

volatile or Atomic



that is the question



volatile or Atomic



that is the question



Visibility problem

```
boolean flag = true
```

thread-1

thread-2

flag = false

```
• while (flag) {  
    // processing  
}
```



Visibility problem

```
boolean flag = true
```

thread-1

thread-2

flag = false

```
while (flag) {  
    // processing  
}
```



Visibility problem

```
boolean flag = true
```

thread-1

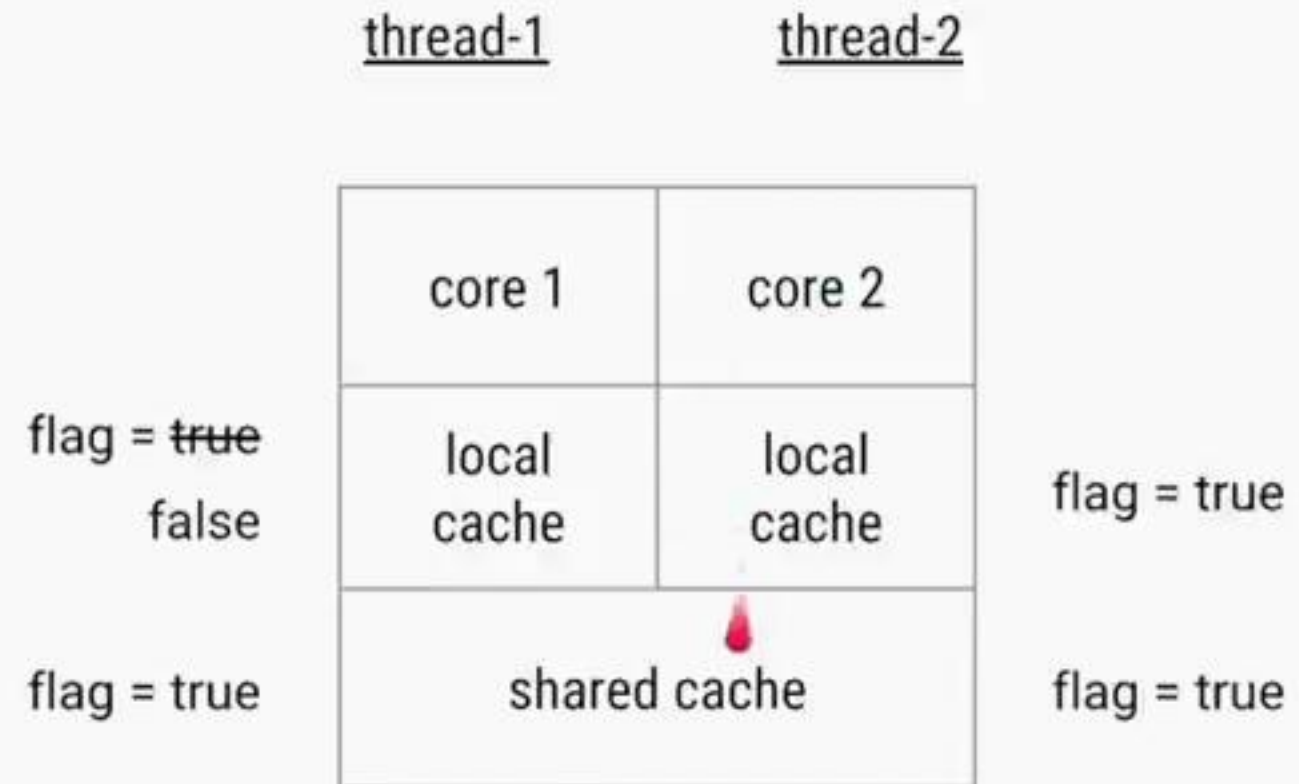
thread-2

flag = false

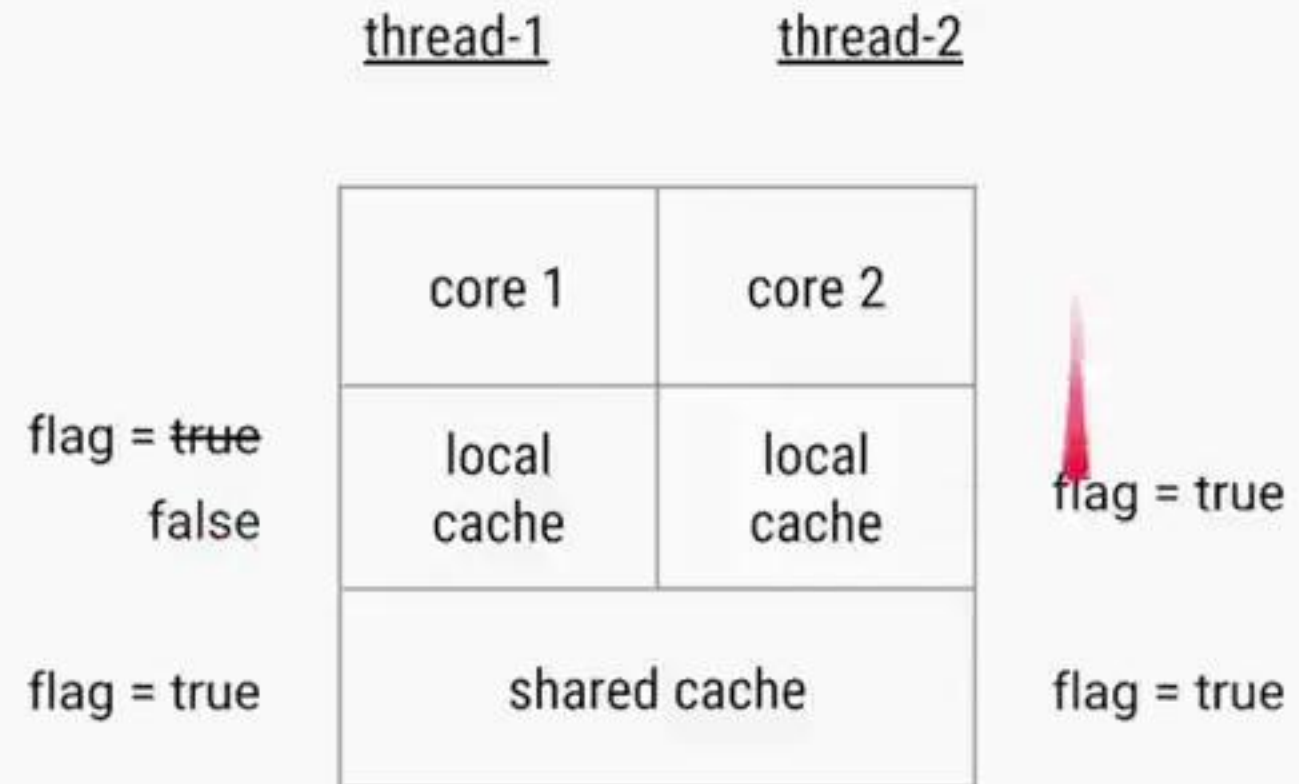
```
while (flag) {  
    // processing  
}
```



Visibility problem



Visibility problem



Visibility problem

```
volatile boolean flag = true
```

thread-1

thread-2

flag = false

```
while (flag) {  
    // processing  
}
```



Visibility problem

```
volatile boolean flag = true
```

thread-1

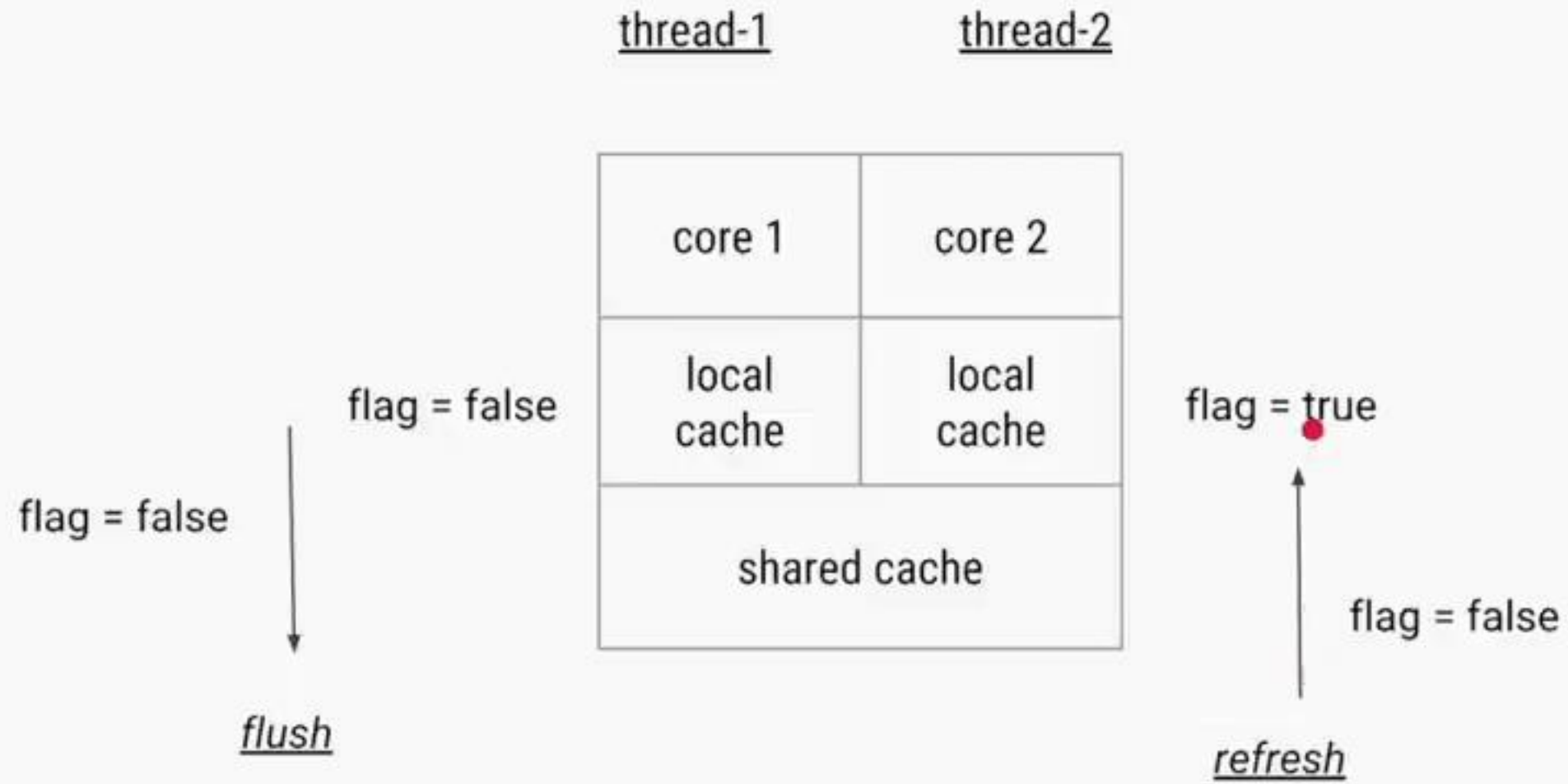
thread-2

flag = false

```
while (flag) {  
    // processing  
}
```



Visibility problem




Synchronization problem

```
int value = 1;
```


thread-1

value++

A vertical line with a downward-pointing arrowhead at the bottom, representing the execution flow of thread-1.

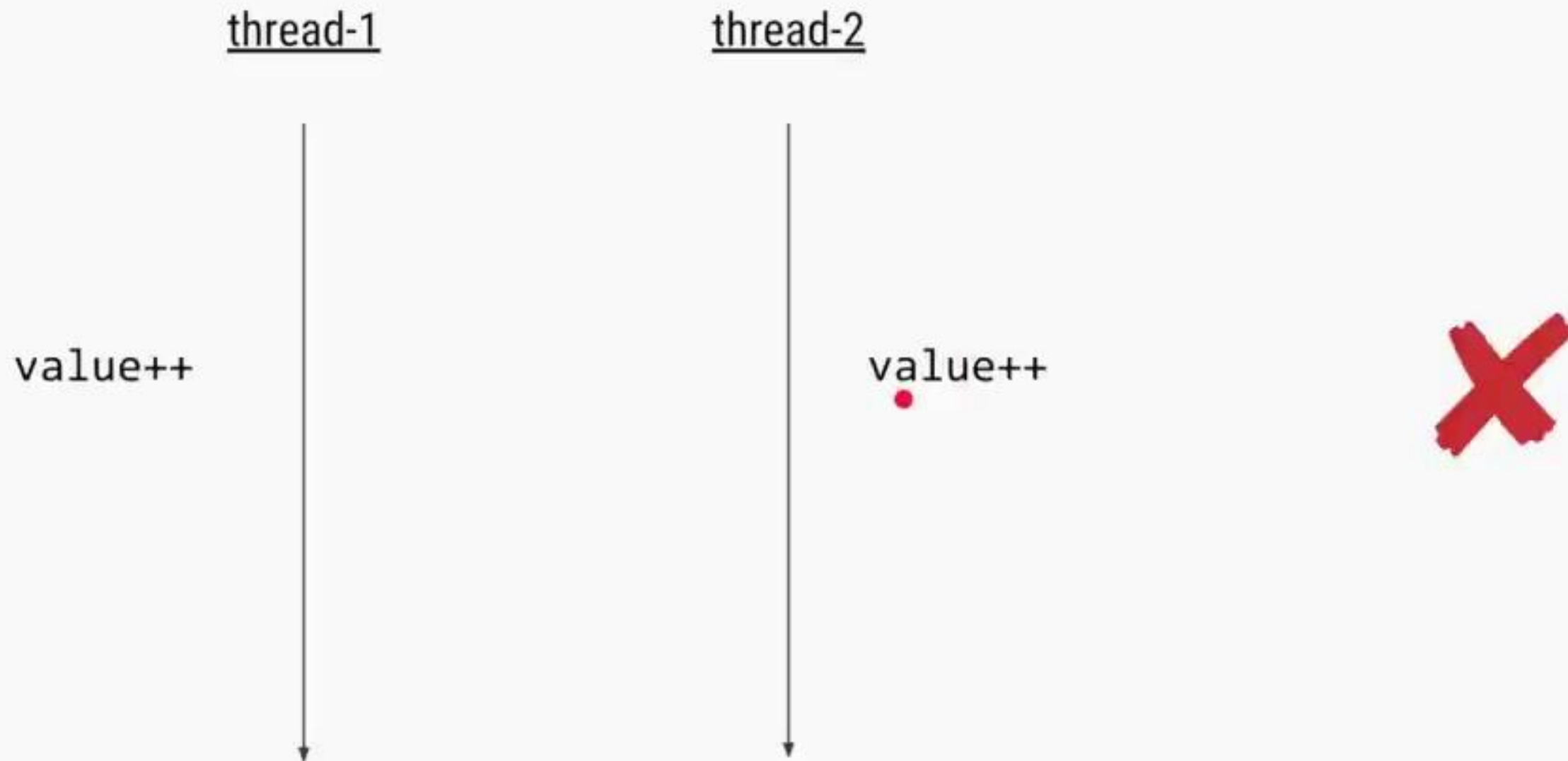
thread-2

value++

A vertical line with a downward-pointing arrowhead at the bottom, representing the execution flow of thread-2.

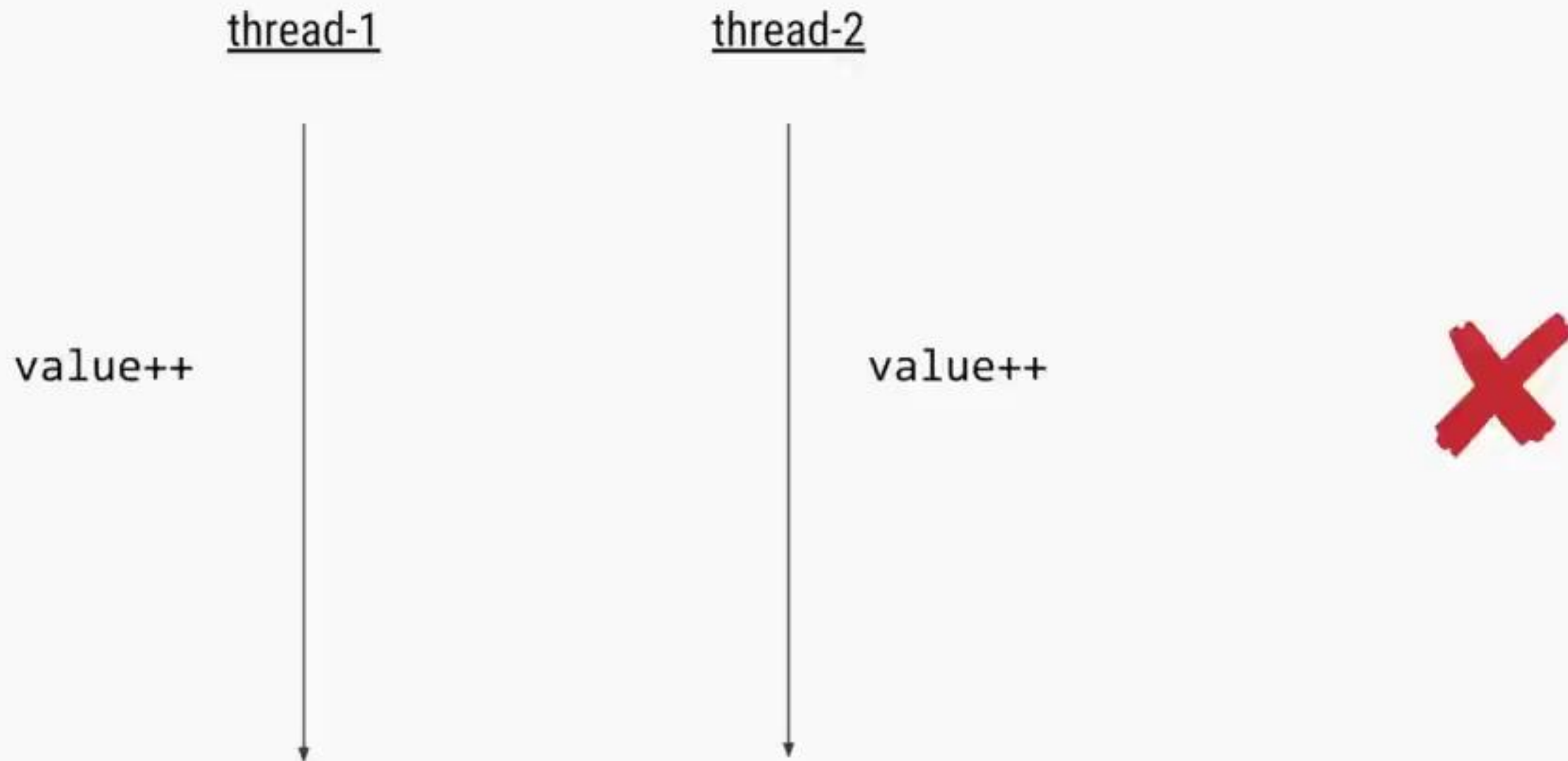
Synchronization problem

```
int value = 1;
```



Synchronization problem

```
volatile int value = 1;
```



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization problem

```
volatile int value = 1;
```

Even with volatile

#	Thread-1	Thread-2
1	Read value (=1)	
2		Read value (=1)
3	Add 1 and write (=2)	
4		Add 1 and write (=2)



Synchronization solutions - #1

```
volatile int value = 1;
```

thread-1

thread-2

```
synchronized (obj) {  
    value++;  
}
```



```
synchronized (obj) {  
    value++;  
}
```



Synchronization solutions - #1

```
volatile int value = 1;
```

thread-1

thread-2

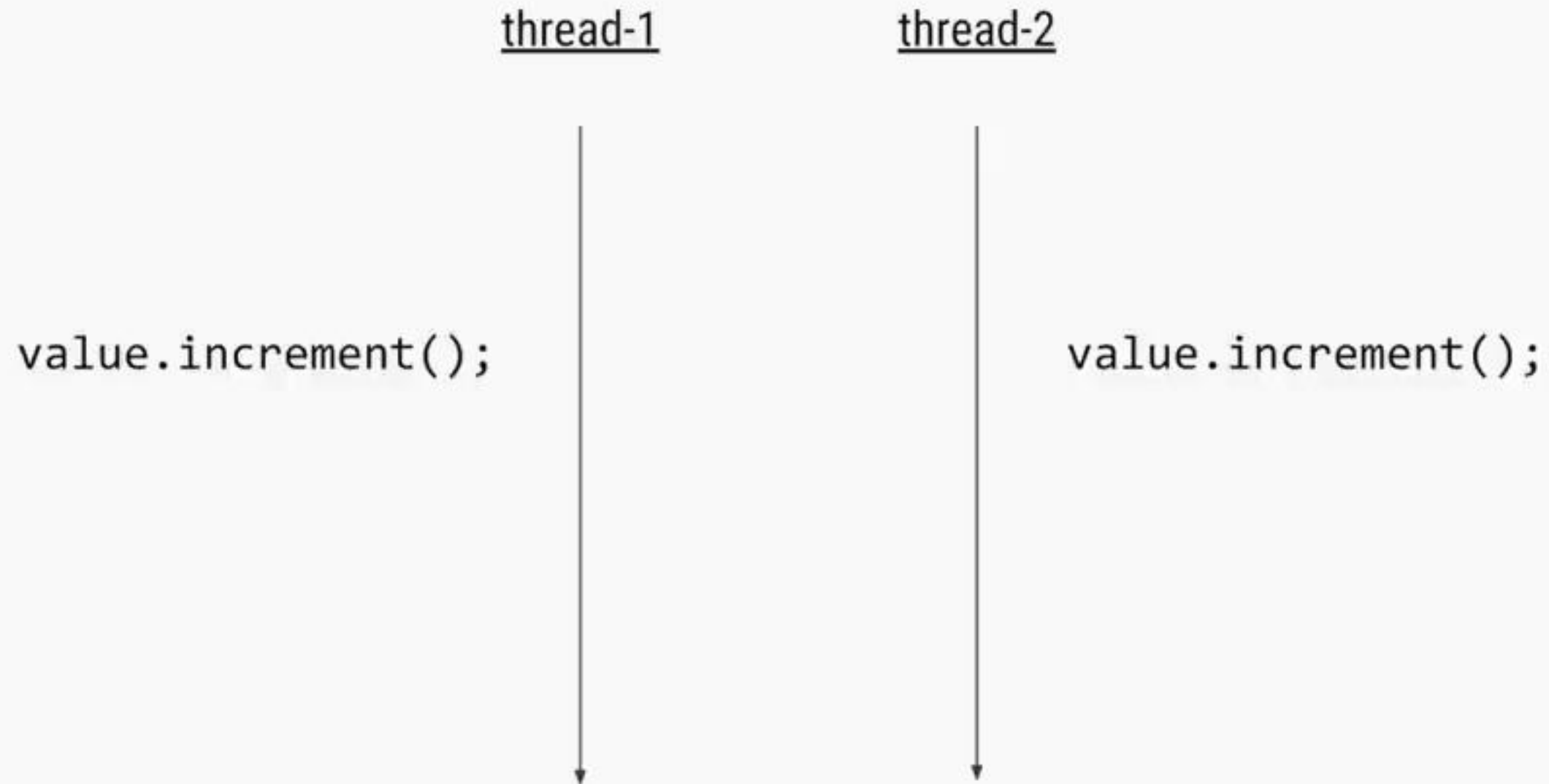
```
synchronized (obj) {  
    value++;  
}
```

```
synchronized (obj) {  
    value++;  
}
```



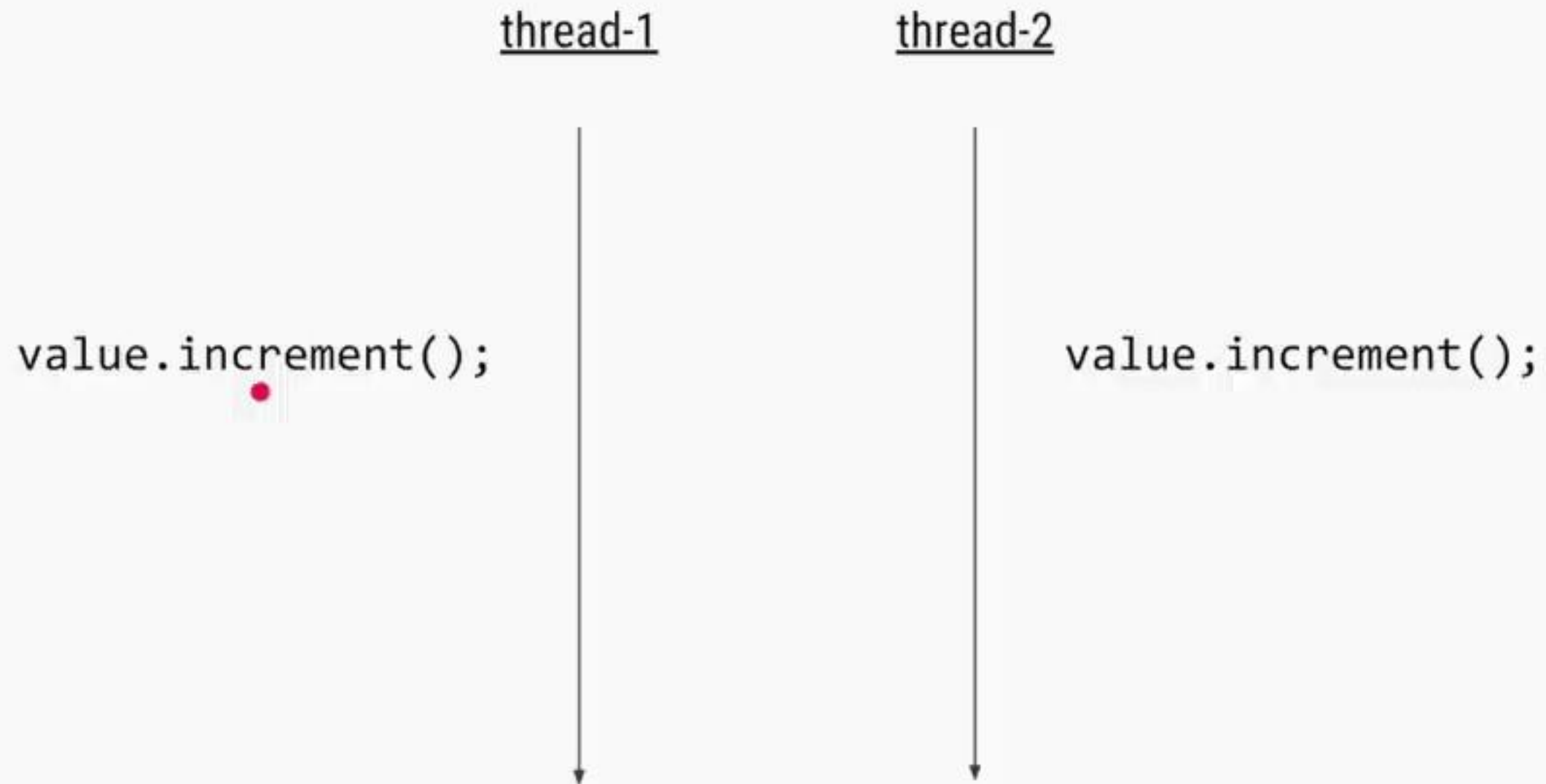
Synchronization solutions - #1

```
AtomicInteger value = new AtomicInteger(1);
```



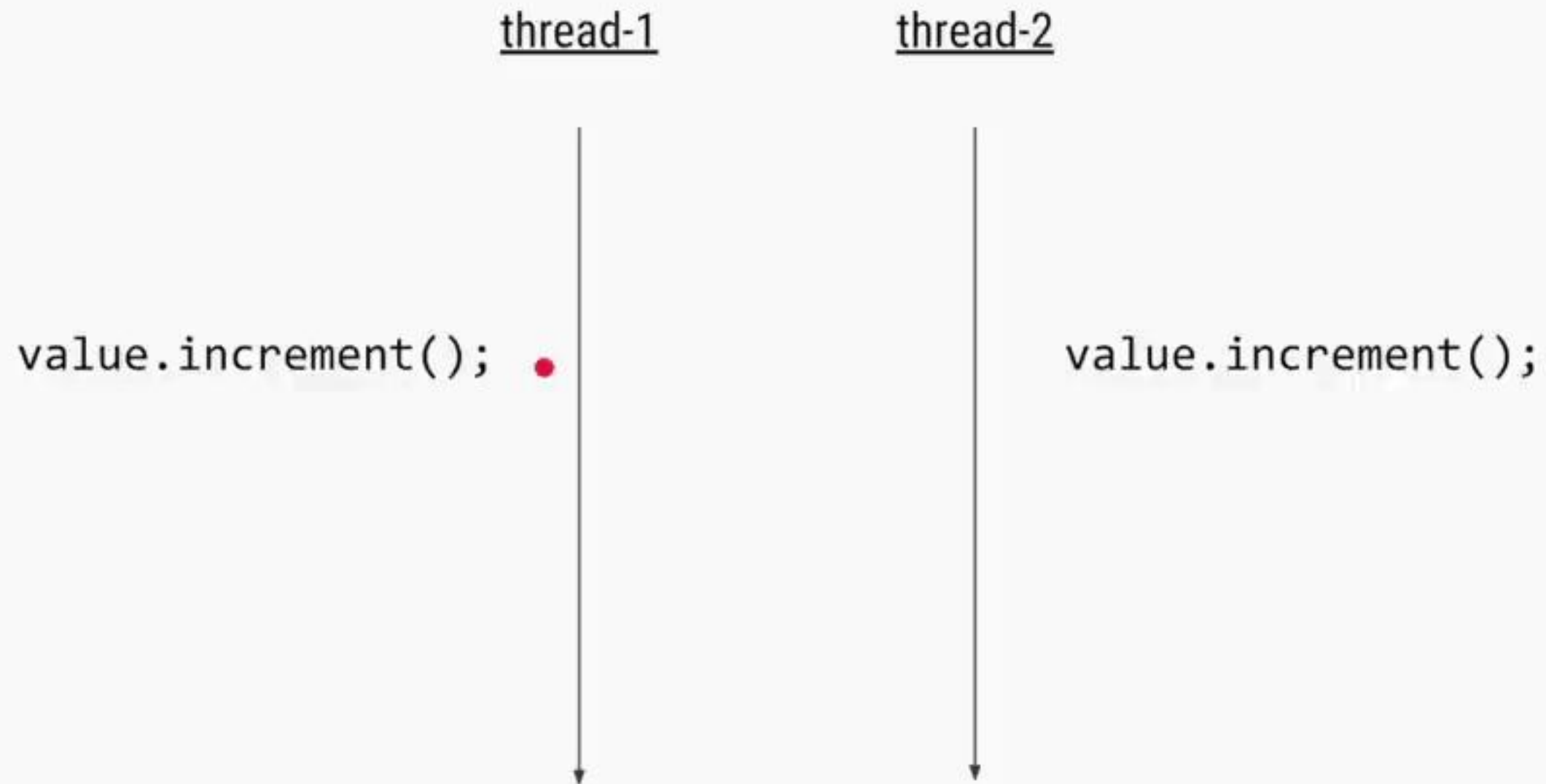
Synchronization solutions - #1

```
AtomicInteger value = new AtomicInteger(1);
```



Synchronization solutions - #1

```
AtomicInteger value = new AtomicInteger(1);
```



Many methods for various compound operations

- `incrementAndGet`
- `decrementAndGet`
- `addAndGet (int delta)`
- `compareAndSet (int expectedValue, int newValue)`

Many methods for various compound operations

- `incrementAndGet`
- `decrementAndGet`
- `addAndGet (int delta)`
- `compareAndSet (int expectedValue, int newValue)`

Compound
Operations

Atomic
variables

Typical Use Cases

Type	Use Case
volatile	Flags
AtomicInteger AtomicLong	Counters
AtomicReference	Caches (building new cache in background and replacing atomically) Used by some internal classes Non-blocking algorithms

Typical Use Cases

Type	Use Case
volatile	Flags
AtomicInteger AtomicLong	Counters
AtomicReference	Caches (building new cache in background and replacing atomically) Used by some internal classes Non-blocking algorithms

Typical Use Cases

Type	Use Case
volatile	Flags
AtomicInteger AtomicLong	Counters
AtomicReference	Caches (building new cache in background and replacing atomically) Used by some internal classes Non-blocking algorithms

Thank you!

CONVERT YOUTUBE VIDEO INTO JPG/PDF IMAGES

WWW.GALLERYMKER.COM