Parallele Programmierung FS25

Exercise Session 4

Jonas Wetzel

Plan für heute

- Organisation
- Nachbesprechung Exercise 3
- Theory Recap
- Intro Exercise 4
- Exam Questions
- Kahoot

Organisation

- Mein Name ist Jonas Wetzel
- Meine Website (Materialien und Inhalt der Übungen): n.ethz.ch/~jwetzel
- Meine Email: jwetzel@ethz.ch
- Discord: @jonas.too

Organisation

- Mein Name ist Jonas Wetzel
- Meine Website (Materialien und Inhalt der Übungen): n.ethz.ch/~jwetzel
- Meine Email: jwetzel@ethz.ch
- Discord: @jonas.too
- Feedback zur Session: https://forms.gle/qiDnqkfSP2NUQGvc9

Organisation

- Feedback zur Session: https://forms.gle/qiDnqkfSP2NUQGvc9
- Falls ihr Feedback möchtet kommt bitte zu mir

Organisation

- Wo sind wir jetzt?

Date	Title
Feb 17	Introduction & Course Overview
Feb 18	Java Recap and JVM Overview
Feb 24	Introduction to Threads and Synchronization (Part I)
Feb 25	Introduction to Threads and Synchronization (Part II)
Mar 3	Introduction to Threads and Synchronization (Part III)
Mar 4	Parallel Architectures: Parallelism on the Hardware Level
Mar 10	Basic Concepts in Parallelism
Mar 11	Divide & Conquer and Executor Service
Mar 17	DAG and ForkJoin Framework
Mar 18	Parallel Algorithms (Part I)
Mar 24	Parallel Algorithms (Part II)
Mar 25	Shared Memory Concurrency, Locks and Data Races
Mar 31	Virtual Threads
Apr 01	Exam Preparation (First Half)

Plan für heute

- Organisation
- **Nachbesprechung Exercise 3**
- Theory Recap
- Intro Exercise 4
- Exam Questions
- Kahoot

Counter

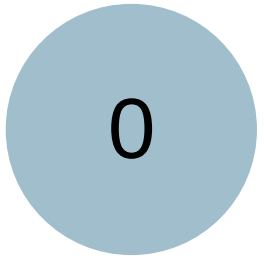
Let’s count number of times a given event occurs

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

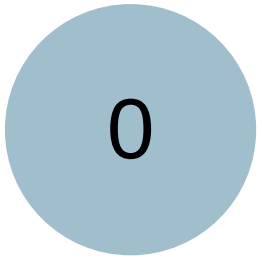
```
// background threads
for (int i = 0; i < numIterations; i++) {
  // perform some work

  counter.increment();
}

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

10 iterations each

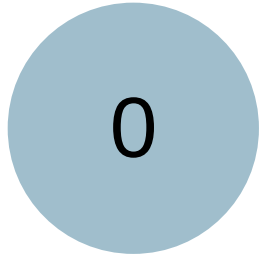
Thread 1



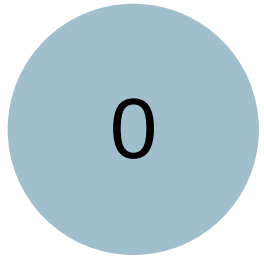
Counter



Thread 2



Thread 3

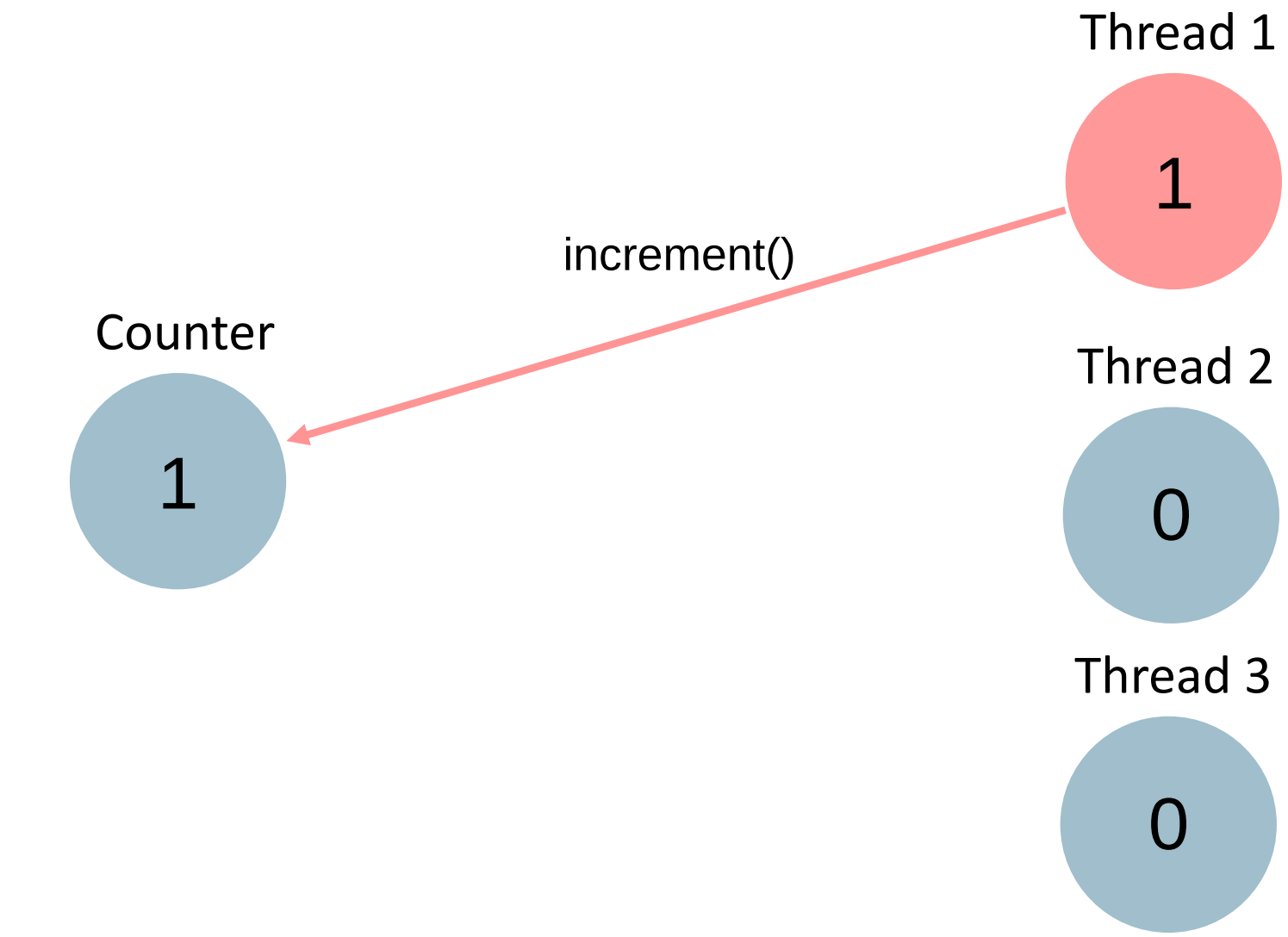


value of the
shared Counter

number of times
`increment()` is called

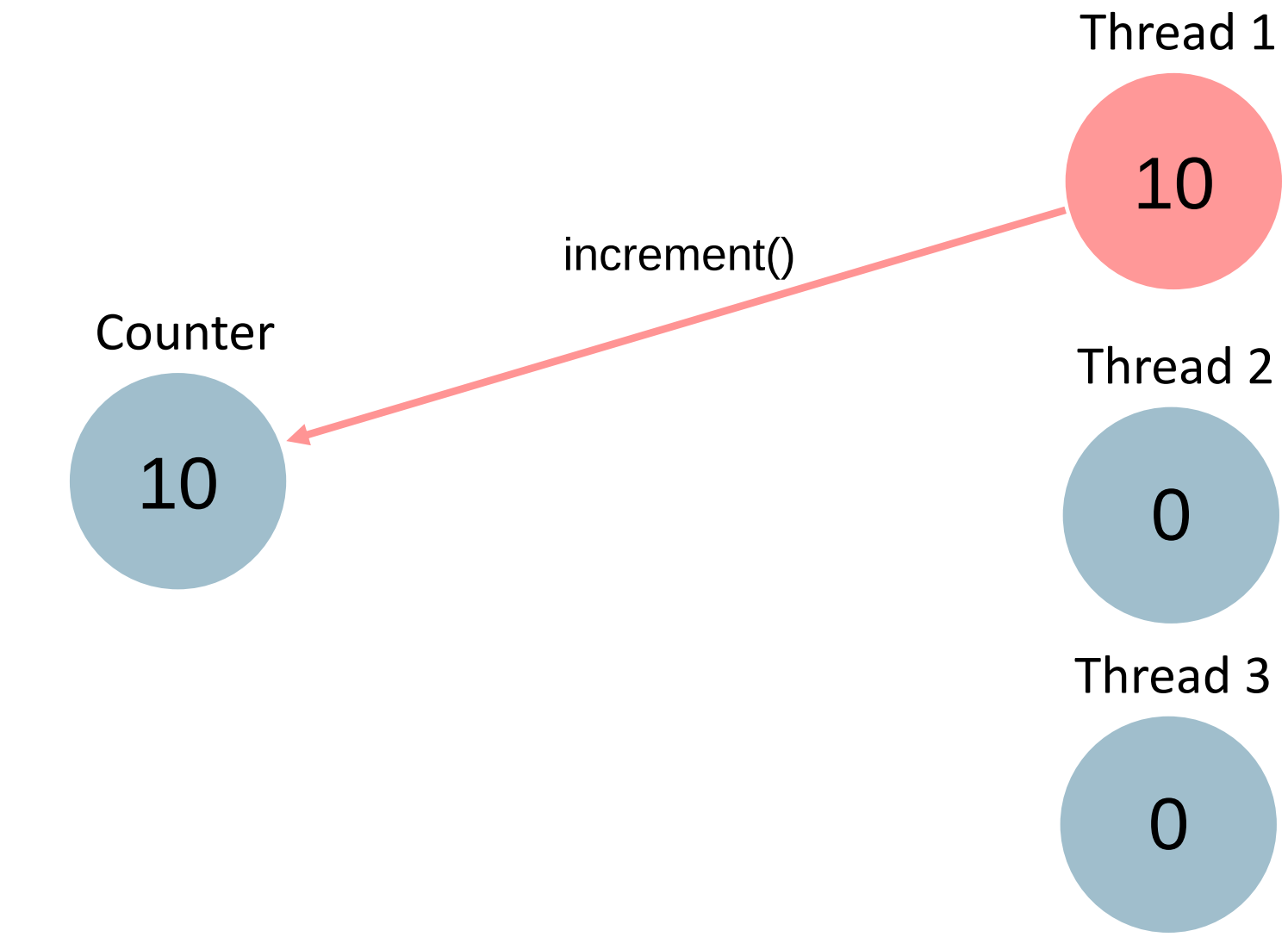

Thread 1
0
Counter
0
Thread 2
0
Thread 3
0
value of the
shared Counter
number of times
increment() is called


Thread 1
1
Counter
0
Thread 2
0
Thread 3
0
value of the
shared Counter
number of times
increment() is called



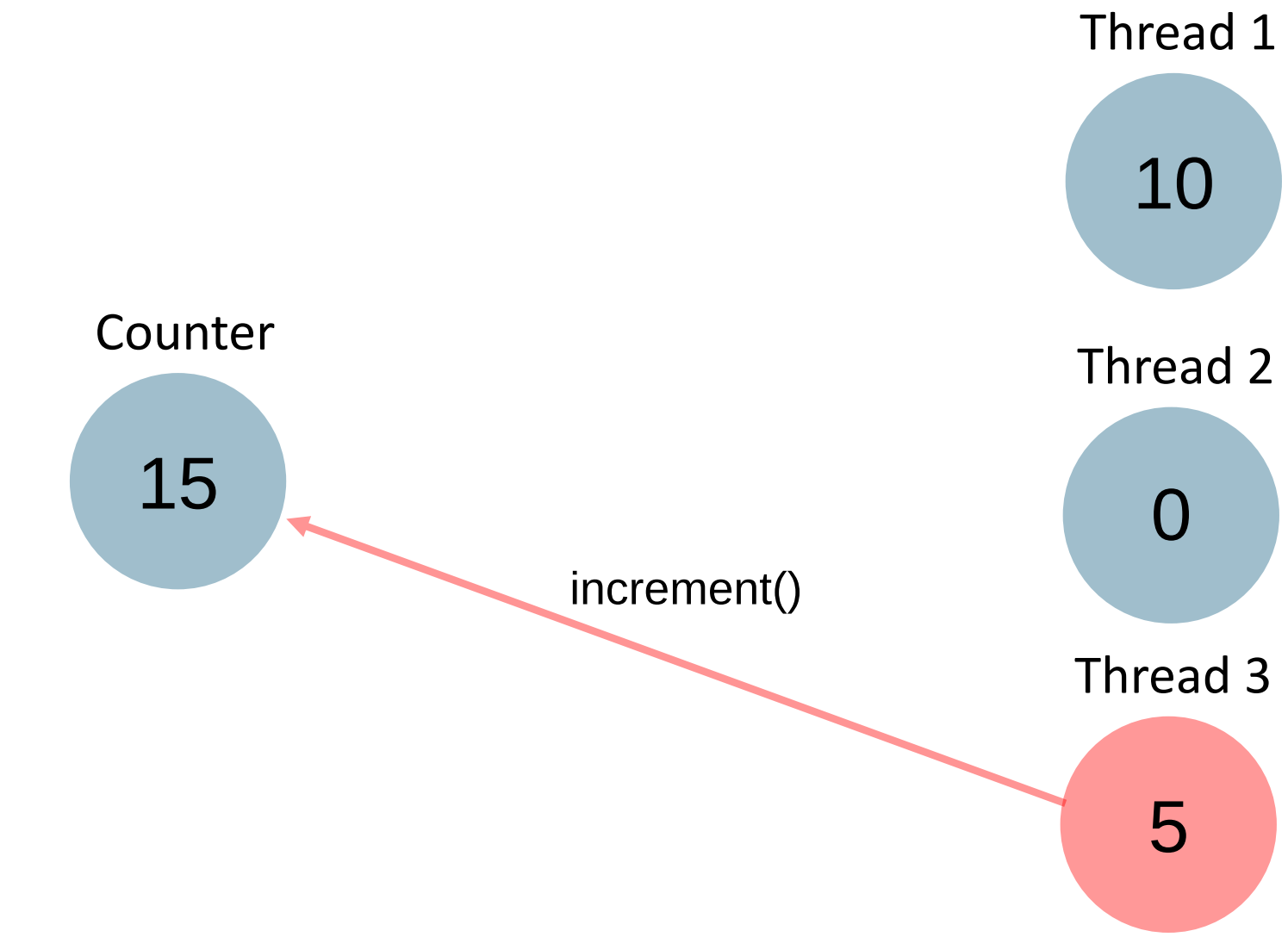
value of the
shared Counter

number of times
`increment()` is called



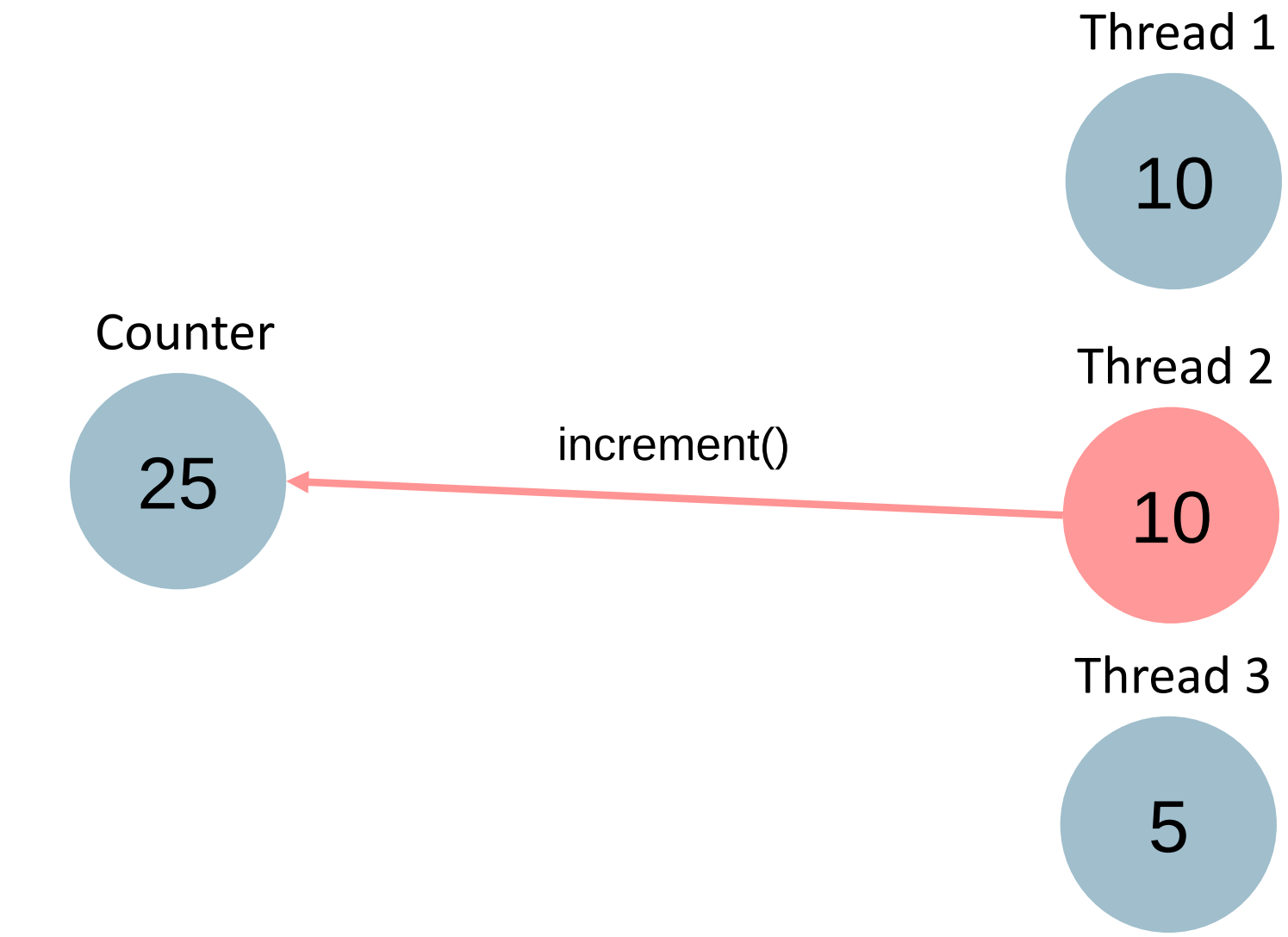
value of the
shared Counter

number of times
`increment()` is called



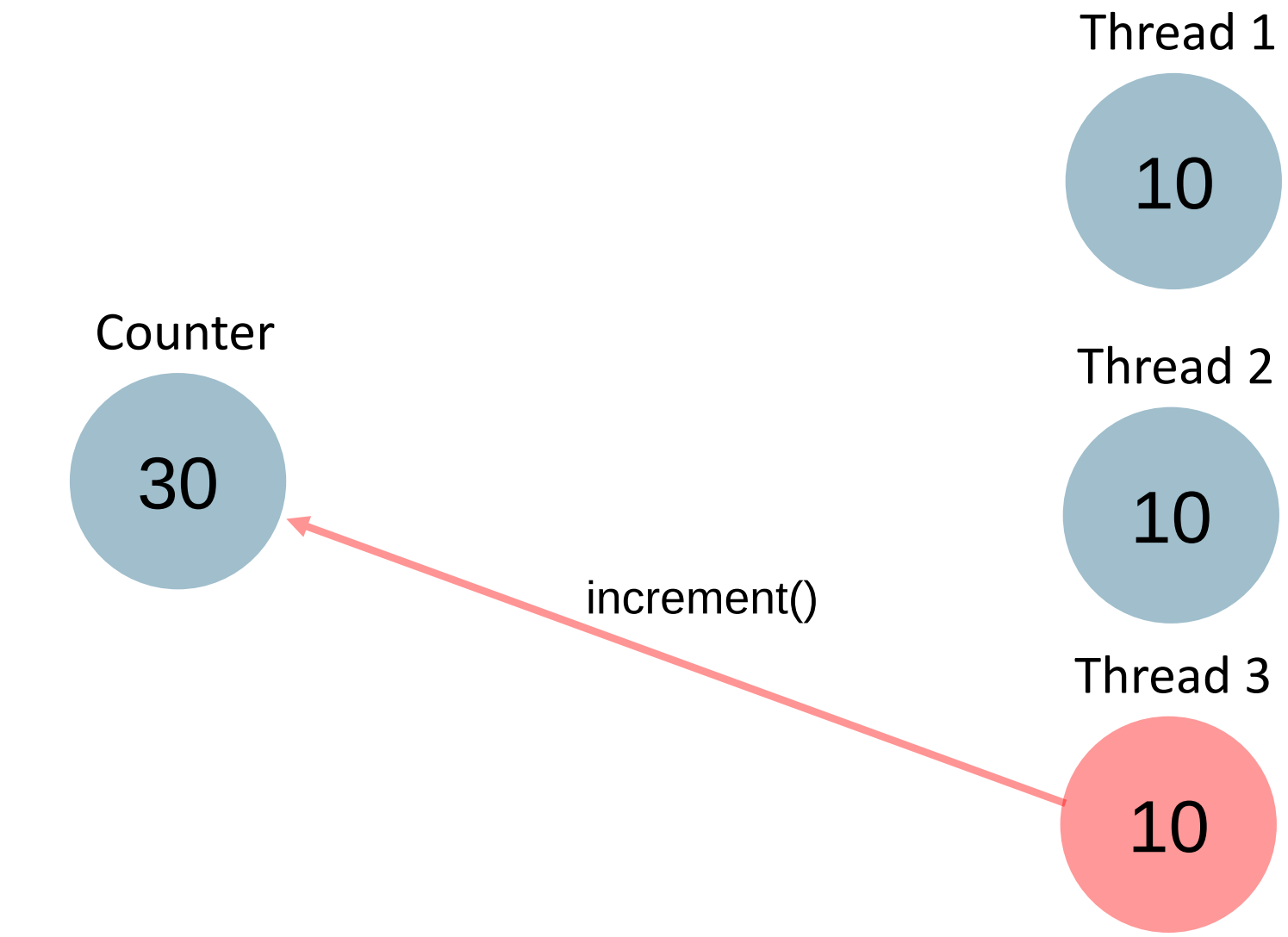
value of the
shared Counter

number of times
`increment()` is called



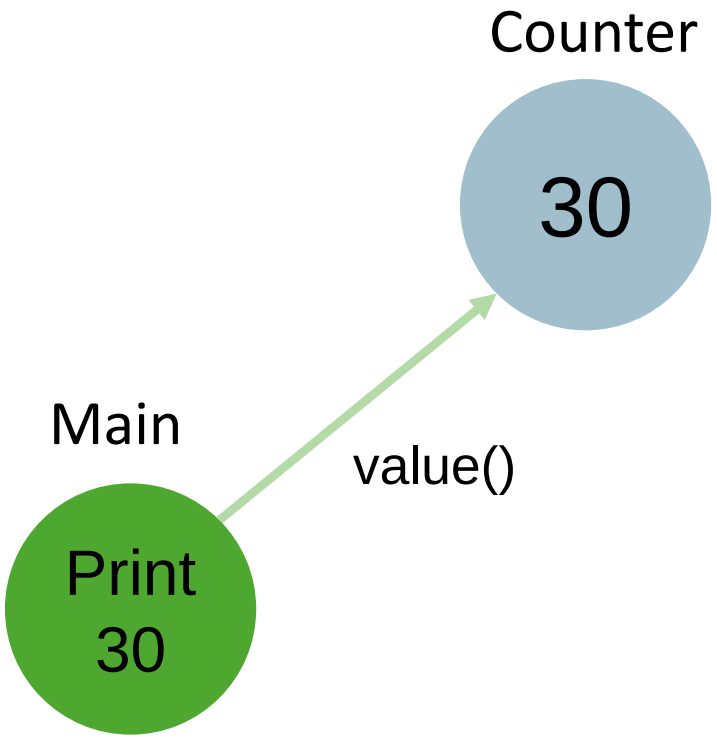
value of the shared Counter

number of times `increment()` is called



value of the
shared Counter

number of times
`increment()` is called



read the Counter value

value of the shared Counter

number of times `increment()` is called

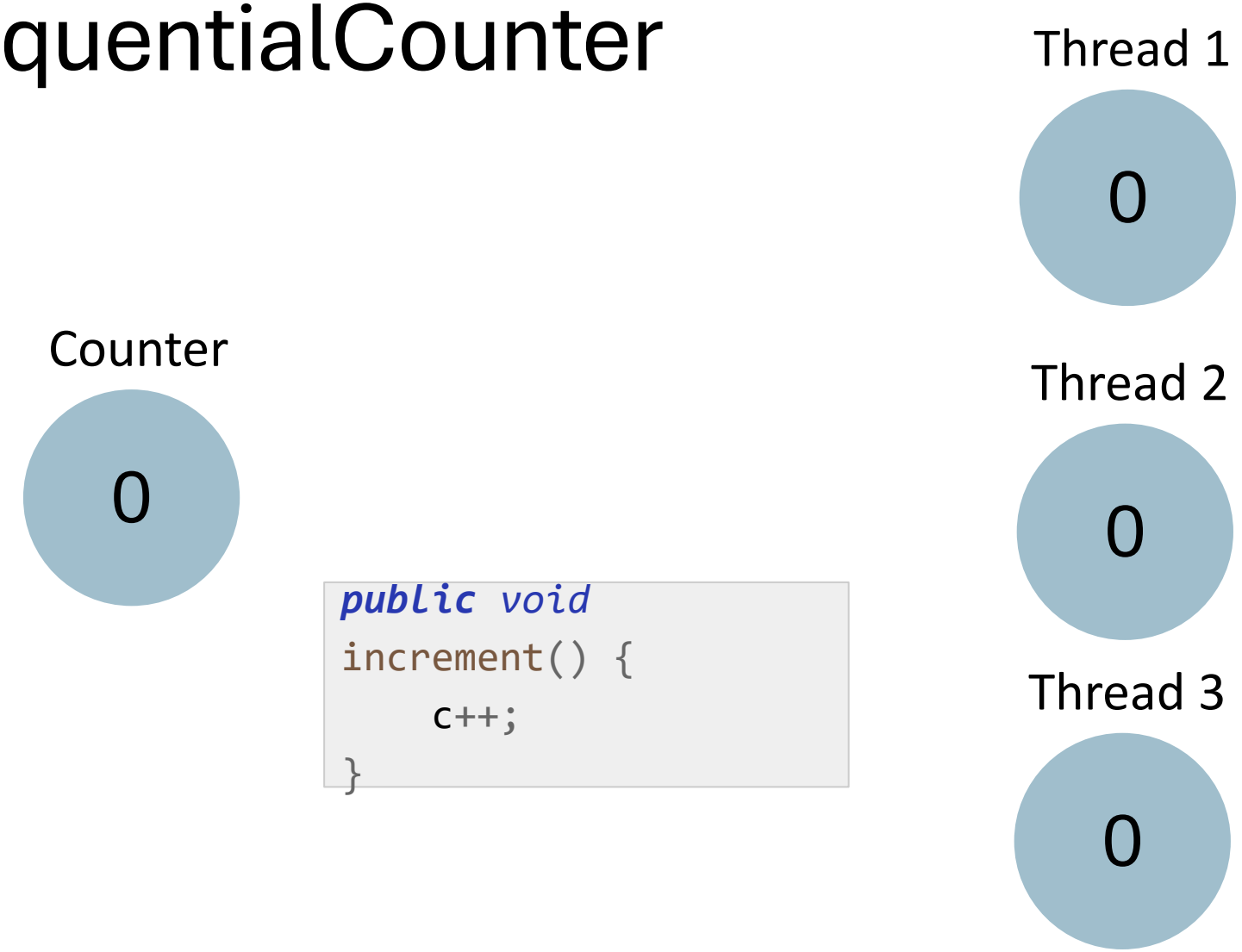
Task A: SequentialCounter

```
public class SequentialCounter implements Counter {


    public void increment() {
        ??
    }

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

Task A: SequentialCounter

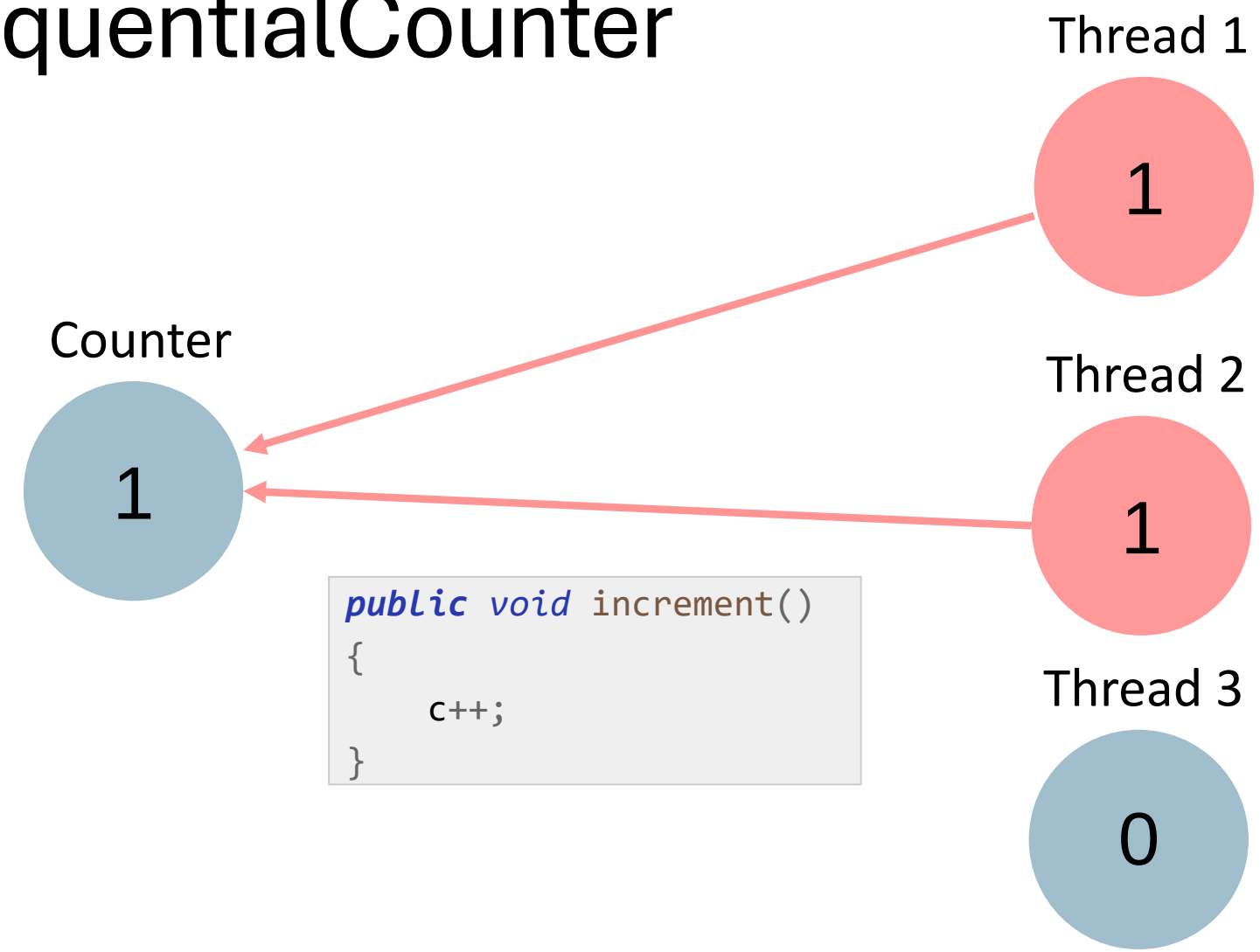
```
public class SequentialCounter implements Counter {
    private int c = 0;

    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?

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

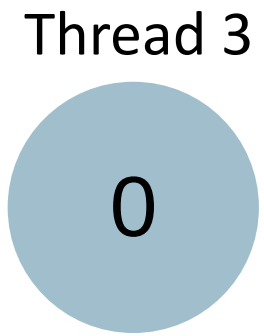
Task A: SequentialCounter

conflicting access!

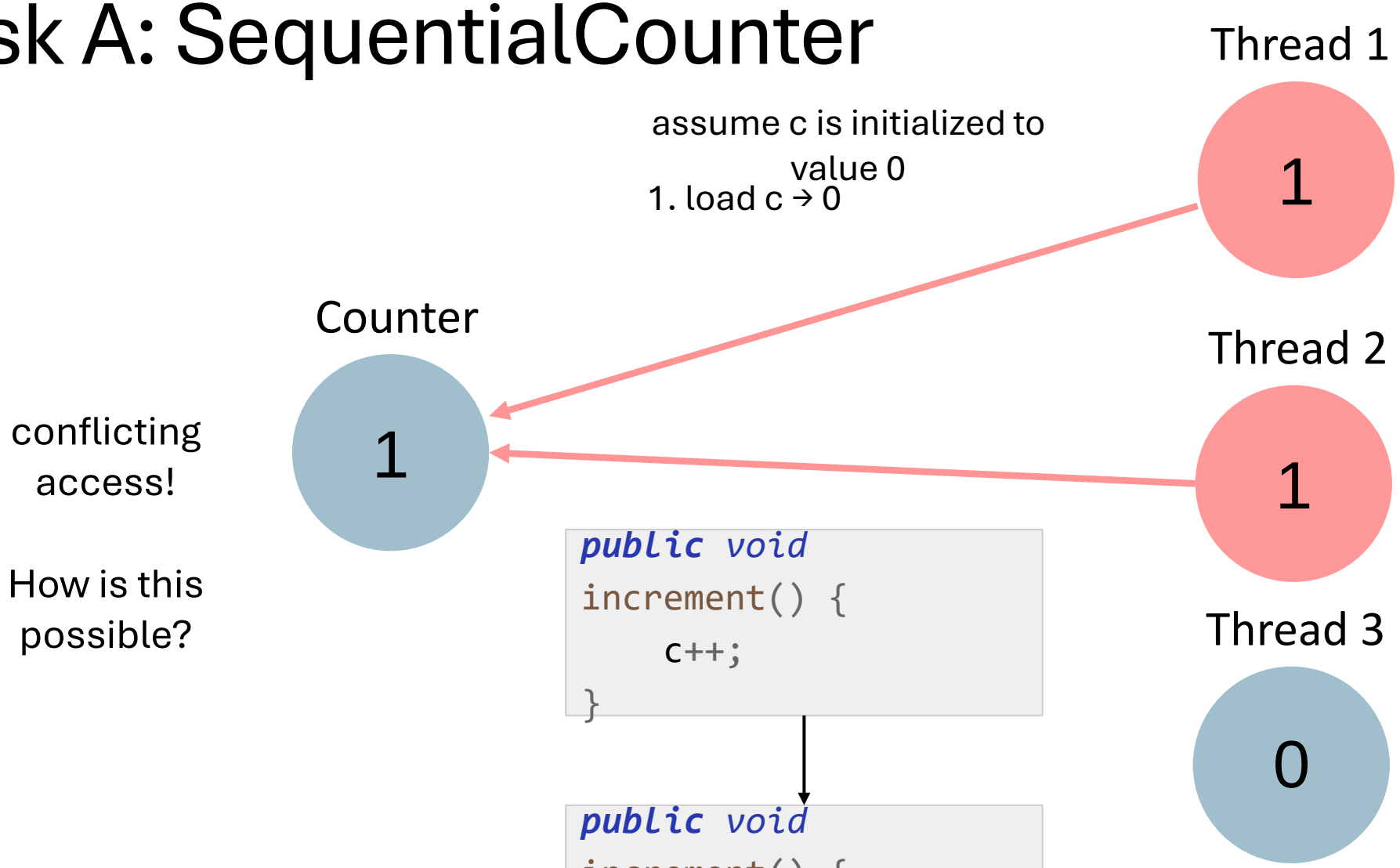
How is this possible?

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

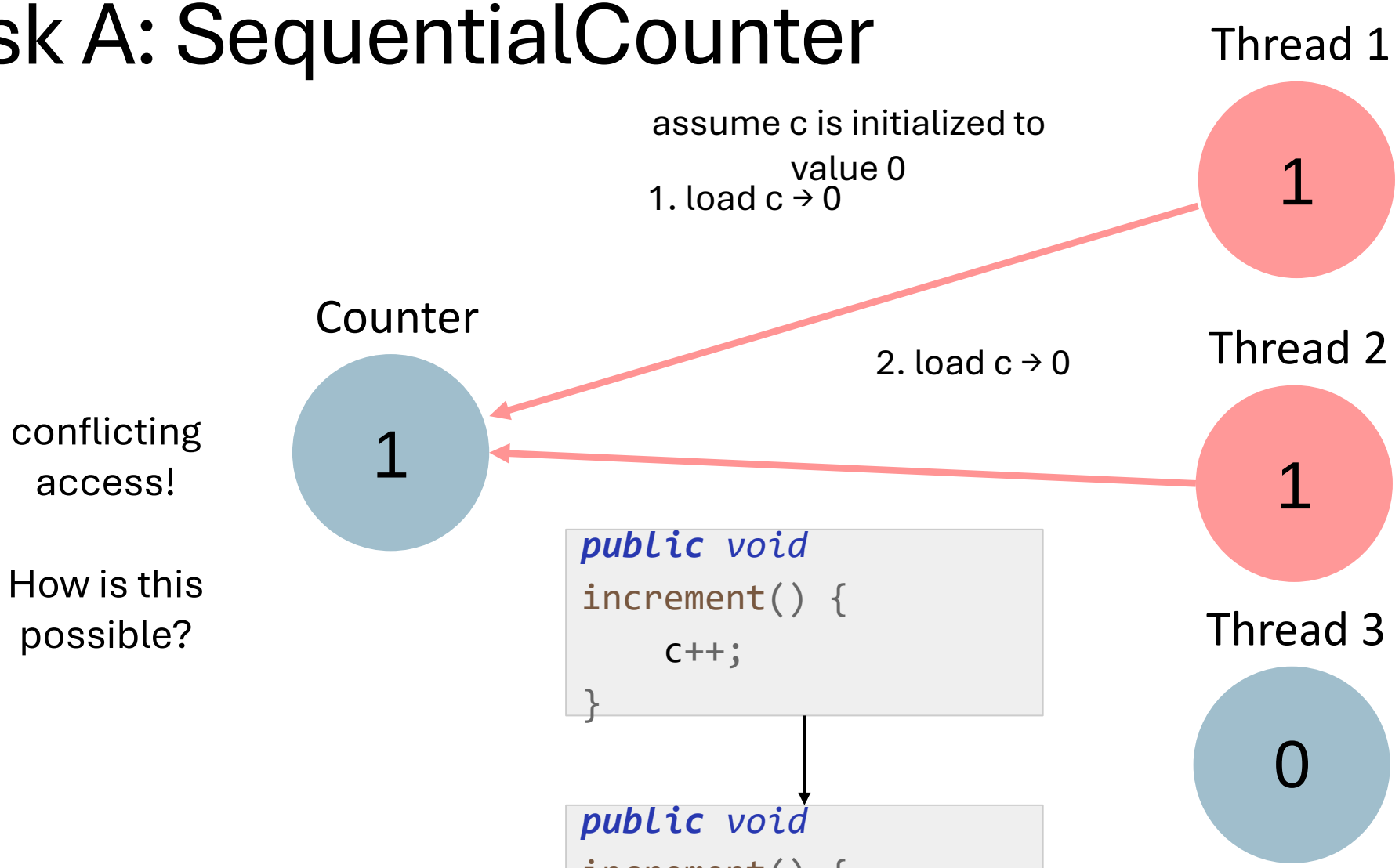
```
public void
increment() {
    c = c + 1;
}
```



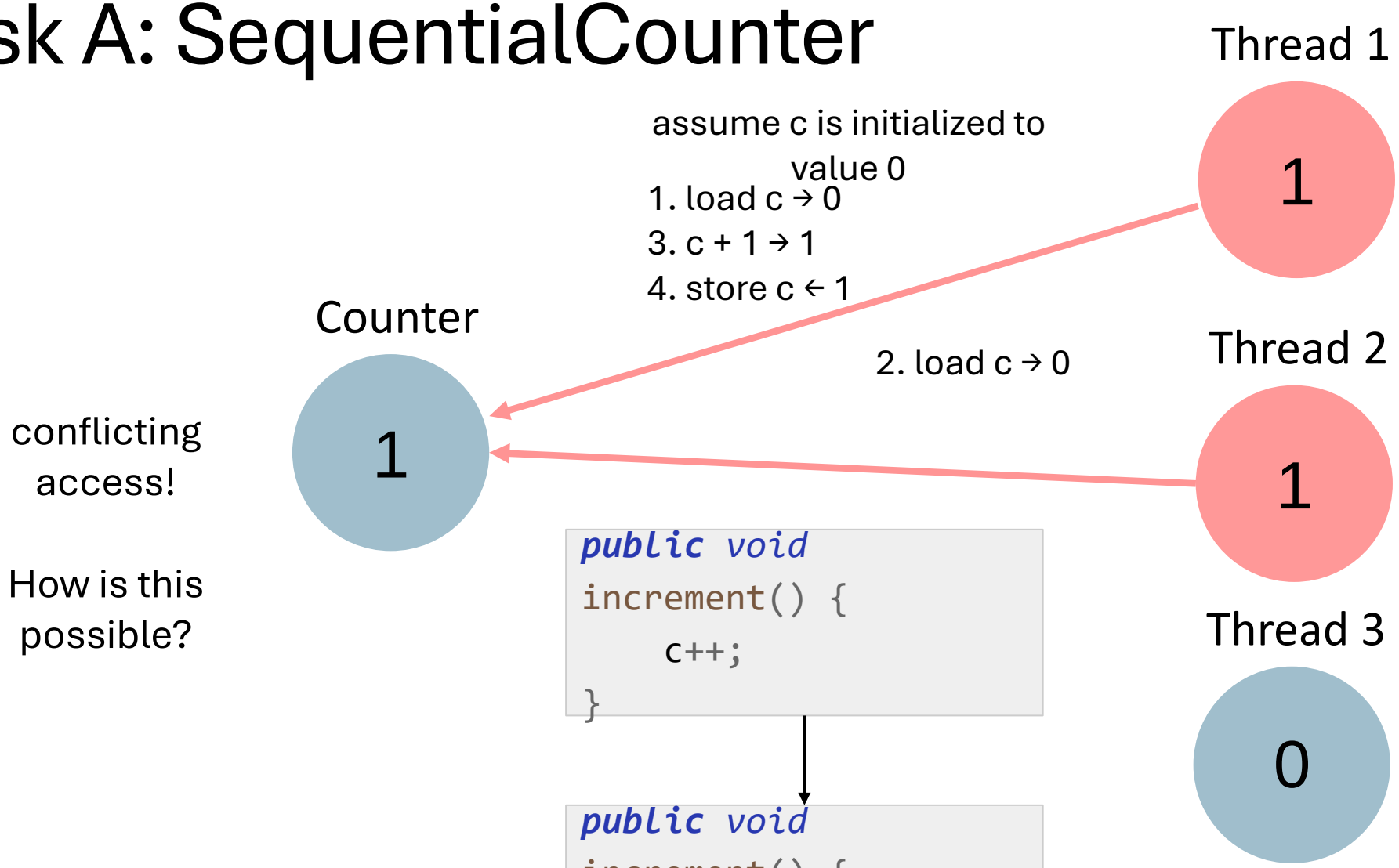
Task A: SequentialCounter



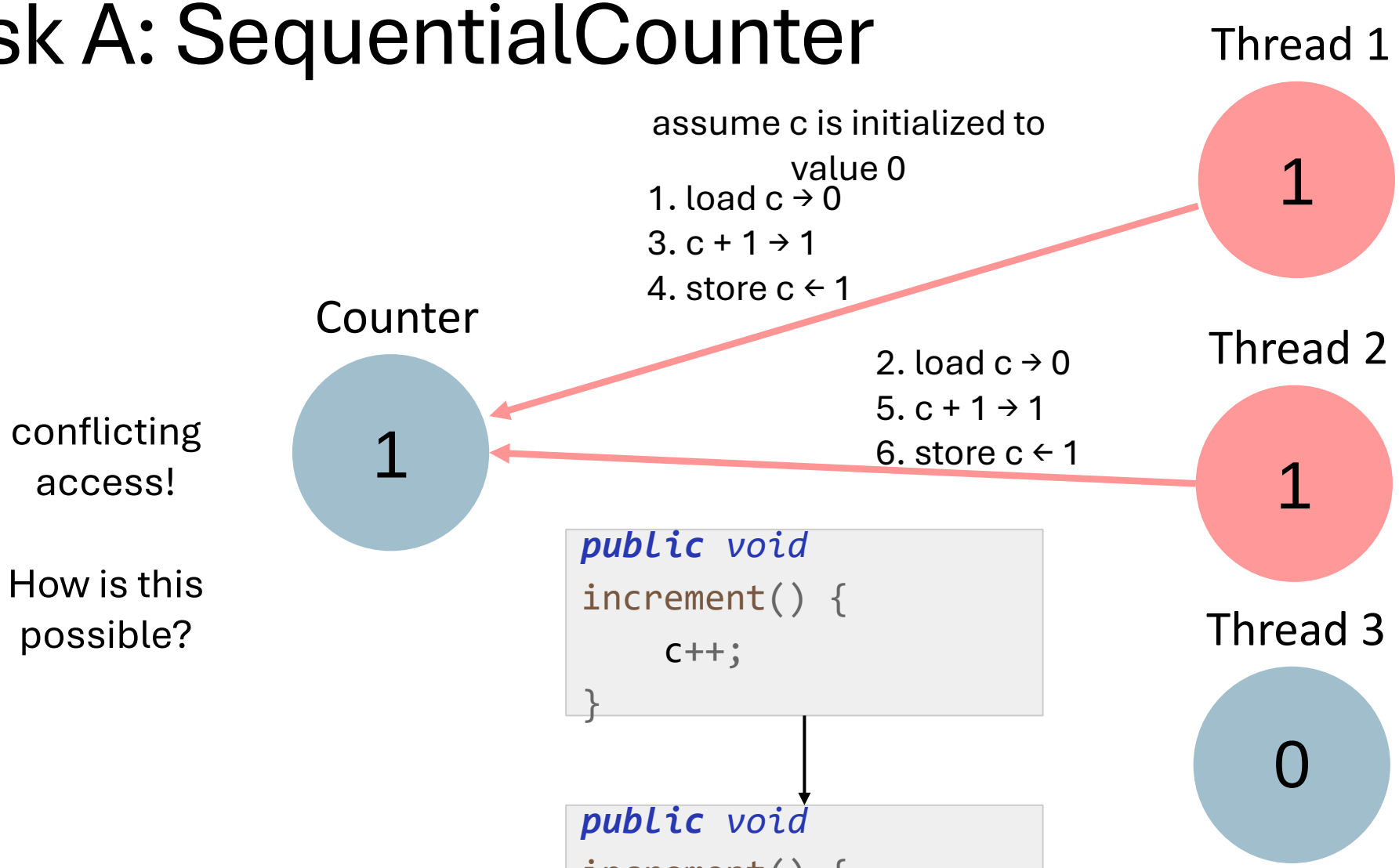
Task A: SequentialCounter



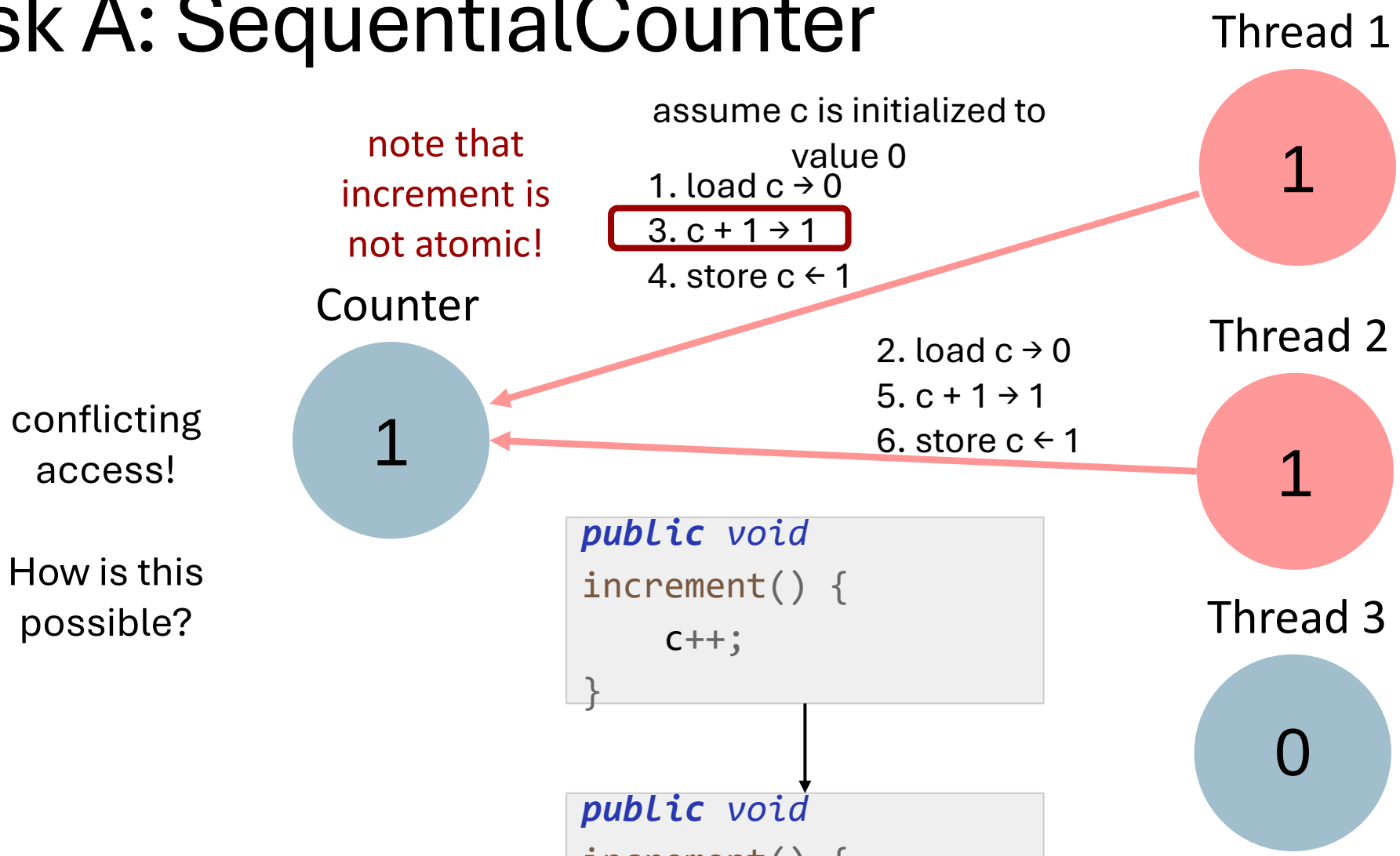
Task A: SequentialCounter



Task A: SequentialCounter



Task A: SequentialCounter



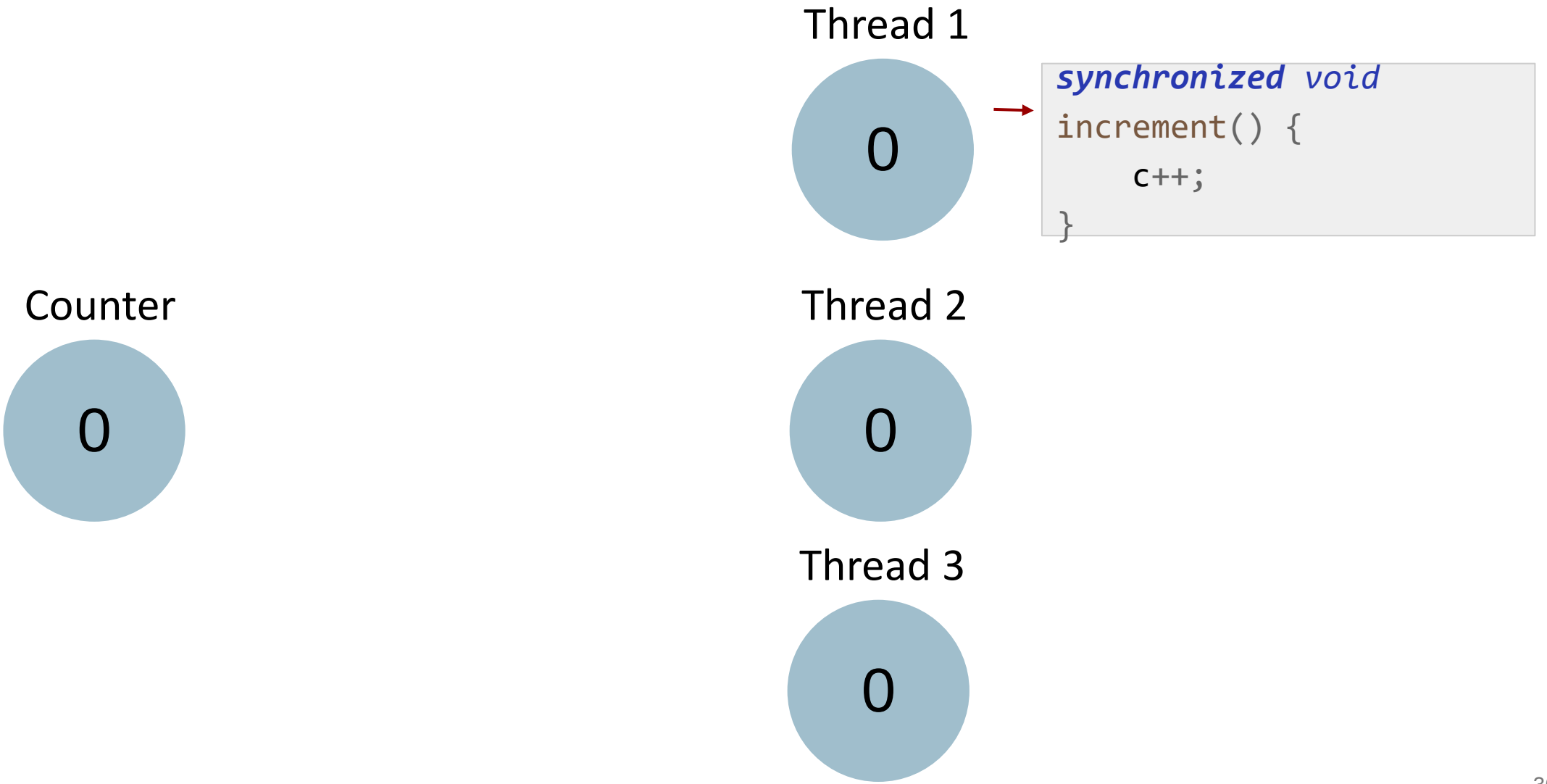
Task B: SynchronizedCounter

```
public class SynchronizedCounter implements Counter {


    public void increment() {
        ??
    }

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

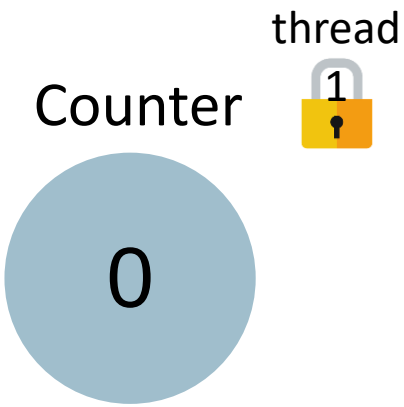
Task B: SynchronizedCounter

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

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

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

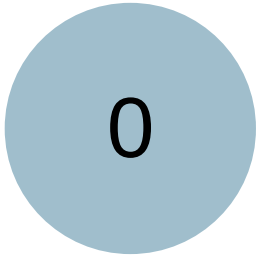
Task B: SynchronizedCounter



Task B: SynchronizedCounter

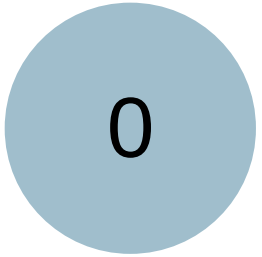


Thread 1



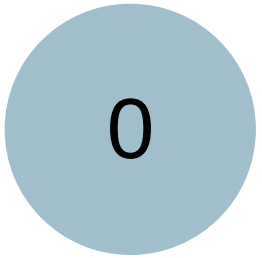

```
synchronized void
increment() {
    c++;
}
```

Thread 2



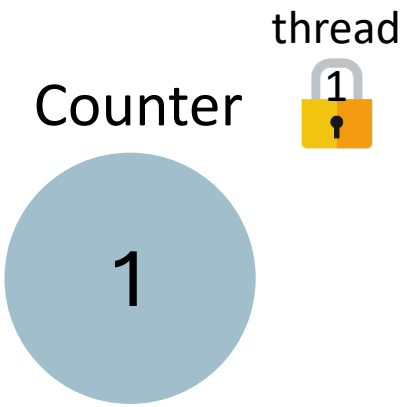

```
synchronized void
increment() {
    c++;
}
```

Thread 3



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



Thread 1

1

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

Thread 2

0

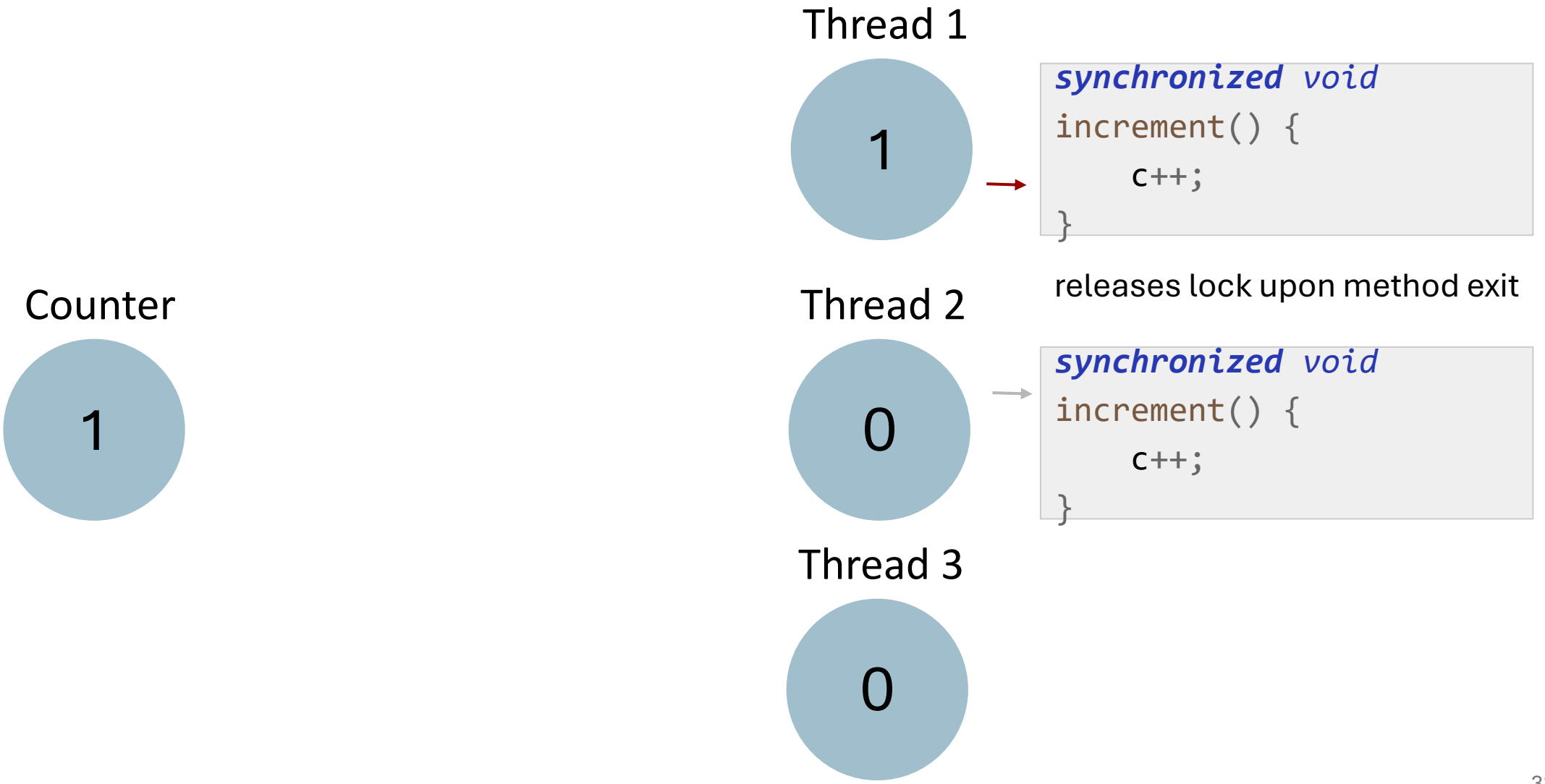
```
synchronized void
increment() {
    c++;
}
```

Thread 3

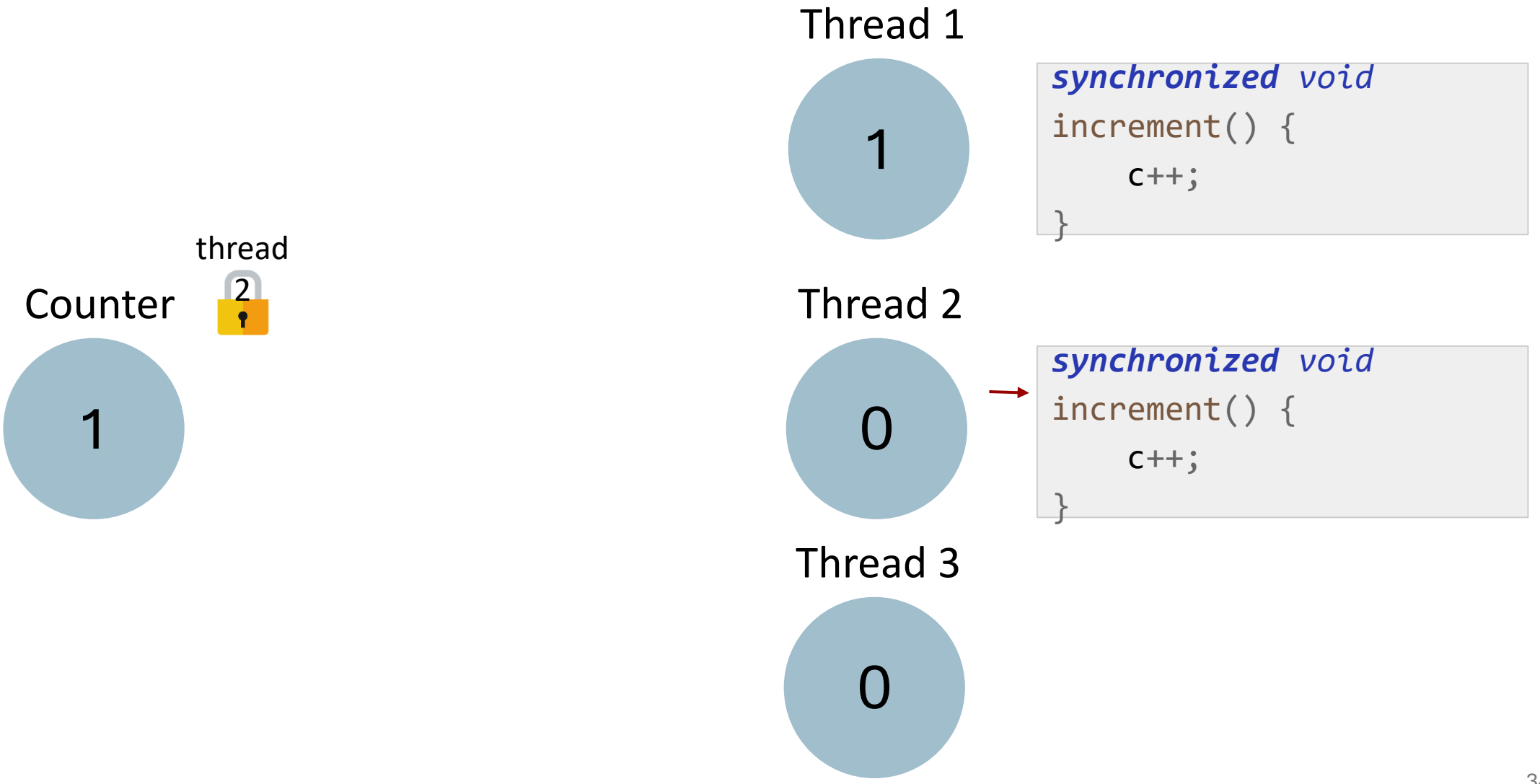
0

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



Task B: SynchronizedCounter



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

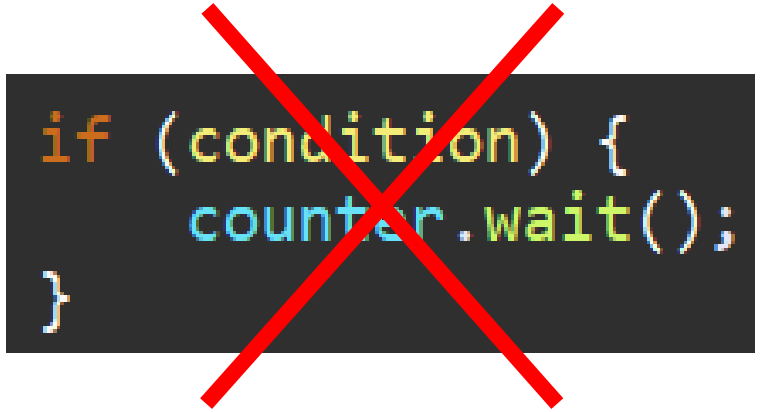
`notify`: Thread must own object's lock to call `notify`

`notify`: Wake one (arbitrary) thread from object's "waiting list"

`notifyAll`: Wake all threads

Wait and Notify Recap

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



Spurious wake-ups and notifyAll()

→ `wait` has to be in a `while` loop

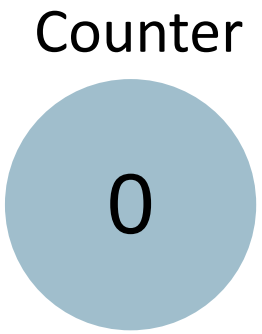
Spurious Wake-ups

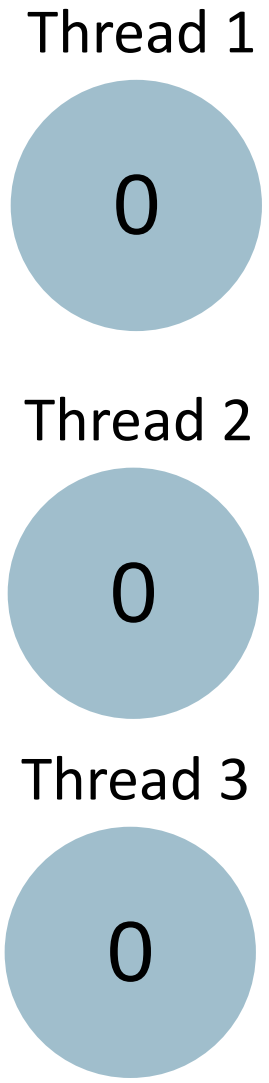
- A waiting thread is woken up without being **explicitly** notified through `notify` or `notifyAll`.

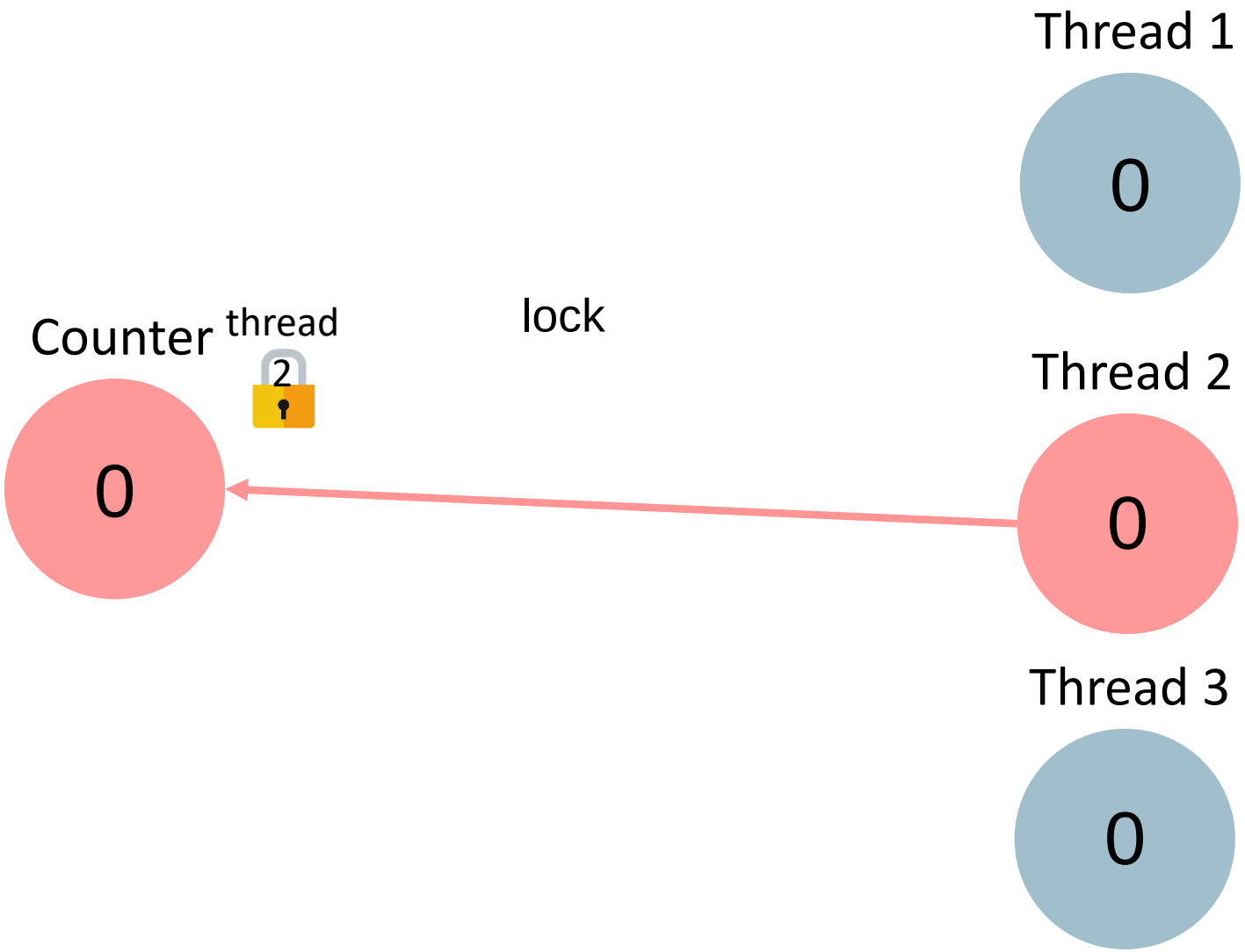
- Reasons: **various**
  - JVM Implementation: one thread is “nudged for housekeeping”
  - Performance (OS dependent): to avoid unnecessary context switches, to rebalance CPU load

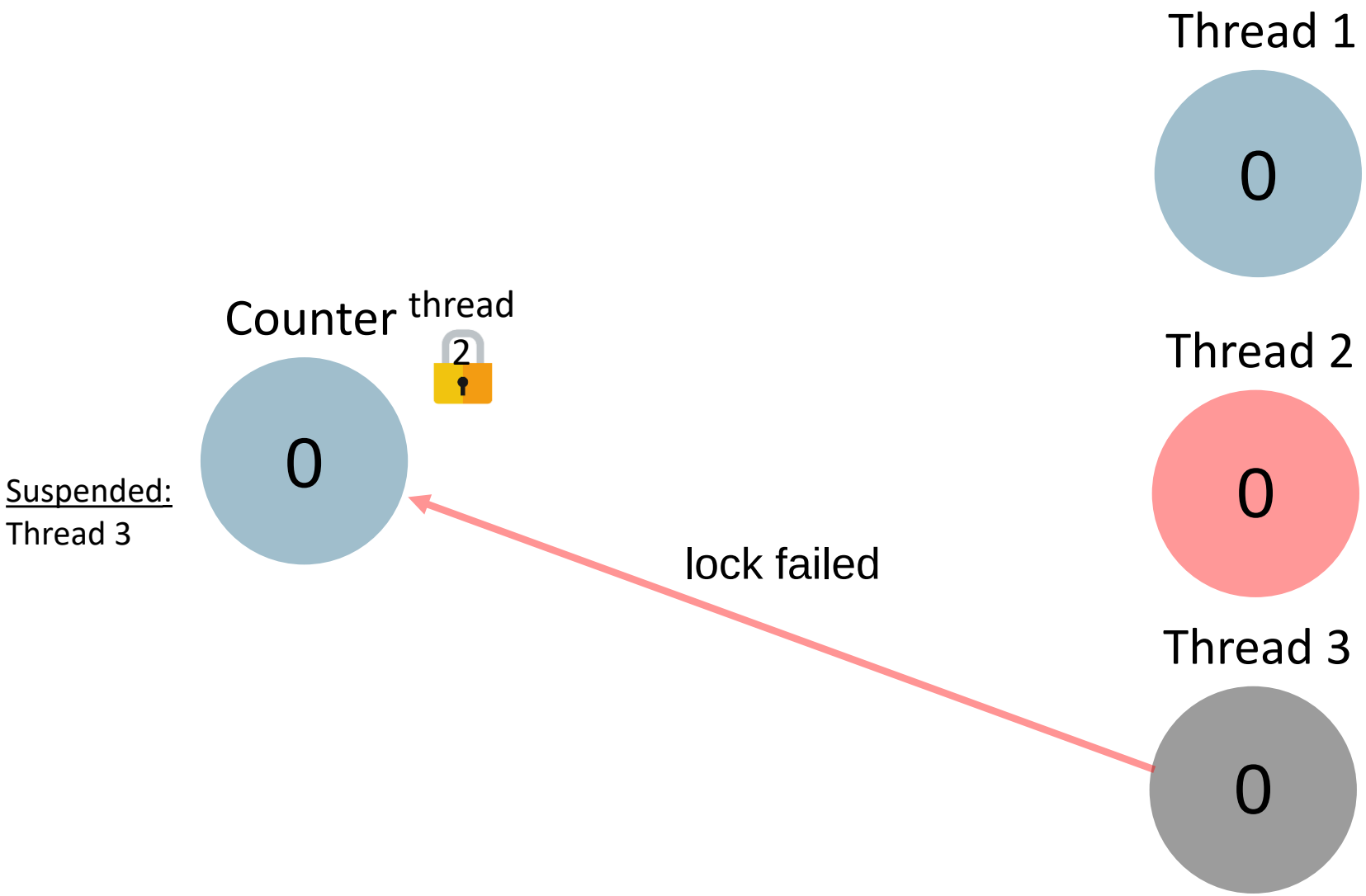
- Java does not prevent them. It expects you to handle them.

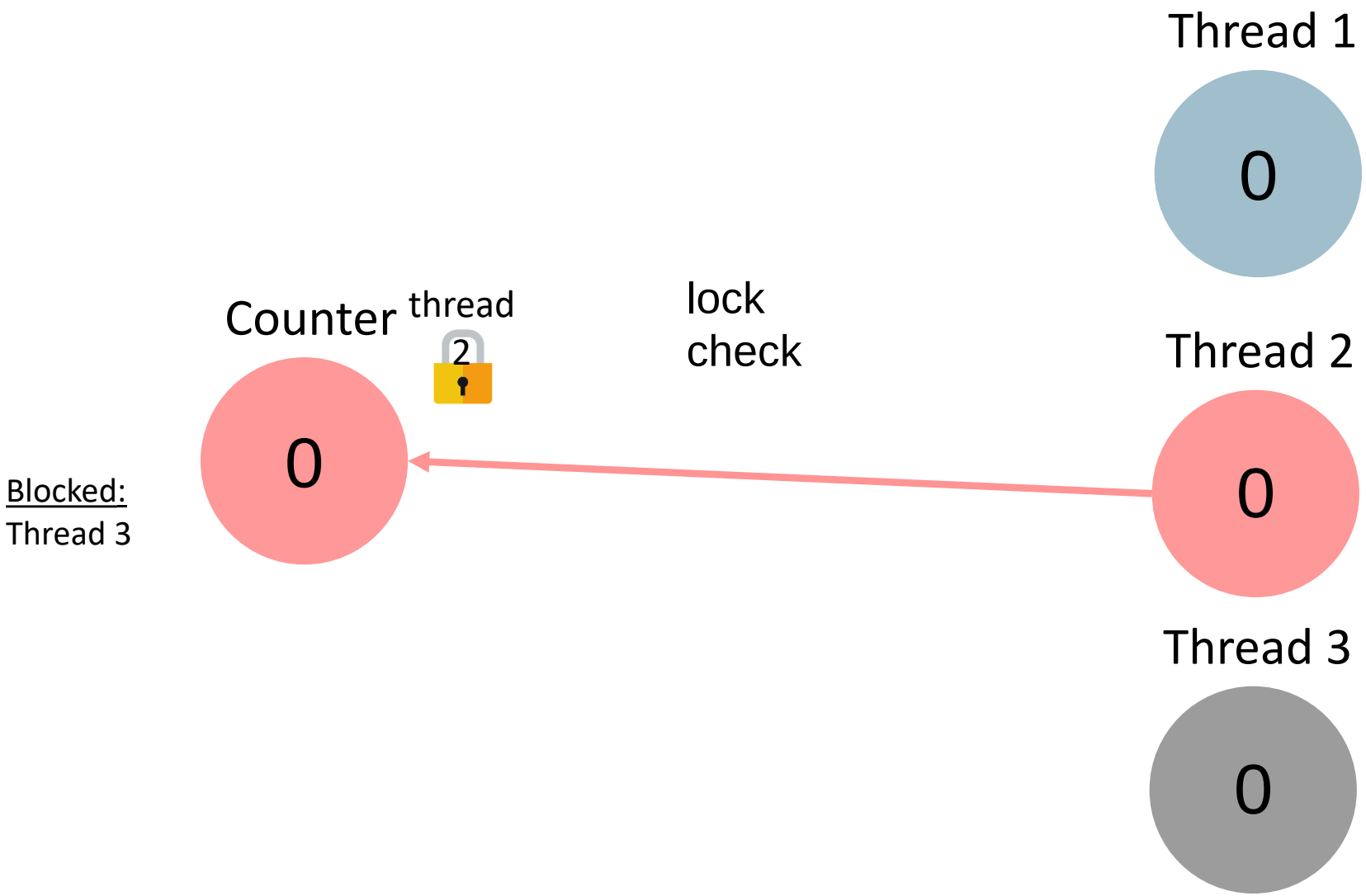
Thread 1 must increment first!

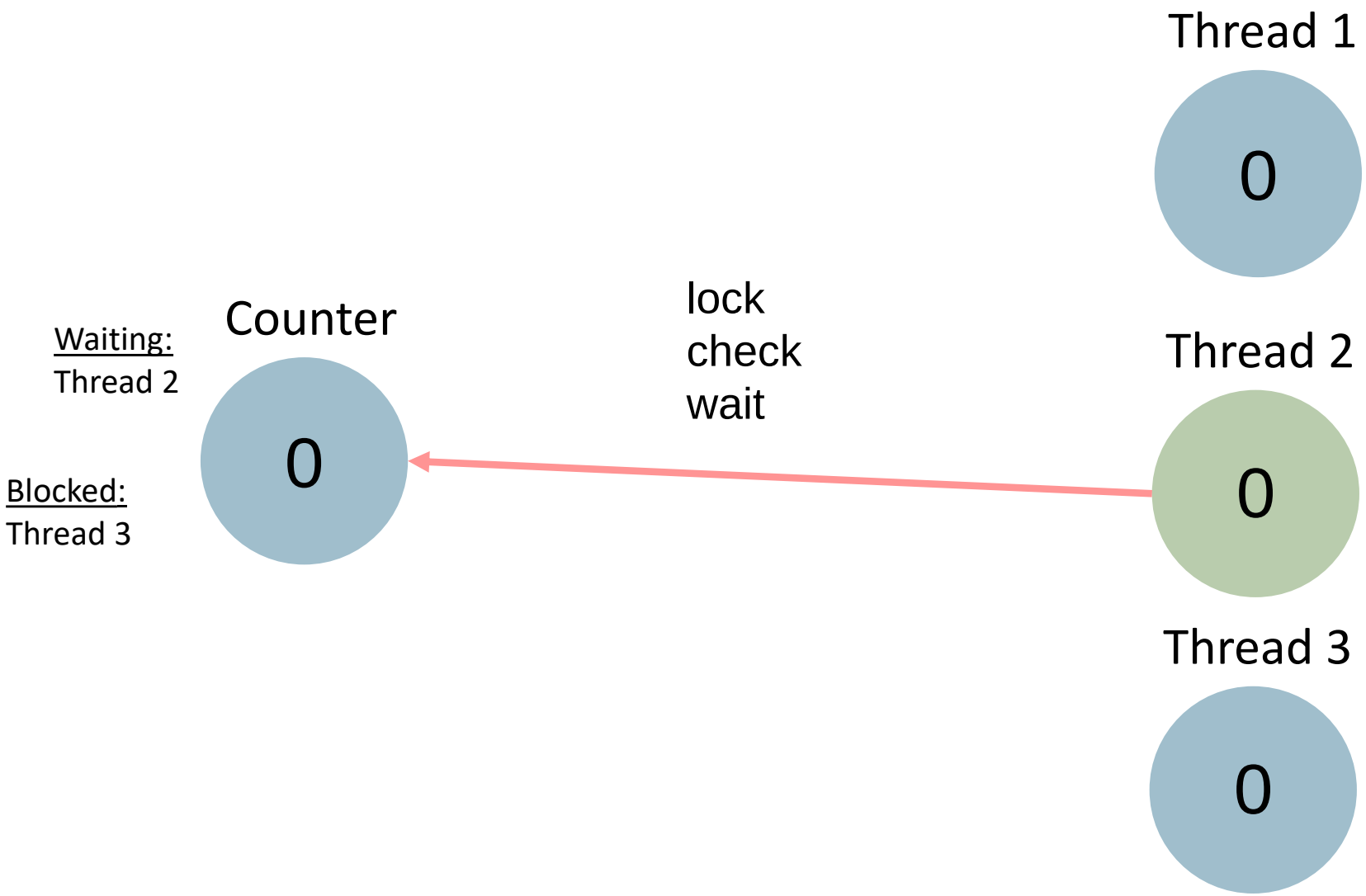


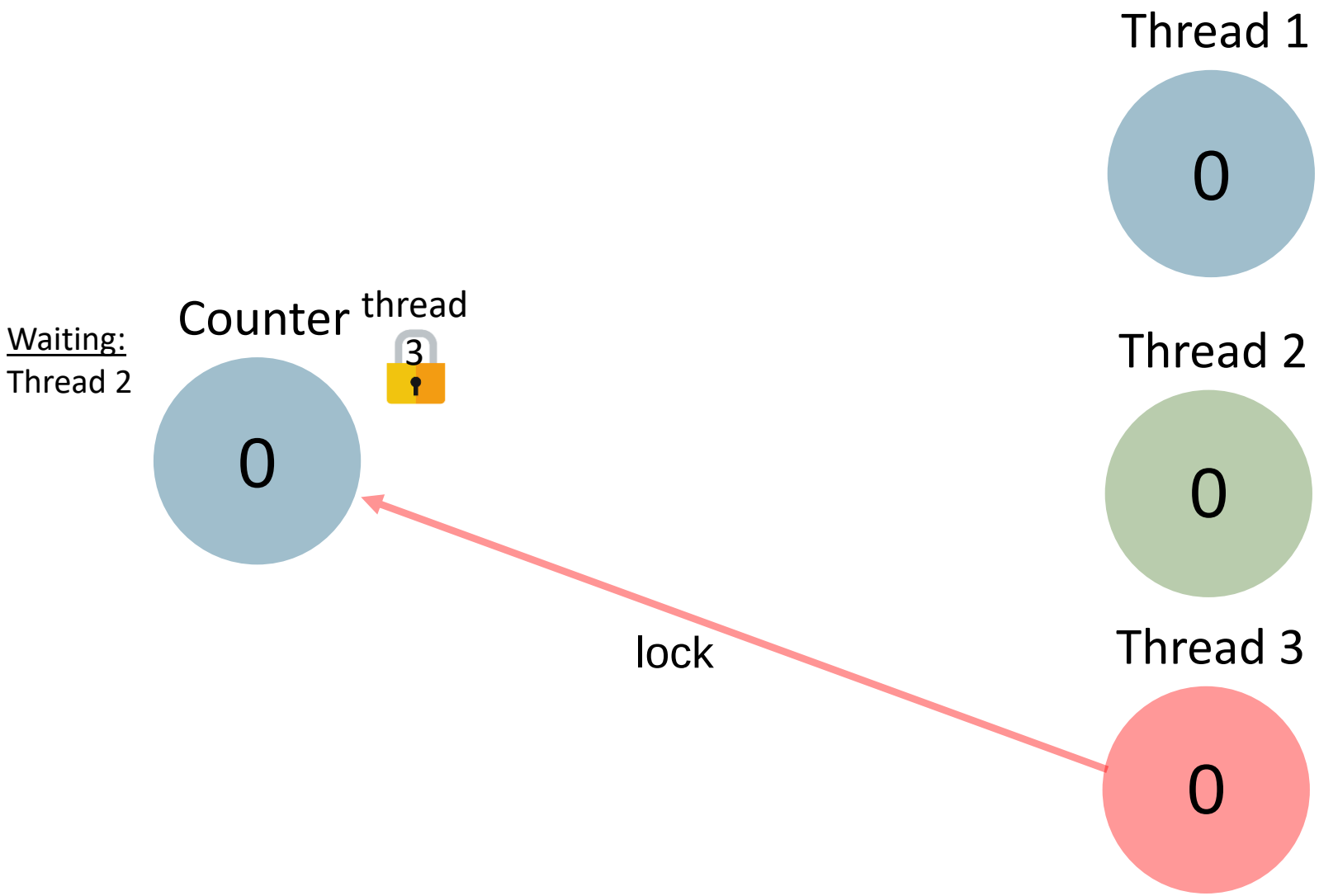


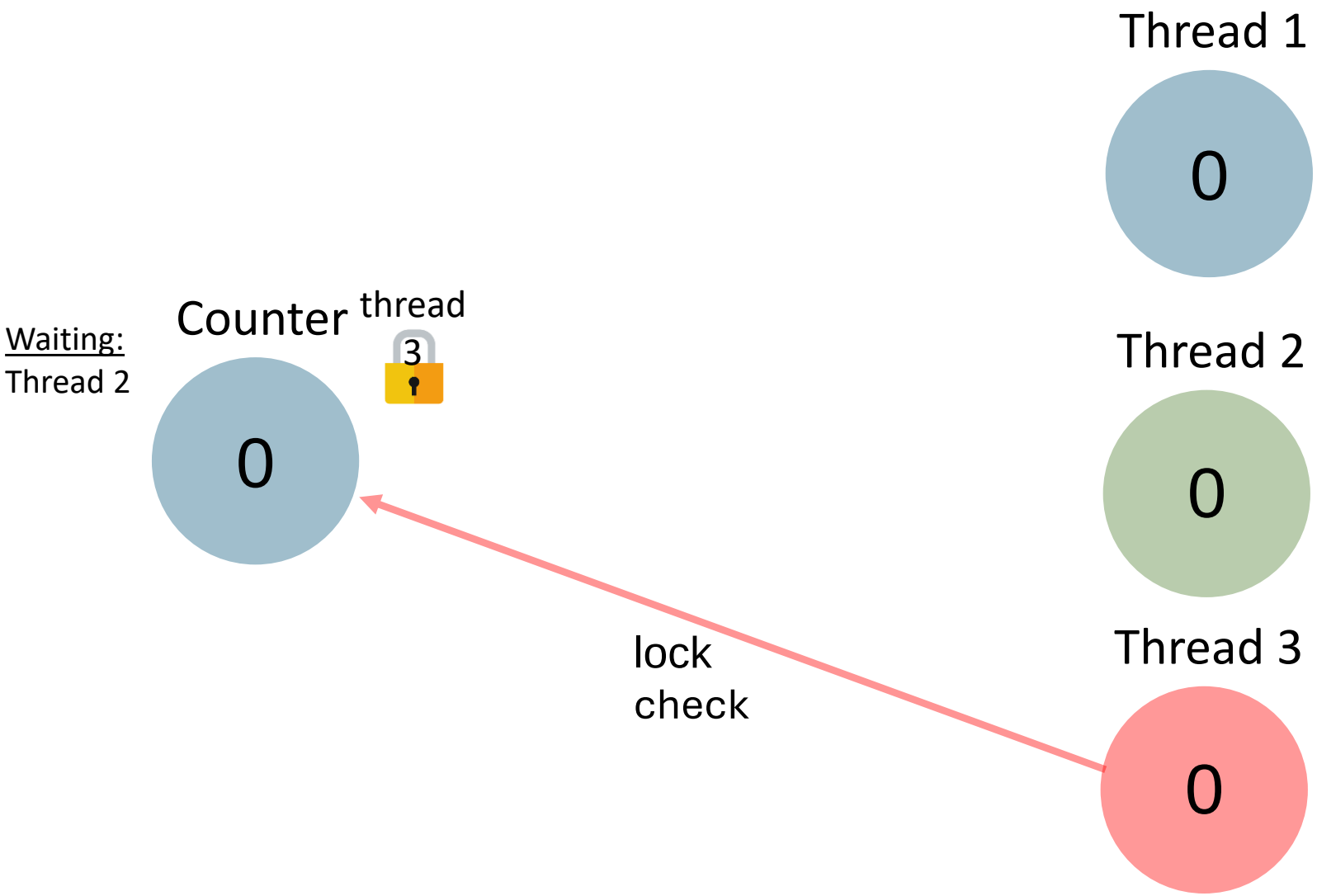

Thread 1
0
Counter
thread
2
lock
Thread 2
0
0
Thread 3
0

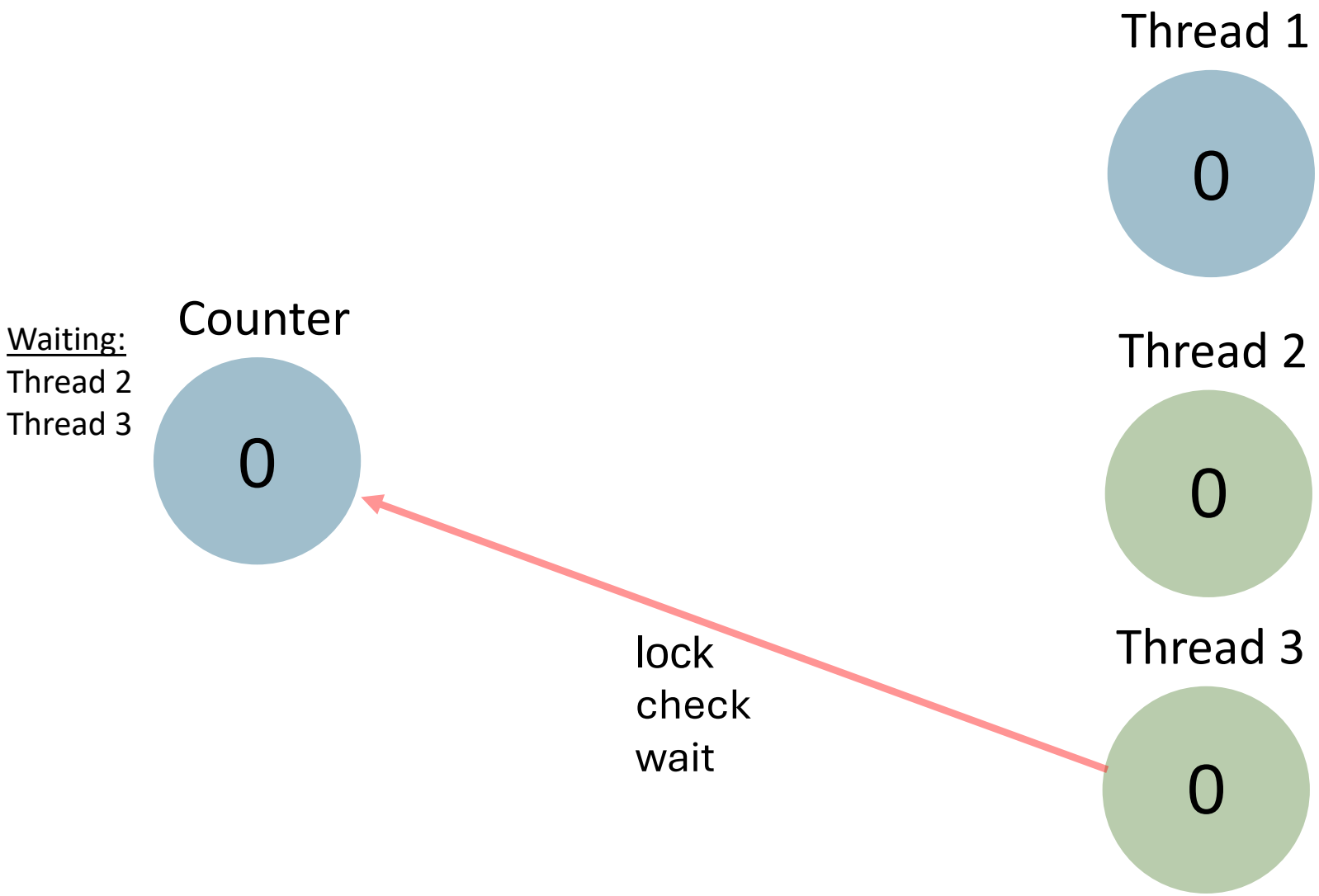

Thread 1
0
Counter
thread
2
0
Thread 2
0
Suspended:
Thread 3
lock failed
Thread 3
0

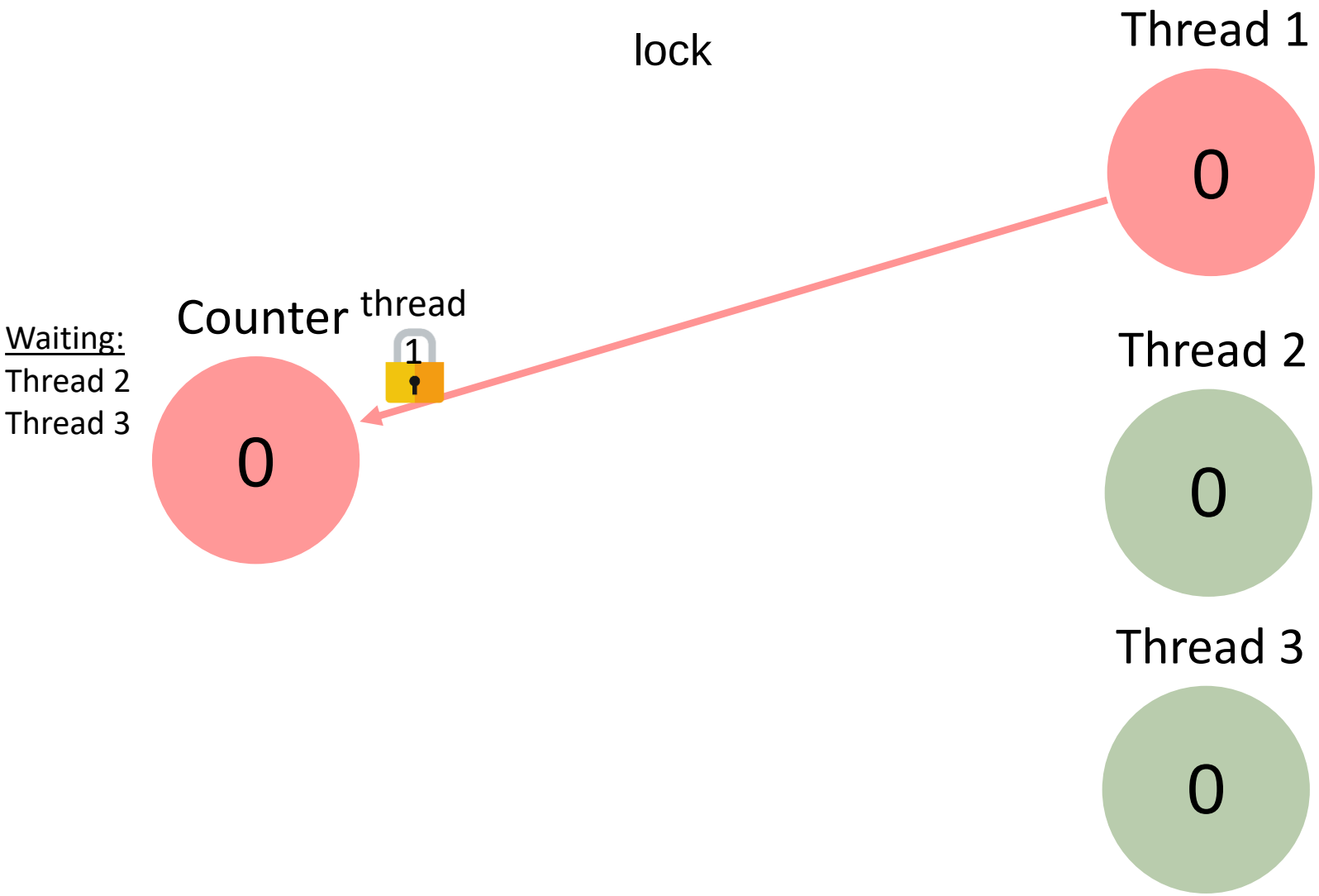

Thread 1
0
Counter
thread
2
lock
check
Thread 2
0
0
Blocked:
Thread 3
Thread 3
0

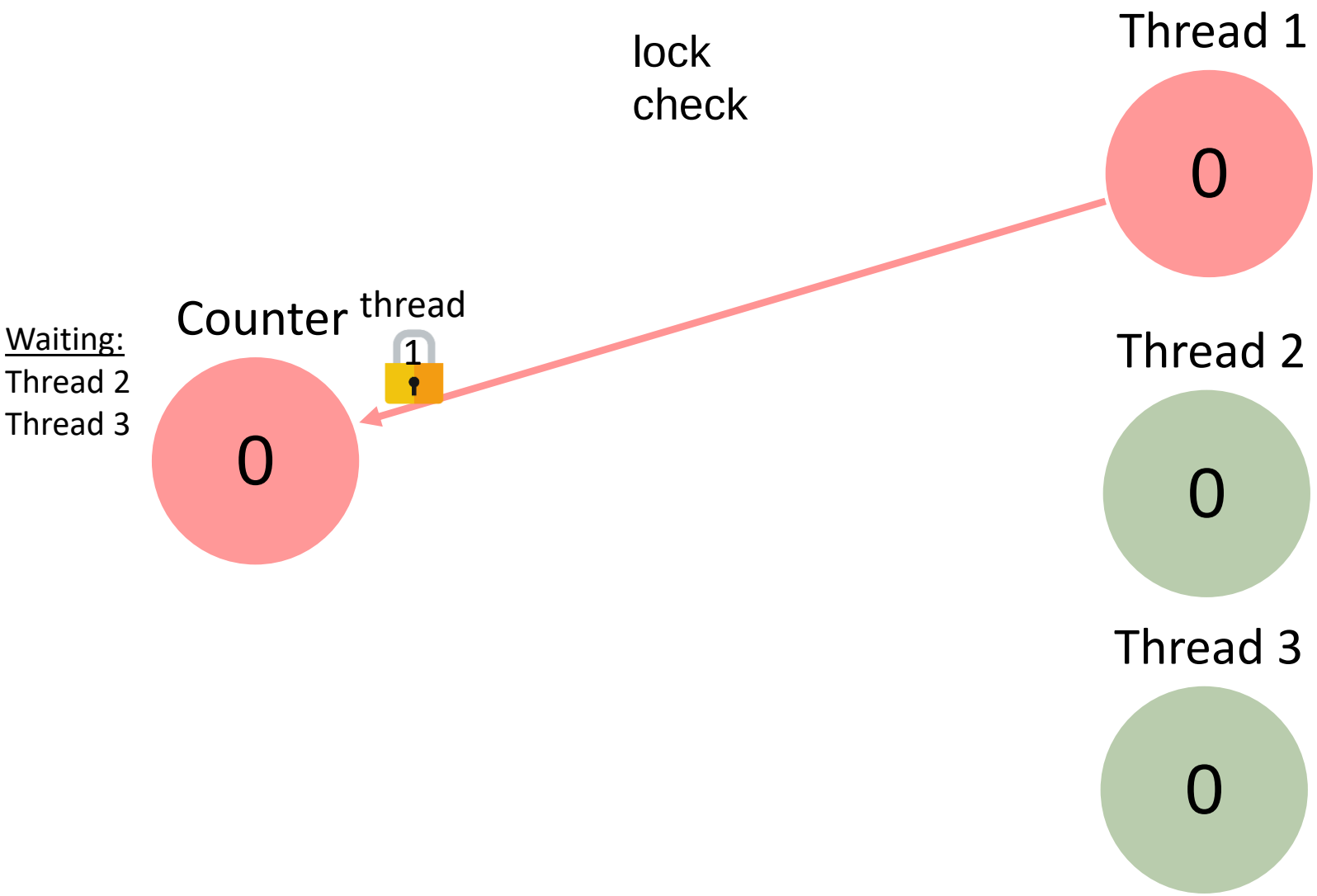

Thread 1
0
Counter
Waiting:
Thread 2
lock
check
wait
Thread 2
0
0
Blocked:
Thread 3
Thread 3
0

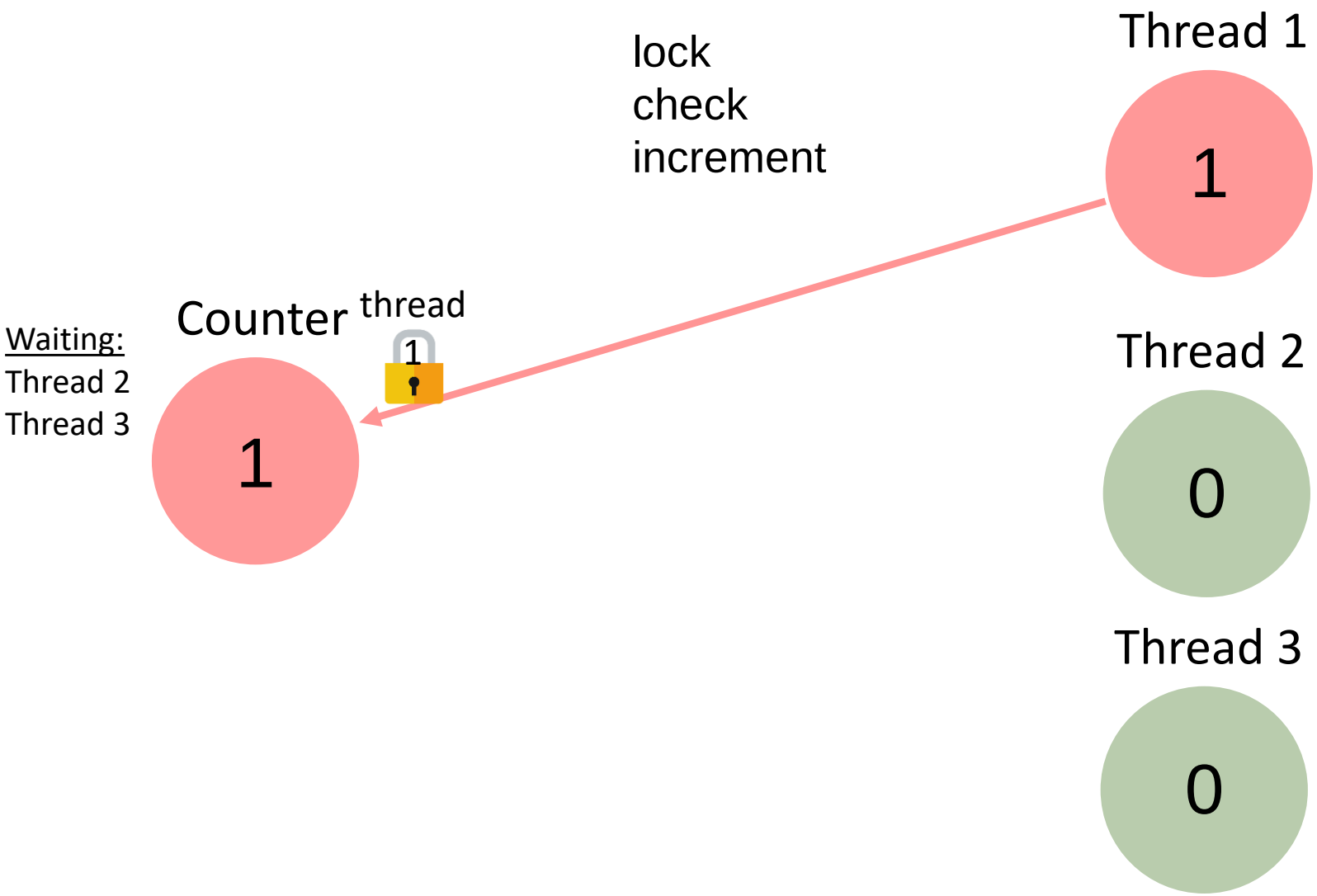

Thread 1
0
Waiting:
Thread 2
Counter
thread
3
0
Thread 2
0
lock
Thread 3
0

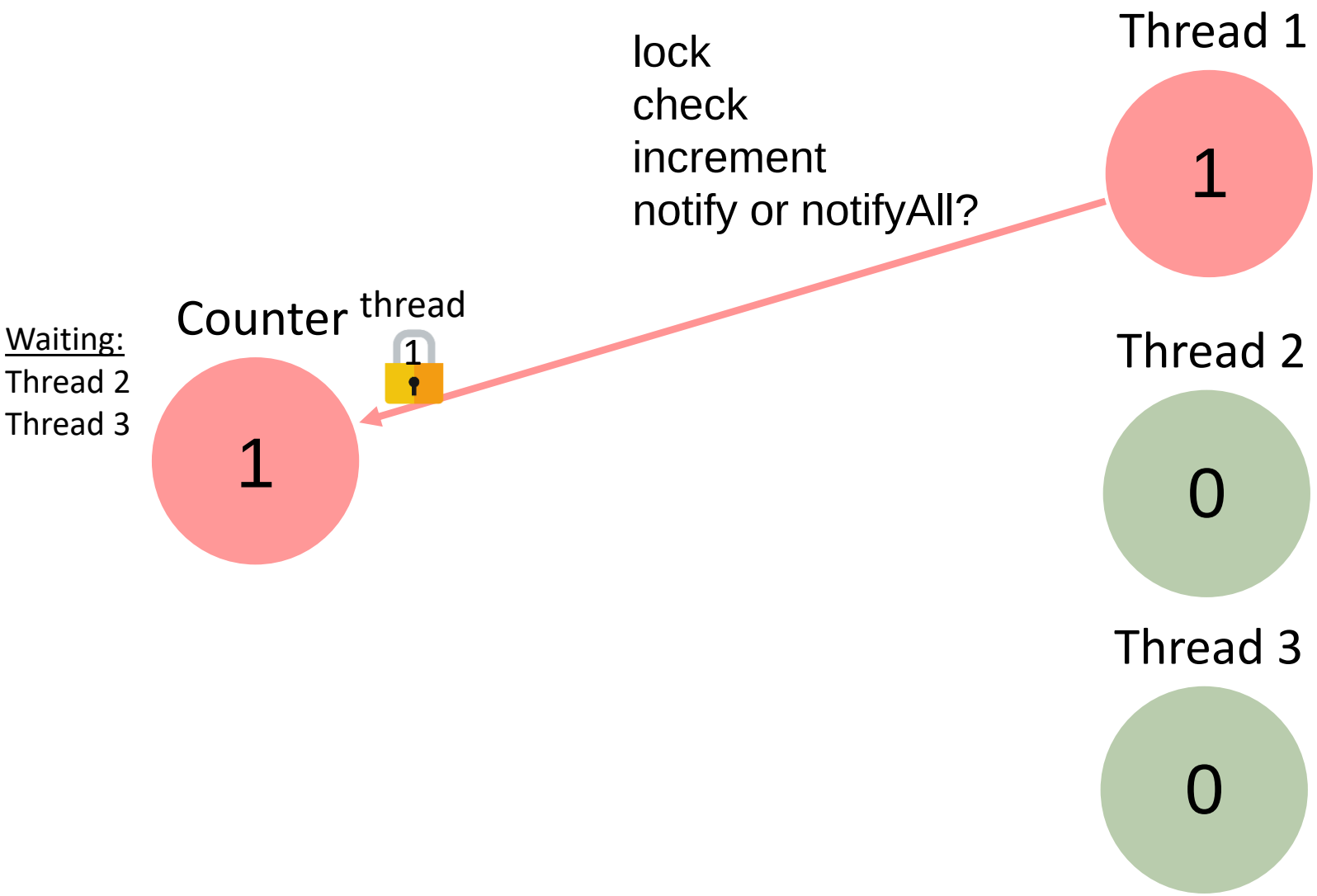

Thread 1
0
Waiting:
Thread 2
Counter
thread
3
0
Thread 2
0
lock
check
Thread 3
0

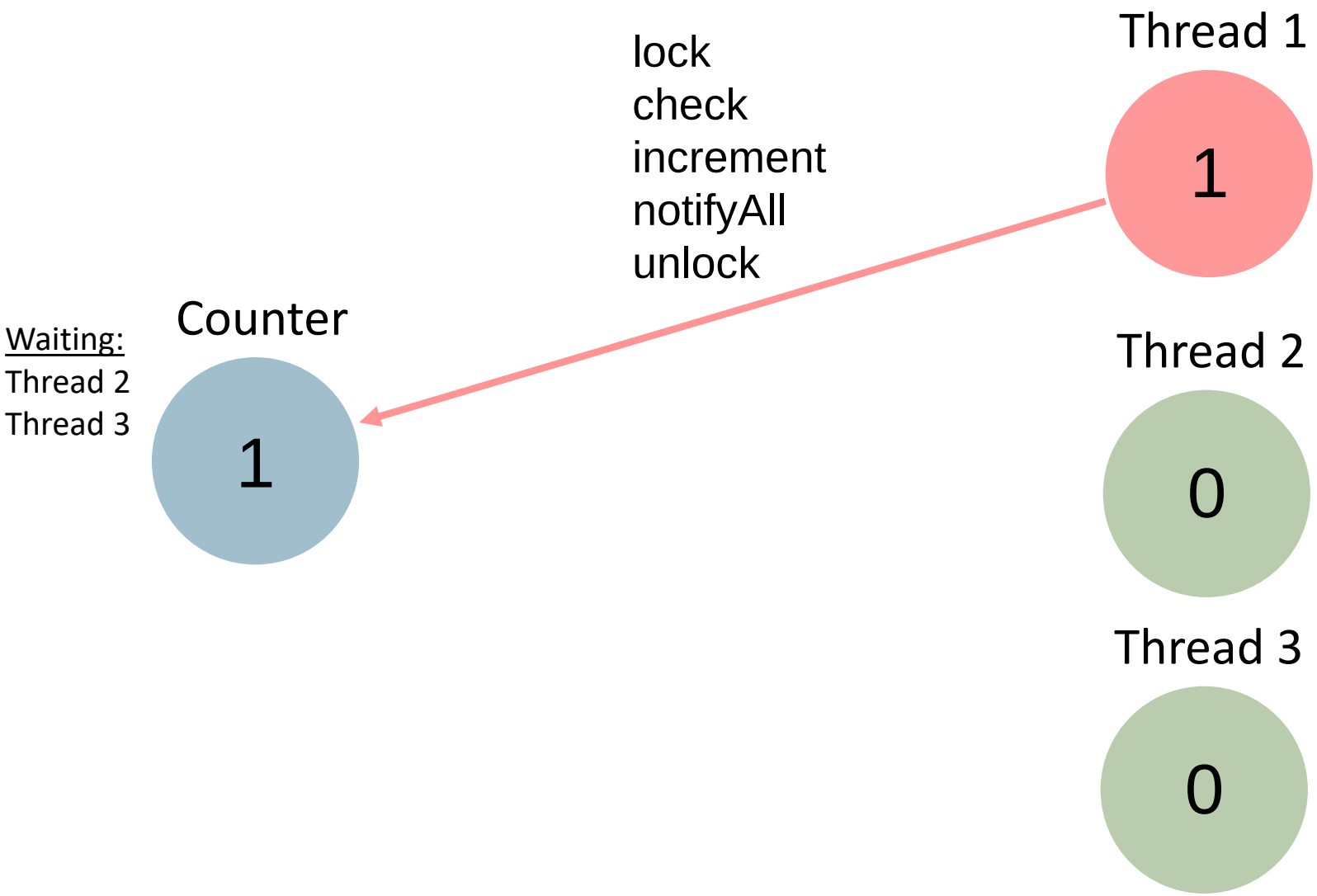

Thread 1
0
Counter
Waiting:
Thread 2
Thread 3
0
Thread 2
0
lock
check
wait
Thread 3
0

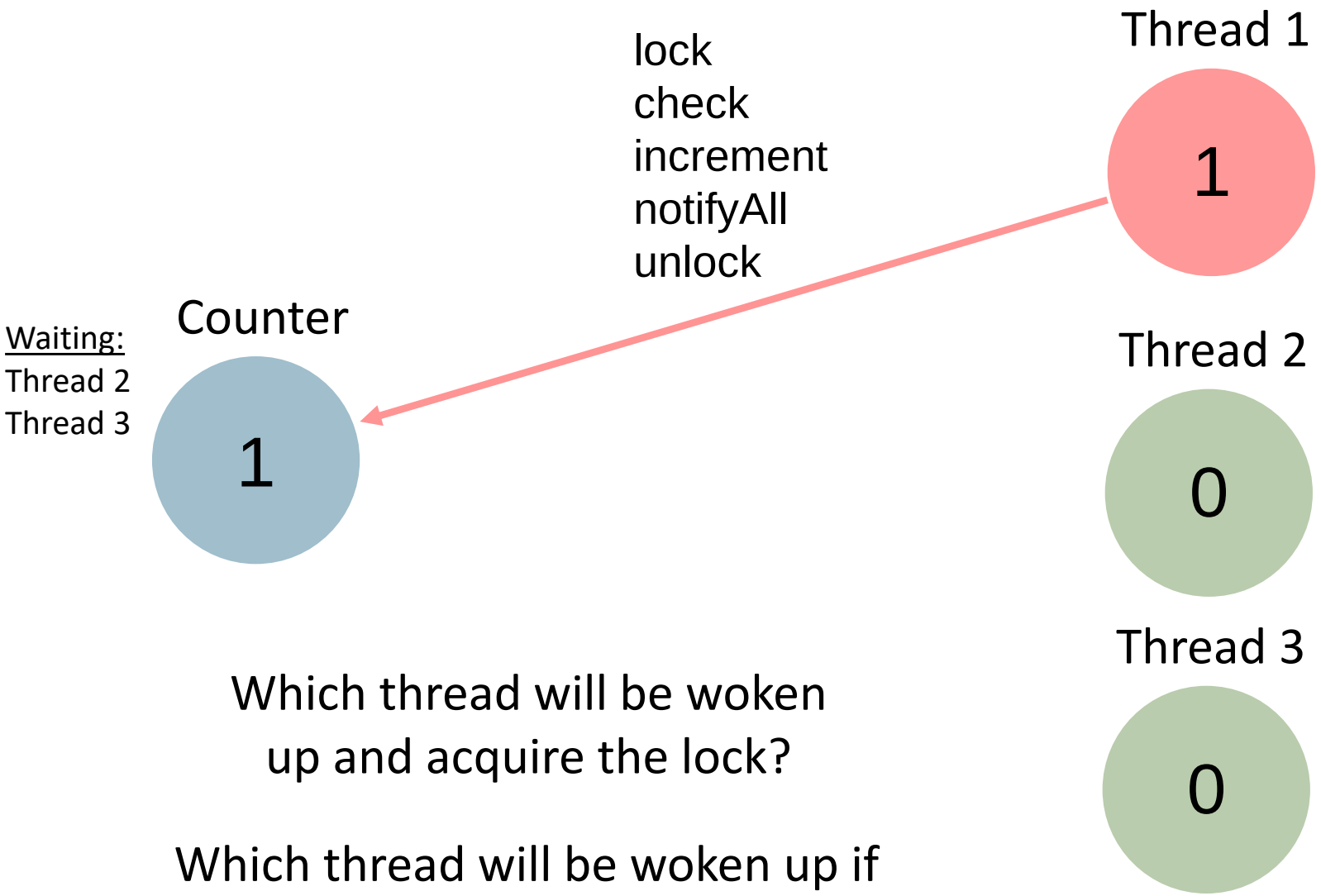

lock
Thread 1
0
Counter
thread
1
Waiting:
Thread 2
Thread 3
0
Thread 2
0
Thread 3
0


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


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


Thread 1
lock
check
increment
notifyAll
unlock
1
Counter
Waiting:
Thread 2
Thread 3
1
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();`

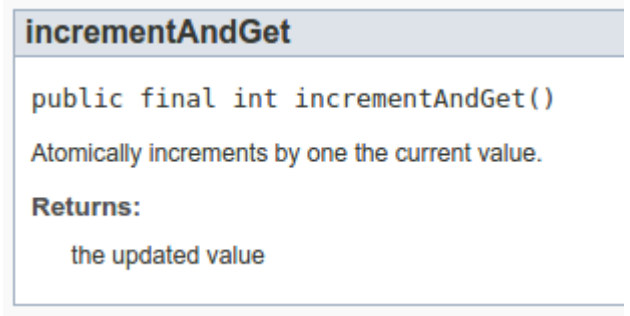
Task E: AtomicCounter

```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	AtomicInteger
1. load c → 0 2. c + 1 → 1 3. store c ← 1 ← `c++;`	`c.getAndIncrement();` →
not atomic	**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
());
```

- See code exercise 3

Plan für heute

- Organisation
- Nachbesprechung Exercise 3
- **Theory Recap**
- Intro Exercise 4
- Exam Questions
- Kahoot

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 all stages require the same 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).

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

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

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?**

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?
	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:

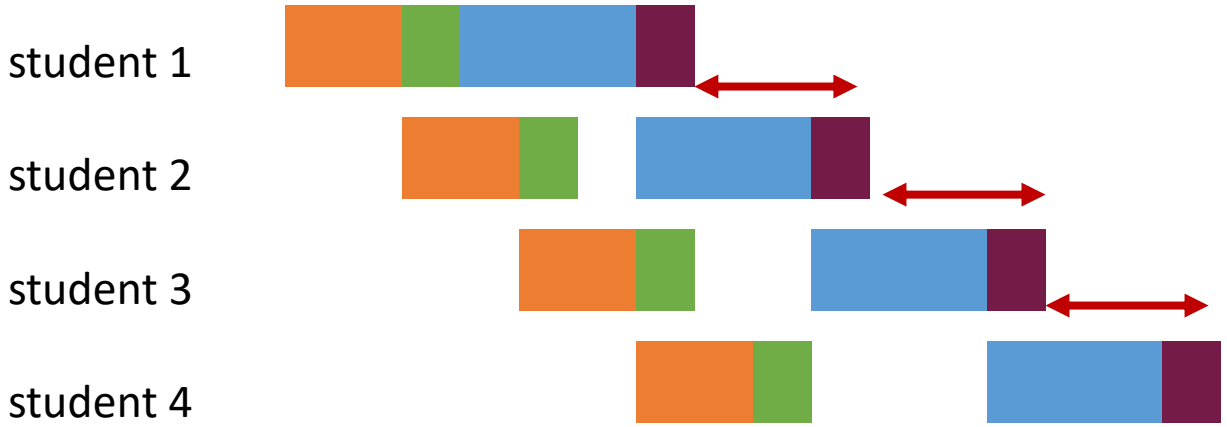
1) Reading book **A** takes 80 minutes

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?
	1 student per 120 minutes	280 min
		320 min
	Without lead in (indefinite 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?
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

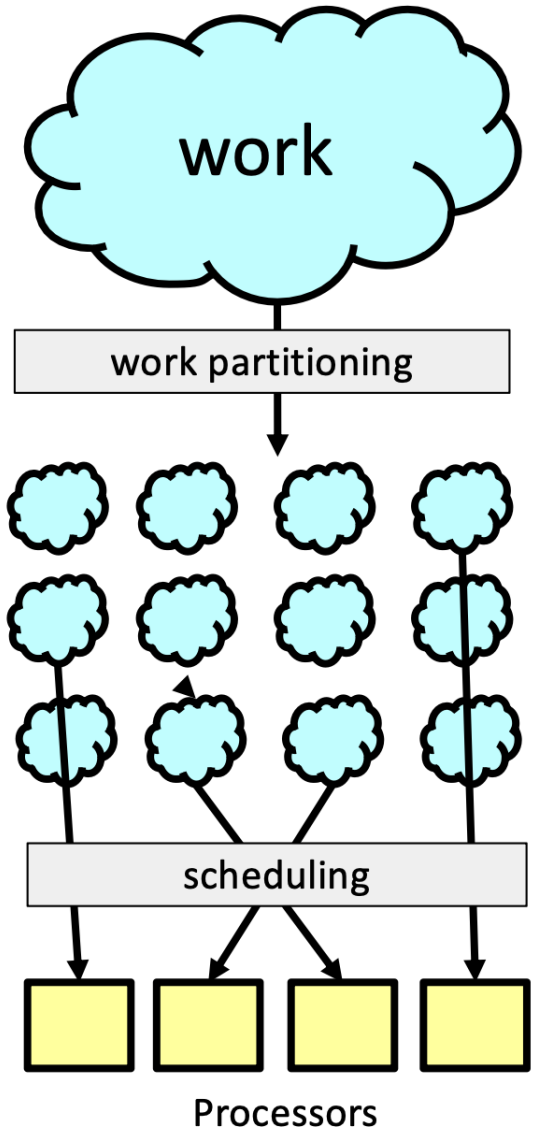
3) Reading book **C** takes 120 minutes

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

Work Partitioning & Scheduling

- **work partitioning**
  - **split up** work into **parallel tasks/threads**
  - (done by user)
  - A task is a unit of work
  - also called: **task/thread decomposition**
- **scheduling**
  - assign tasks to processors
  - (typically done by the system)
  - goal: full utilization

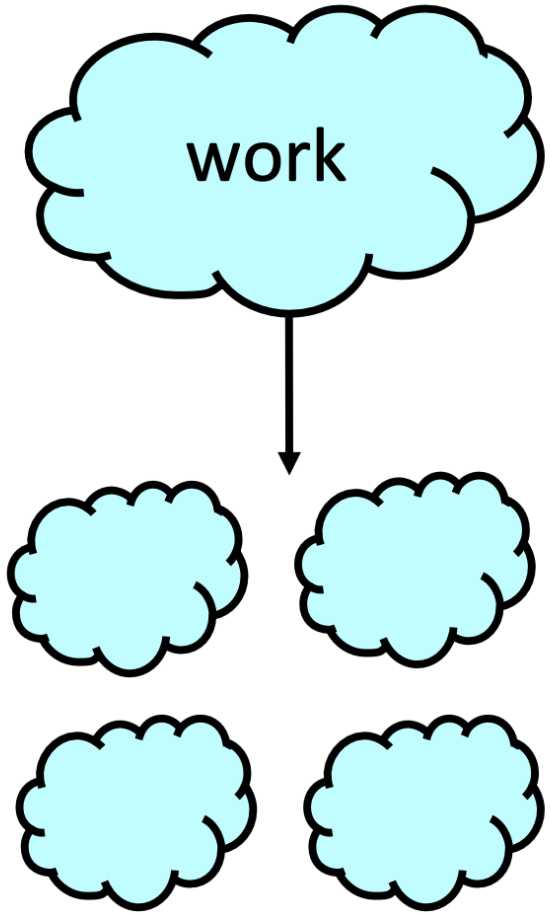
  (no processor is ever idle)



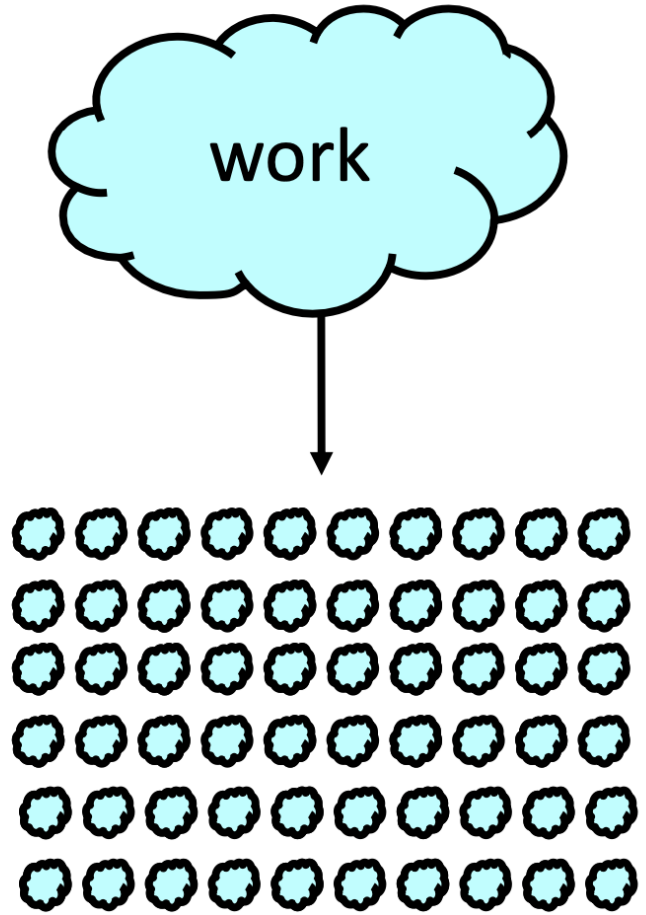
# of chunks should be larger than the # of processors

Credits to Gamal Hassan PProg FS24

Task/Thread Granularity



Coarse granularity



Fine granularity

Coarse vs Fine granularity

- **Fine granularity:**
  - more portable

  (can be executed in machines with more processors)
  - better for scheduling
  - but: if scheduling overhead is comparable to a single task → overhead dominates

Task granularity guidelines

- As small as possible
- but, significantly bigger than scheduling overhead
  - system designers strive to make overheads small

Parallel Performance

Sequential execution time: T_1

Execution time T_p on **p** CPUs

- $T_p = T_1 / p$ (perfection)
- $T_p > T_1 / p$ (performance loss, what normally happens)
- $T_p < T_1 / p$ (sorcery!)

(parallel) Speedup

(parallel) speedup S_p on **p** CPUs:

$$S_p = T_1 / T_p$$

- $S_p = p$ → linear speedup (perfection)
- $S_p < p$ → sub-linear speedup (performance loss)
- $S_p > p$ → super-linear speedup (sorcery!)

- **Efficiency: S_p / p**

Speedup

Sub-linear: usually

Super-linear:

not possible in theory

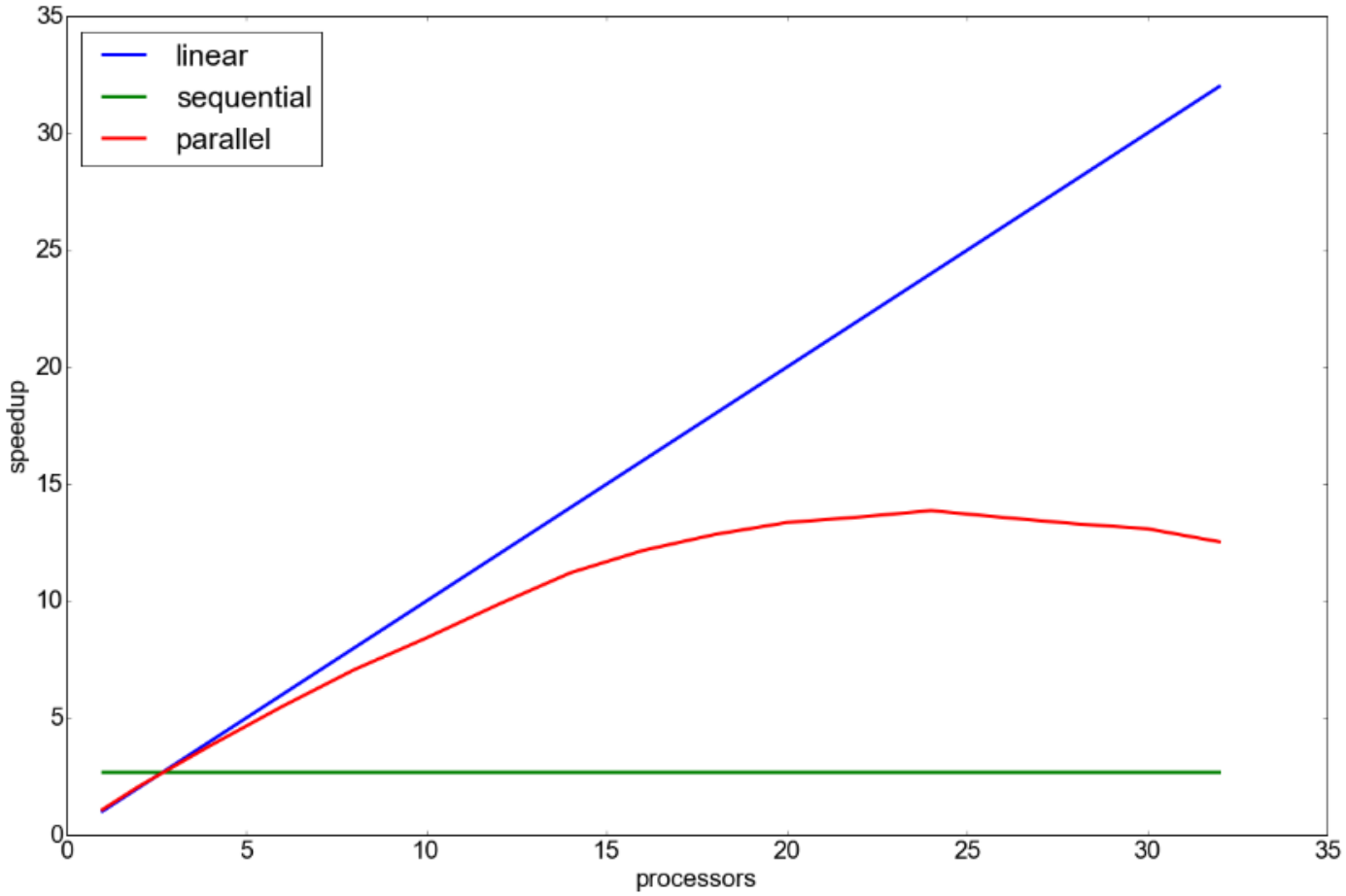


Figure 4.1: A typical graph comparing actual to linear speedup

Amdahl's Law – Ingredients

Given P workers available to do parallelizable work, the times for sequential execution and parallel execution are:

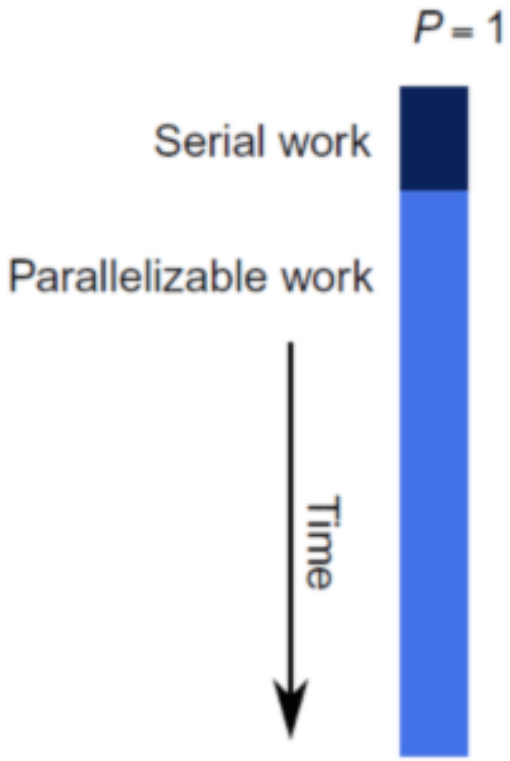
$$T_1 = W_{ser} + W_{par}$$

And this gives a bound on speed-up:

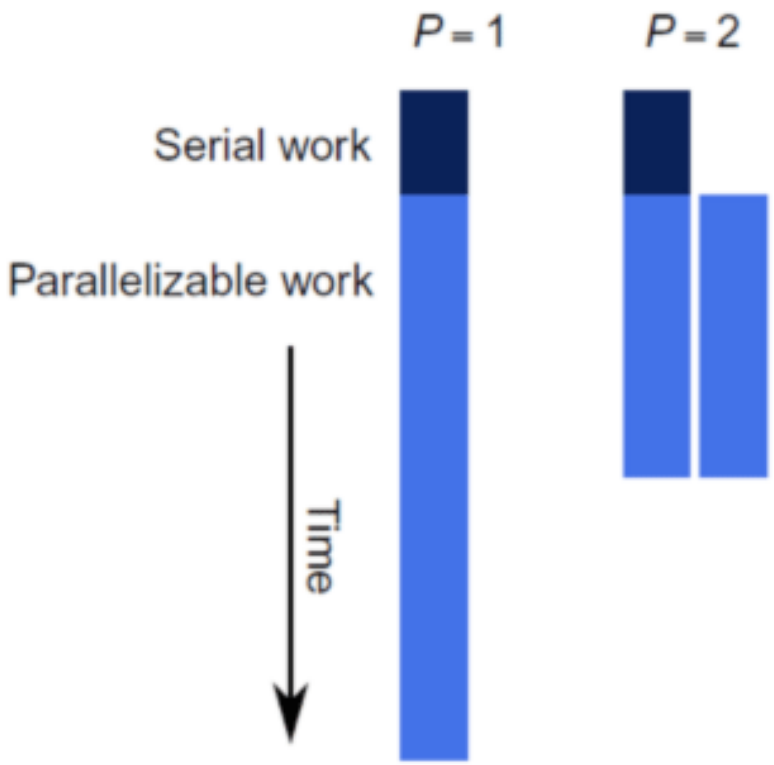
$$T_p \geq W_{ser} + \frac{W_{par}}{P}$$

$$S_p \leq \frac{W_{ser} + W_{par}}{W_{ser} + \frac{W_{par}}{P}}$$

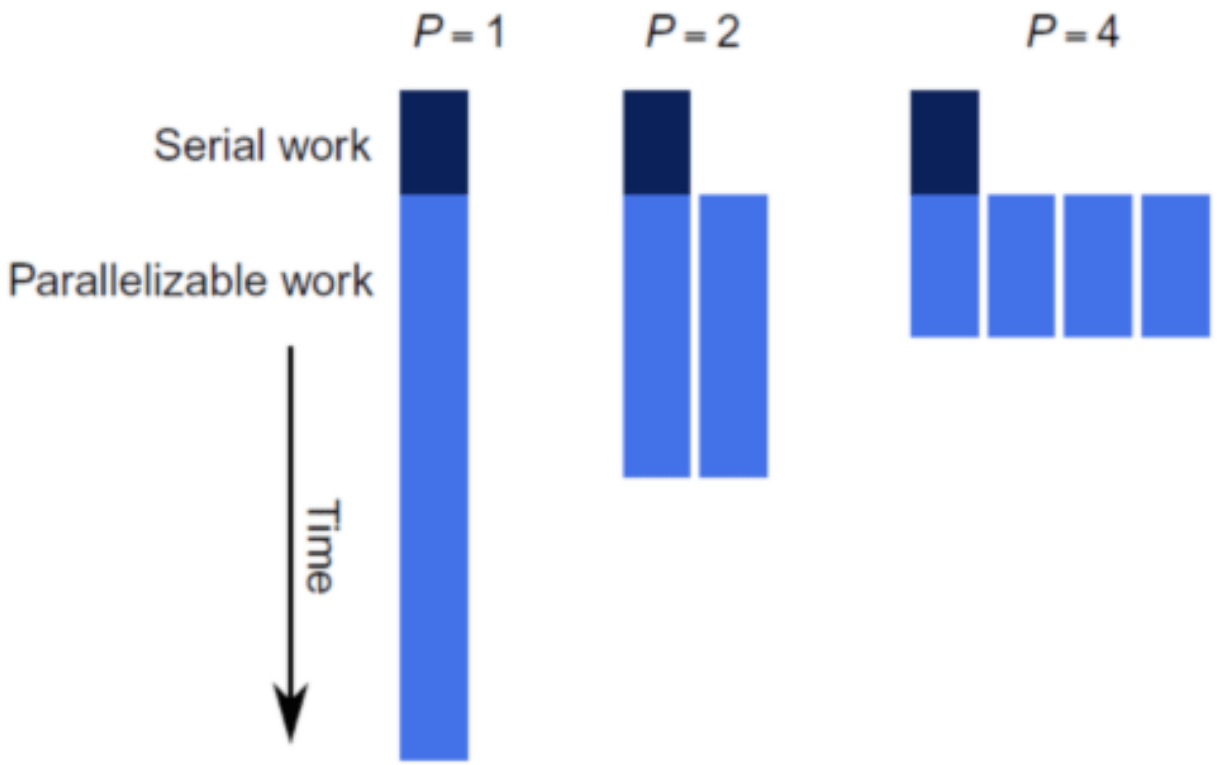
Amdahl's Law Illustrated



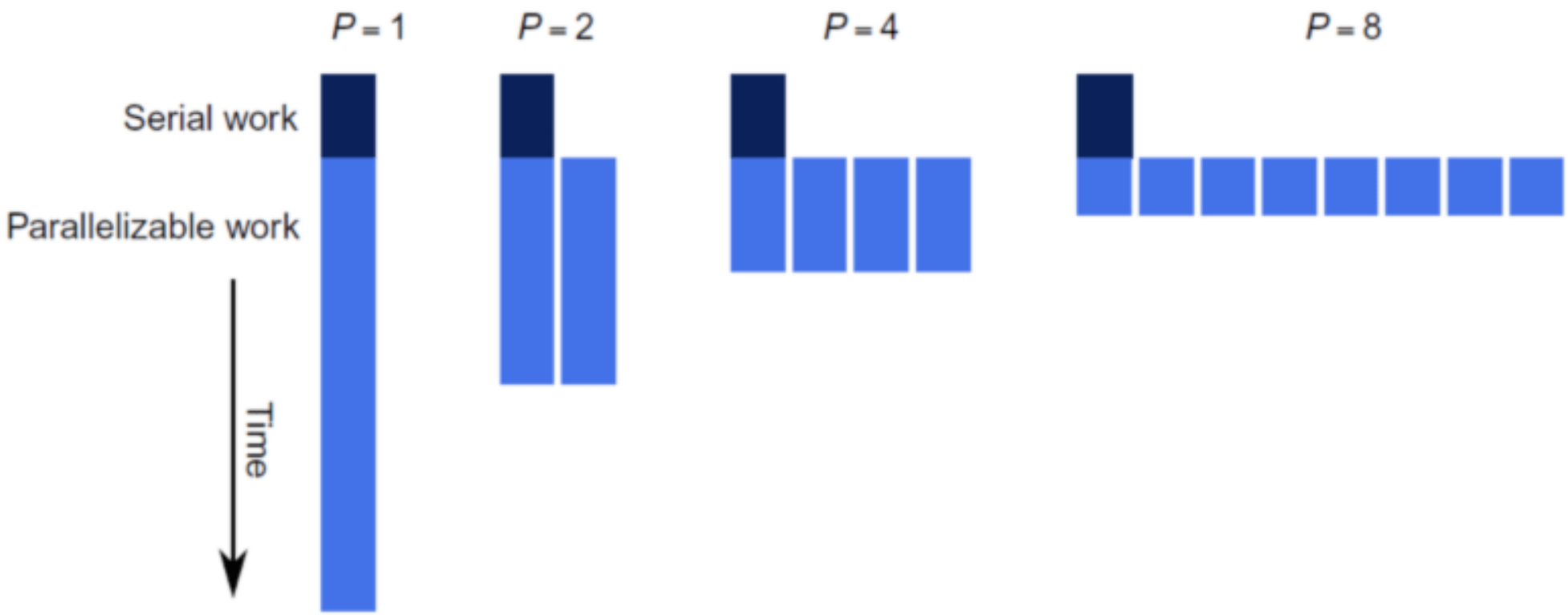
Amdahl's Law Illustrated



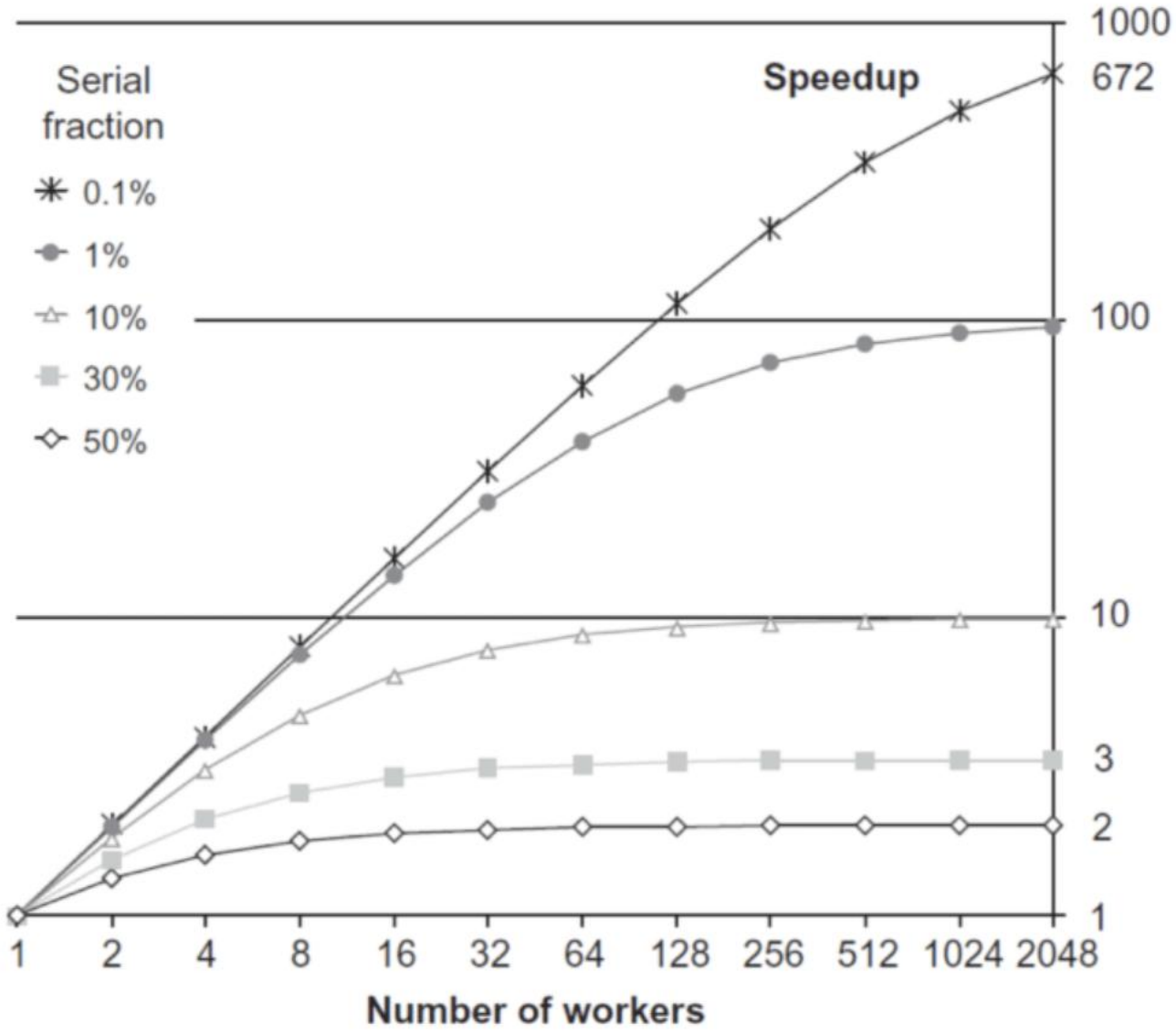
Amdahl's Law Illustrated



Amdahl's Law Illustrated



Speedup


Speedup
Serial fraction
0.1%
1%
10%
30%
50%
1000
672
100
10
3
2
1
1
2
4
8
16
32
64
128
256
512
1024
2048
Number of workers

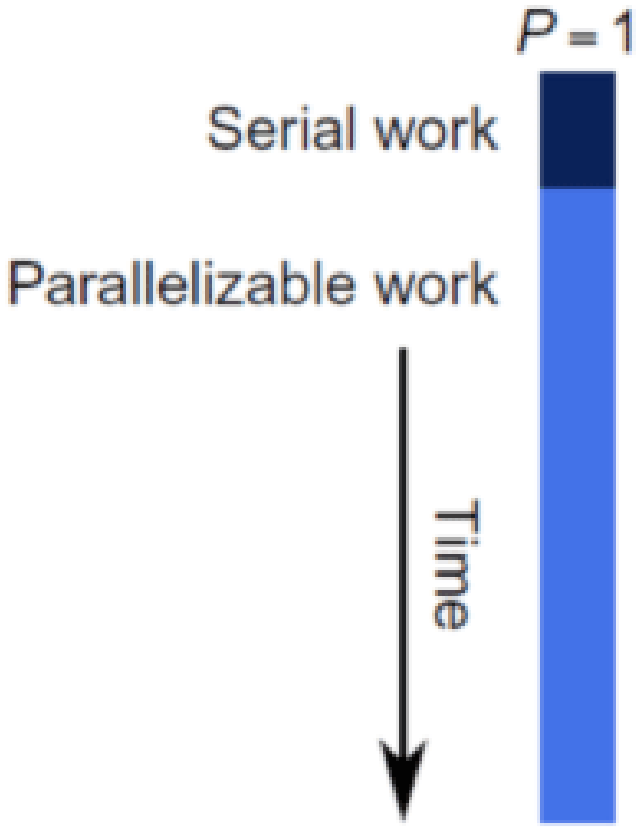
Gustafson's Law

- f: sequential part (no speedup)

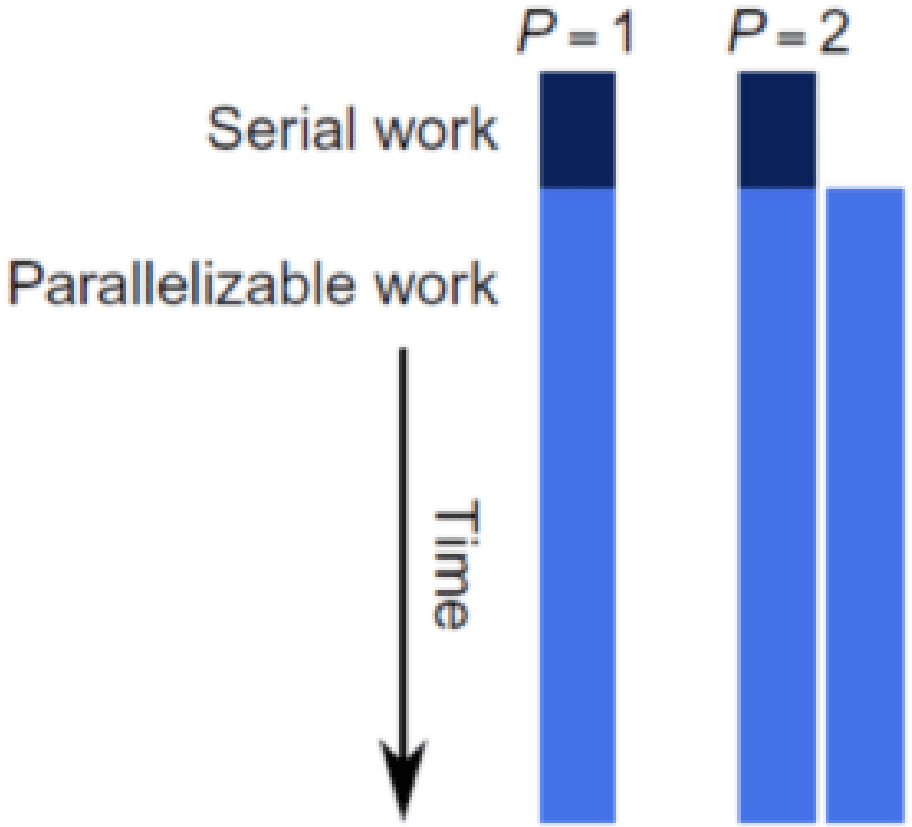
$$W = p(1 - f)T_{wall} + fT_{wall}$$

$$\begin{aligned} S_p &= f + p(1 - f) \\ &= p - f(p - 1) \end{aligned}$$

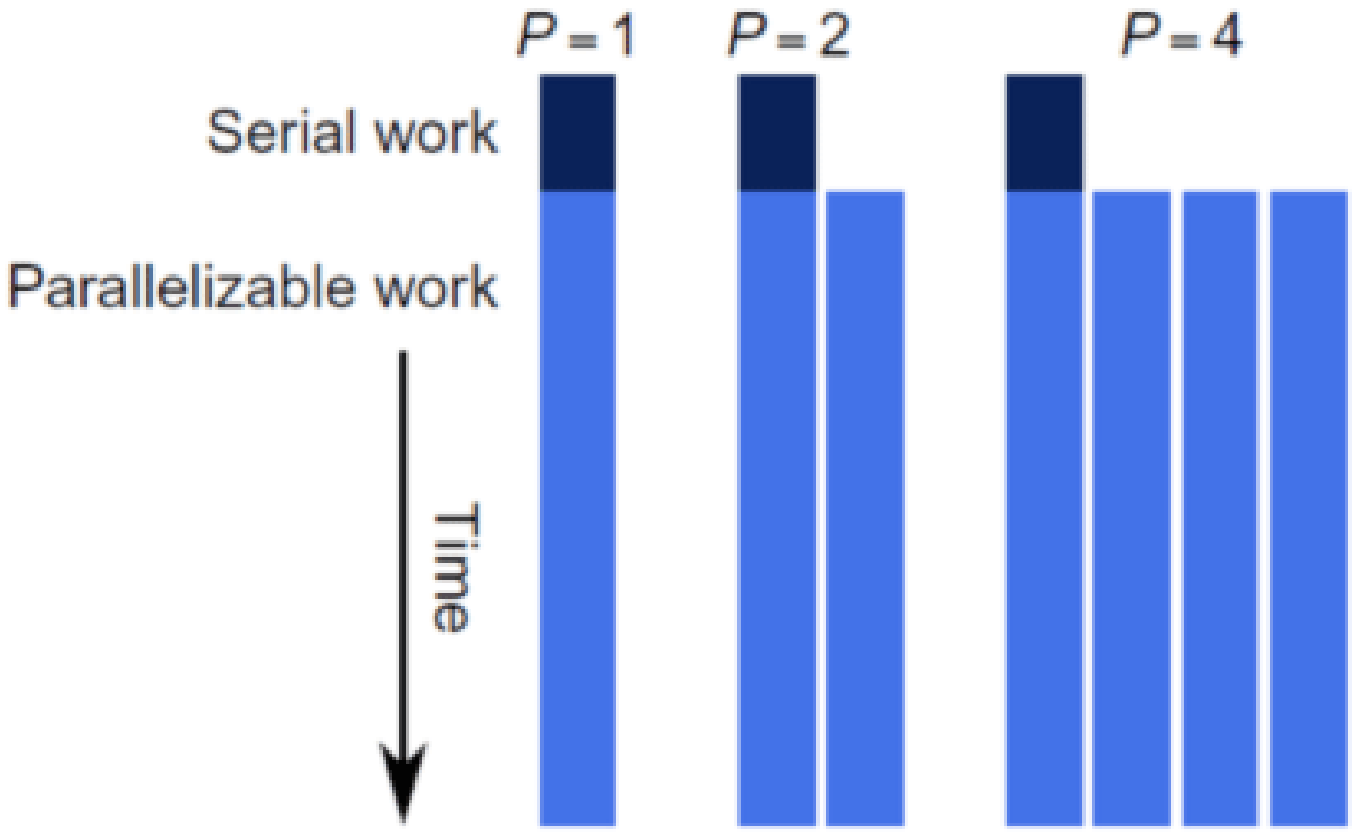
Gustafson's Law



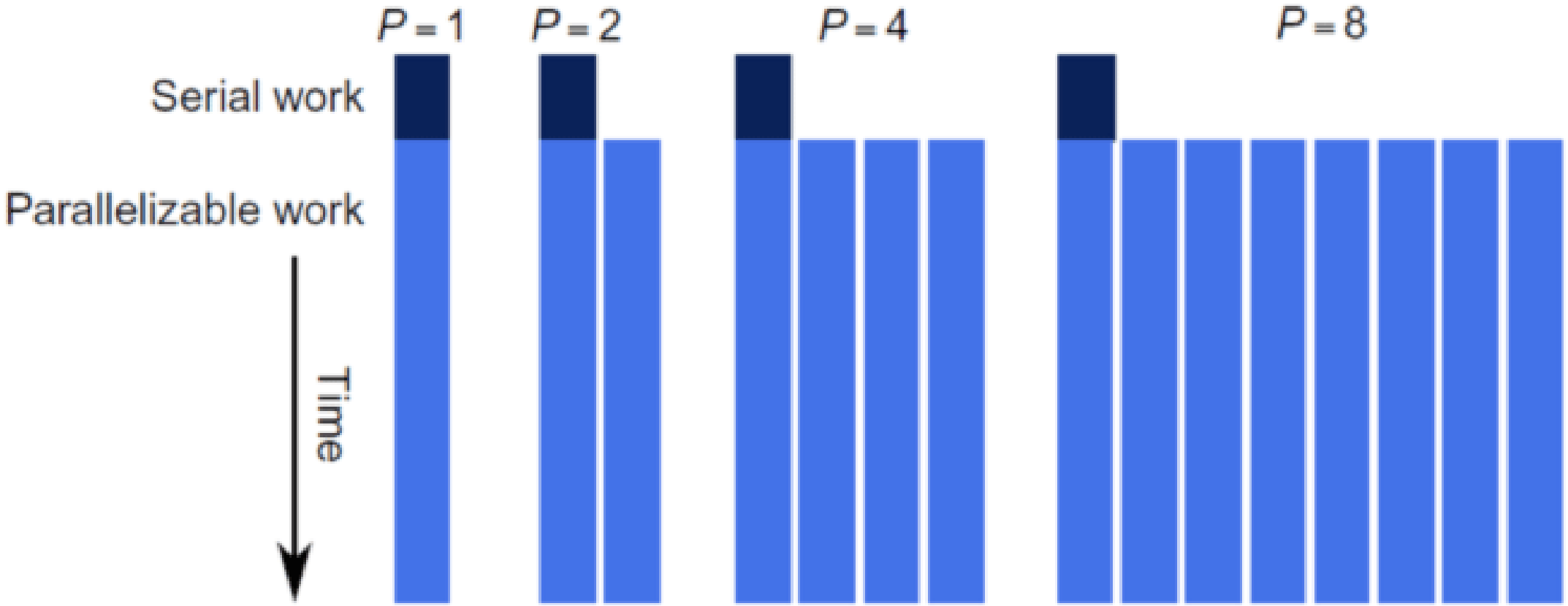
Gustafson's Law



Gustafson's Law



Gustafson's Law



Summary

T_1: Time on one processor
T_P: Time on P processors
T_∞: Time on "infinite" processors ($\lim_{P \to \infty} T_P$)

$S_P = \frac{T_1}{T_P}$: Speedup generated using P processors

- **Amdahls Law** assumes a fixed workload in variable time it states:

$$S_P = \frac{T_1}{T_P} \leq \frac{1}{f + \frac{1-f}{P}}$$

- **Gustafsons Law** assumes a fixed time window, but measures the speedup in workload terms:

$$S_P \leq f + P(1 - f)$$

Plan für heute

- Organisation
- Nachbesprechung Exercise 3
- Theory Recap
- **Intro Exercise 4**
- Exam Questions
- Kahoot

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 issue either

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

for each cycle. 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
```

Plan für heute

- Organisation
- Nachbesprechung Exercise 3
- Theory Recap
- Intro Exercise 4
- **Exam Questions**
- Kahoot

Past Exam Task

Pipelining.
Sie leiten eine Produktionslinie in einer Fabrik, bei der die Produkte vier Montagephasen durchlaufen. Jede Phase hat eine spezifische Bearbeitungszeit:

(A) Material schneiden: 5 Minuten pro Produkt.

(B) Teile zusammenbauen: 10 Minuten pro Produkt.

(C) Lackieren: 15 Minuten pro Produkt.

(D) Qualitätsprüfung: 7 Minuten pro Produkt.

Pipelining.
You are managing a factory production line where products go through four stages of assembly. Each stage has a specific processing time:

(A) Cutting materials: 5 min per product.

(B) Assembling parts: 10 min per product.

(C) Painting: 15 min per product.

(D) Quality check: 7 min per product.

Exam, FS 2024

Past Exam Task

i. Was ist der **Throughput** der Produktionslinie? **Spezifizieren Sie die Einheiten.**

*What is the **throughput** of the production line? **Specify the units.*** (1)

ii. Was ist die **Latency** der Produktionslinie? **Spezifizieren Sie die Einheiten.**

*What is the **latency** of the production line? **Specify the units.*** (1)

Exam, FS 2024

Past Exam Task

i. Was ist der **Throughput** der Produktionslinie? **Spezifizieren Sie die Einheiten.** *What is the **throughput** of the production line? **Specify the units.*** (1)

$$\text{Throughput} = 60/15 = 4 \text{ products per hour}$$

or

$$\text{Throughput} = 1/15 = 1 \text{ product/15min}$$

ii. Was ist die **Latency** der Produktionslinie? **Spezifizieren Sie die Einheiten.** *What is the **latency** of the production line? **Specify the units.*** (1)

$$\text{Latency} = 5 + 10 + 15 + 7 = 37 \text{ minutes}$$

Exam, FS 2024

Past Exam Task

Kreuzen Sie alle korrekten Aussagen über die Ausführung von Java `Threads` an.

Mark all correct statements regarding the execution of Java `Threads`.

- ○ Die start() Methode in `t = new Thread(); t.start()` ruft automatisch auch die run() methode auf.

  The start() method in `t = new Thread(); t.start()` automatically also calls the run() method.

- ○ Die run() Methode in `t = new Thread(); t.run()` erzeugt einen neuen Thread und führt diesen aus.

  The run() method in `t = new Thread(); t.run()` creates a new thread and executes the thread.

- ○ Ein Codeblock mit mehreren Threads wird immer deterministisch ausgeführt. D.h. der Output ist immer exakt der gleiche.

  A codeblock with several threads is always executed deterministically. That means the output is always the same.

- ○ Ein komplett serieller Codeblock kann zur Beschleunigung auf mehreren Prozessoren ausgeführt werden.

  A fully serial block of code can be run on multiple processors to speedup execution.

Rep. Exam, FS 2023

Past Exam Task

Kreuzen Sie alle korrekten Aussagen über die Ausführung von Java `Threads` an.

Mark all correct statements regarding the execution of Java `Threads`.

√ **Die start() Methode in `t = new Thread(); t.start()` ruft automatisch auch die run() methode auf.**

The start() method in `t = new Thread(); t.start()` automatically also calls the run() method.

○ Die run() Methode in `t = new Thread(); t.run()` erzeugt einen neuen Thread und führt diesen aus.

The run() method in `t = new Thread(); t.run()` creates a new thread and executes the thread.

○ Ein Codeblock mit mehreren Threads wird immer deterministisch ausgeführt. D.h. der Output ist immer exakt der gleiche.

A codeblock with several threads is always executed deterministically. That means the output is always the same.

○ Ein komplett serieller Codeblock kann zur Beschleunigung auf mehreren Prozessoren ausgeführt werden.

A fully serial block of code can be run on multiple processors to speedup execution.

Rep. Exam, FS 2023

Past Exam Task

(c) Wozu dient die `join()` Methode in Java `Threads`? *What is the purpose of the `join()` method in Java `Threads`?.* (2)

- ◯ Um eine Prioritätenreihenfolge zwischen mehreren Threads zu erzwingen. *To enforce a priority order among multiple threads.*
- ◯ Um das von dem aktuellen Thread gehaltene `Lock` freizugeben. *To release the lock held by the current thread.*
- ◯ Um die Ausführung des aktuellen Threads anzuhalten, bis der Thread, den er `joined`, abgeschlossen ist. *To pause the current thread's execution until the thread it `joins` completes.*
- ◯ Um die Kontrolle an einen anderen Thread zu übergeben, ohne auf dessen Abschluss zu warten. *To transfer control to another thread without waiting for its completion.*

Rep. Exam, FS 2023

Past Exam Task

(c) Wozu dient die `join()` Methode in Java `Threads`? *What is the purpose of the `join()` method in Java `Threads`?.* (2)

- ○ Um eine Prioritätenreihenfolge zwischen mehreren Threads zu erzwingen. *To enforce a priority order among multiple threads.*
- ○ Um das von dem aktuellen Thread gehaltene `Lock` freizugeben. *To release the lock held by the current thread.*
- √ **Um die Ausführung des aktuellen Threads anzuhalten, bis der Thread, den er `joined`, abgeschlossen ist.** *To pause the current thread's execution until the thread it `joins` completes.*
- ○ Um die Kontrolle an einen anderen Thread zu übergeben, ohne auf dessen Abschluss zu warten. *To transfer control to another thread without waiting for its completion.*

Rep. Exam, FS 2023

iv.	In Java kann ein Thread sich ein Monitor Lock eines Objekts nur einmal erwerben.	*In Java, a thread can acquire the monitor lock of an object only once.*	(1)

True False

iv. In Java kann ein Thread sich ein Monitor Lock eines Objekts nur einmal erwerben.

In Java, a thread can acquire the monitor lock of an object only once. (1)

True False

Locks sind reentrant!

vi. Wenn ein Java Thread versucht in einen `synchronized` Block einzutreten, welcher bereits von einem anderen Thread ausgeführt wird, geht der Thread in den `BLOCKED` Zustand.

When a Java thread tries to enter a ***`synchronized`*** *block which is already being executed by another thread, the thread enters the* ***`BLOCKED`*** *state.*

(1)

True False

vi. Wenn ein Java Thread versucht in einen `synchronized` Block einzutreten, welcher bereits von einem anderen Thread ausgeführt wird, geht der Thread in den `BLOCKED` Zustand.

When a Java thread tries to enter a ***`synchronized`*** *block which is already being executed by another thread, the thread enters the* ***`BLOCKED`*** *state.*

(1)

True False

viii. Die `notify()` Methode gibt dem Aufwecken von Threads, die am längsten warten, **keine** Priorität.

*The `notify()` method does **not** prioritise waking up threads which have waited the longest.* (1)

True False

viii. Die `notify()` Methode gibt dem Aufwecken von Threads, die am längsten warten, **keine** Priorität.

*The `notify()` method does **not** prioritise waking up threads which have waited the longest.* (1)

True False

vii. Ein Thread, der auf `Object obj` wartet, kann **nur** aufgeweckt werden, wenn ein anderer Therad `obj.notify()` oder `obj.notifyAll()` aufruft.

*A thread waiting on `Object obj` can **only** be woken up when another thread calls the `obj.notify()` or `obj.notifyAll()` methods.*

(1)

True False

vii. Ein Thread, der auf `Object obj` wartet, kann **nur** aufgeweckt werden, wenn ein anderer Therad `obj.notify()` oder `obj.notifyAll()` aufruft.

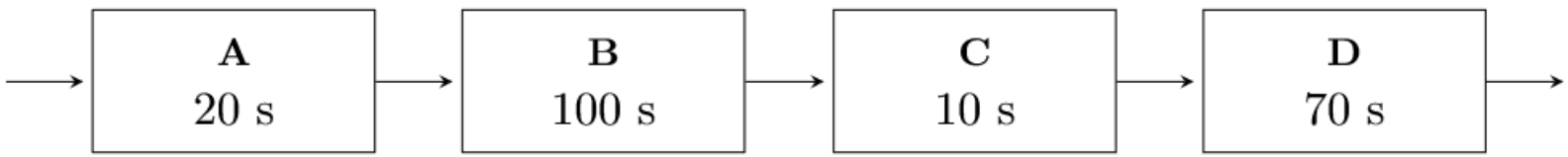
*A thread waiting on `Object obj` can **only** be woken up when another thread calls the `obj.notify()` or `obj.notifyAll()` methods.* (1)

True False

Spurious wake-ups! Deswegen ist wait() immer in einer while loop.

(b) Betrachten Sie die unten gezeigte Pipeline. Es gibt mehrere parallele Eingänge und A, B, C und D sind unterschiedliche Funktionseinheiten.

Consider the pipeline shown below. There are multiple parallel inputs and A, B, C, and D are different functional units.

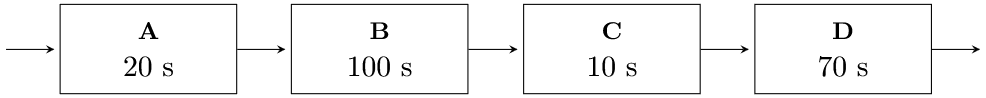


i. Ist die Pipeline balanced? Begründen Sie Ihre Antwort. Sie müssen keine Zahl oder Berechnung angeben.

Is the pipeline balanced? Justify your answer. You do not need to provide a number or calculation. (2)

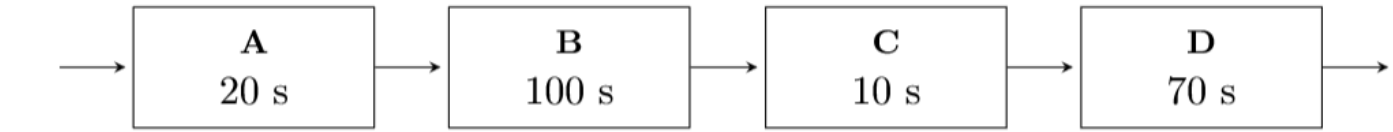
(b) Betrachten Sie die unten gezeigte Pipeline. Es gibt mehrere parallele Eingänge und A, B, C und D sind unterschiedliche Funktionseinheiten.

Consider the pipeline shown below. There are multiple parallel inputs and A, B, C, and D are different functional units.



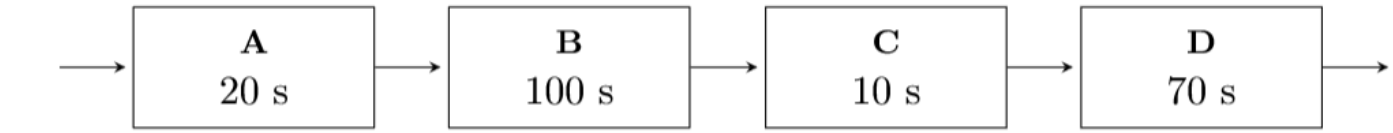
i. Ist die Pipeline balanced? Begründen Sie Ihre Antwort. Sie müssen keine Zahl oder Berechnung angeben. *Is the pipeline balanced? Justify your answer. You do not need to provide a number or calculation.* (2)

Nein, nicht jede stage hat die selbe Zeit.



ii. Wie lange wird es dauern, die gesamte Pipeline (d.h. die Stufen A-B-C-D) *15* Mal auszuführen? Gehen Sie davon aus, dass wir die Pipeline so parallelisieren, dass jede Stufe parallel zu den anderen Stufen laufen kann, jedoch nicht zu sich selbst. Achten Sie darauf, Ihre Antwort in den entsprechenden Einheiten anzugeben.

How long will it take to run the whole pipeline (i.e., stages A-B-C-D) 15 times? Assume that we parallelize the pipeline such that each stage can run in parallel with the other stages, but not with itself. Make sure to leave your answer in the appropriate units. (3)



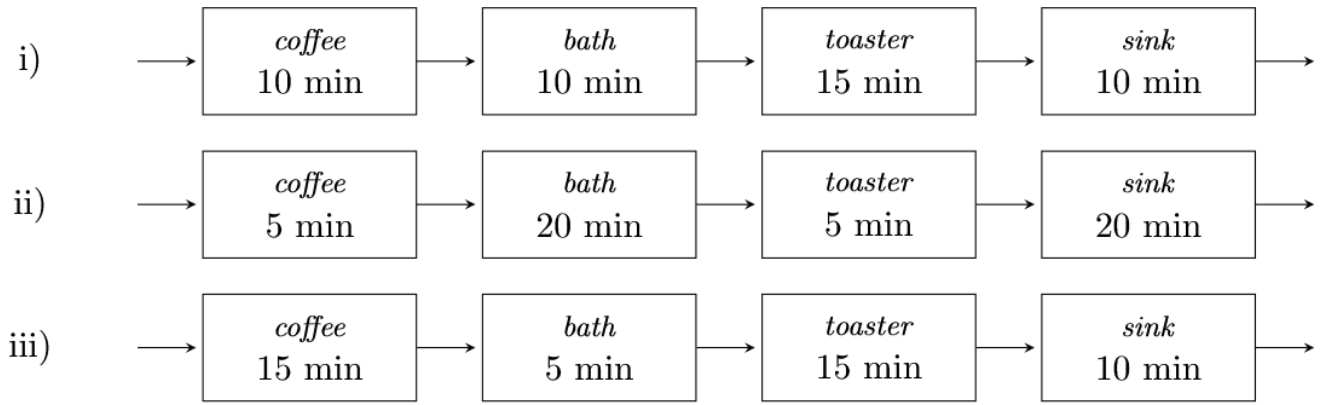
ii. Wie lange wird es dauern, die gesamte Pipeline (d.h. die Stufen A-B-C-D) *15* Mal auszuführen? Gehen Sie davon aus, dass wir die Pipeline so parallelisieren, dass jede Stufe parallel zu den anderen Stufen laufen kann, jedoch nicht zu sich selbst. Achten Sie darauf, Ihre Antwort in den entsprechenden Einheiten anzugeben.

How long will it take to run the whole pipeline (i.e., stages A-B-C-D) 15 times? Assume that we parallelize the pipeline such that each stage can run in parallel with the other stages, but not with itself. Make sure to leave your answer in the appropriate units. (3)

- Let T:= “duration between the completion of two subsequent instances". We have that T = 100s (duration of longest stage).
  - -> throughput
- For the total duration we have: duration for 15 instances := D = “Latency for first instance + (15-1) T = 200s + 14 times 100s = 1600s

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*

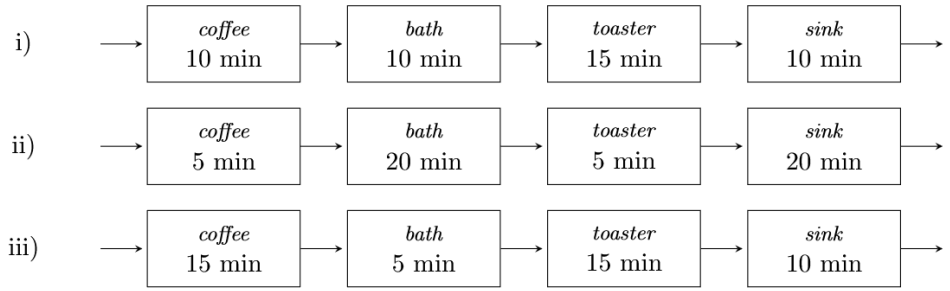


(a) Was ist der Durchsatz ("throughput") der einzelnen Pipelines pro Stunde in dieser WG? *What is the throughput of each pipeline per hour in this living community?* (3)

i) ii) iii)

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*



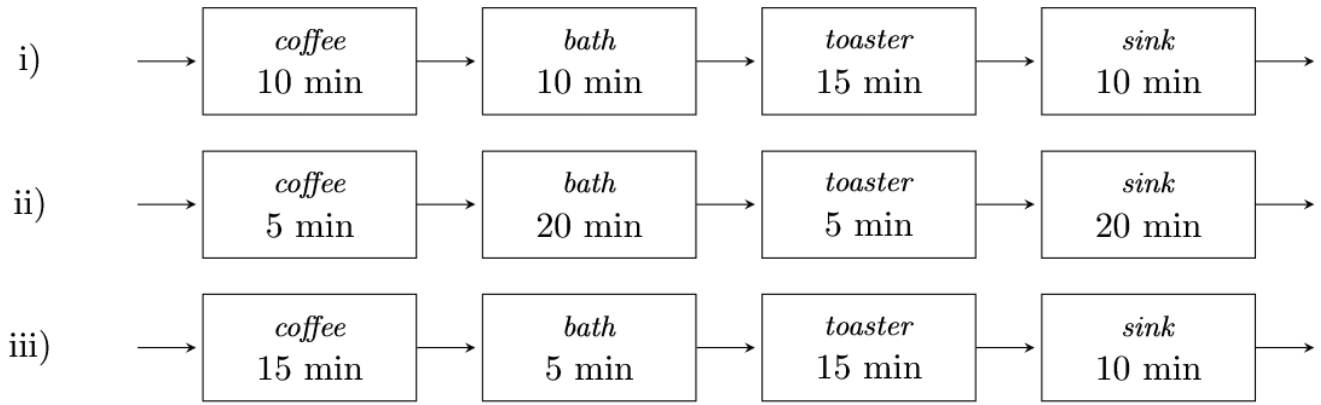
(a) Was ist der Durchsatz ("throughput") der einzelnen Pipelines pro Stunde in dieser WG? *What is the throughput of each pipeline per hour in this living community?* (3)

i) ii) iii)

- 1.) 1 Student /15 Minuten = 4 Studenten/h
- 2.) 1 Student/20Minuten = 3 Studenten/h
- 3.) 1 Student /15 Minuten = 4 Studenten/h

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

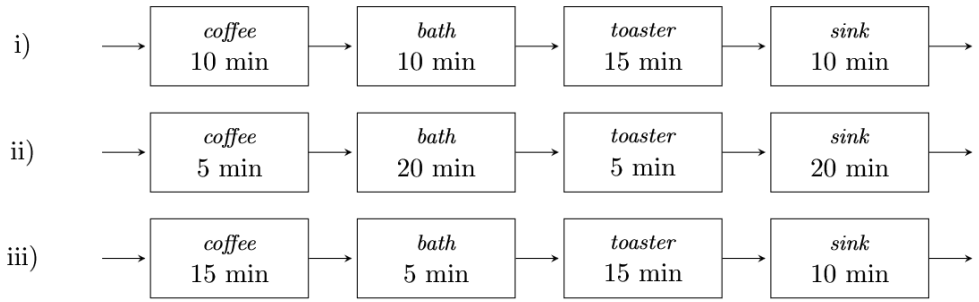
For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*



(b) Das Ziel ist, dass alle drei Studenten so schnell wie möglich fertig sind. Rechnen sie die Zeit aus, die es benötigt bis alle drei Studenten die Pipelines durchlaufen haben. Begründen Sie Ihre Antwort und geben Sie alle Rechenschritte an. Welche Pipeline(s) ist (sind) die schnellste(n)?

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*



(b) Das Ziel ist, dass alle drei Studenten so schnell wie möglich fertig sind. Rechnen sie die Zeit aus, die es benötigt bis alle drei Studenten die Pipelines durchlaufen haben. Begründen Sie Ihre Antwort und geben Sie alle Rechenschritte an. Welche Pipeline(s) ist (sind) die schnellste(n)?

Zuerst berrechnen wir die Zeit, welche die erste Person benötigt um fertig zu sein:

1. $10 + 10 + 15 + 10 = 45$
2. $5 + 20 + 5 + 20 = 50$
3. $15 + 5 + 15 + 10 = 45$

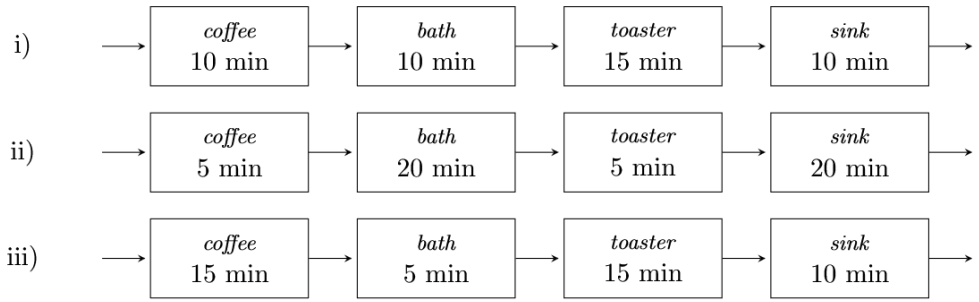
Für jeden zusätzlichen Student, benötigen wir nun die längste Station (1 -> 15', 2 -> 20', 3 -> 15') mehr. Also:

1. $45 + 2 * 15 = 75$
2. $50 + 2 * 20 = 90$
3. $45 + 2 * 15 = 75$

Somit sehen wir das (1) und (3) die schnellsten Pipelines sind.

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

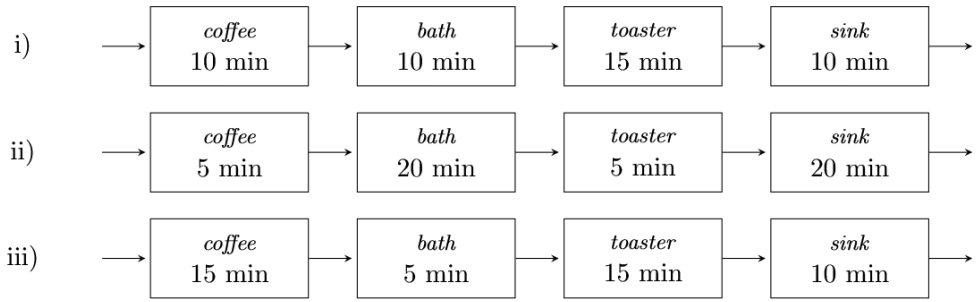
For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*



(d) Aufgrund von einer Pandemie ist es den Studenten leider nicht erlaubt eines der 4 gemeinsamen Objekte zu benutzen bevor der vorhergehende Student komplett fertig ist (d.h. mit dem "sink" Schritt abgeschlossen hat). Die Studenten haben jedoch die Möglichkeit, jeden Schritt um 10 Minuten zu verlängern um die Objekte zu desinfizieren, damit die nächste Person das Objekt unmittelbar nach der Desinfektion verwenden kann. Welche Pipeline ist am schnellsten mit dieser neuen Regelung und lohnt es sich für die Studenten diese zu implementieren? Aus Solidarität muss der letzte Student ebenso alles desinfizieren. Begründen Sie Ihre Antwort und geben Sie alle Rechenschritte an.

Pipelines. **Nehmen Sie für alle nachfolgenden Fragen an, dass die drei Studenten kein Objekt gleichzeitig benutzen wollen.** Die folgenden drei Zeilen zeigen drei mögliche separate Pipelines:

For all of the following question, assume that the three students do not want to use an object at the same time. *The following three lines show three possible separate pipelines:*



(d) Aufgrund von einer Pandemie ist es den Studenten leider nicht erlaubt eines der 4 gemeinsamen Objekte zu benutzen bevor der vorhergehende Student komplett fertig ist (d.h. mit dem "sink" Schritt abgeschlossen hat). Die Studenten haben jedoch die Möglichkeit, jeden Schritt um 10 Minuten zu verlängern um die Objekte zu desinfizieren, damit die nächste Person das Objekt unmittelbar nach der Desinfektion verwenden kann. Welche Pipeline ist am schnellsten mit dieser neuen Regelung und lohnt es sich für die Studenten diese zu implementieren? Aus Solidarität muss der letzte Student ebenso alles desinfizieren. Begründen Sie Ihre Antwort und geben Sie alle Rechenschritte an.

Wenn wir alles sequentiell abarbeiten:

1. $10 + 10 + 15 + 10 = 45 \rightarrow 45 * 3 = 135$
2. $5 + 20 + 5 + 20 = 50 \rightarrow 50 * 3 = 150$
3. $15 + 5 + 15 + 10 = 45 \rightarrow 45 * 3 = 135$

So sehen die Pipelines mit Desinfektion aus:

1. $20 + 20 + 25 + 20 = 85 \rightarrow 85 + 2 * 25 = 135$
2. $15 + 30 + 15 + 30 = 90 \rightarrow 90 + 2 * 30 = 150$
3. $25 + 15 + 25 + 20 = 85 \rightarrow 85 + 2 * 25 = 135$

Da sich die Zeiten nicht unterscheiden, spielt es keine Rolle, ob die Regel implementiert ist oder nicht.


QUIZ

Feedback

- Falls ihr Feedback möchtet sagt mir bitte Bescheid!
- Schreibt mir eine Mail oder auf Discord

Danke

- Bis nächste Woche!