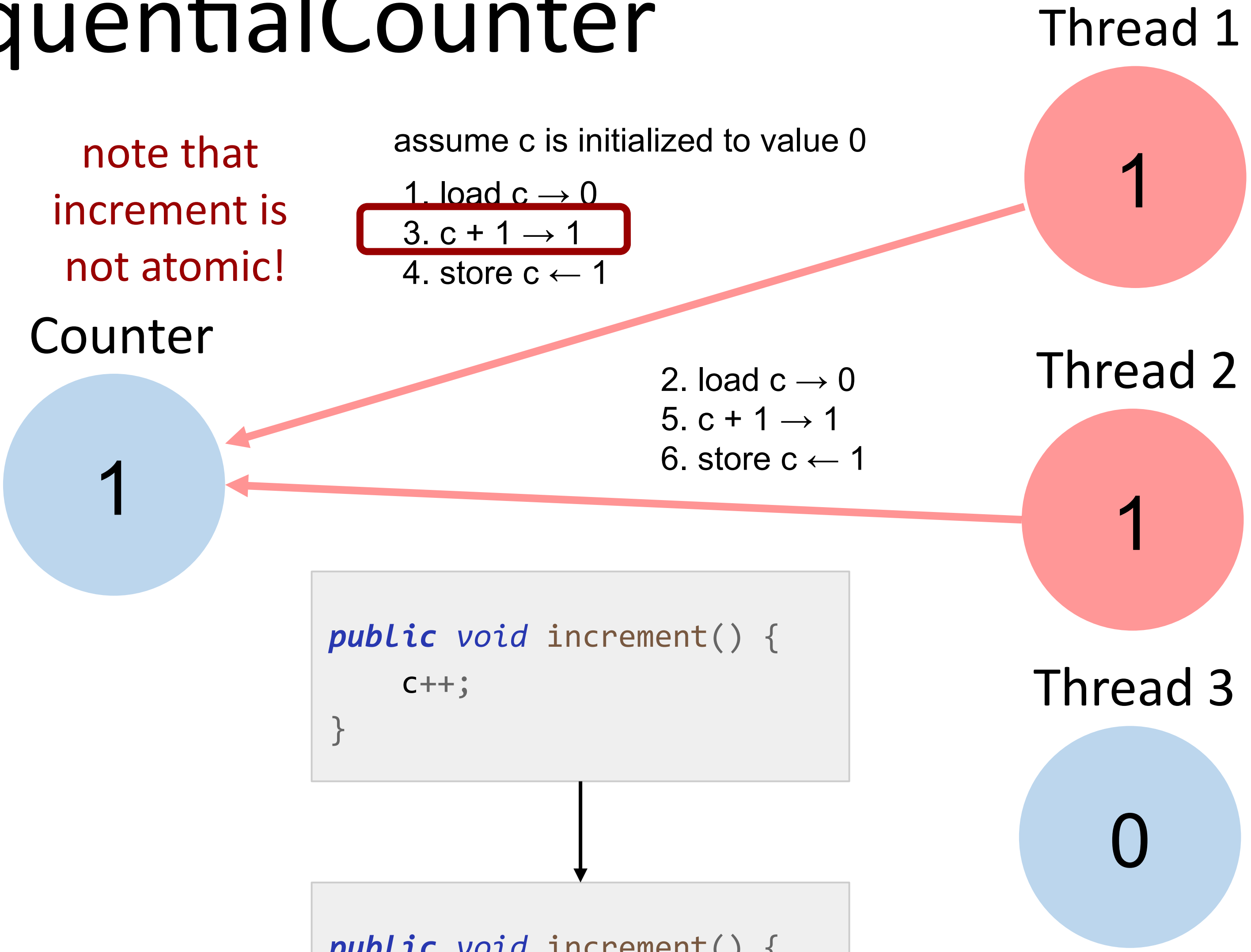
conflicting access!

How is this possible?

```
public void increment() {
    c++;
}
```

```
public void increment() {
    c = c + 1;
}
```

Task A: SequentialCounter



conflicting access!

How is this possible?

```
public void increment() {
    c++;
}
```

```
public void increment() {
    c = c + 1;
}
```

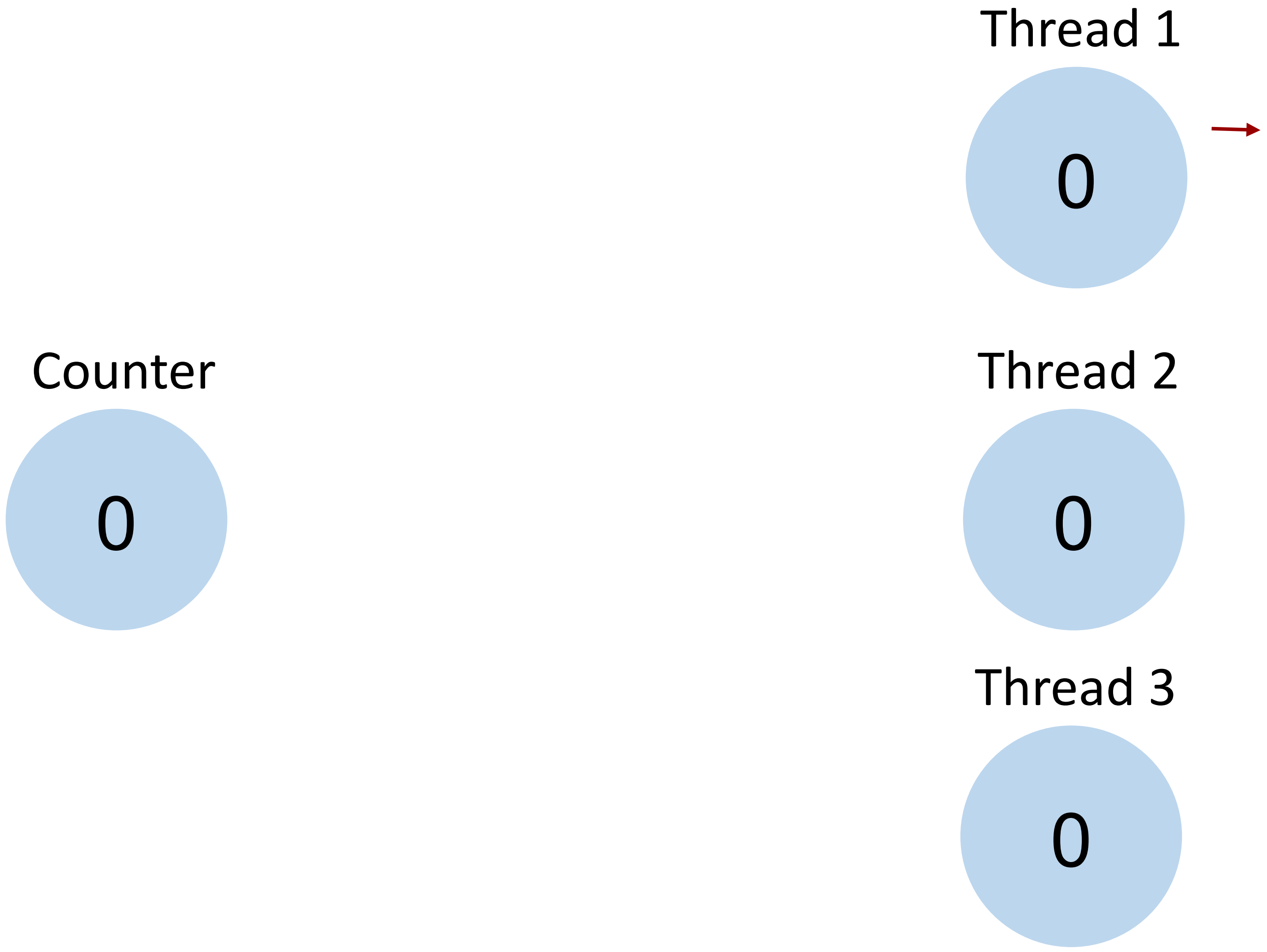
Task B: SynchronizedCounter

```
public class SynchronizedCounter implements Counter {


    public void increment() {
        ??
    }

    public int value() {
        ??
    }
}
```

Task B: SynchronizedCounter

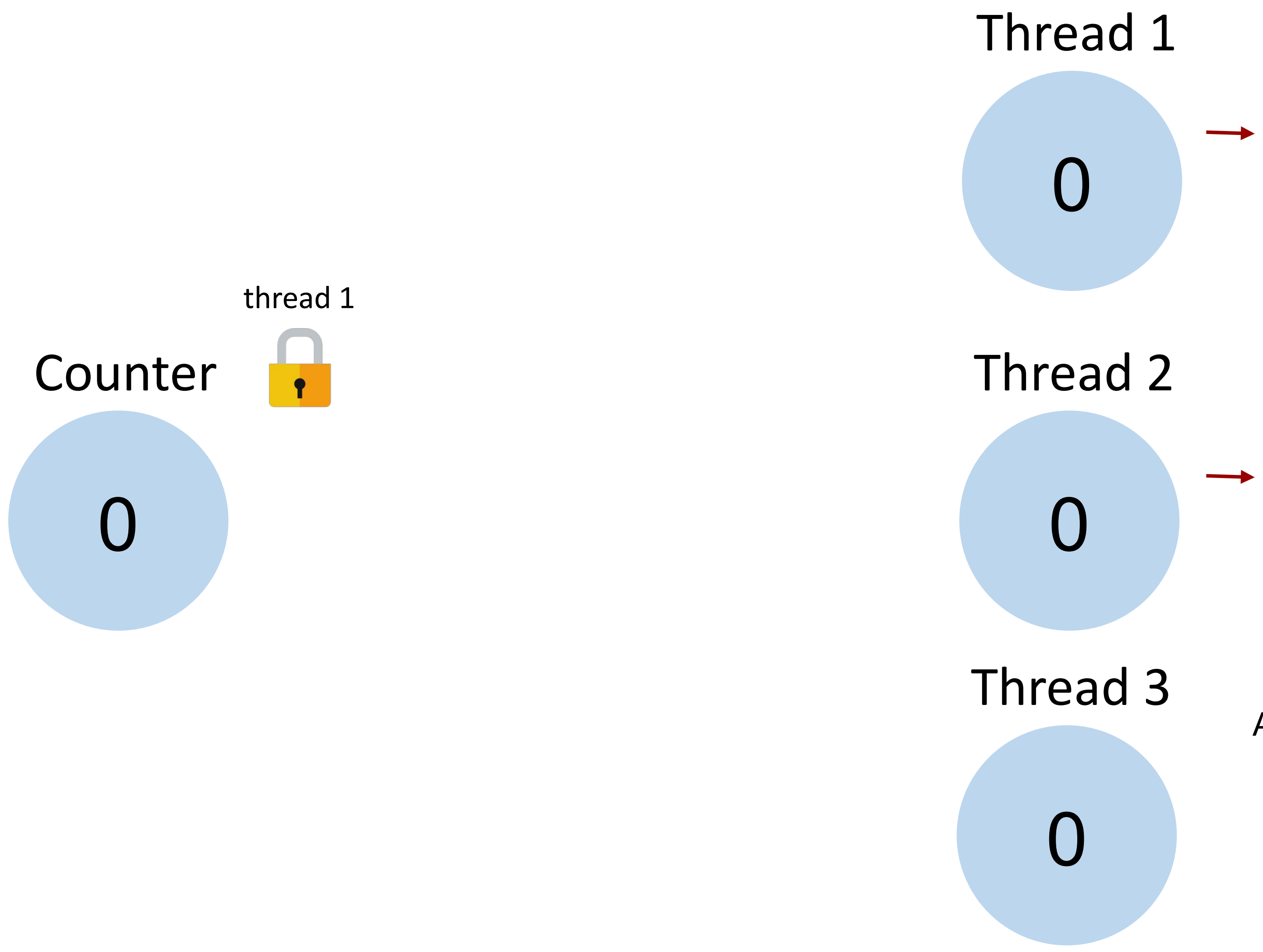
```
public class SynchronizedCounter implements Counter {
    private int c = 0;

    public synchronized void increment() {
        c++;
    }

    public synchronized int value() {
        return c;
    }
}
```

Task B: SynchronizedCounter



```
synchronized void increment() {
    c++;
}
```

Task B: SynchronizedCounter

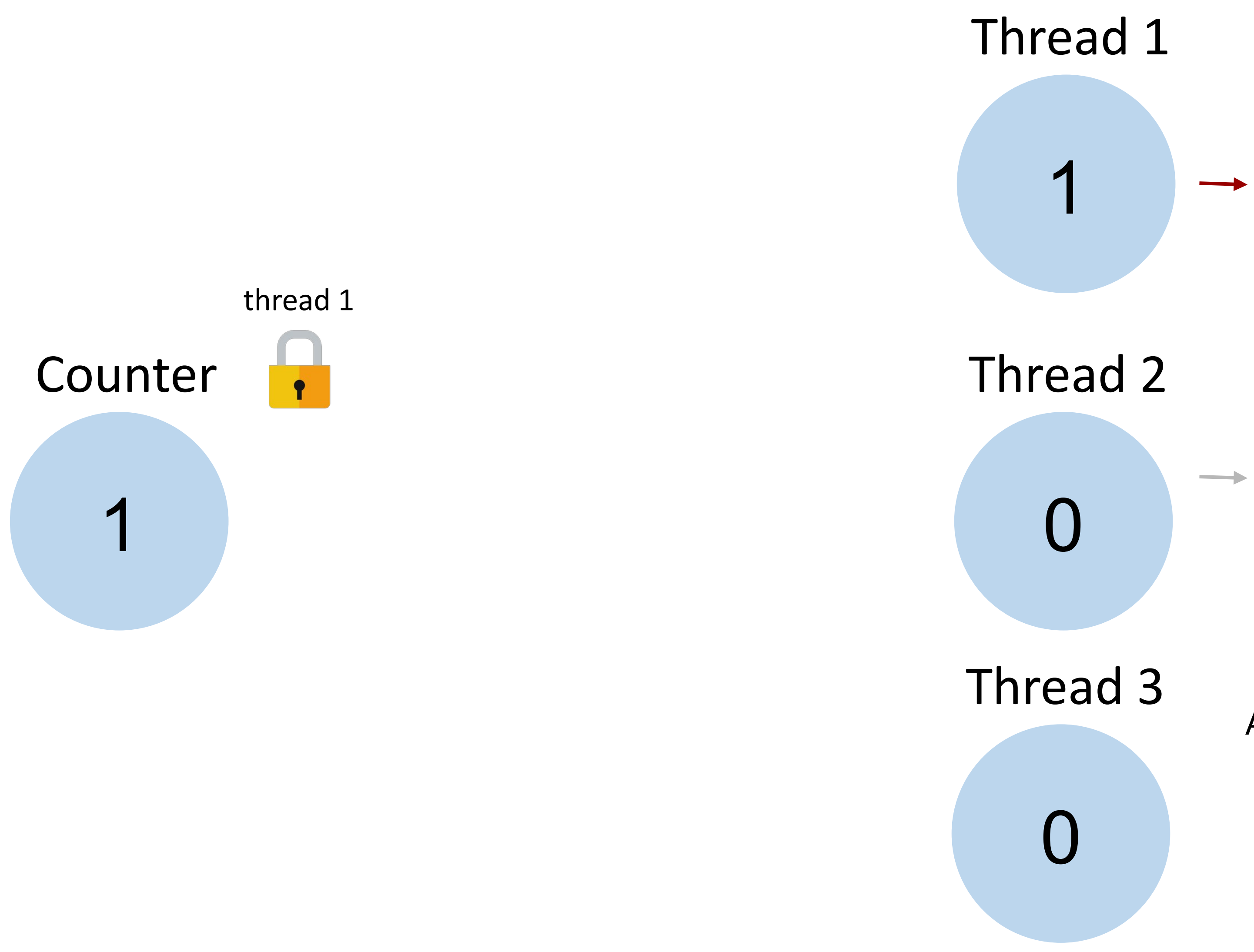


```
synchronized void increment() {
    c++;
}
```

```
synchronized void increment() {
    c++;
}
```

Thread 2 tries to acquire lock on counter. As the lock is already acquired by thread 1 the thread 2 suspends its execution.

Task B: SynchronizedCounter

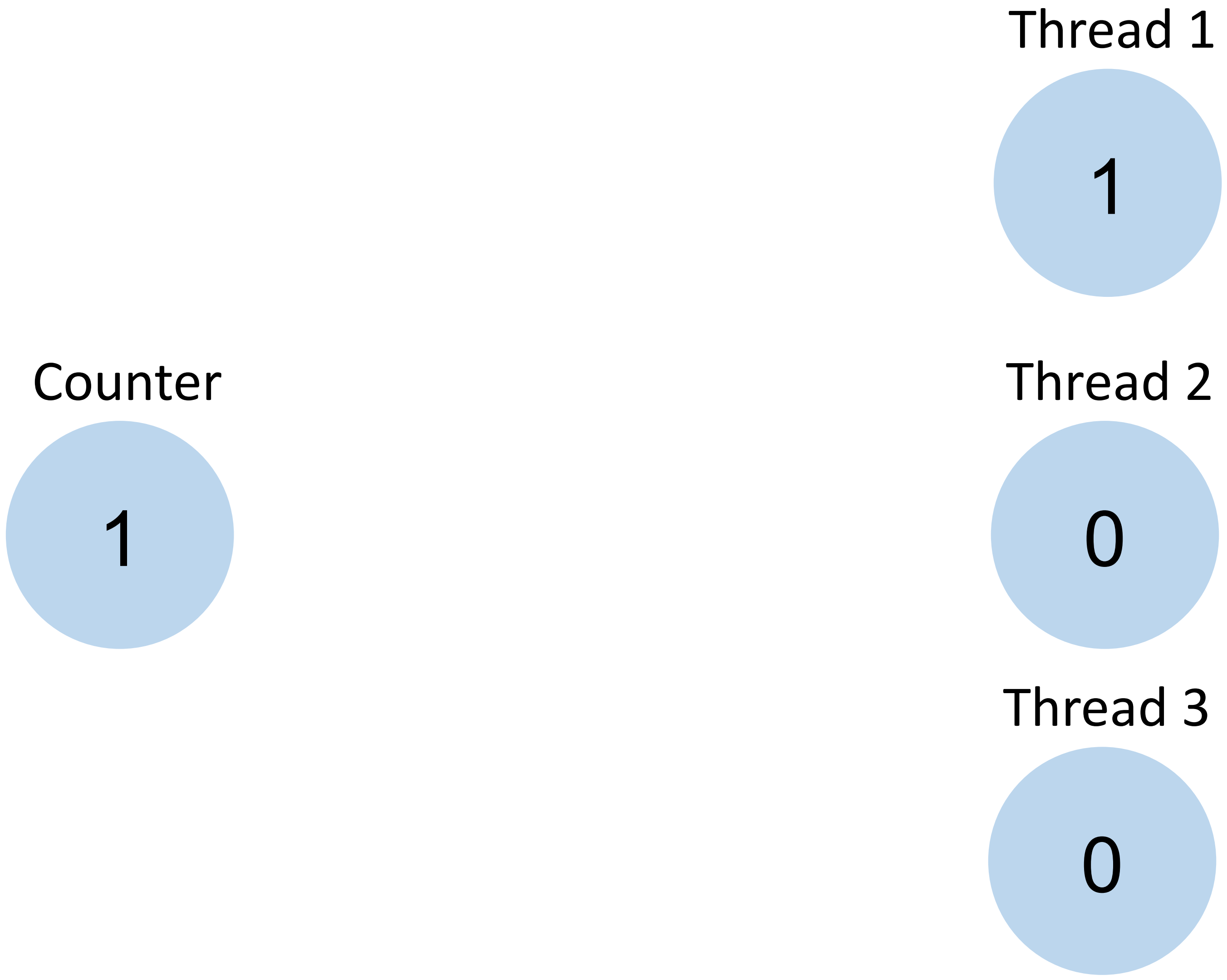


```
synchronized void increment() {
    c++;
}
```

```
synchronized void increment() {
    c++;
}
```

Thread 2 tries to acquire lock on counter. As the lock is already acquired by thread 1 the thread 2 suspends its execution.

Task B: SynchronizedCounter

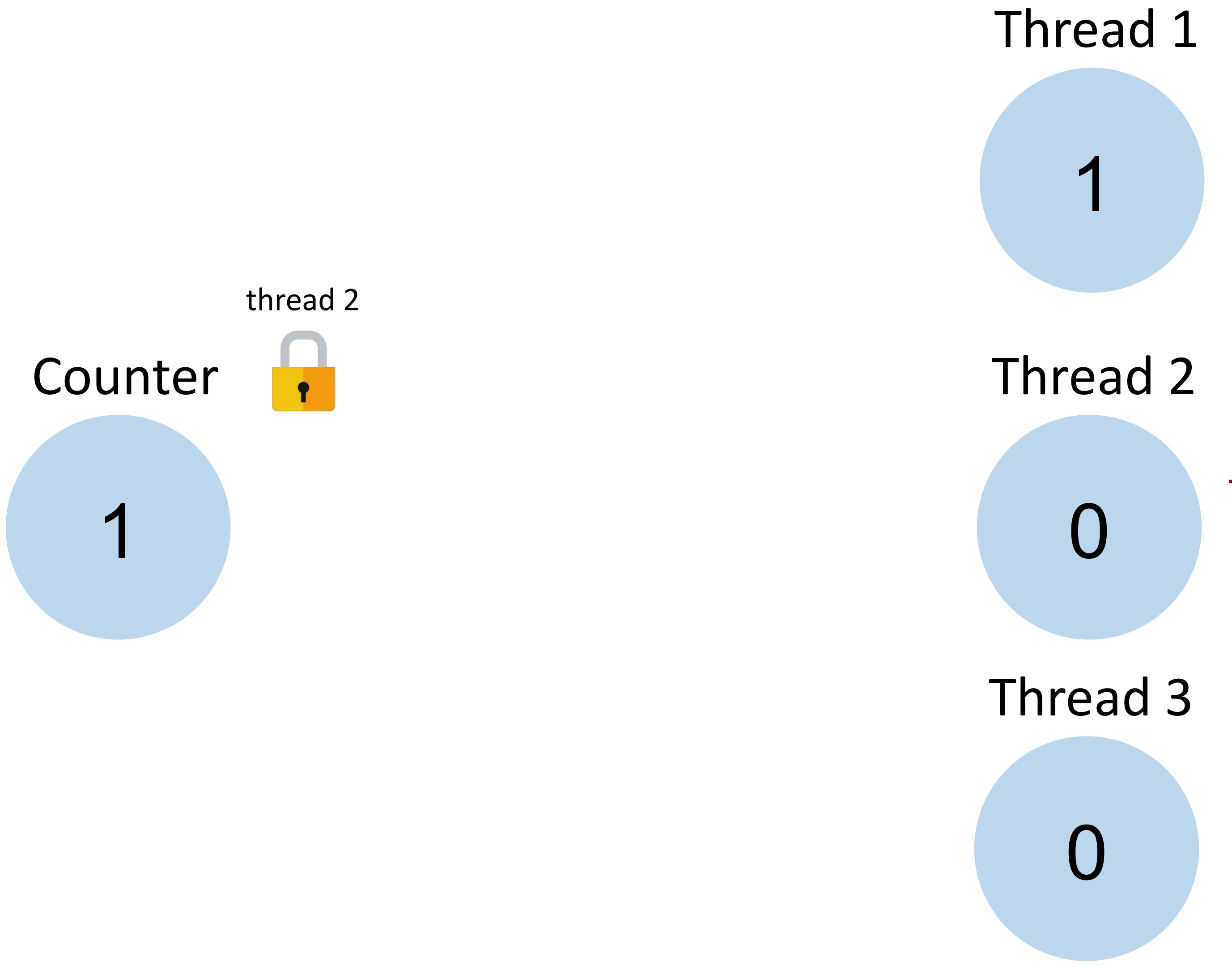


```java
synchronized void increment() {
    c++;
}
```

releases lock upon method exit

```java
synchronized void increment() {
    c++;
}
```

Task B: SynchronizedCounter



```
synchronized void increment() {
    c++;
}
```

```
synchronized void increment() {
    c++;
}
```

Task D

- Implement a FairThreadCounter that ensures that different threads increment the Counter in a round-robin fashion. That is, two threads with ids 1 and 2 would increment the value in the following order 1, 2, 1, 2, 1, 2, etc. You should implement the scheduling using the wait and notify methods.
- (Optional) Extend your implementation to work with arbitrary number of threads (instead of only 2) that increment the counter in round-robin fashion.

Wait and Notify Recap

Object (lock) provides `wait` and `notify` methods

(any object is a lock)

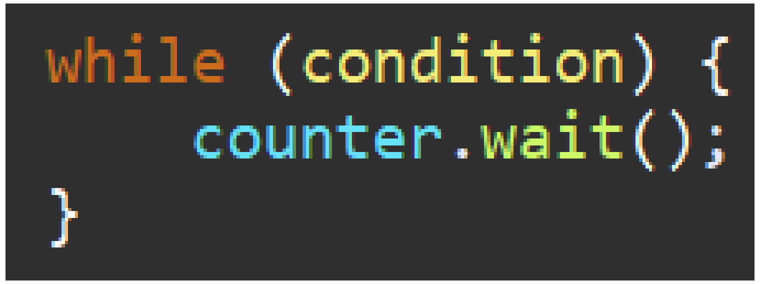
`wait`: Thread must own object’s lock to call `wait`
  thread releases lock and is added to “waiting list” for that object
  thread waits until `notify` is called on the object

`notify`: Thread must own object’s lock to call `notify`

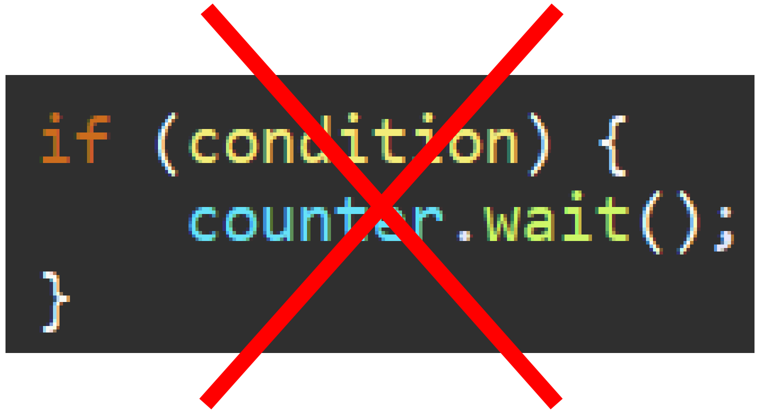
`notify`: Wake one (arbitrary) thread from object’s “waiting list”

`notifyAll`: Wake all threads

Wait and Notify Recap



```
while (condition) {
    counter.wait();
}
```

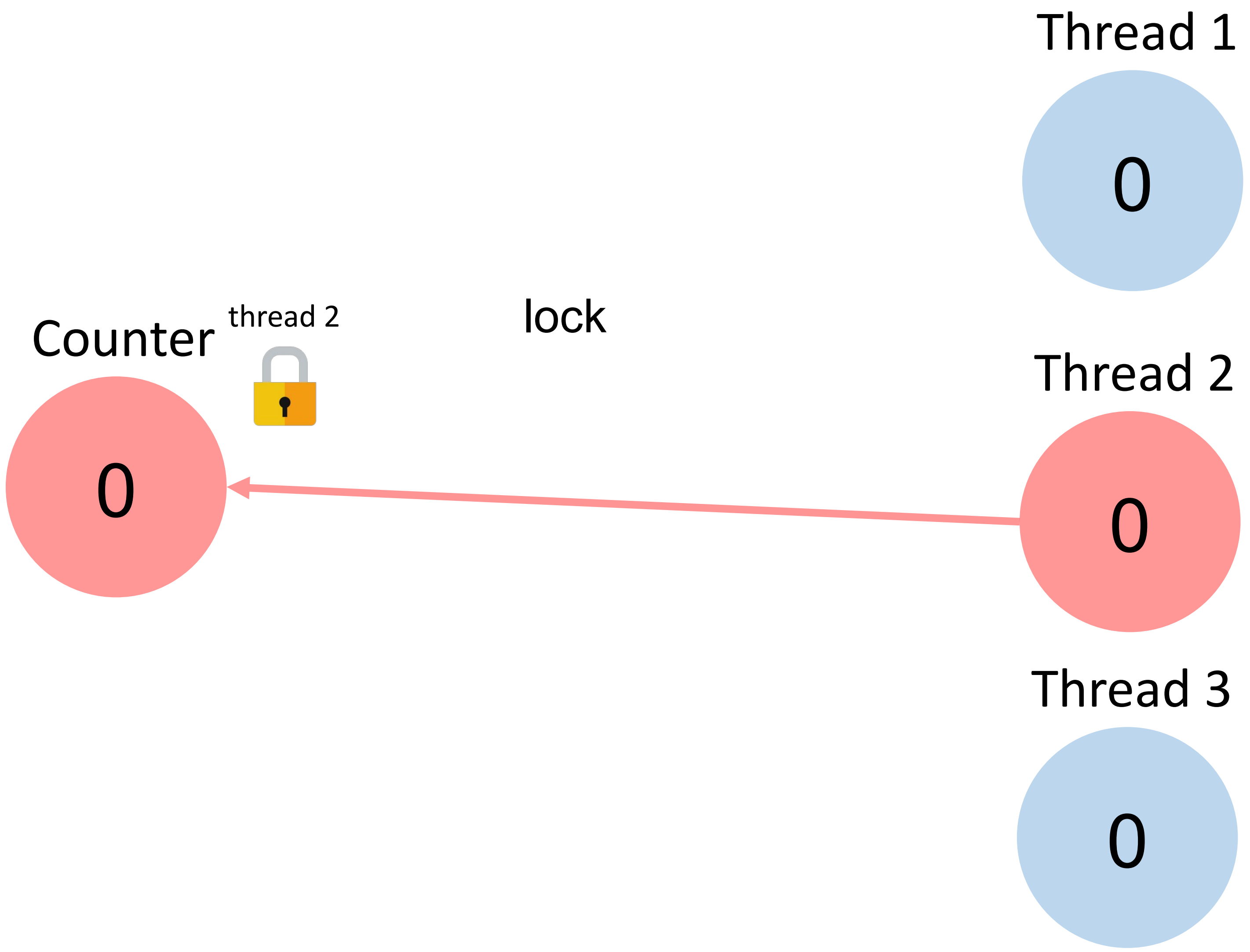


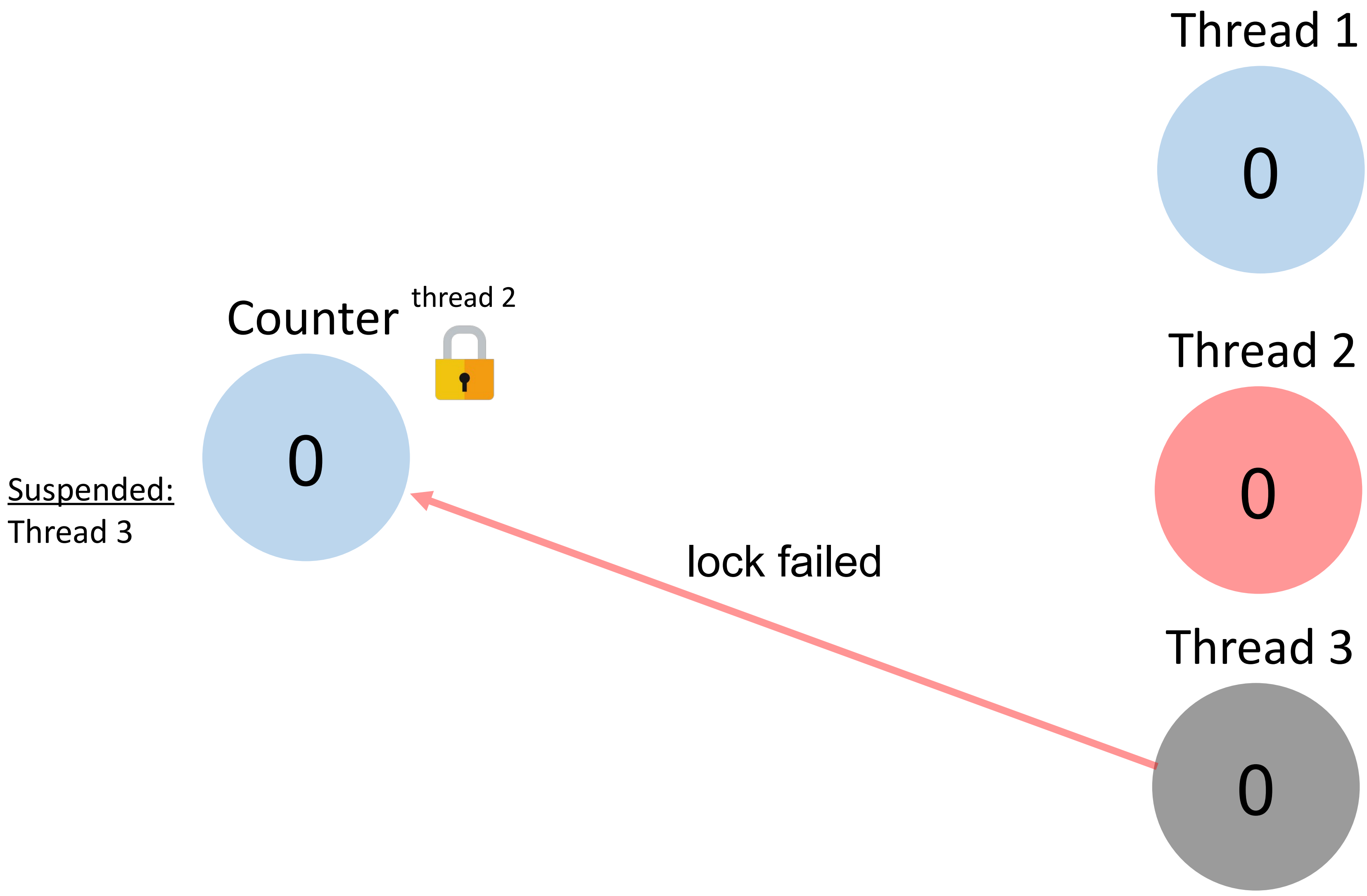
```
if (condition) {
    counter.wait();
}
```

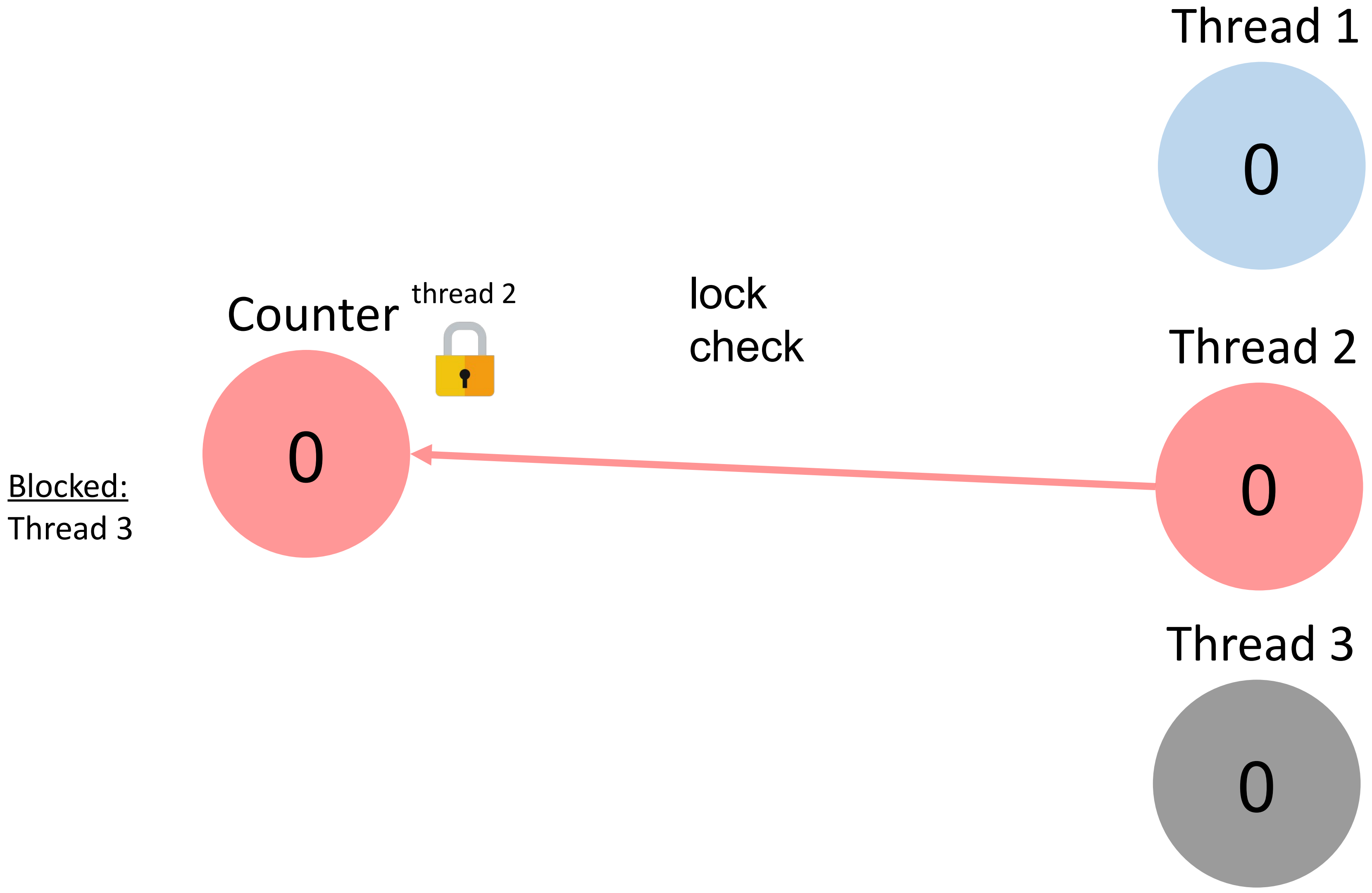
Spurious wake-ups and notifyAll()
→ `wait` has to be in a `while` loop

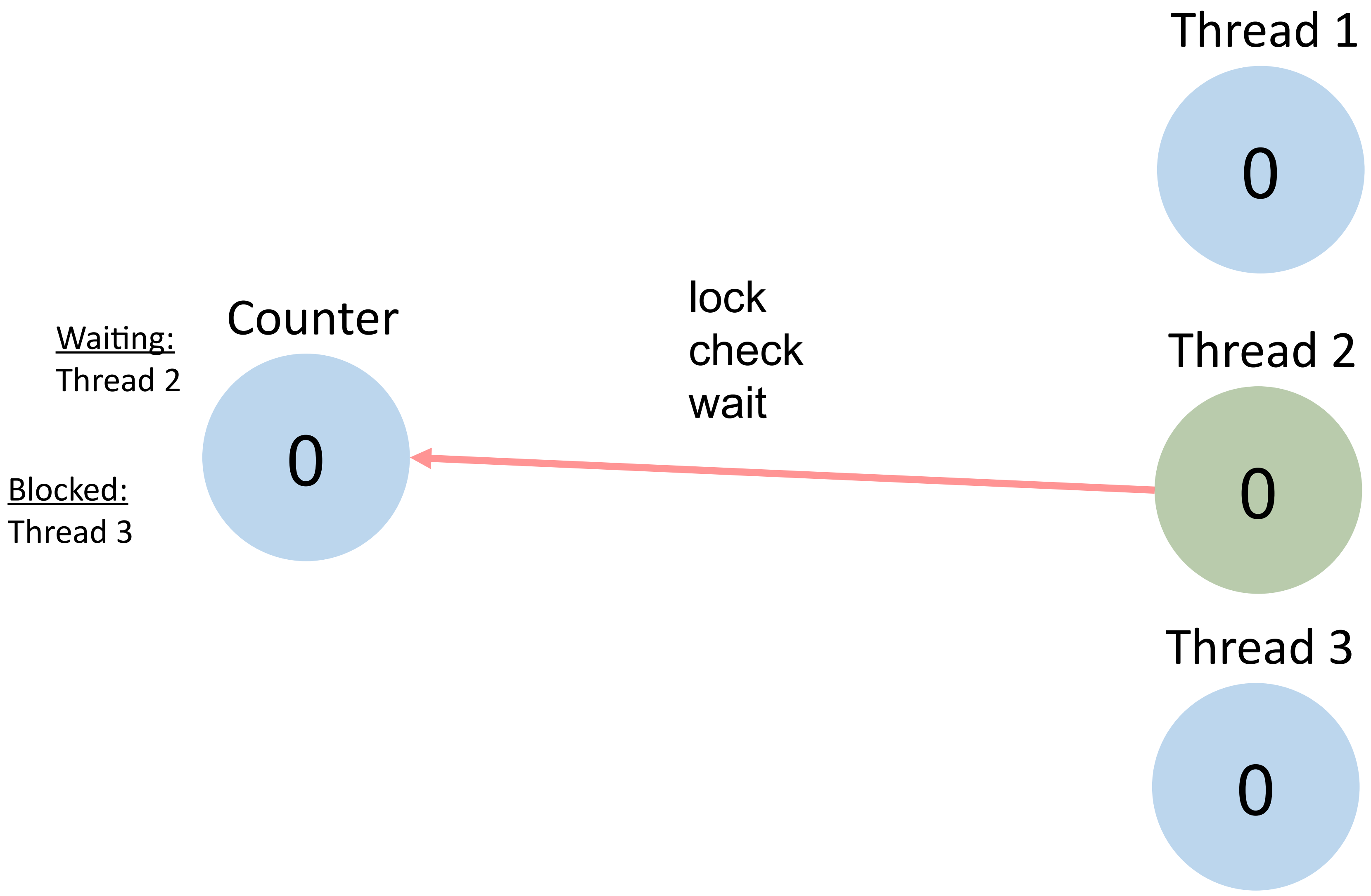
Thread 1 must increment first!

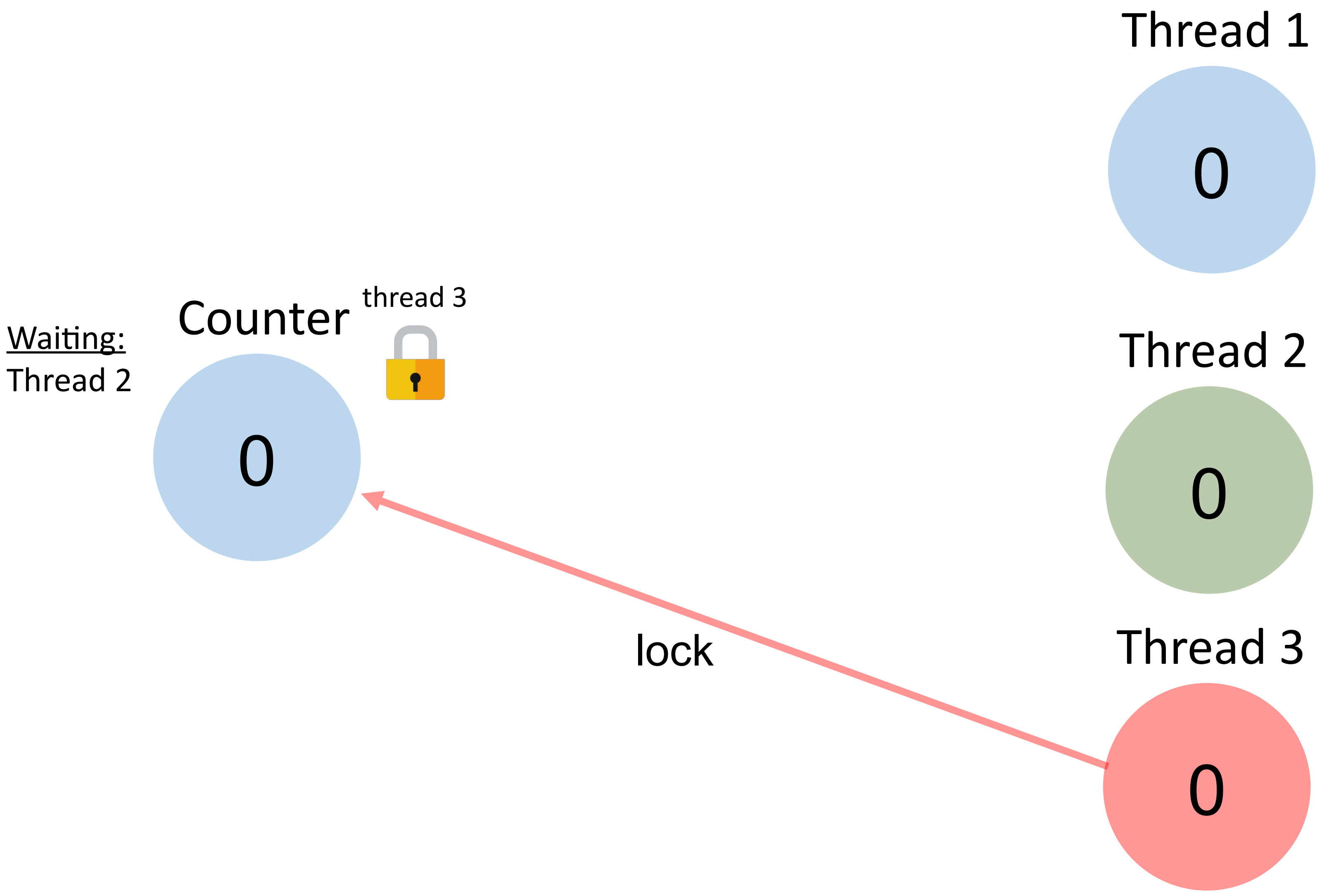
Counter
0
Thread 1
0
Thread 2
0
Thread 3
0

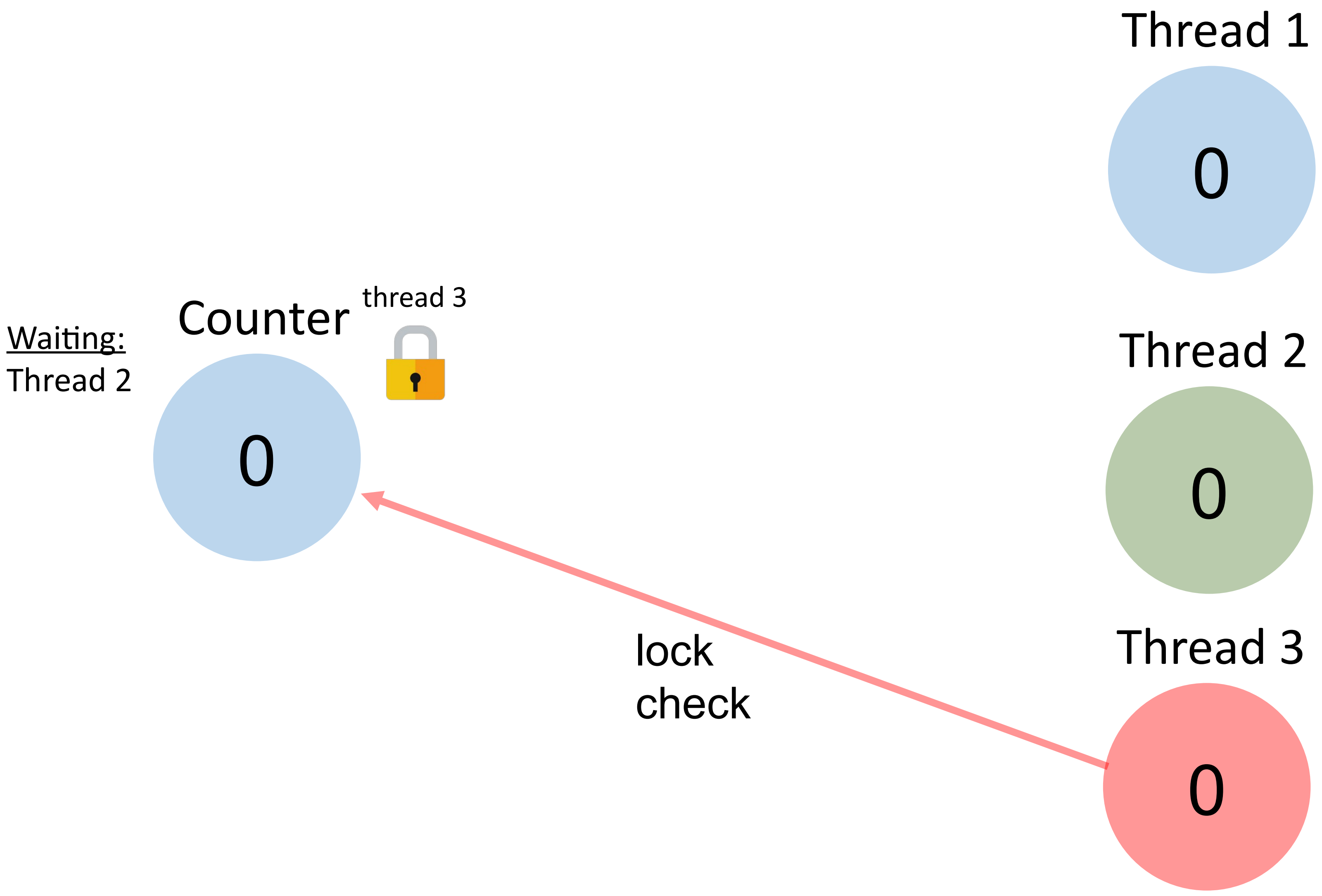

Thread 1
0
Counter
thread 2
lock
Thread 2
0
0
Thread 3
0

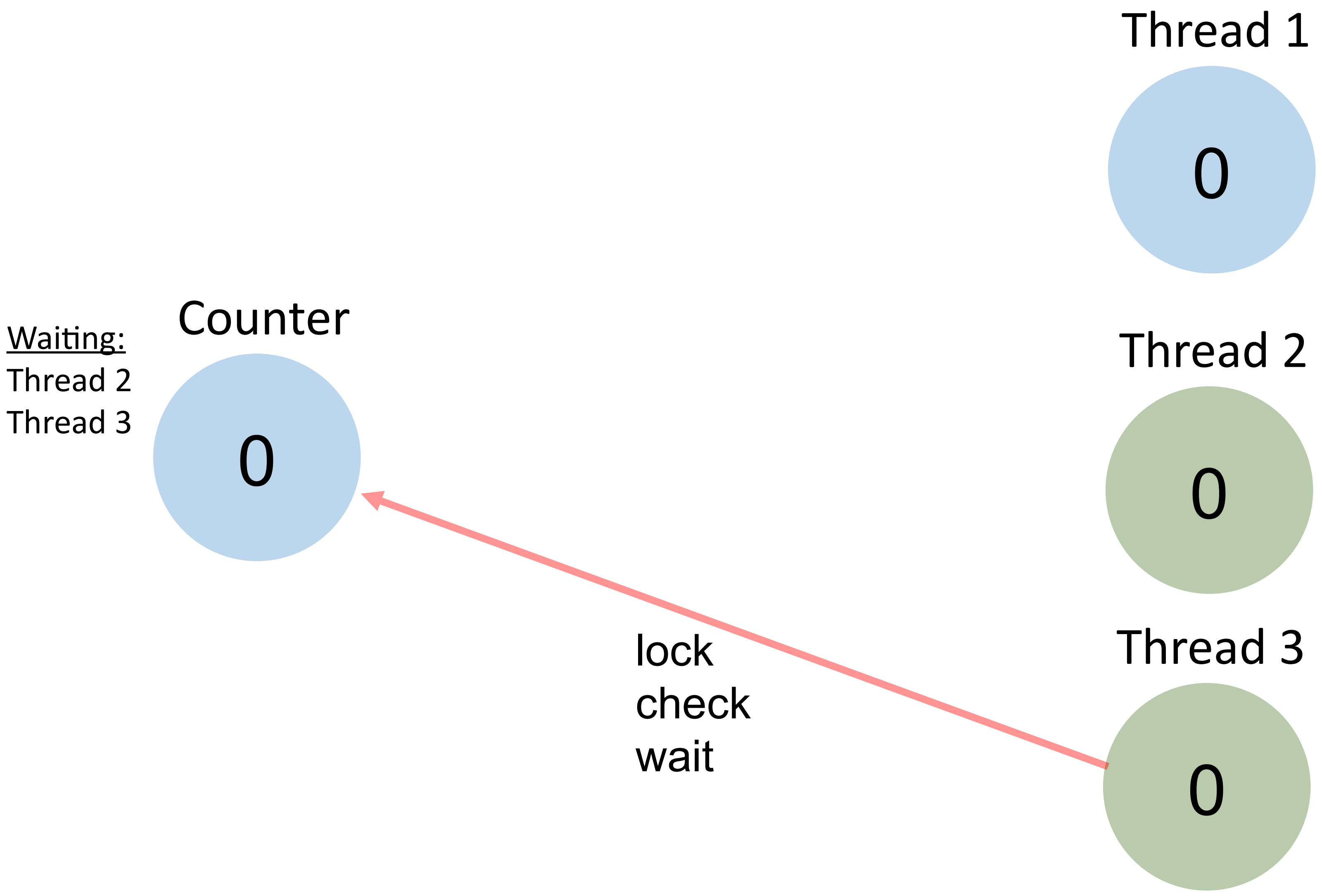

Thread 1
0
Counter
thread 2
0
Thread 2
0
Suspended:
Thread 3
lock failed
Thread 3
0

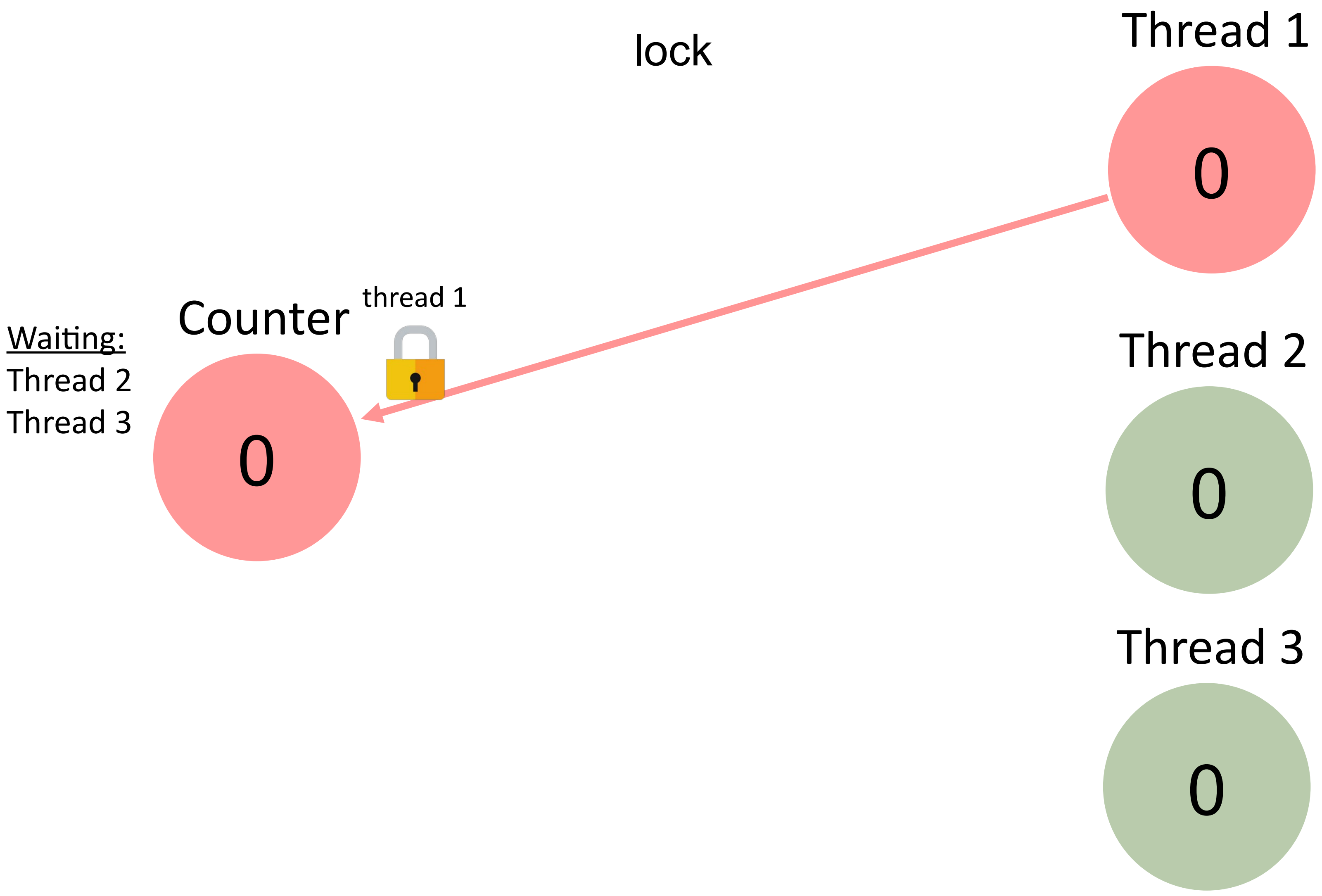

Thread 1
0
Counter
thread 2
lock
check
Thread 2
0
0
Blocked:
Thread 3
Thread 3
0

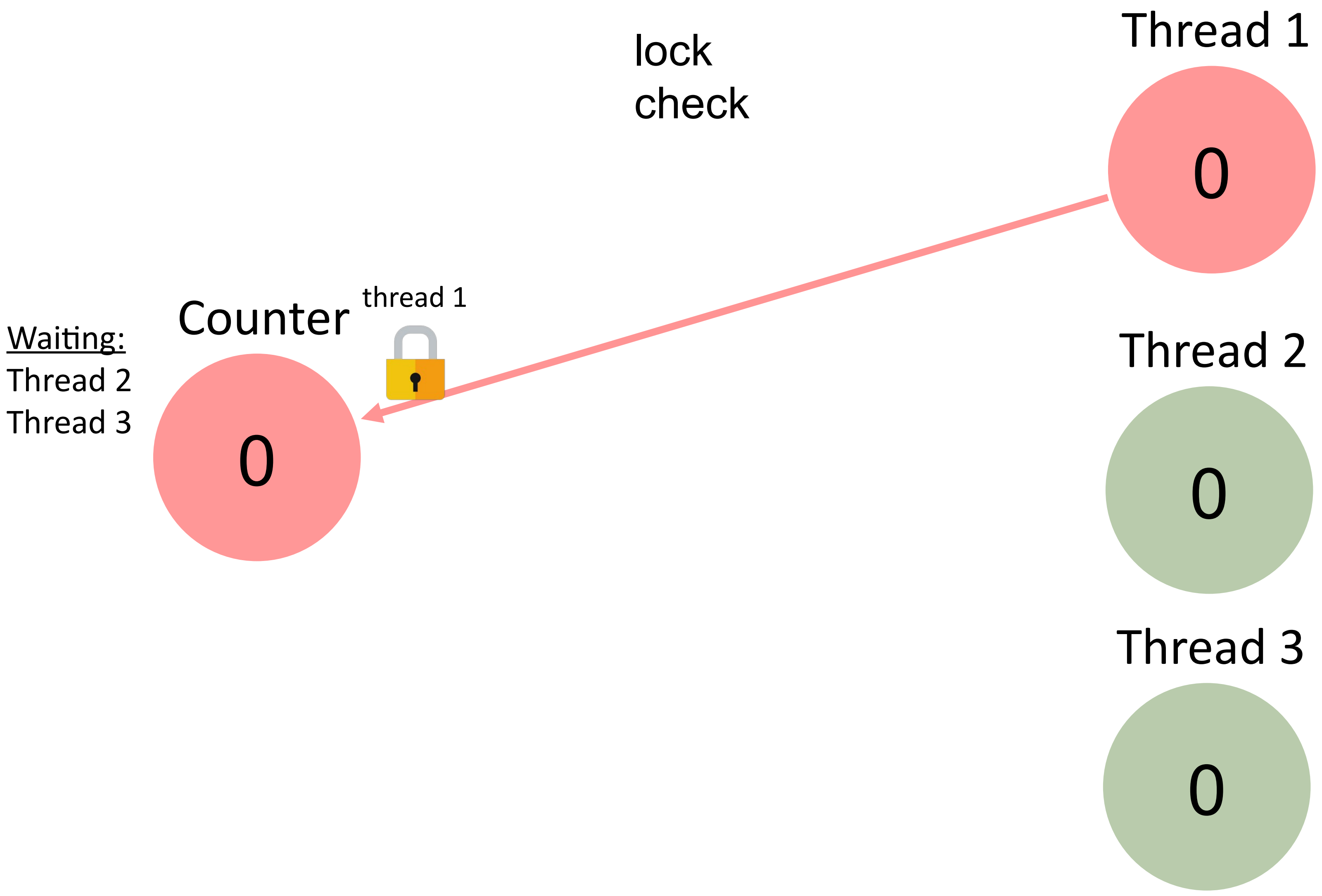

Thread 1
0
Counter
Waiting:
Thread 2
0
Blocked:
Thread 3
lock
check
wait
Thread 2
0
Thread 3
0

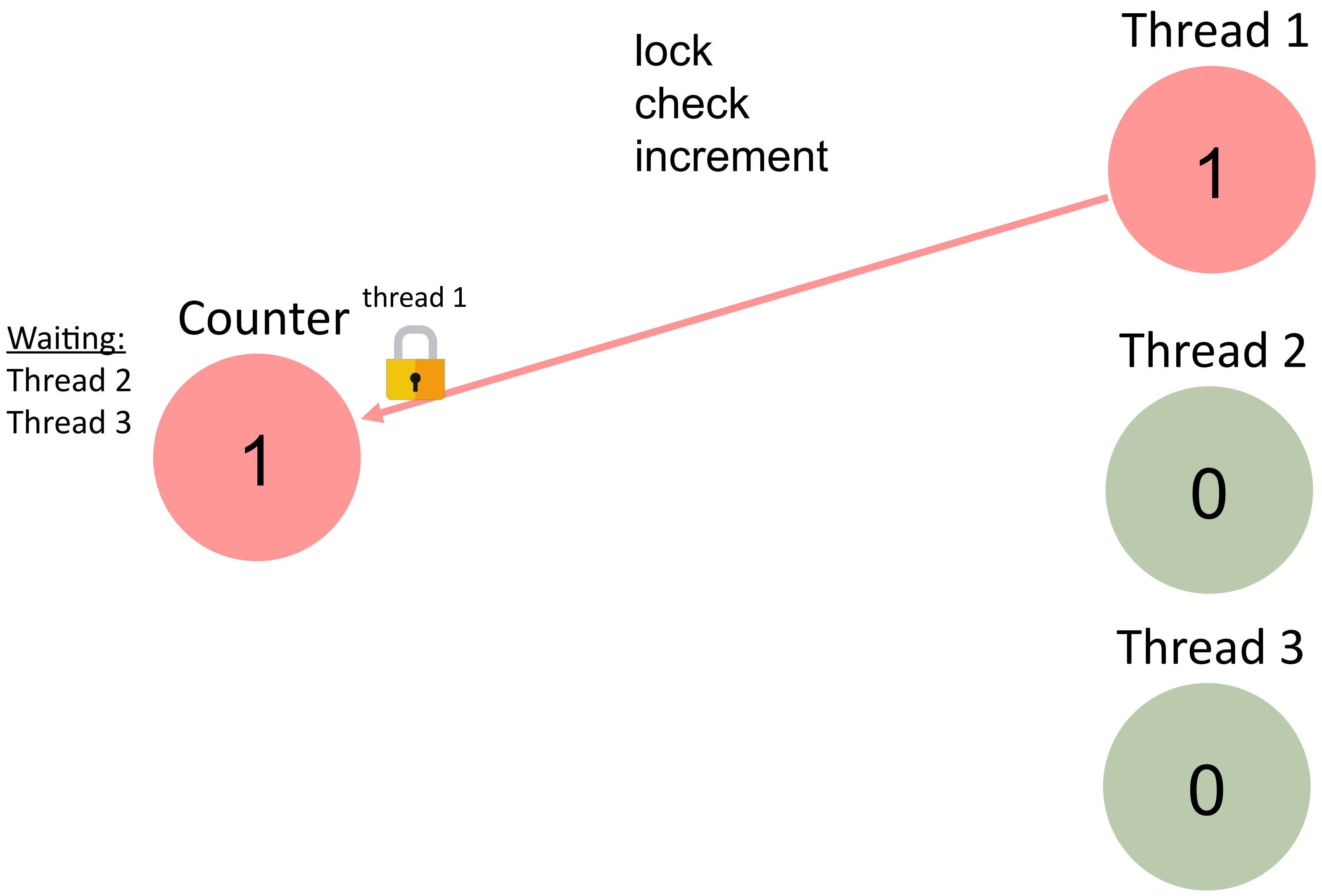

Waiting:
Thread 2
Counter
thread 3
0
Thread 1
0
Thread 2
0
lock
Thread 3
0

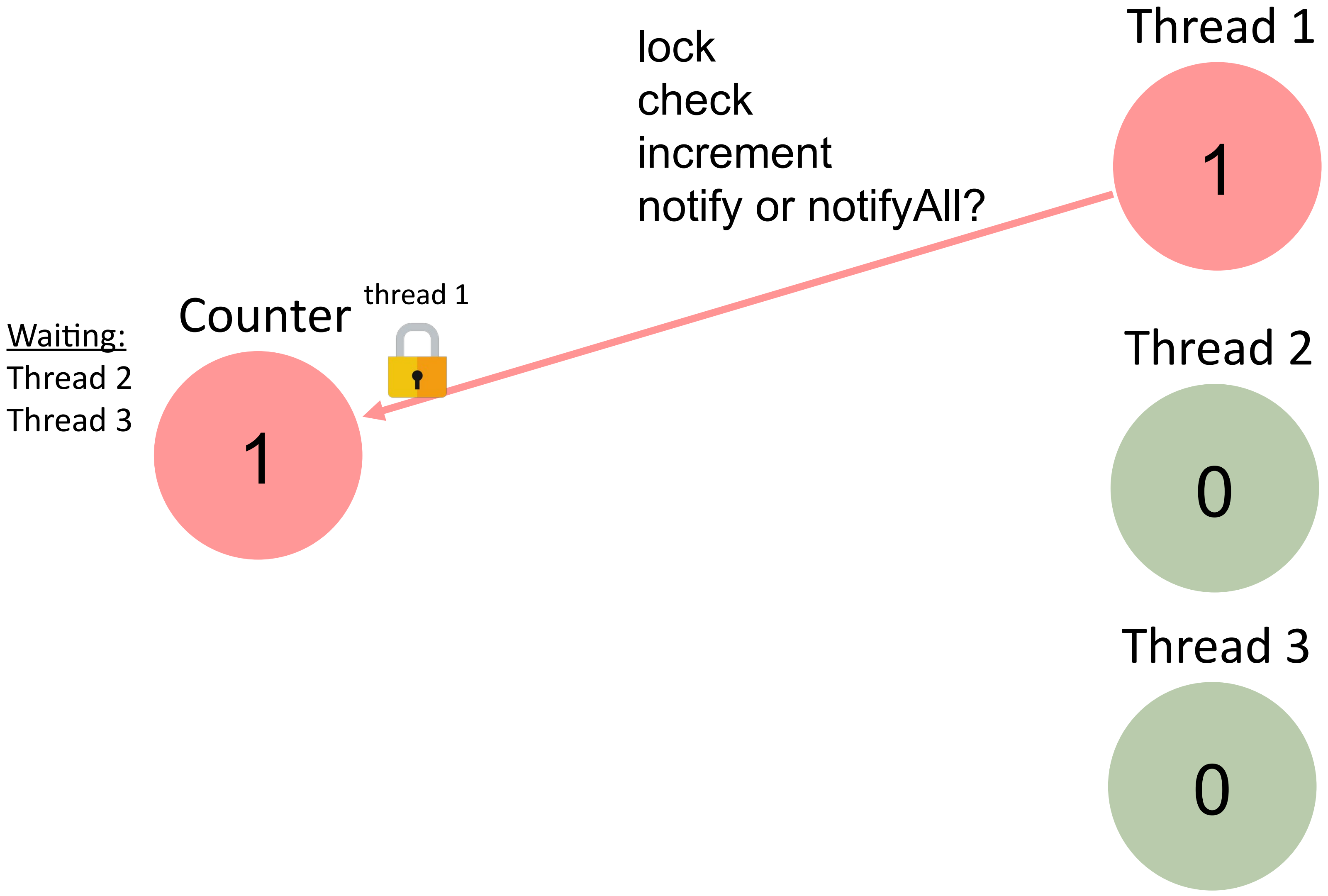

Thread 1
0
Waiting:
Thread 2
Counter
thread 3
0
Thread 2
0
lock
check
Thread 3
0

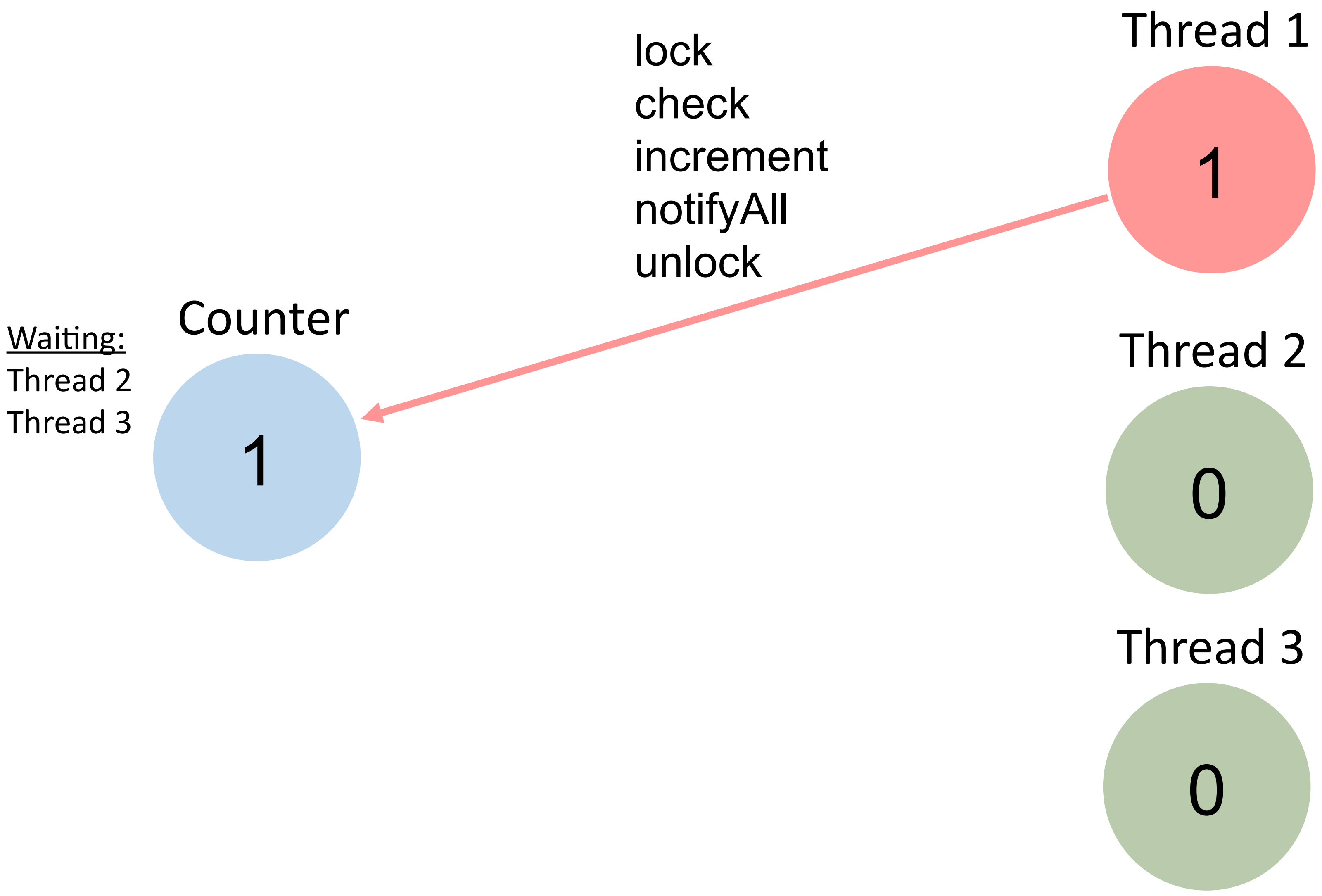

Thread 1
0
Counter
Waiting:
Thread 2
Thread 3
0
Thread 2
0
lock
check
wait
Thread 3
0

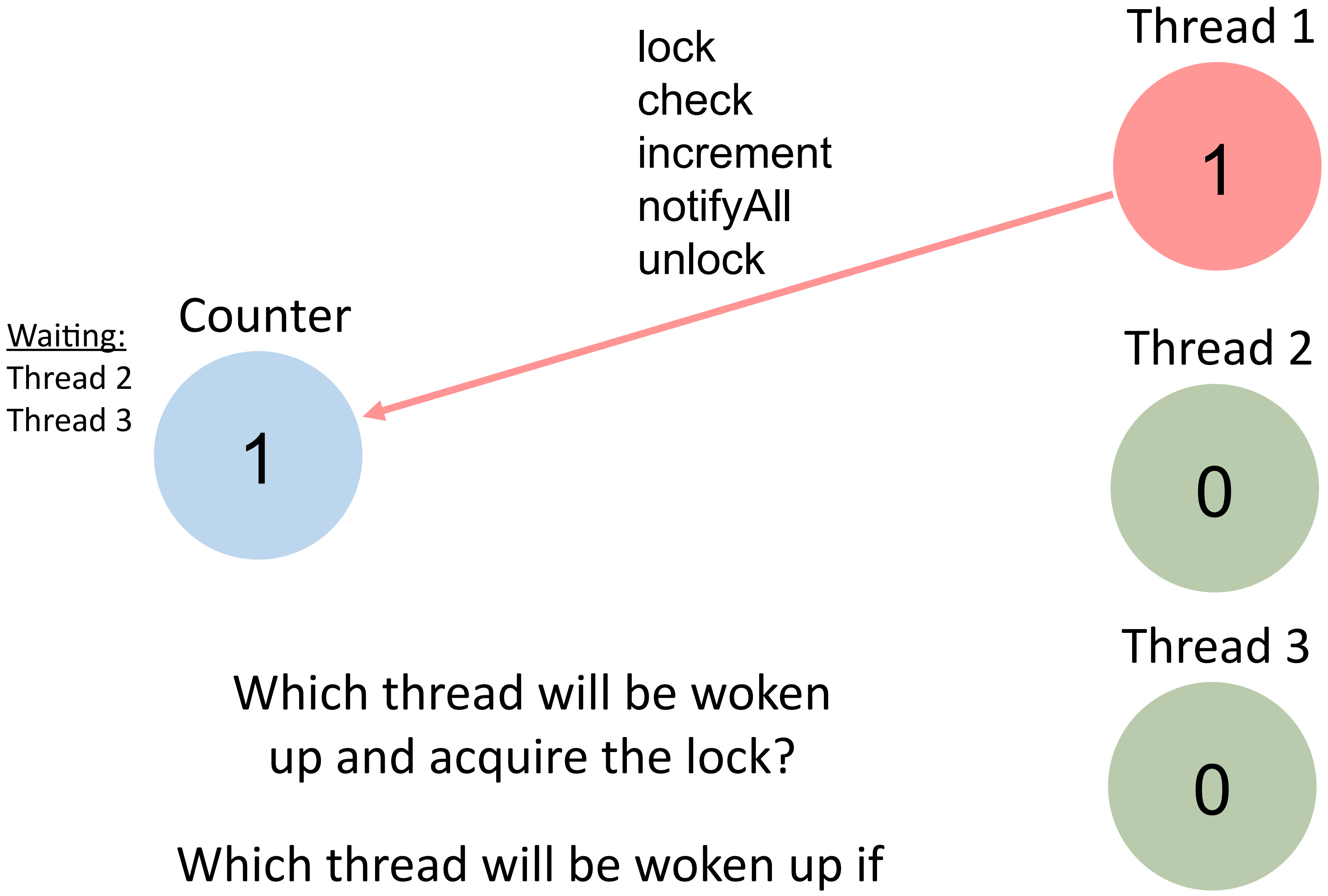

lock
Thread 1
0
Thread 2
0
Thread 3
0
Counter
thread 1
0
Waiting:
Thread 2
Thread 3


lock
check
Thread 1
0
Counter
thread 1
Waiting:
Thread 2
Thread 3
0
Thread 2
0
Thread 3
0


lock
check
increment
Thread 1
1
Counter
thread 1
Waiting:
Thread 2
Thread 3
1
Thread 2
0
Thread 3
0


lock
check
increment
notify or notifyAll?
Thread 1
1
Waiting:
Thread 2
Thread 3
Counter
thread 1
1
Thread 2
0
Thread 3
0

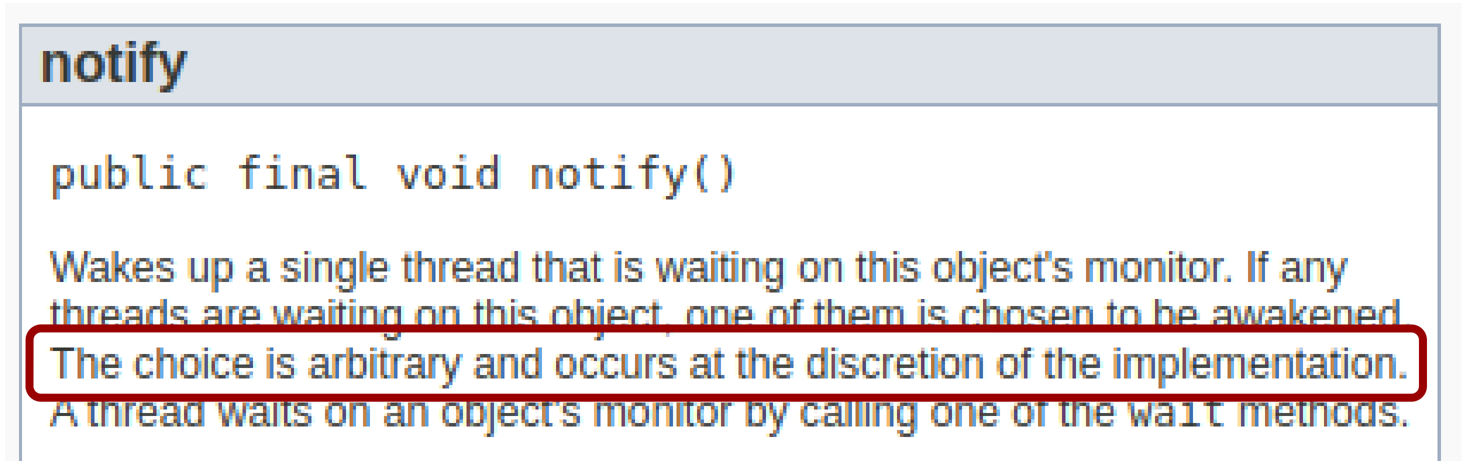

lock
check
increment
notifyAll
unlock
Thread 1
1
Counter
1
Waiting:
Thread 2
Thread 3
Thread 2
0
Thread 3
0



Which thread will be woken up and acquire the lock?

Which thread will be woken up if we use notify instead of notifyAll?

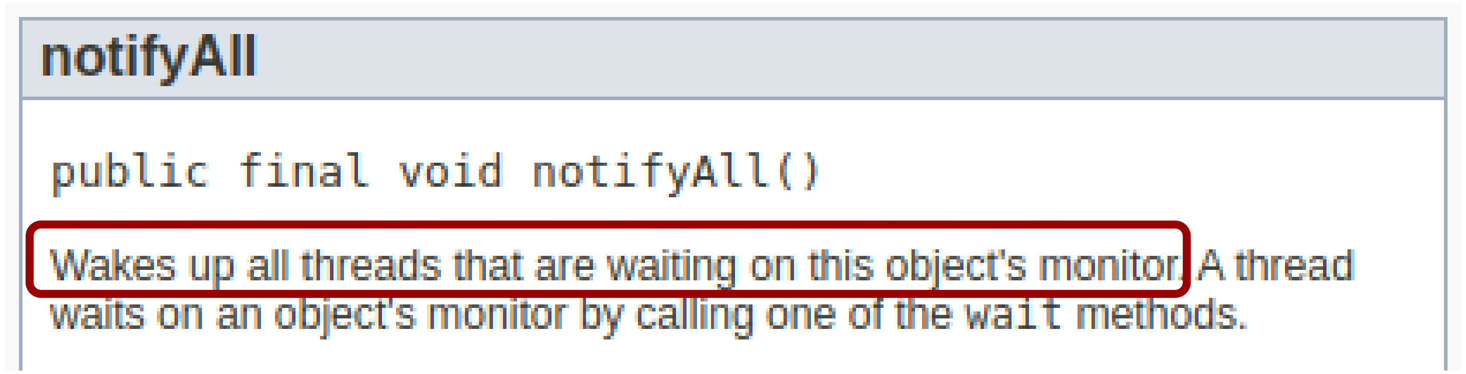
How to find the difference between notify vs notifyAll?



notify

```
public final void notify()
```

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. A thread waits on an object's monitor by calling one of the `wait` methods.



notifyAll

```
public final void notifyAll()
```

Wakes up all threads that are waiting on this object's monitor. A thread waits on an object's monitor by calling one of the `wait` methods.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Object.html

Task E: AtomicCounter

```
public class AtomicCounter implements Counter {


    public void increment() {
        ??
    }

    public int value() {
        ??
    }
}
```

Task E: AtomicCounter

```
public class AtomicCounter implements Counter {
    private AtomicInteger c = new AtomicInteger(0);

    public void increment() {
        c.incrementAndGet();
    }

    public int value() {
        return c.get();
    }
}
```

Task E: AtomicCounter

```
public class AtomicCounter implements Counter {
    private AtomicInteger c = new AtomicInteger(0);

    public void increment() {
        c.incrementAndGet();
    }

    public int value() {
        return c.get();
    }
}
```

What is the difference?

`int`	`AtomicInteger`
`c++;`	`c.incrementAndGet();`

Task E: AtomicCounter

```java
public class AtomicCounter implements Counter {
    private AtomicInteger c = new AtomicInteger(0);

    public void increment() {
        c.incrementAndGet();
    }

    public int value() {
        return c.get();
    }
}
```

An operation is atomic if no other thread can see it partly executed. Atomic as in “appears indivisible”. However, does not mean it’s implemented as single instruction.

What is the difference?

`int`

1. load c → 0
2. c + 1 → 1
3. store c ← 1

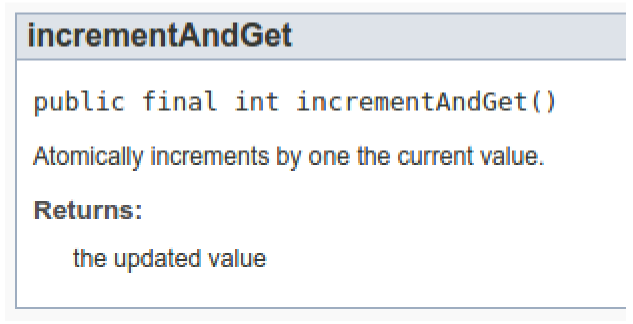
← `c++;`

not atomic

`AtomicInteger`

`c.getAndIncrement();` →

atomic



incrementAndGet

`public final int incrementAndGet()`

Atomically increments by one the current value.

Returns:

the updated value

Post- vs Pre-Increment

Post-Increment

```
int i = 0;
AtomicInteger c = new AtomicInteger(0);

System.out.println(i++);
System.out.println(c.getAndIncrement());
```

Pre-Increment

```
int i = 0;
AtomicInteger c = new AtomicInteger(0);

System.out.println(++i);
System.out.println(c.incrementAndGet());
```

Exercise 4: Pipelining Recap

Pipelining: Main Concepts Recap

Latency

Throughput

Balanced/Unbalanced Pipeline

Pipelining: Main Concepts Recap

Latency
time needed to perform a given computation
(e.g., process a customer)

Throughput

Balanced/Unbalanced Pipeline

Pipelining: Main Concepts Recap

Latency

time needed to perform a given computation
(e.g., process a customer)

Throughput

amount of work that can be done by a system in a given period of time
(e.g., how many customers can be processed in one minute)

Balanced/Unbalanced Pipeline

Pipelining: Main Concepts Recap

Latency

time needed to perform a given computation
(e.g., process a customer)

Throughput

amount of work that can be done by a system in a given period of time
(e.g., how many customers can be processed in one minute)

Balanced/Unbalanced Pipeline

a pipeline is balanced if each stage takes the same length of time

Library

At UZH the law students have been tasked with writing a legal essay about the philosophy of Swiss law. In order to write the essay, each student needs to read four different books on the subject, denoted as A, B, C and D (in this order).

This exercise is created by Lasse Meinen and is part of the unofficial VIS Prüfungsvorbereitungsworkshop Scripts available at:

https://vis.ethz.ch/de/services/pvw-scripts/

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library

Over at UZH the law students have been tasked with writing a legal essay about the philosophy of Swiss law. In order to write the essay, each student needs to read four different books on the subject, denoted as A, B, C and D (in this order).

Question 1: Let's assume all law students are a bit too competitive and don't return any books before they're done reading all of them. How long will it take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

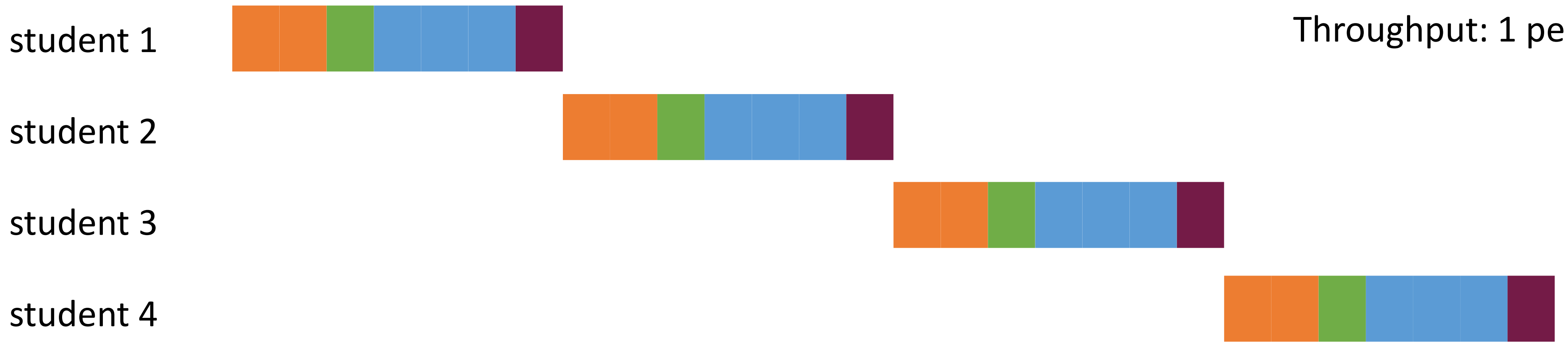
1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library

Total: 4 * 280 min

Latency: 280 min

Throughput: 1 per 280 min



Question 1: Let's assume all law students are a bit too competitive and don't return any books before they're done reading all of them. How long will it take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes

2) Reading book **B** takes 40 minutes

3) Reading book **C** takes 120 minutes

4) Reading book **D** takes 40 minutes

Library

Draw diagrams, as seen before

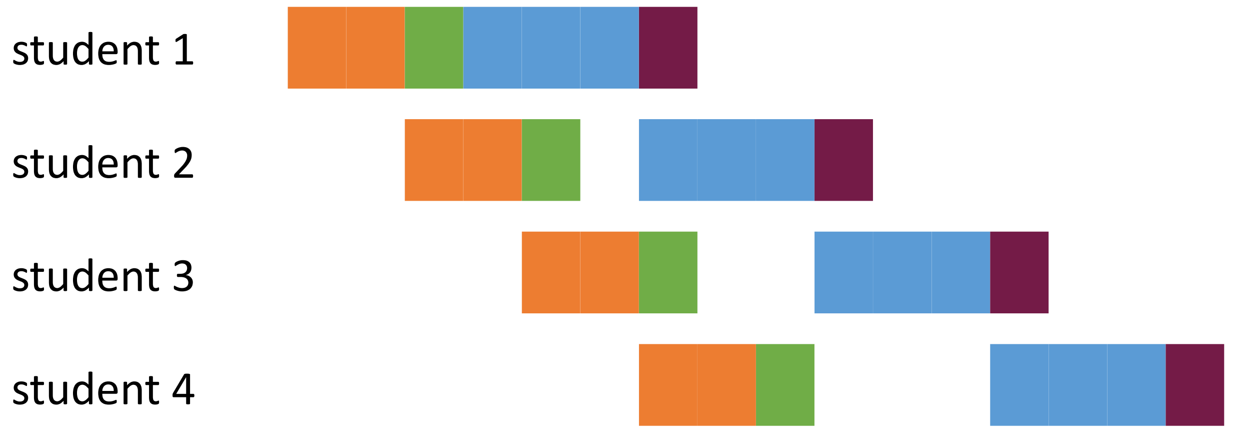
Question 2: The library introduces a ”one book at a time” policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library

Latency?



Question 2: The library introduces a ”one book at a time” policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

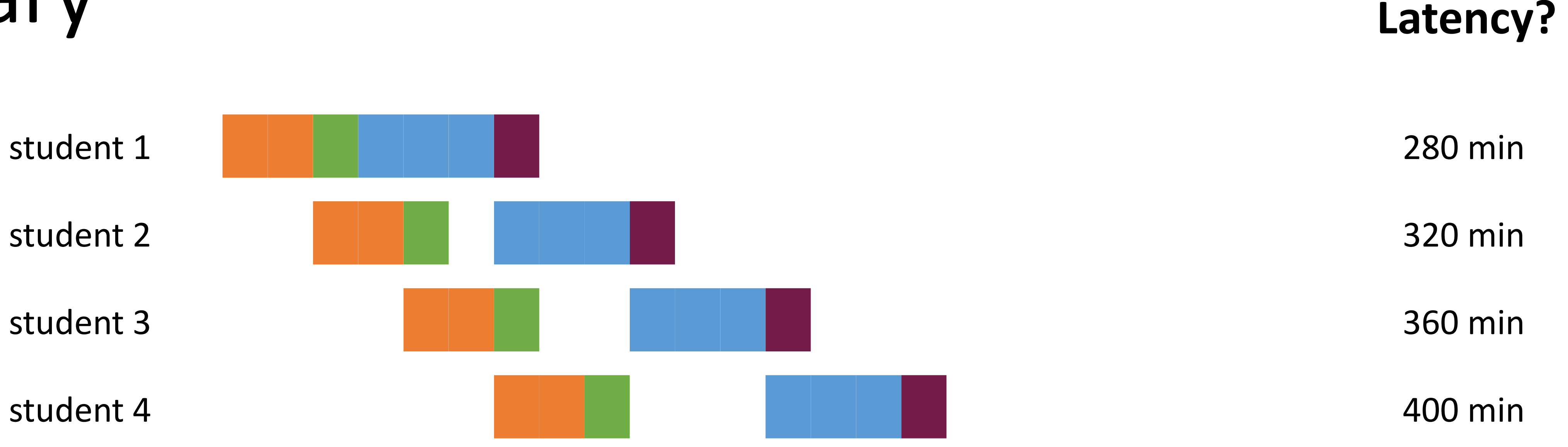
1) Reading book **A** takes 80 minutes

2) Reading book **B** takes 40 minutes

3) Reading book **C** takes 120 minutes

4) Reading book **D** takes 40 minutes

Library



Question 2: The library introduces a "one book at a time" policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

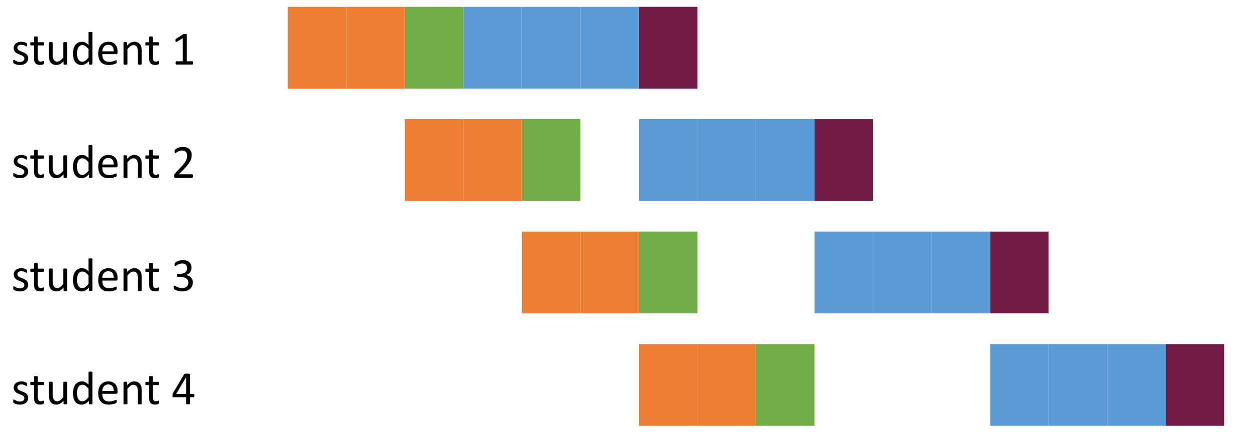
Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library

For this pipeline, latency makes sense only if asked for a particular student, not for the whole pipeline.

Latency?



280 min

320 min

360 min

400 min

Question 2: The library introduces a "one book at a time" policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes

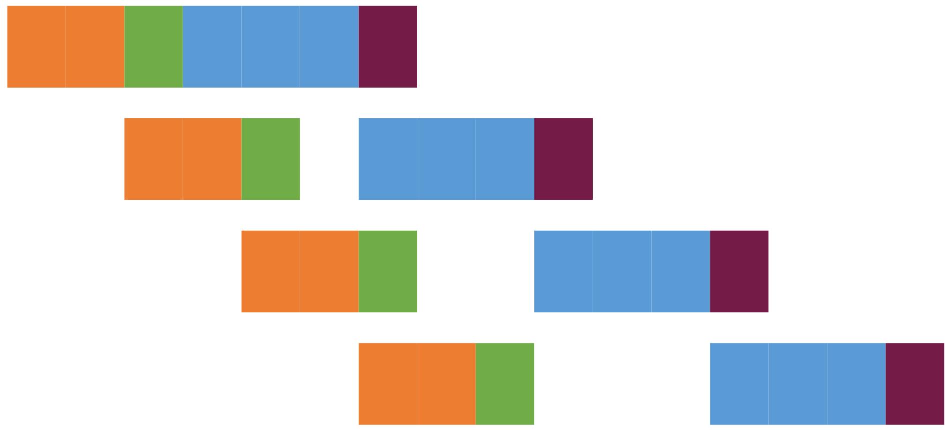
2) Reading book **B** takes 40 minutes

3) Reading book **C** takes 120 minutes

4) Reading book **D** takes 40 minutes

Library

	Balanced?	Throughput?	Latency?
student 1			280 min
student 2			320 min
student 3			360 min
student 4			400 min



Question 2: The library introduces a "one book at a time" policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes

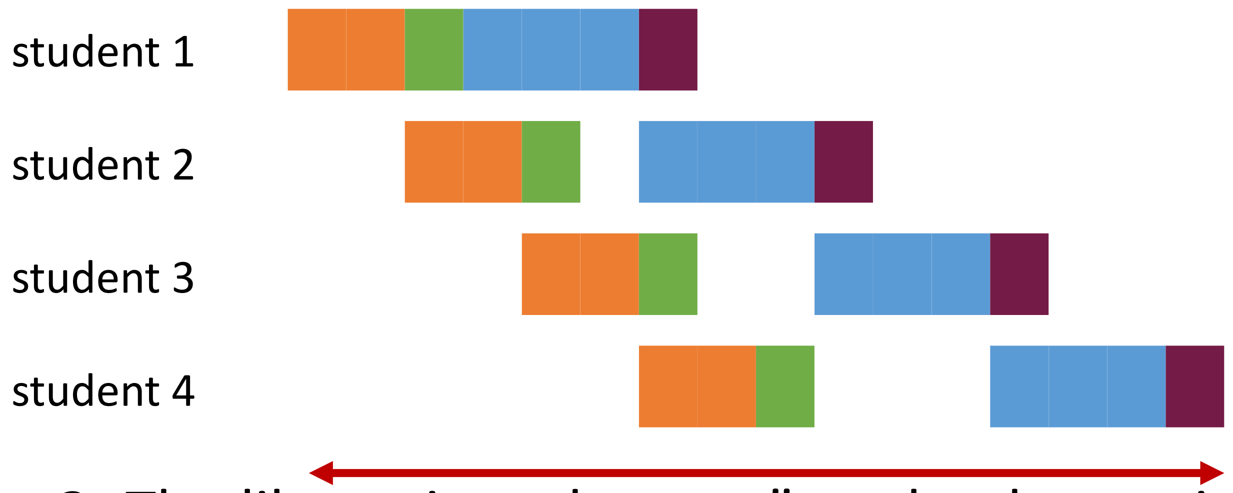
2) Reading book **B** takes 40 minutes

3) Reading book **C** takes 120 minutes

4) Reading book **D** takes 40 minutes

Library

Balanced? **Throughput?** **Latency?**



Throughput?	Latency?
1 student	280 min
per 160 minutes	320 min
With lead in (fixed number of students)	360 min
	400 min

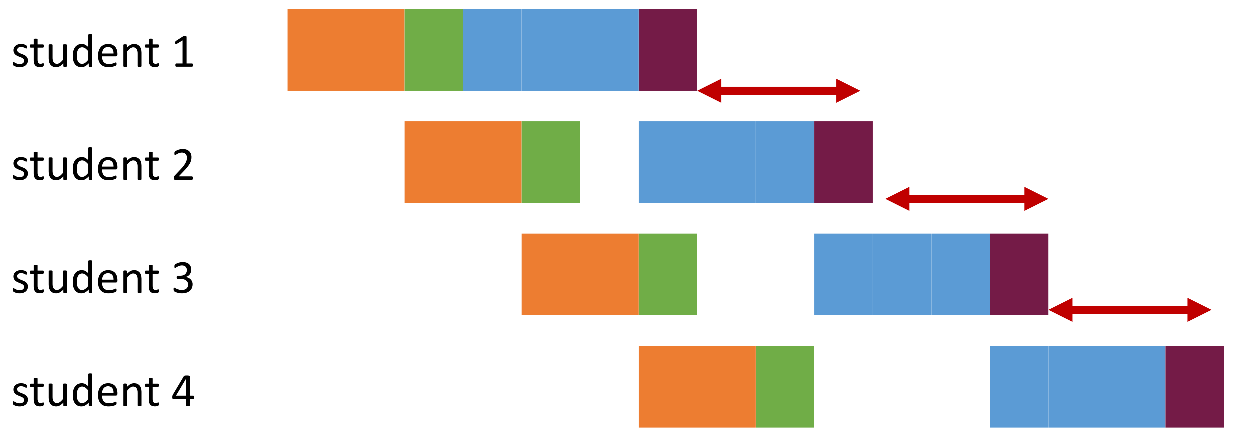
Question 2: The library introduces a ”one book at a time” policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library

Balanced? **Throughput?** **Latency?**



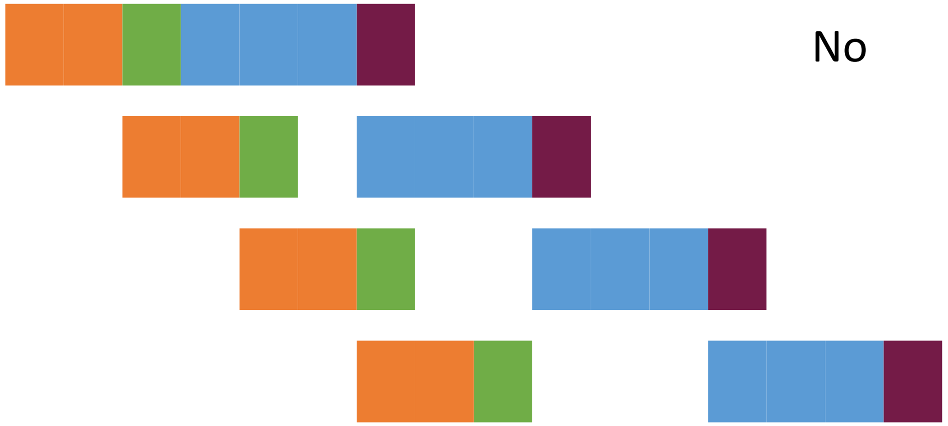
	Throughput?	Latency?
student 1	1 student	280 min
student 2	per 120 minutes	320 min
student 3	Without lead in (indefinite number of students)	360 min
student 4		400 min

Question 2: The library introduces a "one book at a time" policy, i.e., the students have to return a book before they can start on the next one. How long will it now take for 4 students until all of them have started writing their essays?

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes
2) Reading book **B** takes 40 minutes
3) Reading book **C** takes 120 minutes
4) Reading book **D** takes 40 minutes

Library



	Balanced?	Throughput?	Latency?
student 1	No	1 student per 160 minutes	280 min
student 2			320 min
student 3			360 min
student 4			400 min

The pipeline is not balanced since the stages have different length

Every student takes the exact same amount of time to read a book, concretely:

1) Reading book **A** takes 80 minutes

2) Reading book **B** takes 40 minutes

3) Reading book **C** takes 120 minutes

4) Reading book **D** takes 40 minutes

Exercise 4

Task 1 - Pipelining

Bob, Mary, John and Alice

50 min

90 min

15 min

a) Laundry time using sequential order

b) Design a strategy with better laundry time

c) How would the laundry time improve if they bought a new dryer?

Task 2 - Pipelining II

Assume a processor that can each cycle issue either:

- one multiplication instruction with latency 6 cycles
- one addition instruction with latency 3 cycles

How many cycles are required to execute following loops?

```
for (int i = 0; i < data.length; i++)
{
    data[i] = data[i] * data[i];
}
```

```
for (int i = 0; i < data.length; i += 2)
{
    j = i + 1;
    data[i] = data[i] * data[i];
    data[j] = data[j] * data[j];
}
```

```
for (int i = 0; i < data.length; i += 4) {
    j = i + 1;
    k = i + 2;
    l = i + 3;
    data[i] = data[i] * data[i];
    data[j] = data[j] * data[j];
    data[k] = data[k] * data[k];
    data[l] = data[l] * data[l];
}
```

Task 3 - Identify Potential Parallelization

Can we parallelize following two loops using parallel for construct?

```
for (int i=1; i<size; i++) {    // for loop: i from 1 to (size-1)
    if (data[i-1] > 0)          // If the previous value is positive
        data[i] = (-1)*data[i]; // change the sign of this value
}                               // end for loop
```

```
for (int i=0; i<size; i++) {      // for loop: i from 0 to (size-1)
    data[i] = Math.sin(data[i]);  // calculate sin() of the value
}                                 // end for loop
```

https://quizizz.com/admin/quiz/62265bf968631b001e4acbde

Replace link with link to quiz