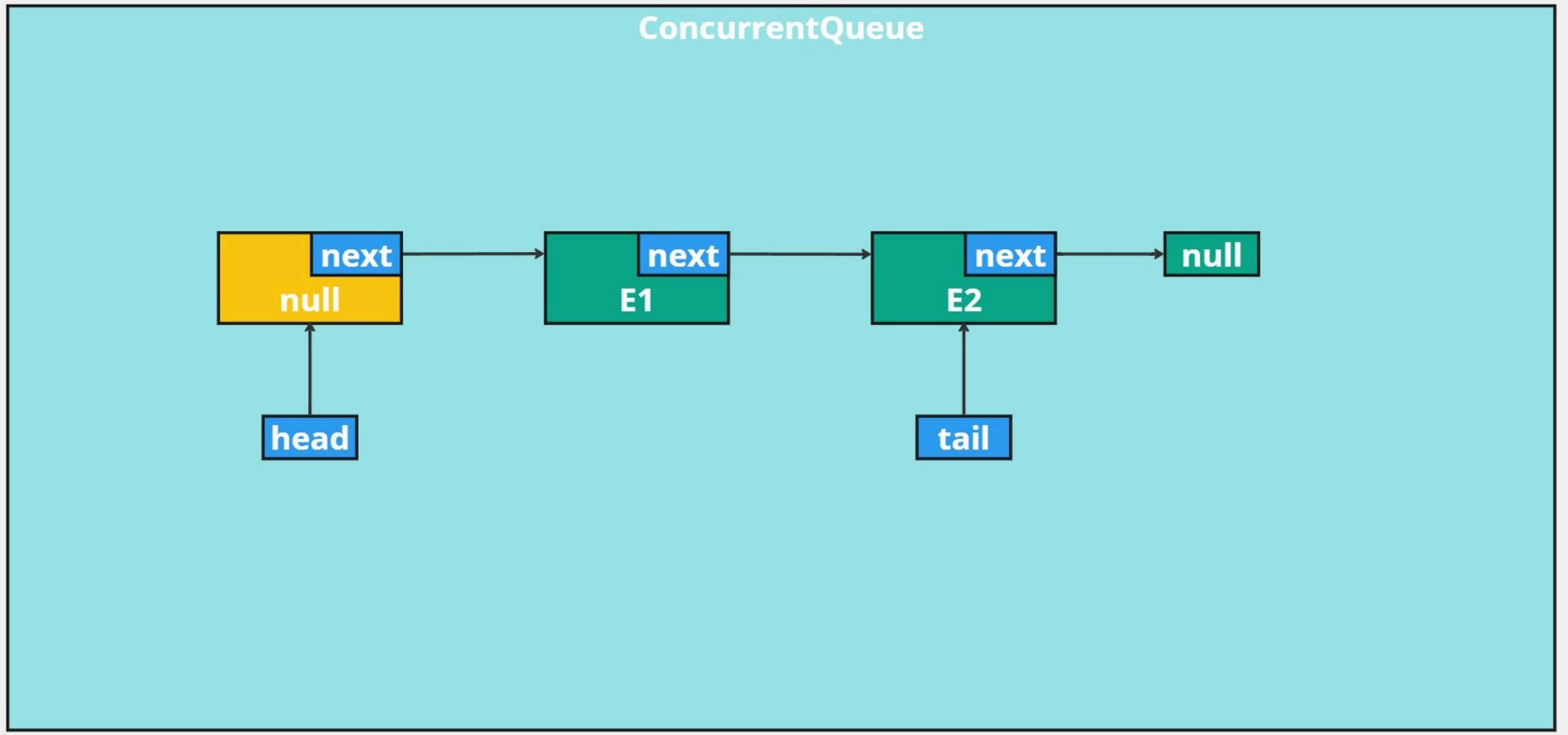
JAVA

МНОГОПОТОЧНОСТЬ.
ОЧЕРЕДЬ МАЙКЛА-СКОТТА



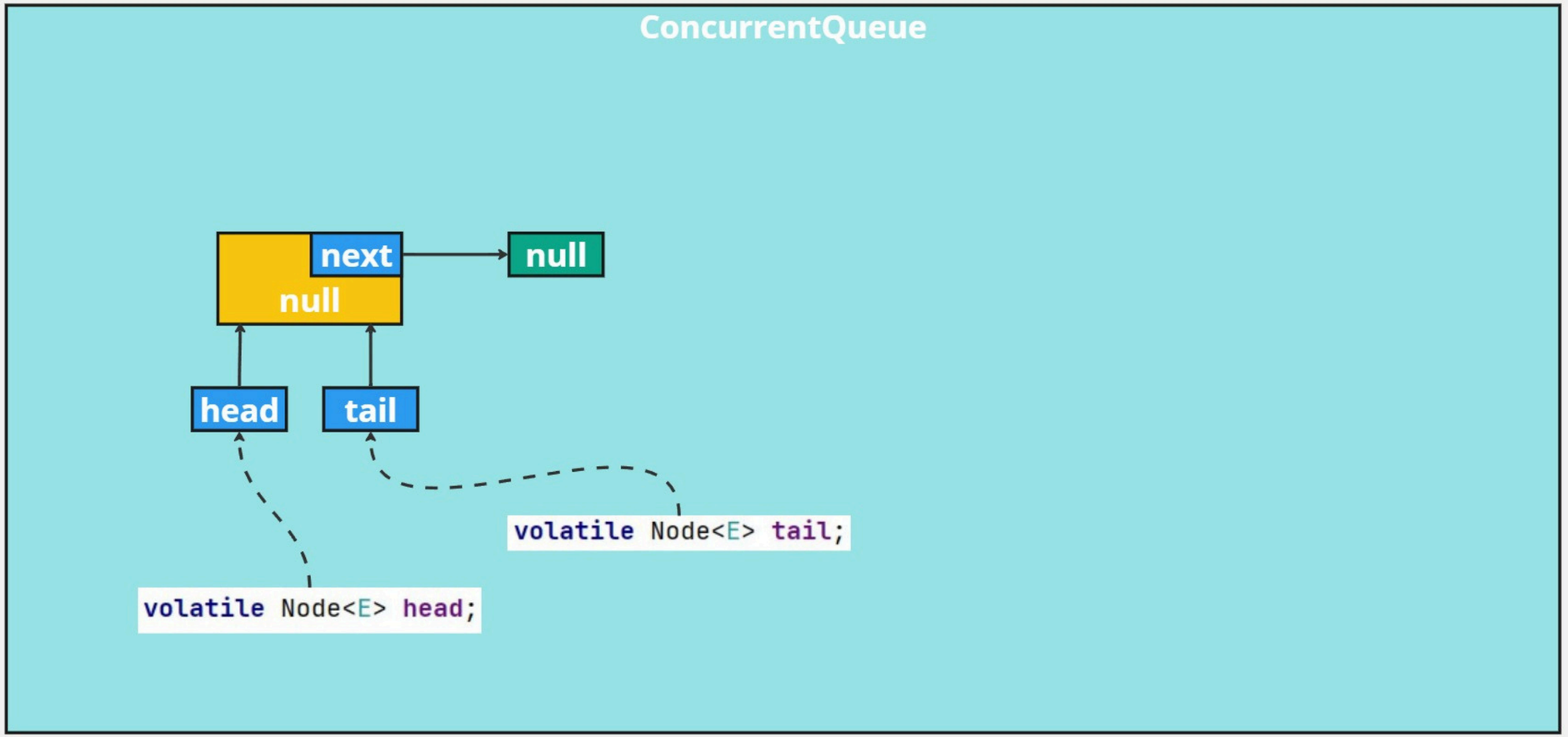
synchronized

ReentrantLock

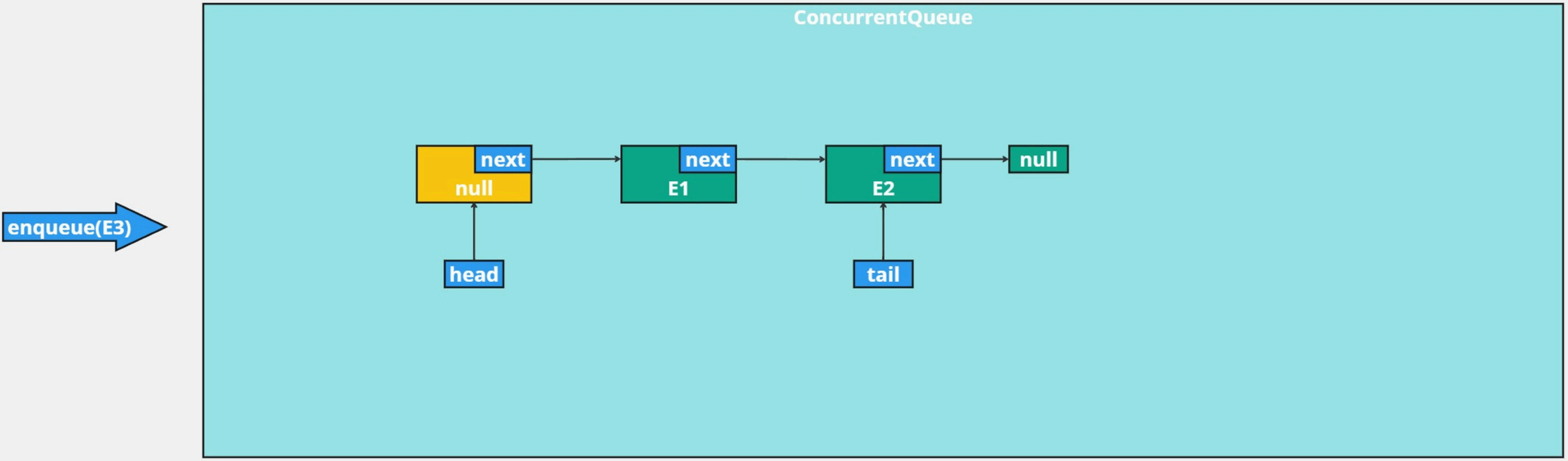
volatile

AtomicReference<E>

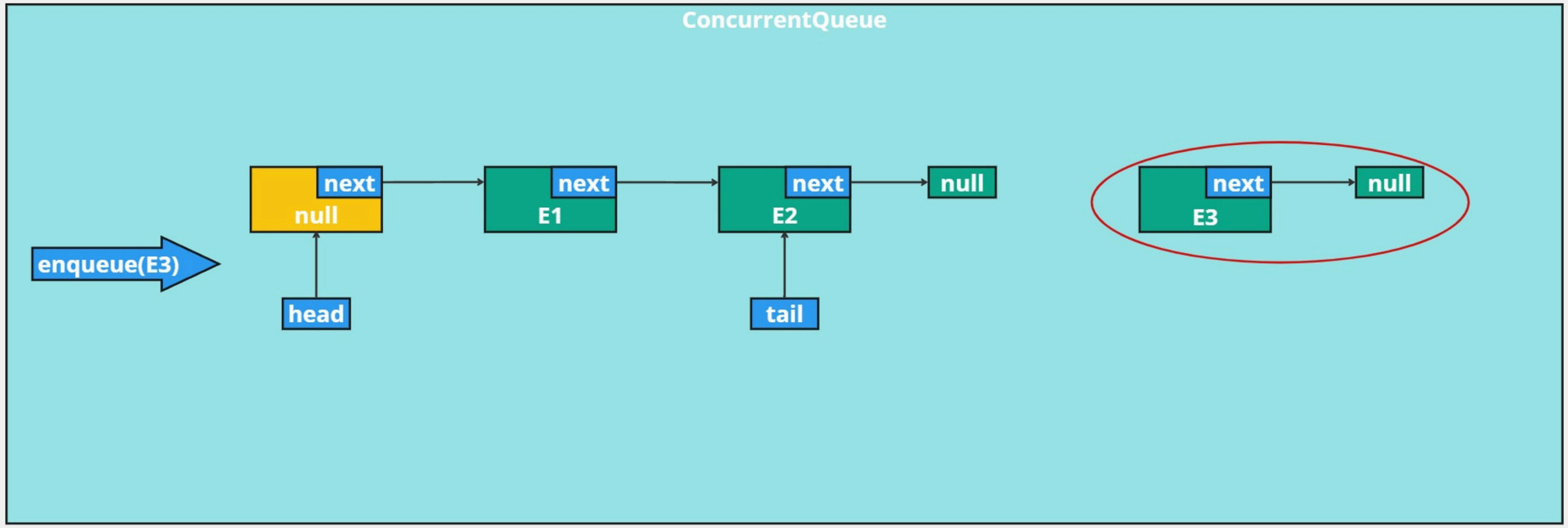
```java
private static final class Node<E> {
    1 usage
    private volatile E value;
    no usages
    private volatile Node<E> next;

    1 usage
    public Node() {


    }

    no usages
    public Node(final E value) {
        this.value = value;
    }
}
```



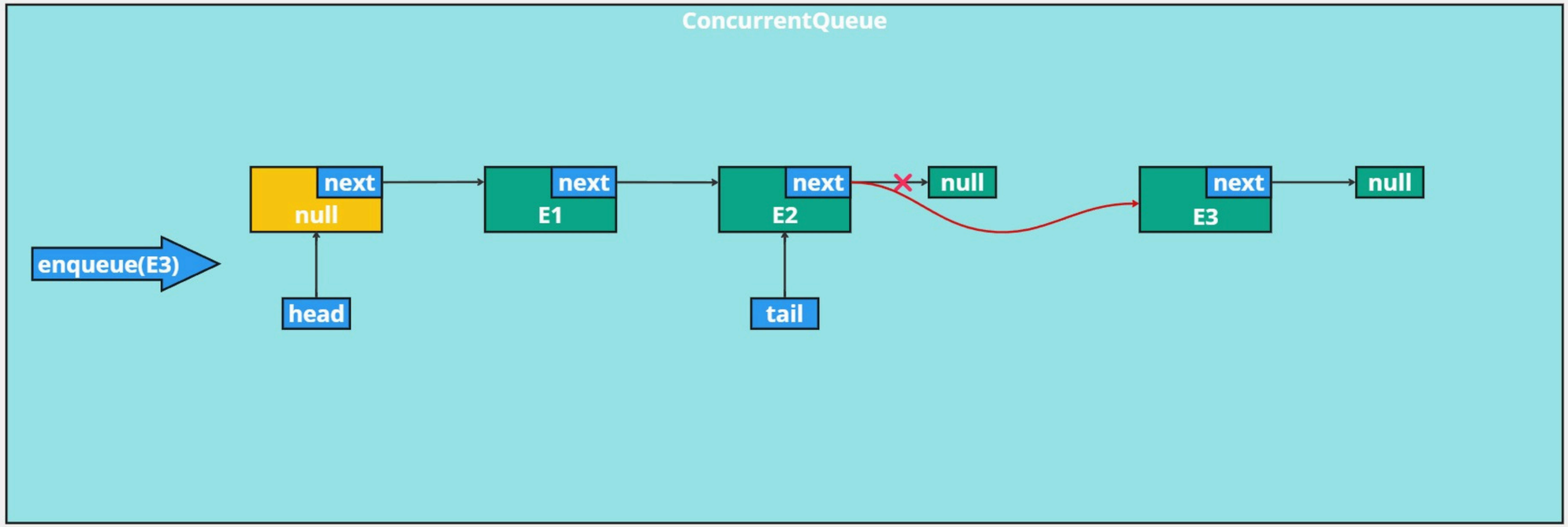

```
public ConcurrentQueue() {
    final Node<E> dummyNode = new Node<>();
    head = dummyNode;
    tail = dummyNode;
}
```



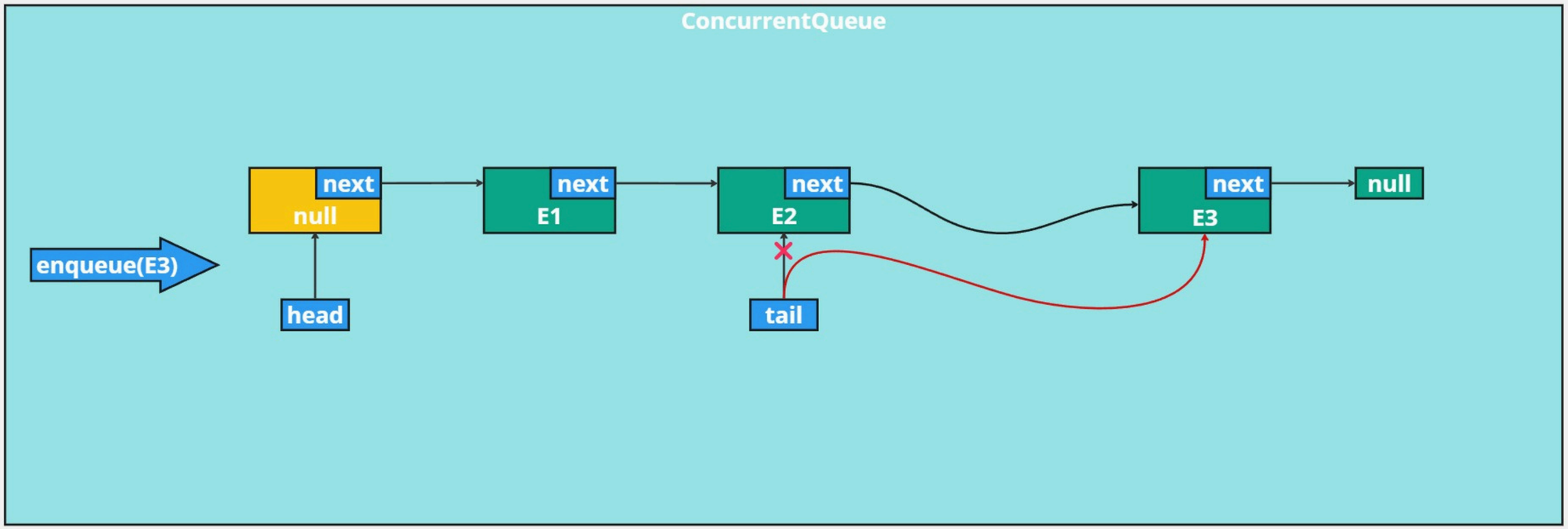

```
public void enqueue(final E element) {
    //create new node
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```



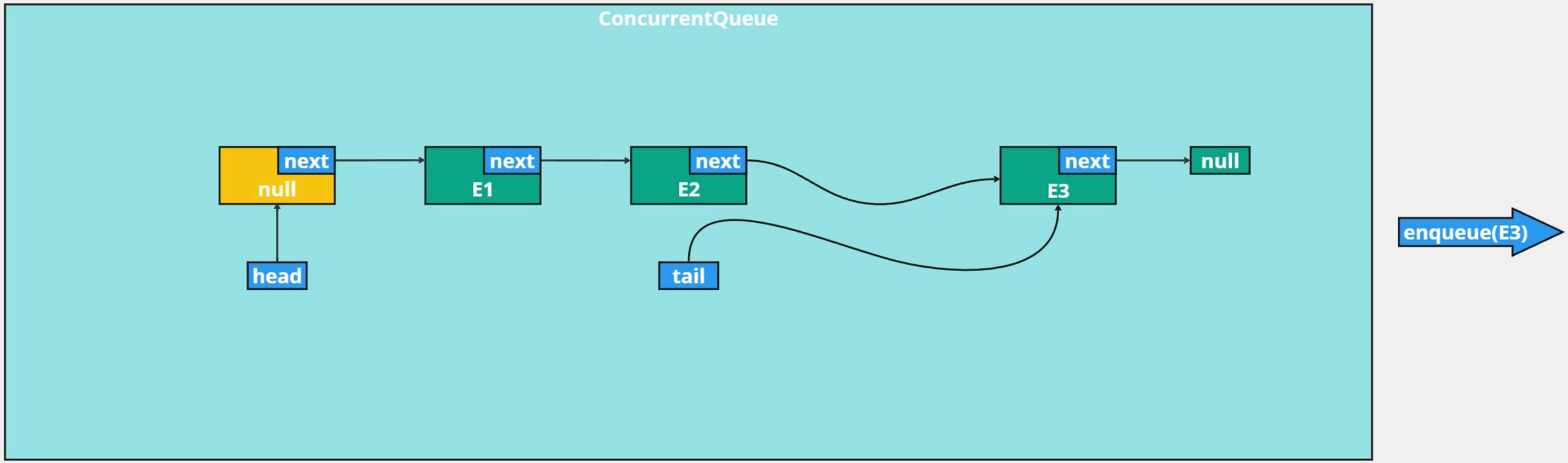

```
public void enqueue(final E element) {
    //create new node
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```



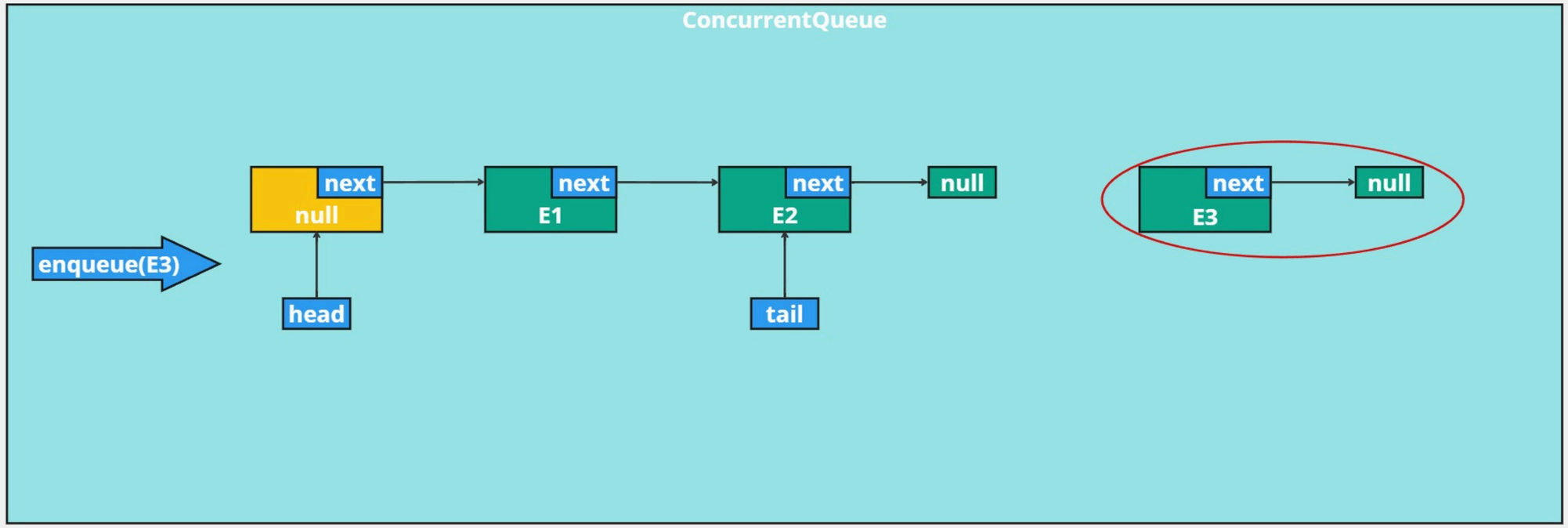

```
public void enqueue(final E element) {
    //create new node
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```



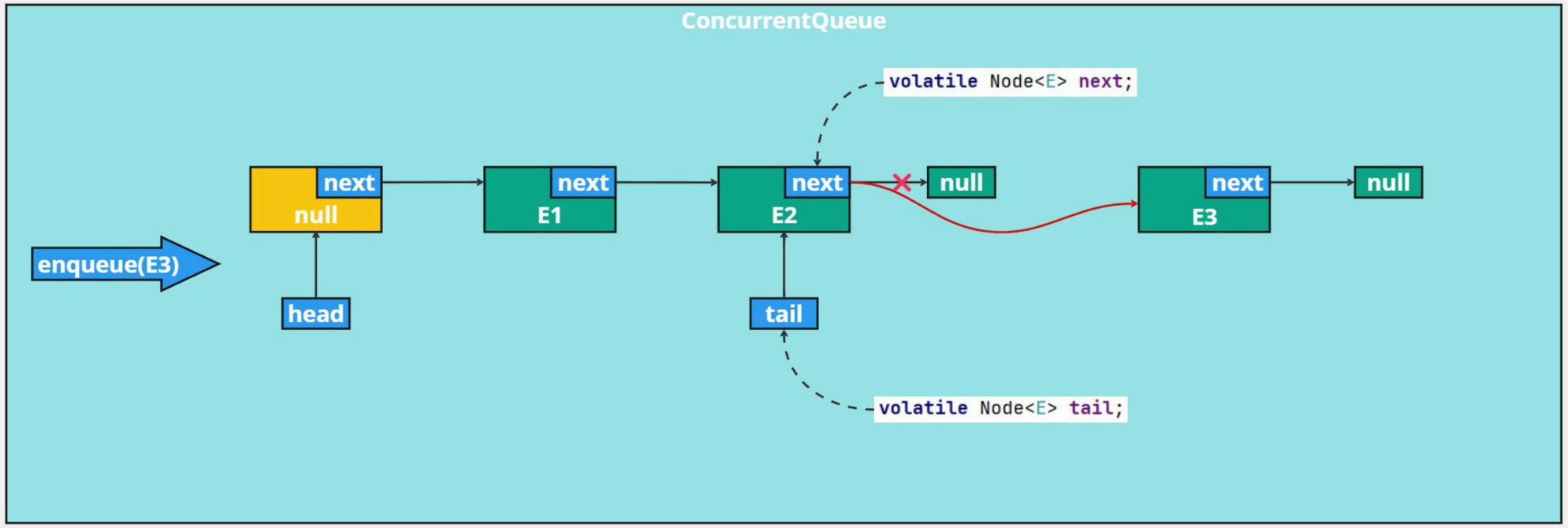

```
public void enqueue(final E element) {
    //create new node
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```



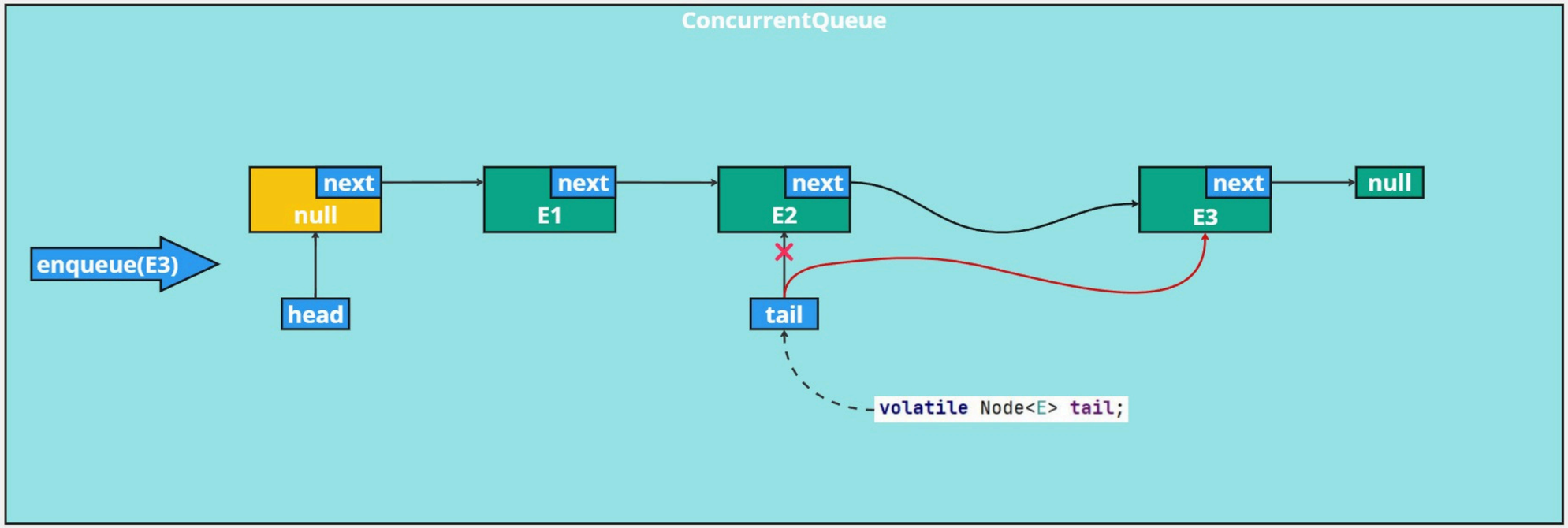

```
public void enqueue(final E element) {
    //create new node
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```



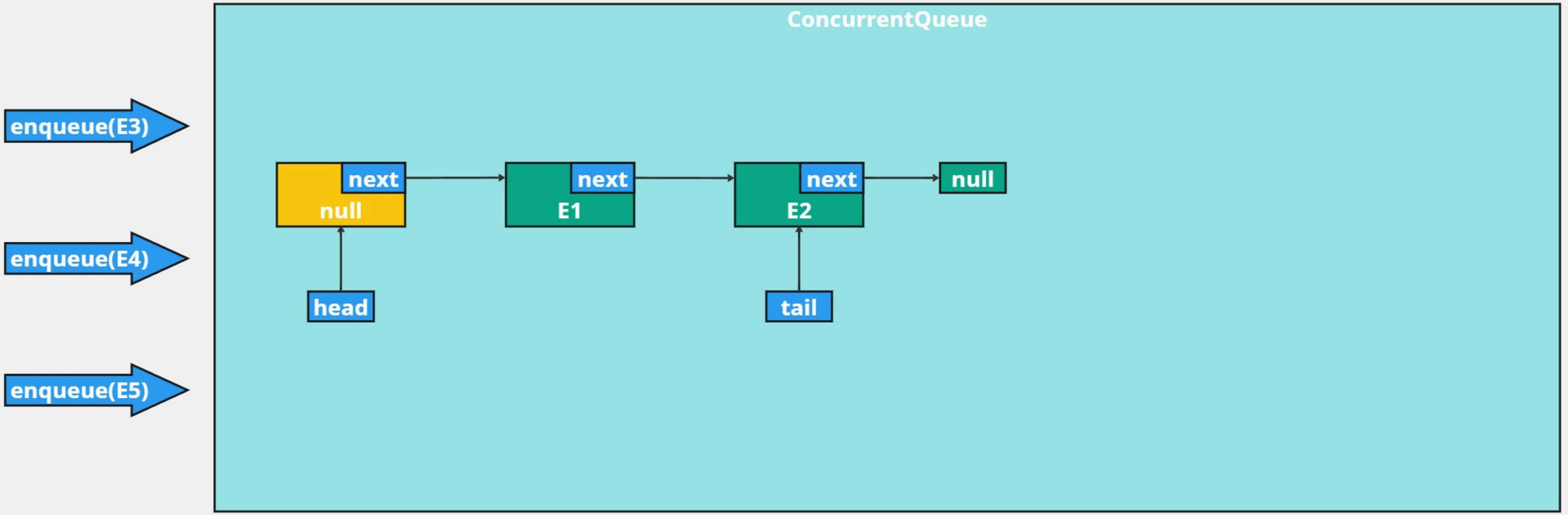

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    //tail.next should be assigned new node
    //tail should be assigned new node
}
```




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    //tail should be assigned new node
}
```




```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

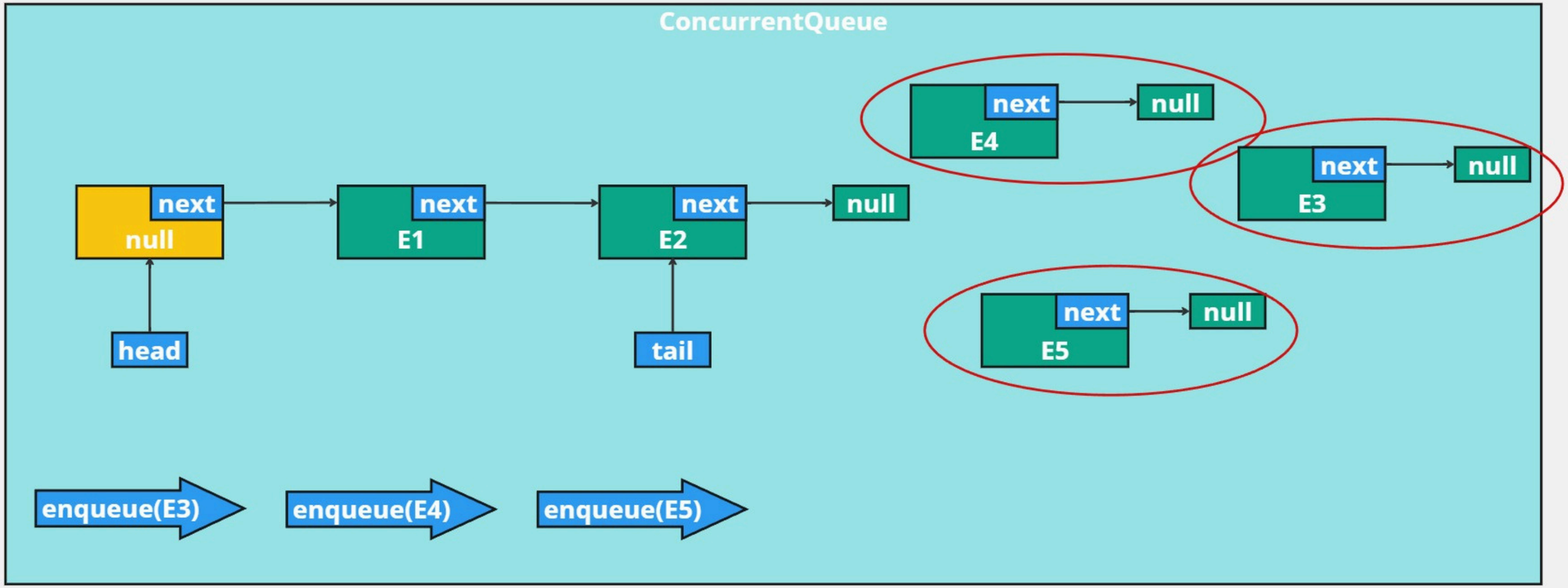
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

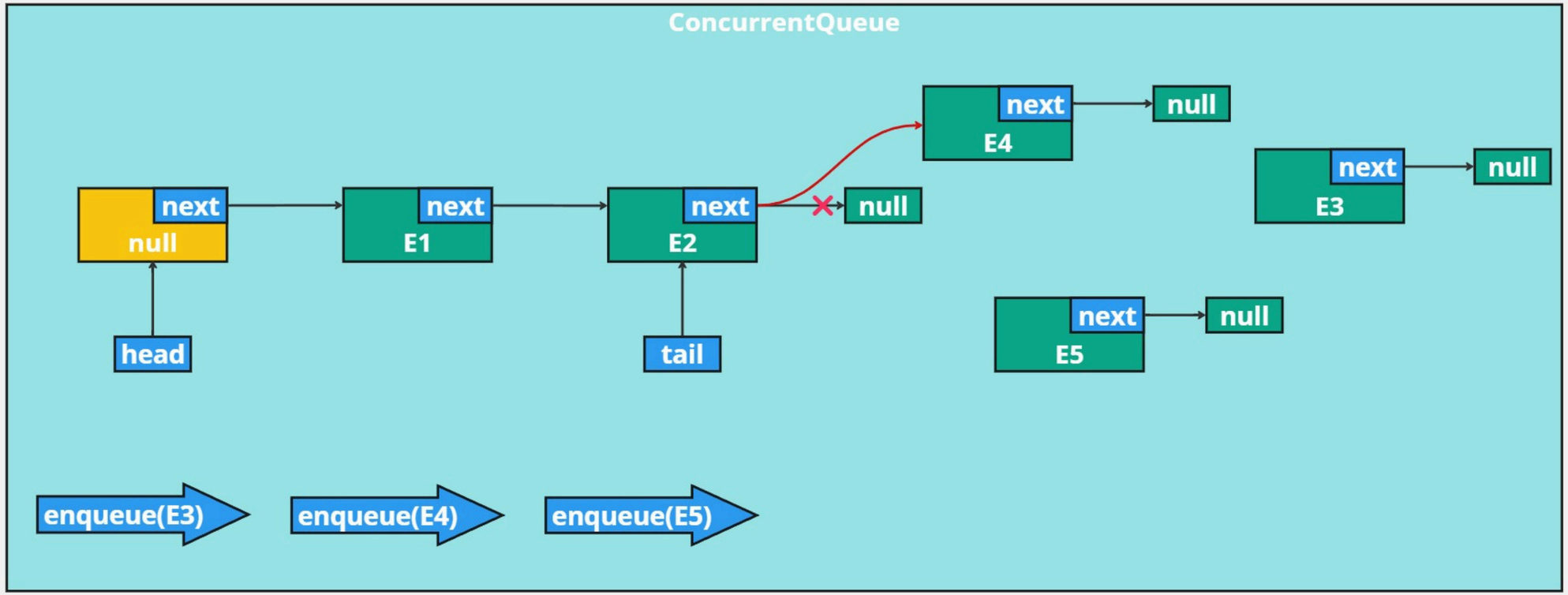
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

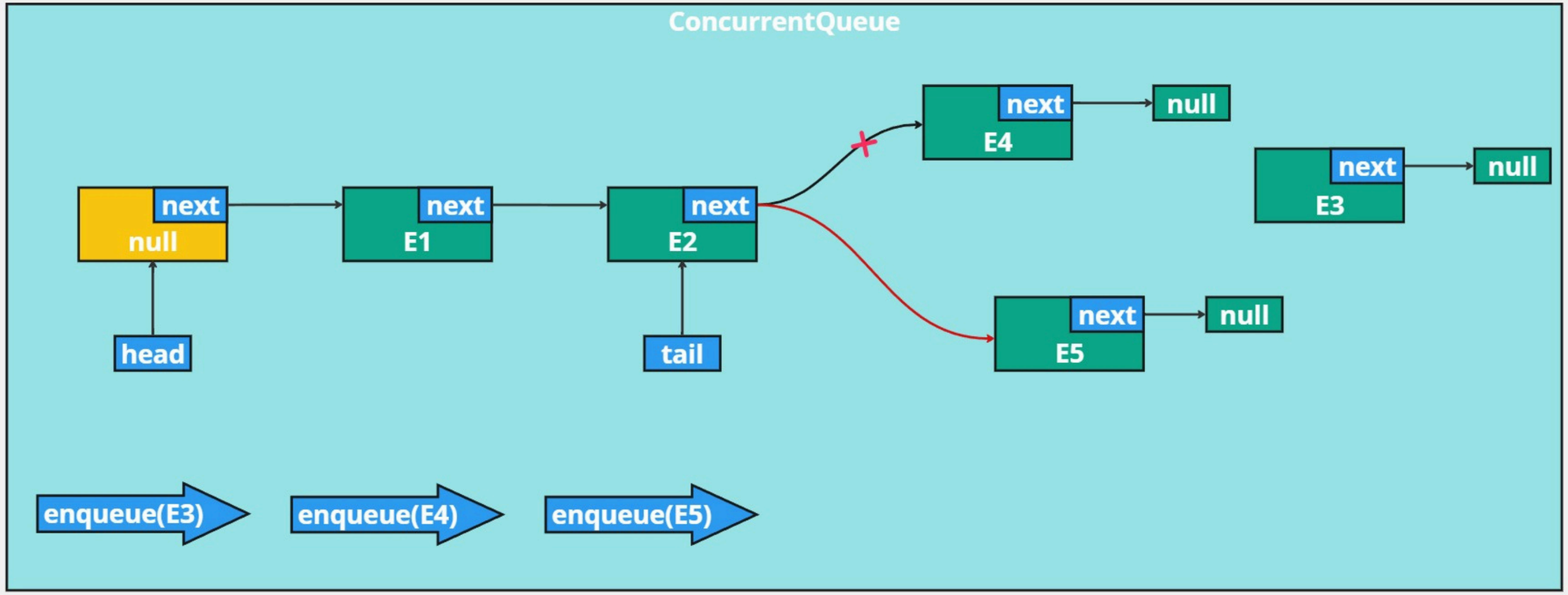
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

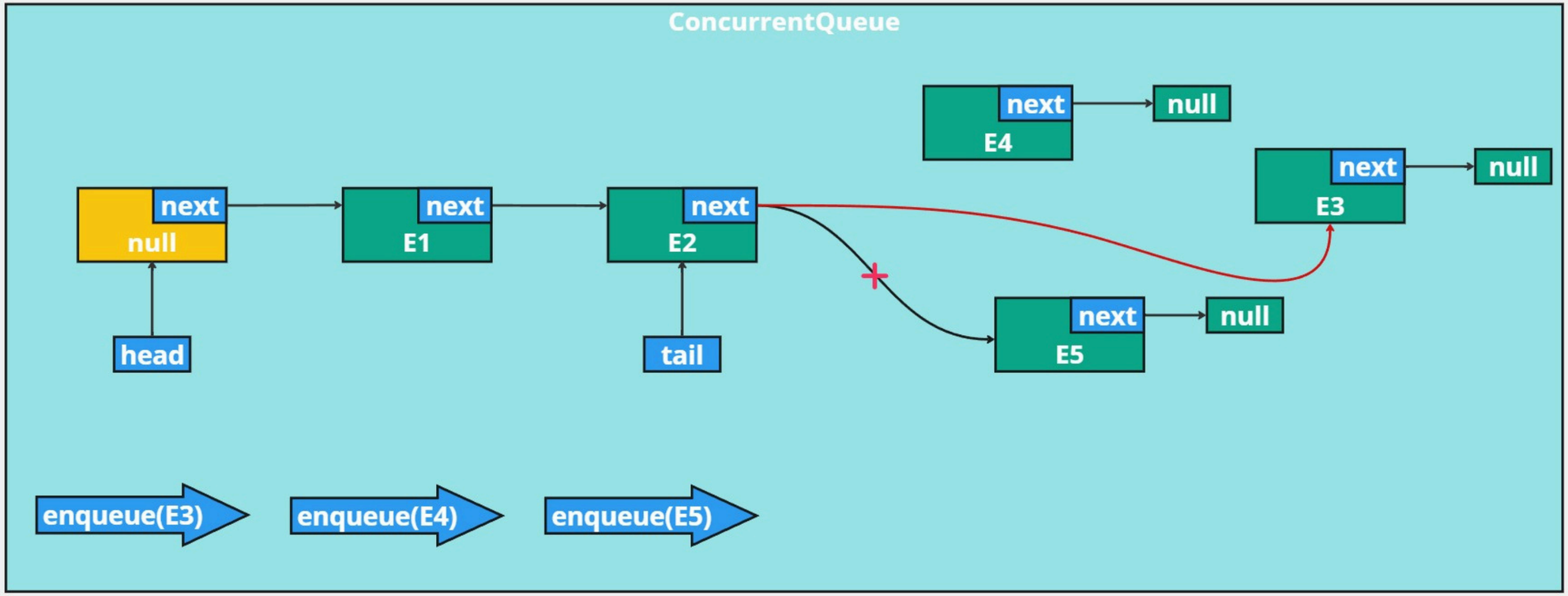
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

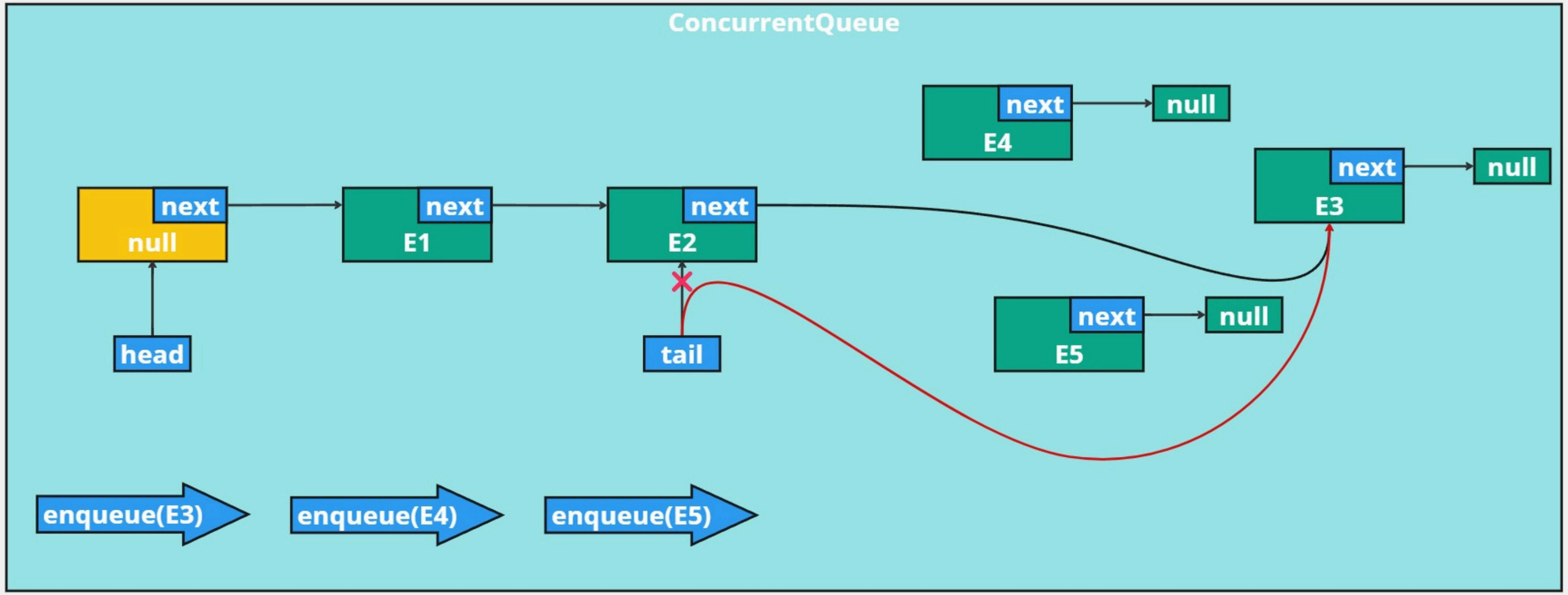
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

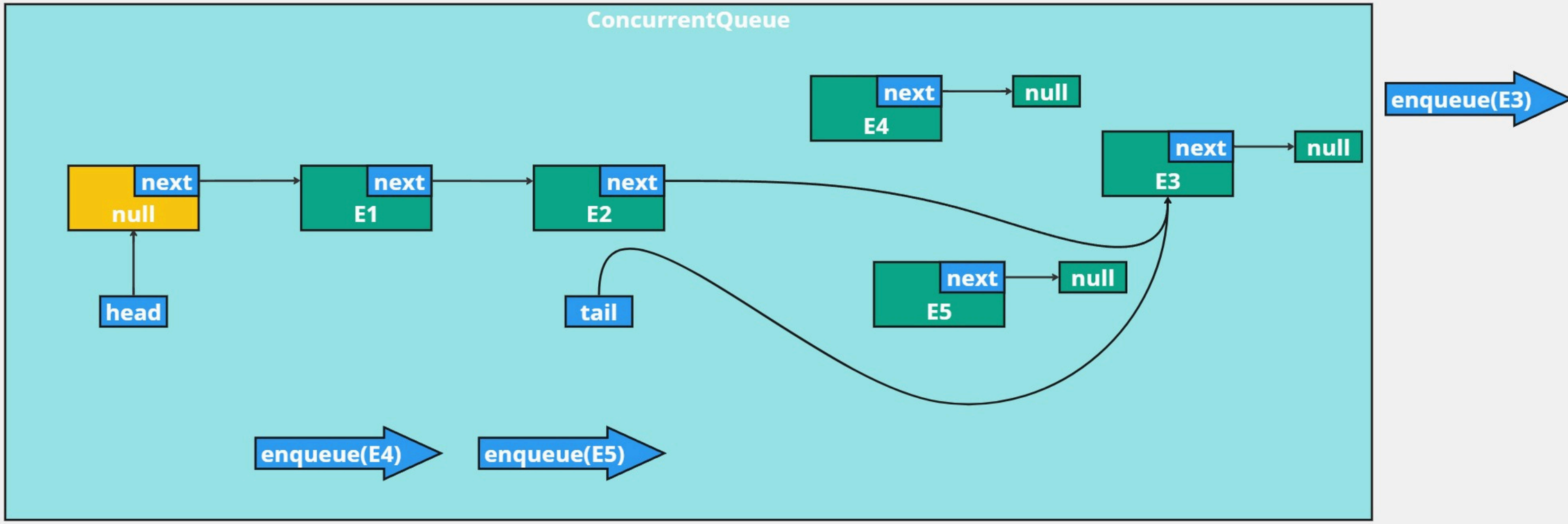
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

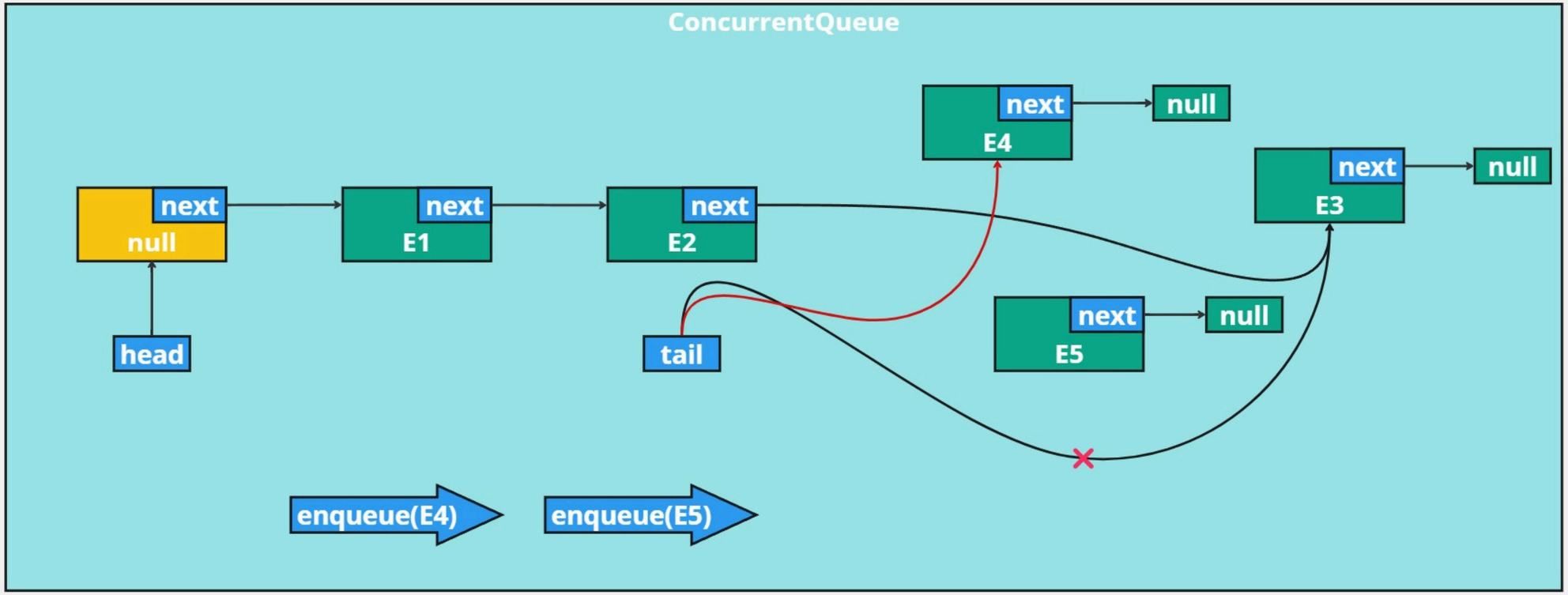
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

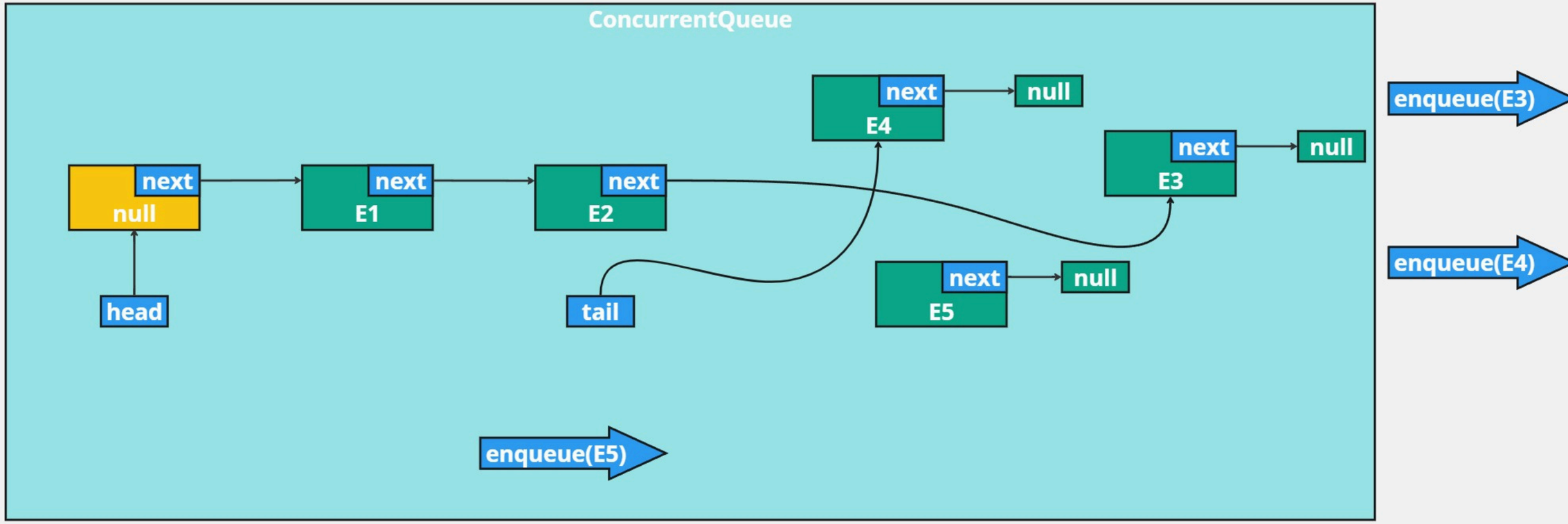
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

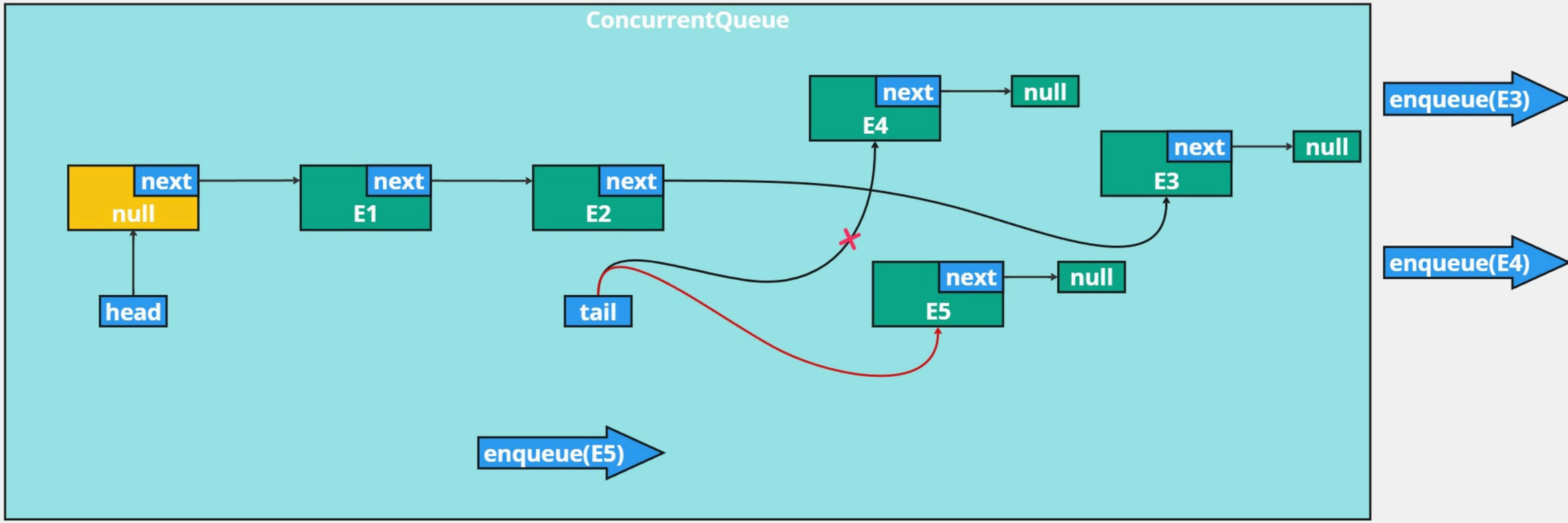
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

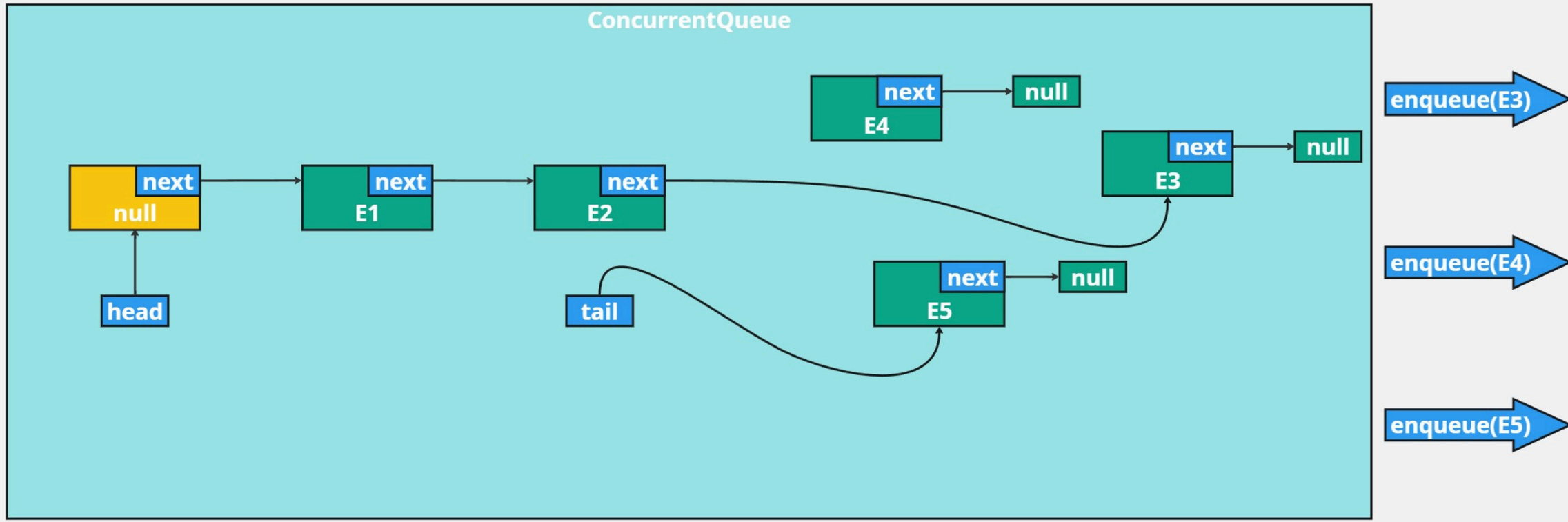
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

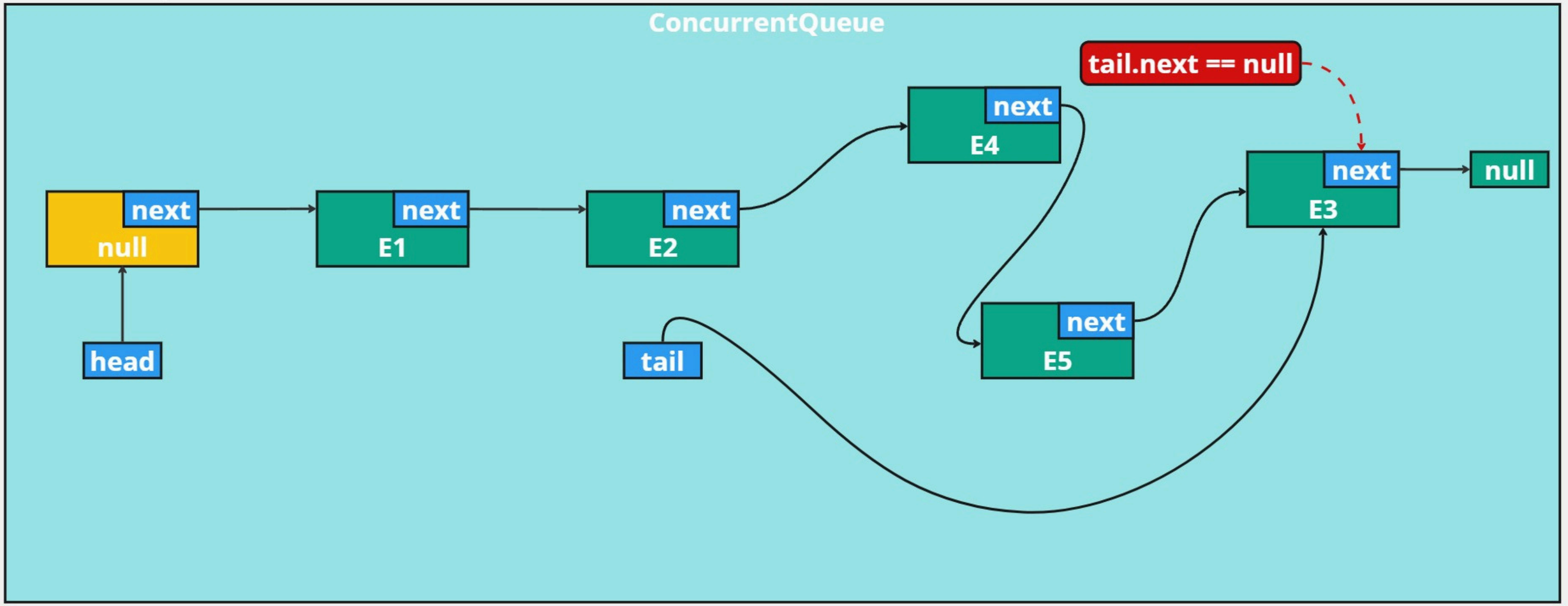
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

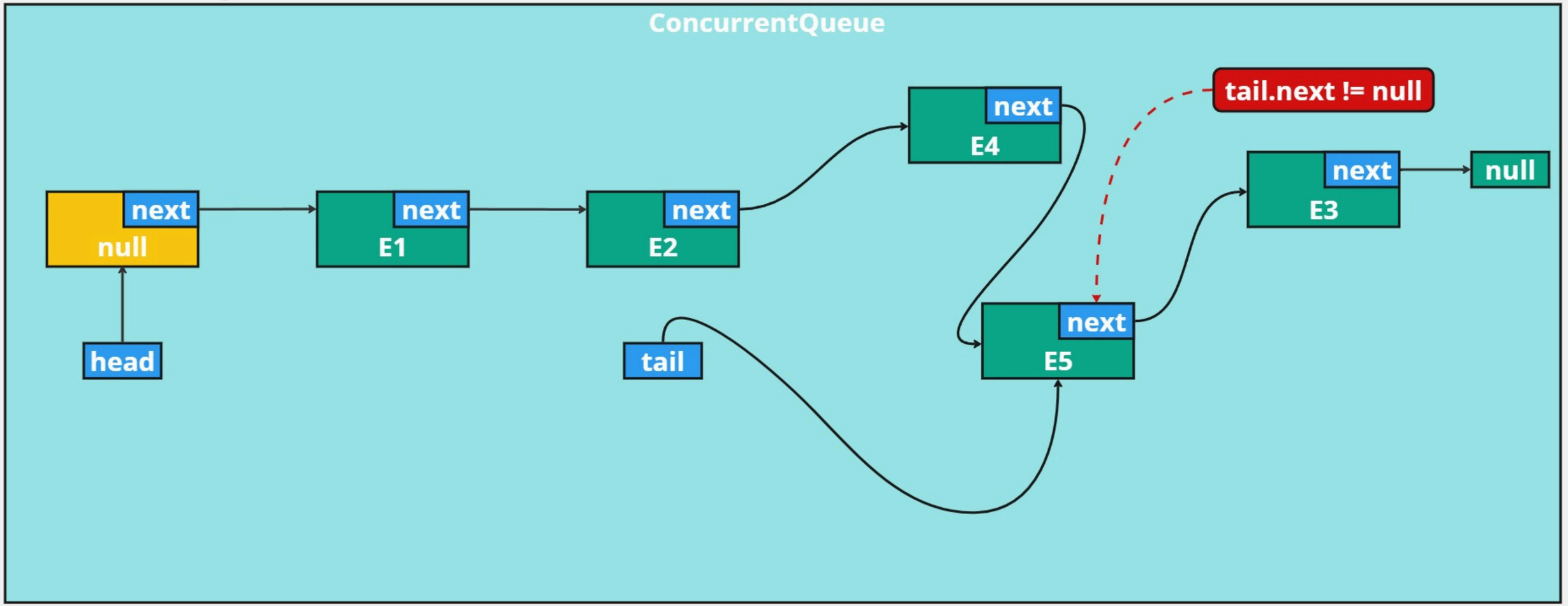
enqueue(E5)

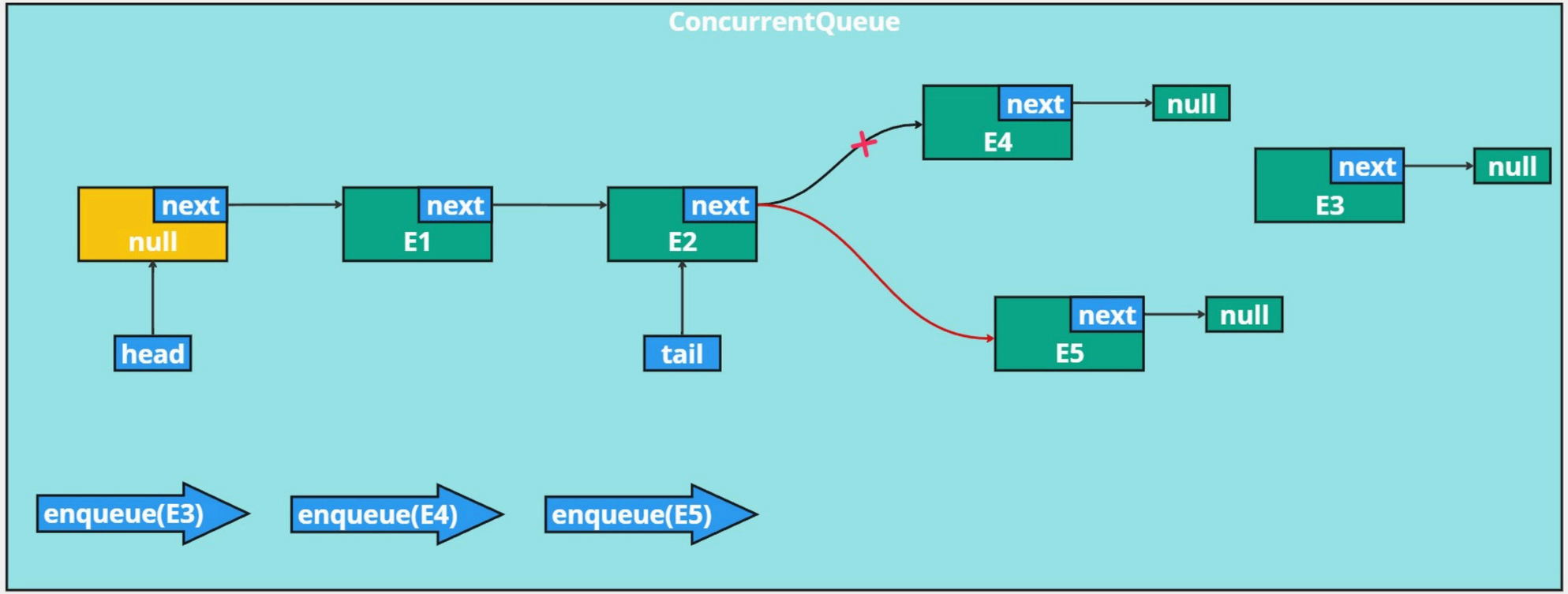
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

Consistency state



Not consistency state





enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

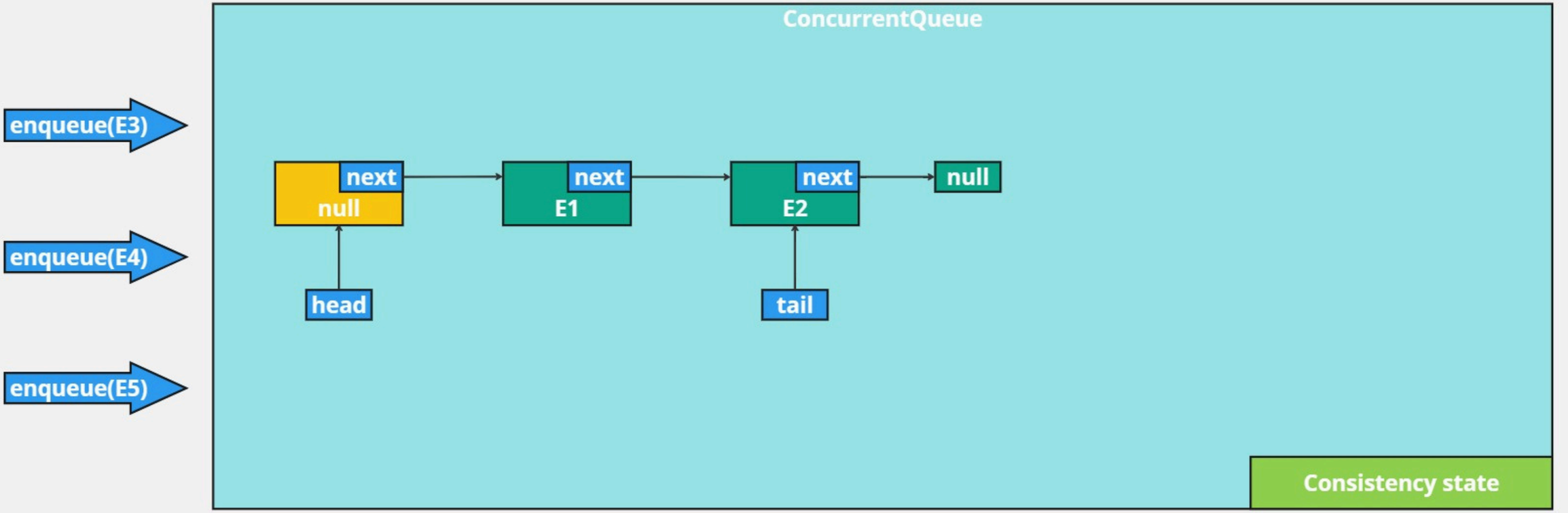
enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

tail.next != null

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    tail.next = newNode;
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

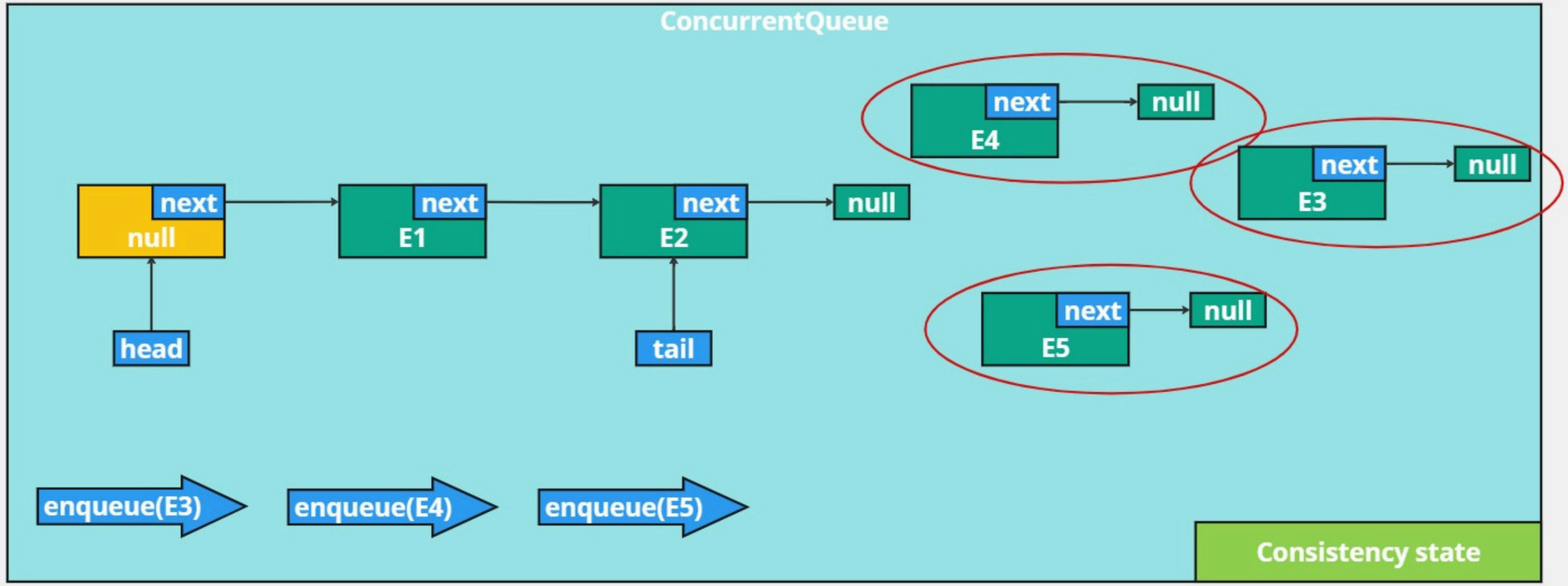
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

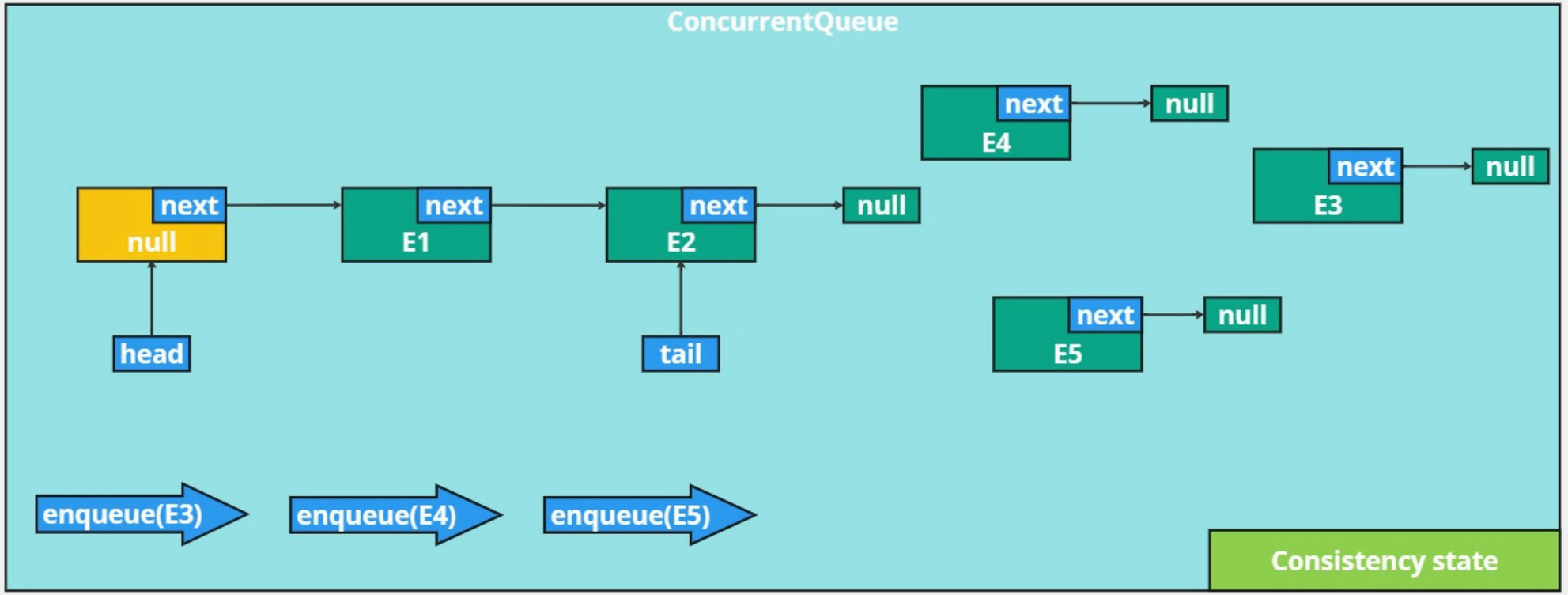
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

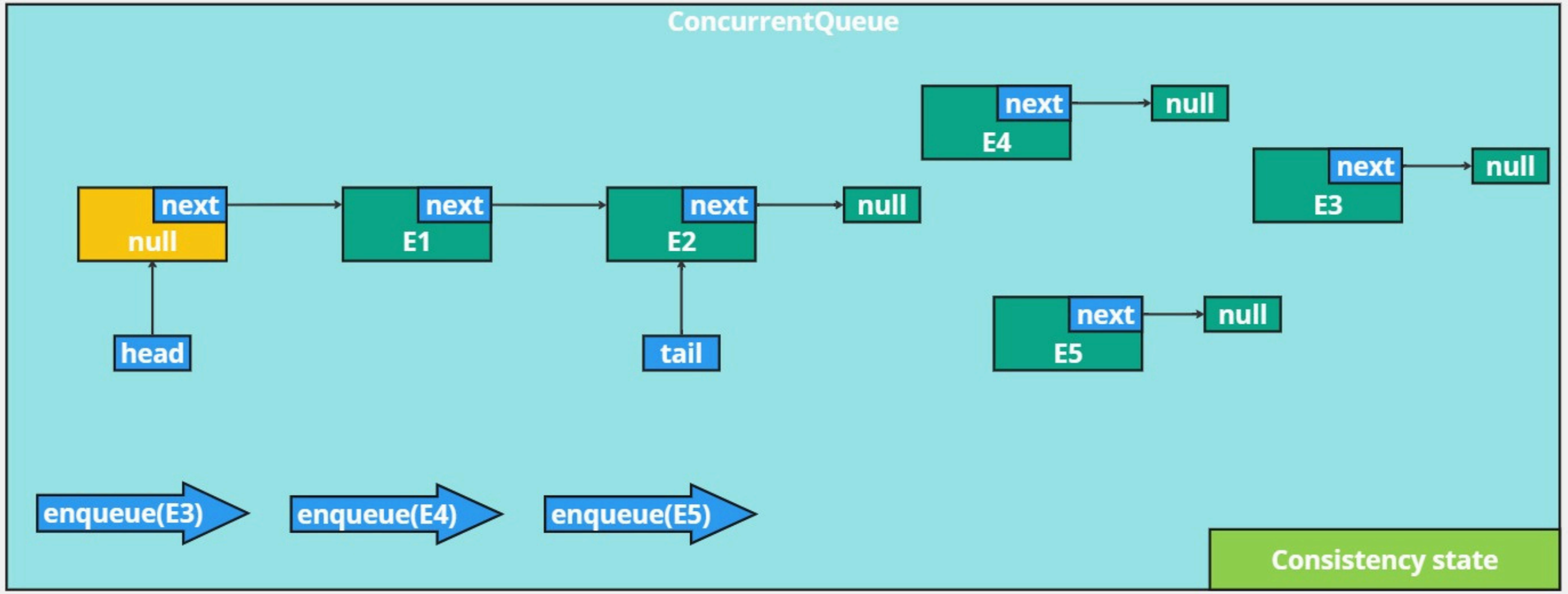
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

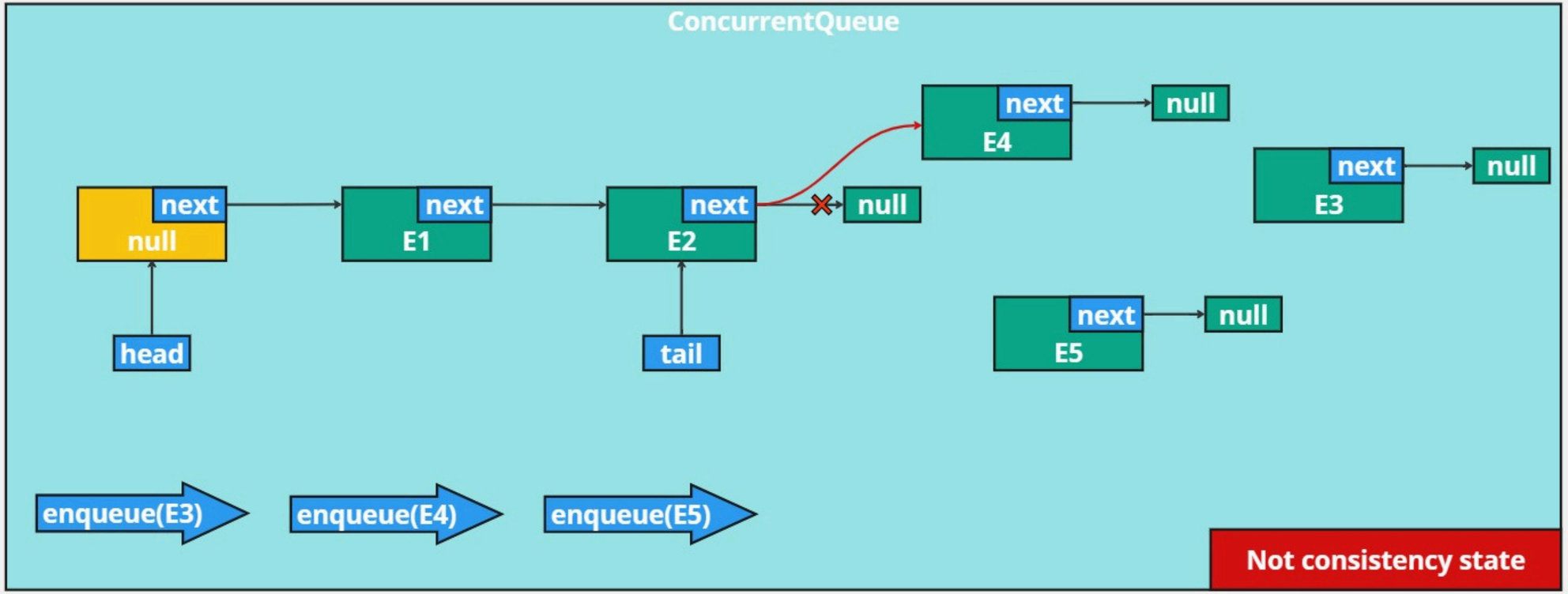
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

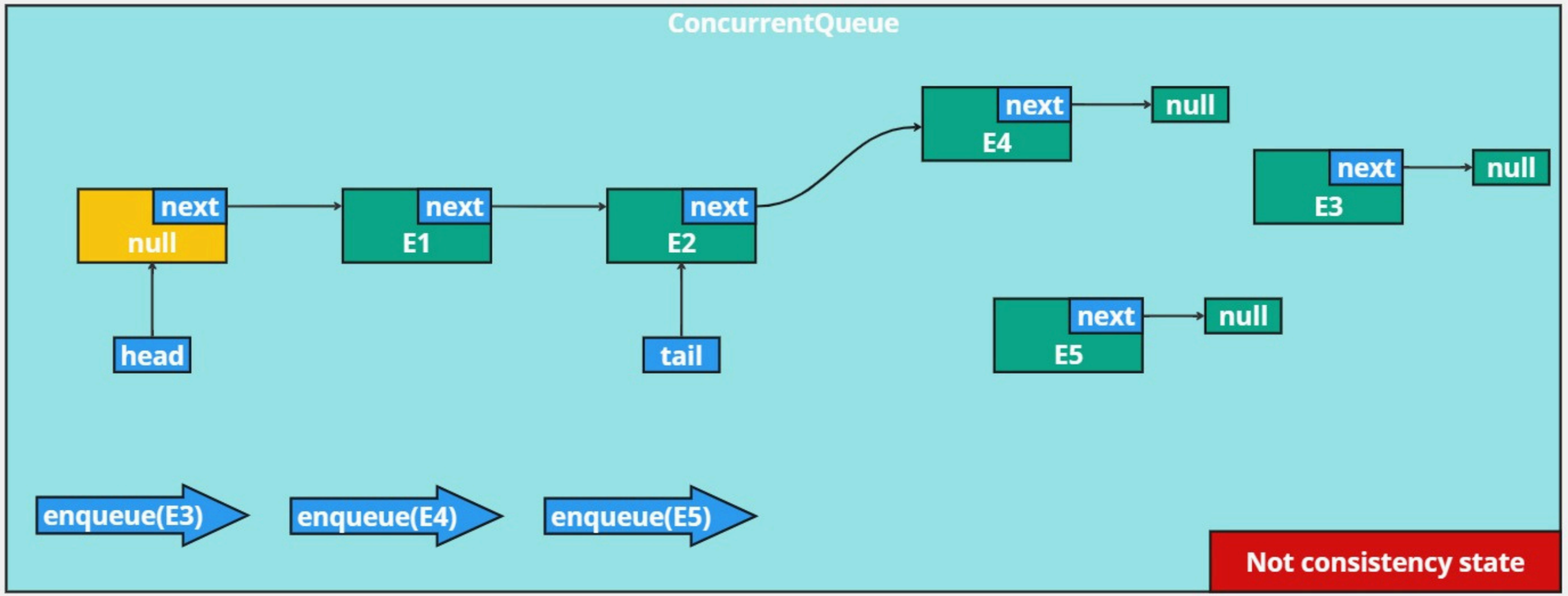
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

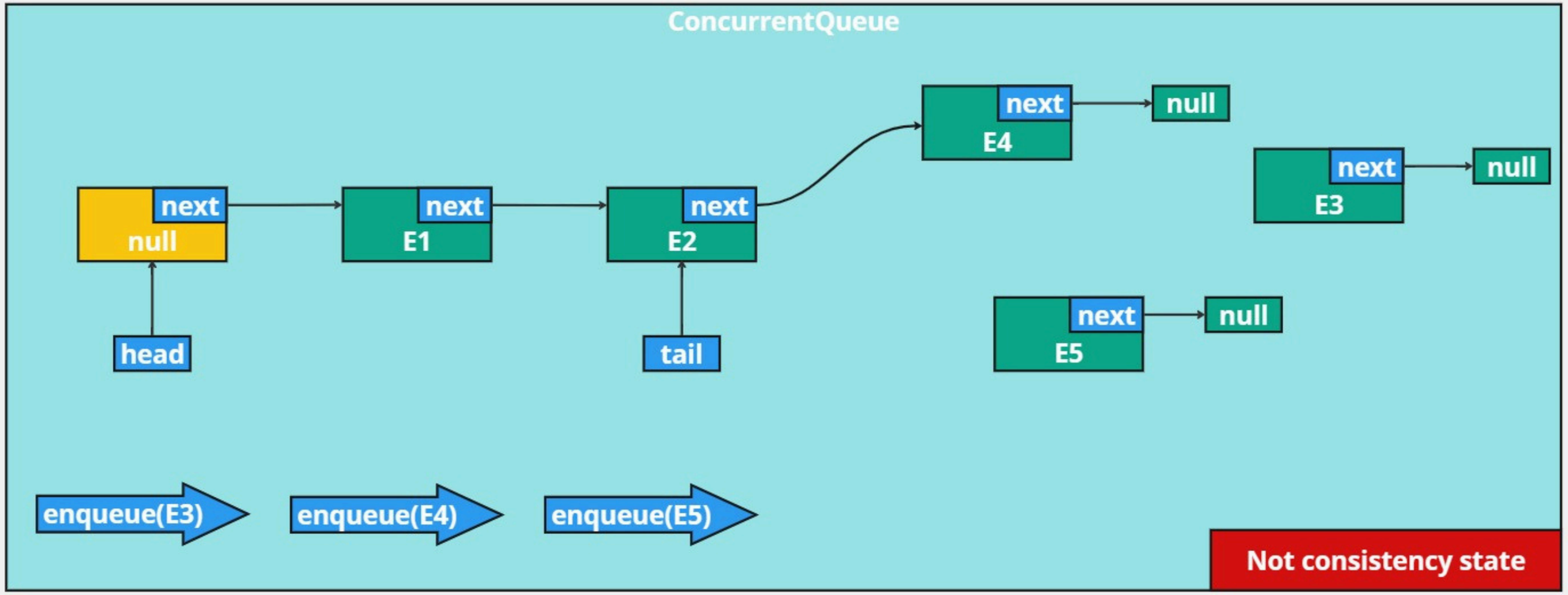
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

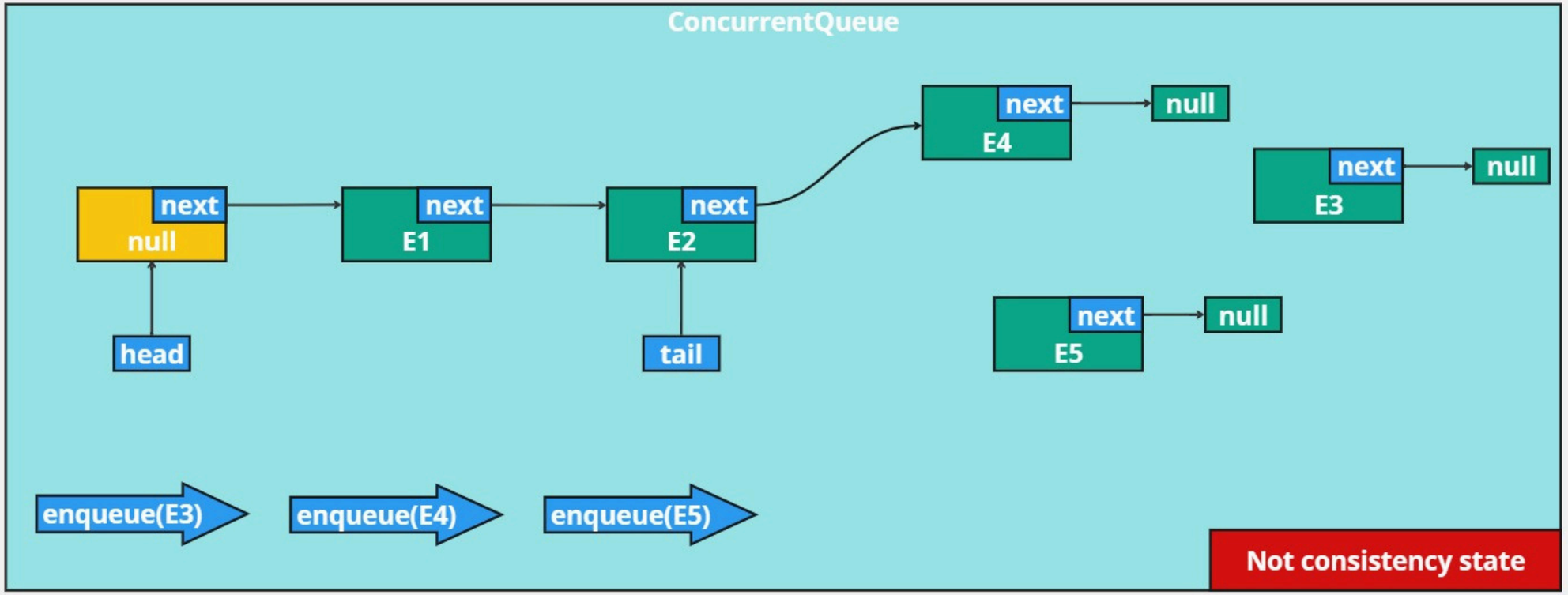
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

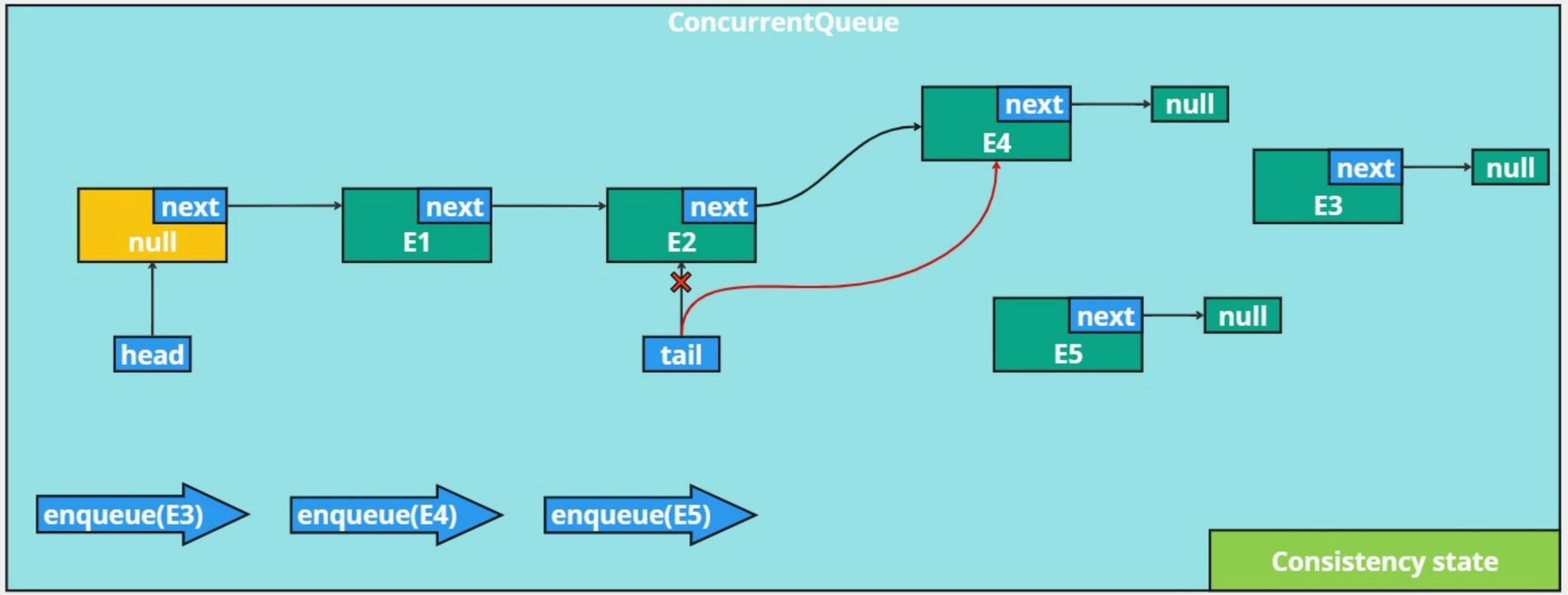
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

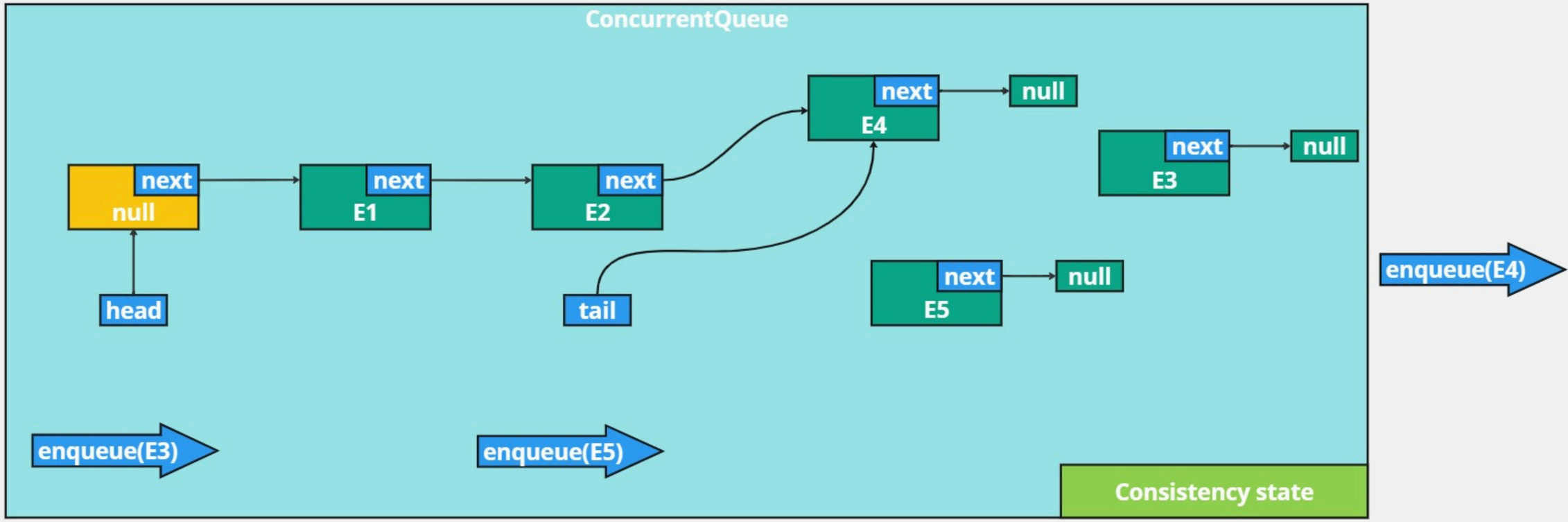
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

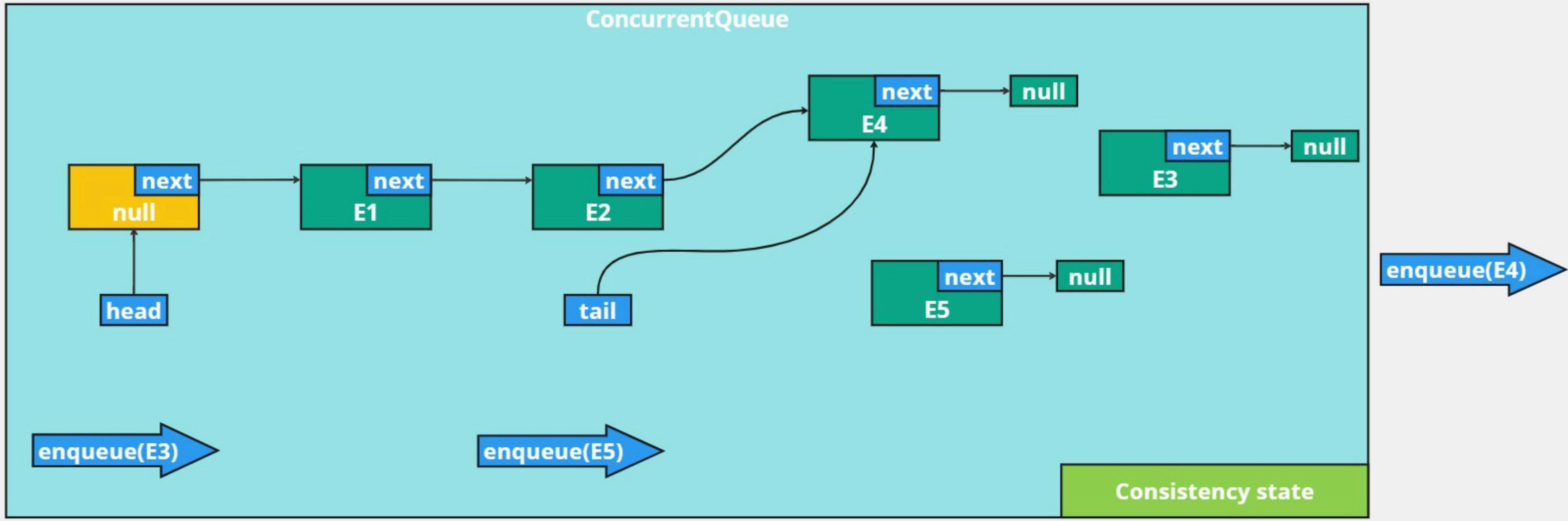
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

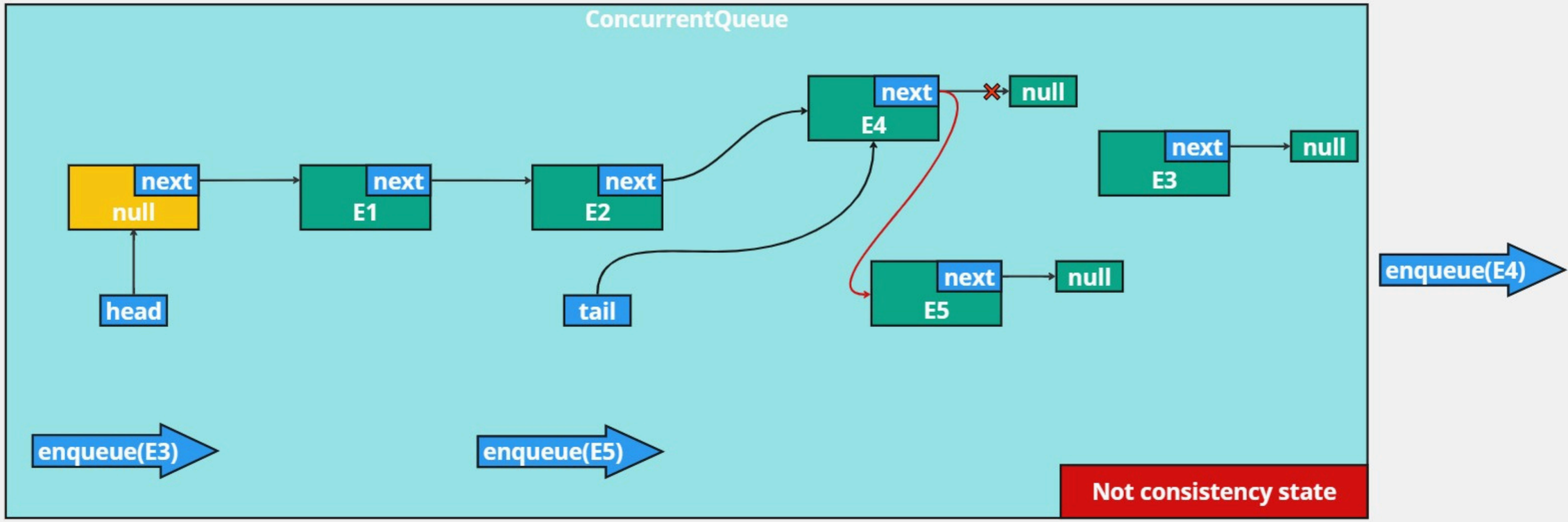
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

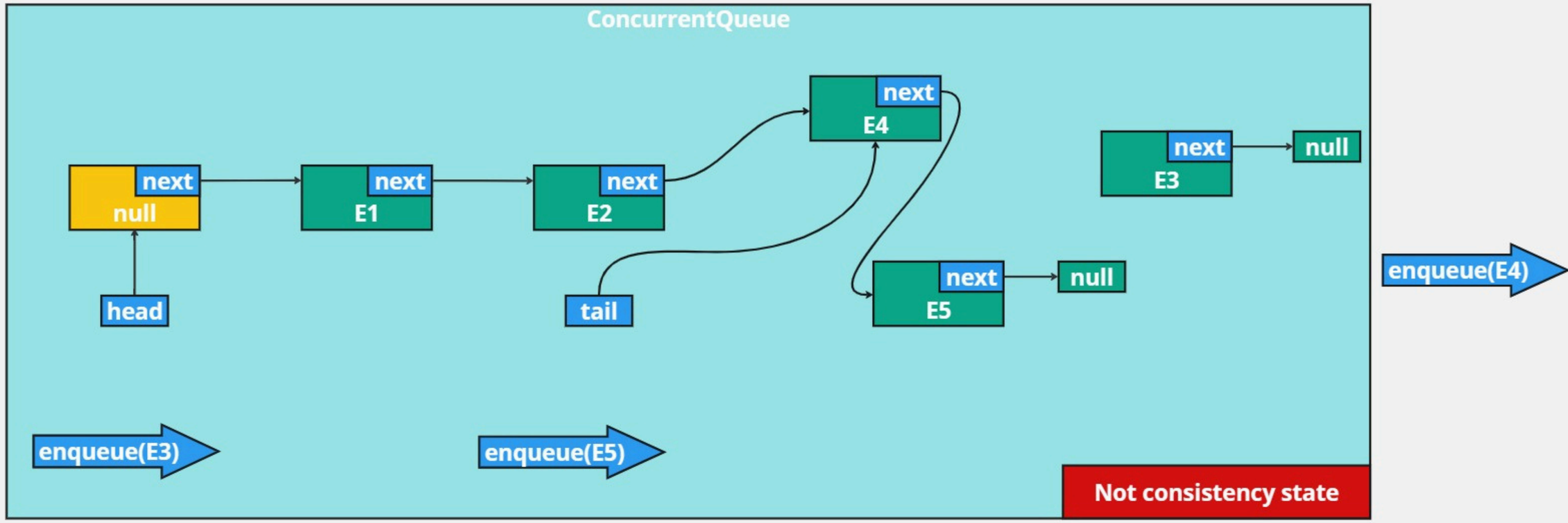
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

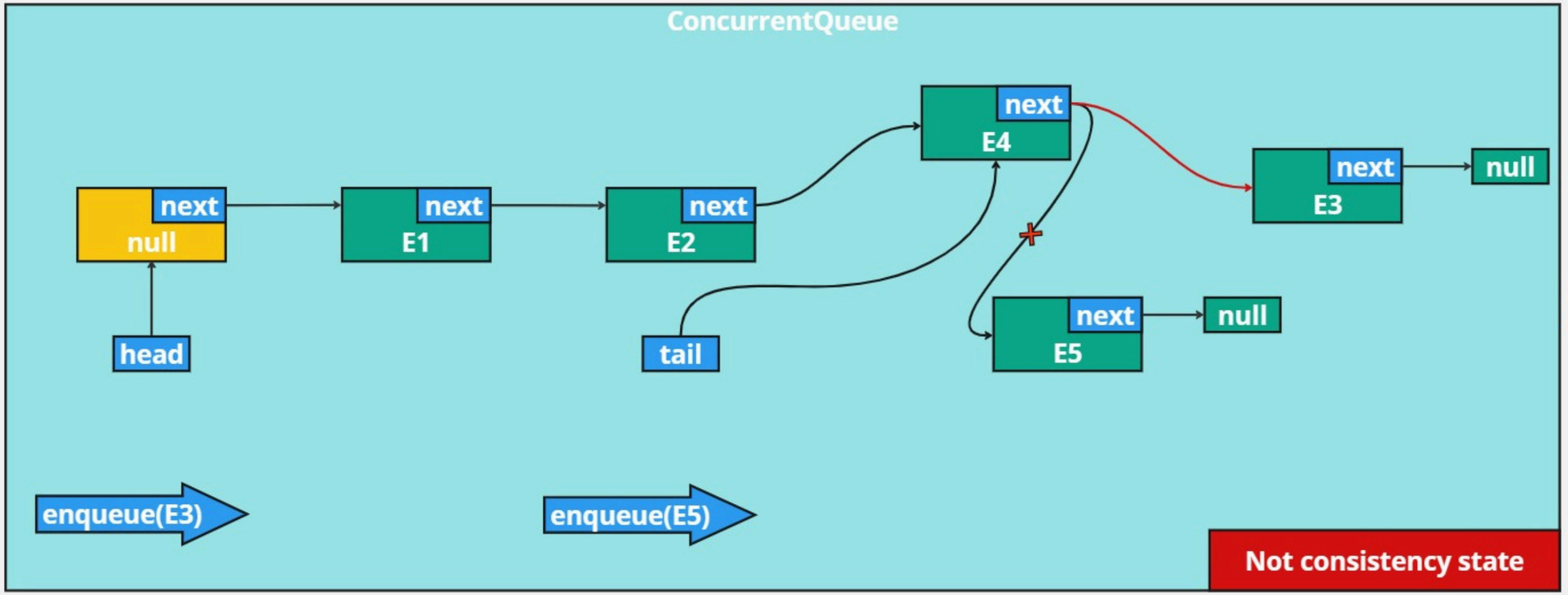
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

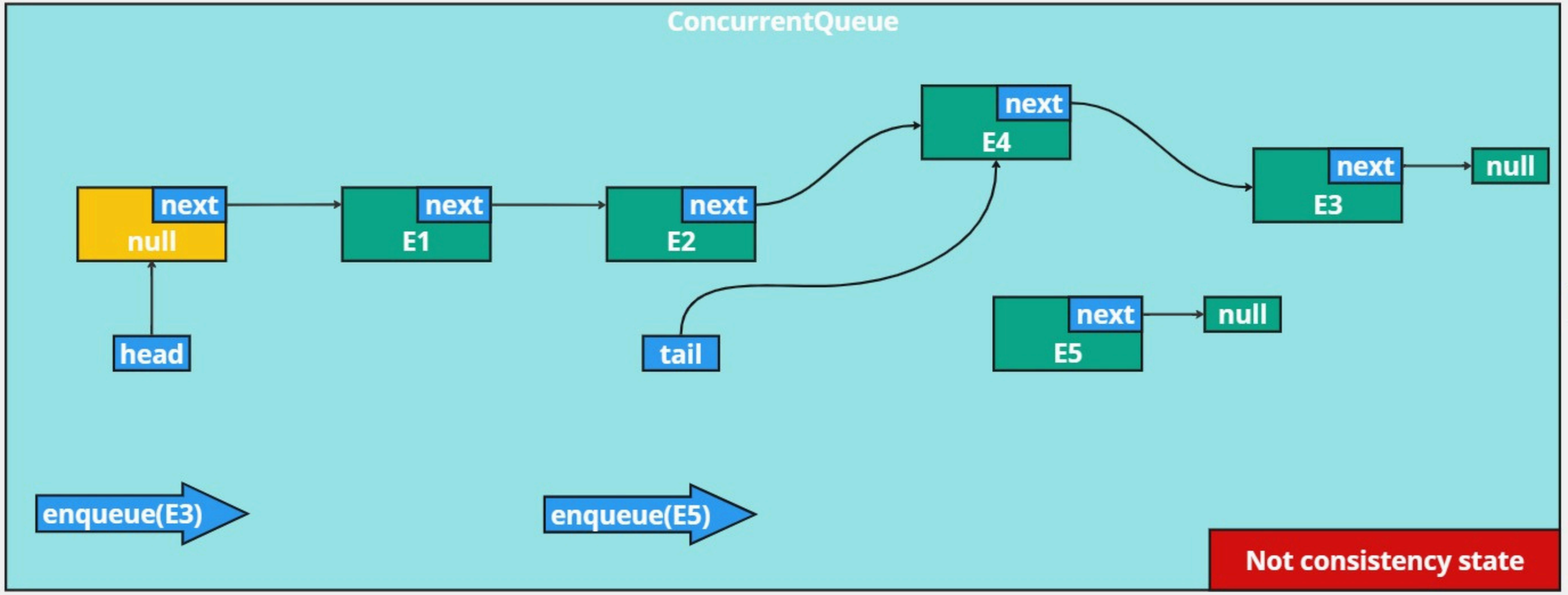
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

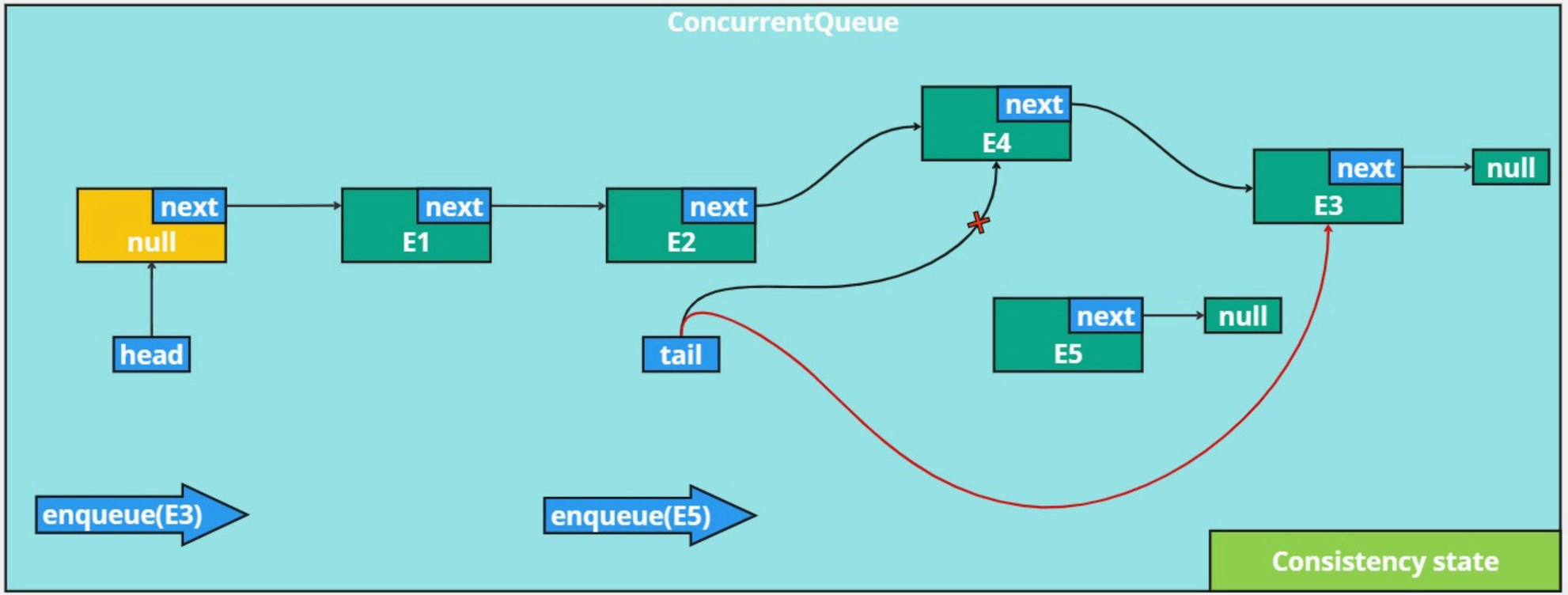
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

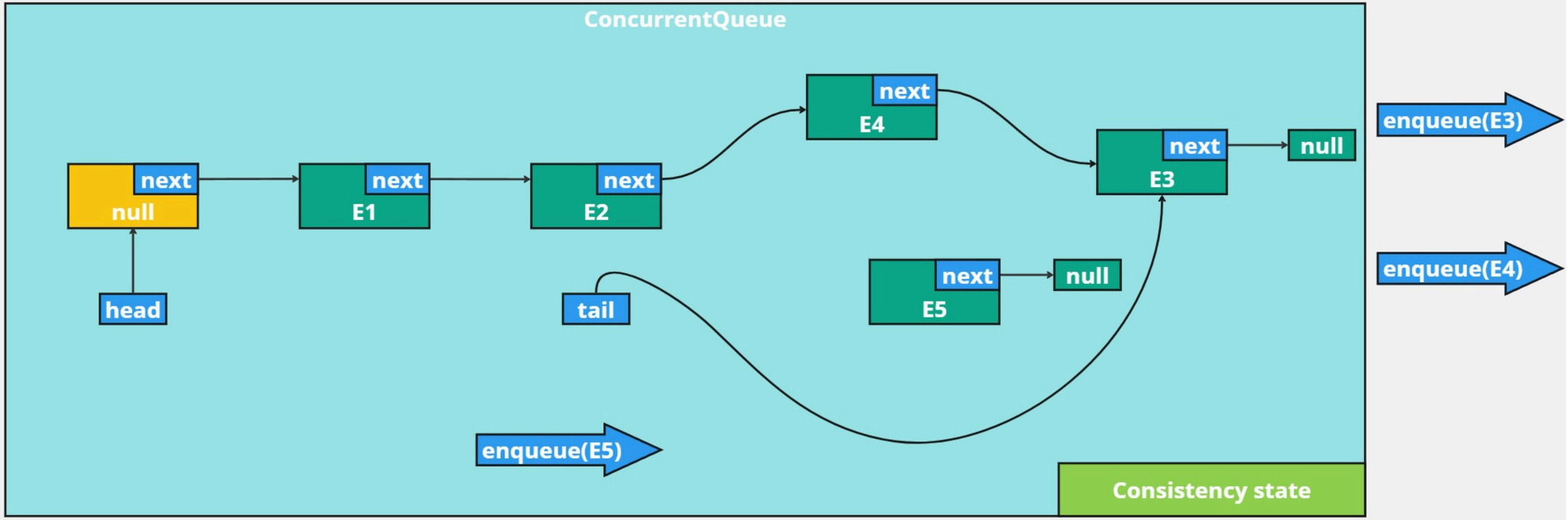
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

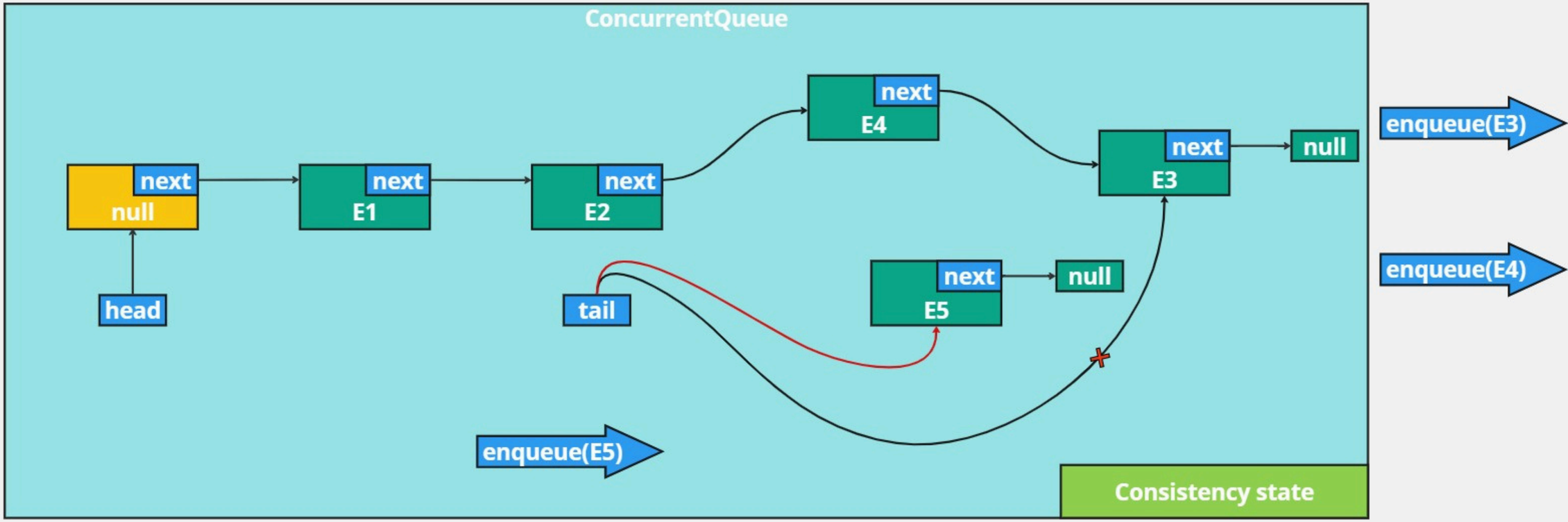
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

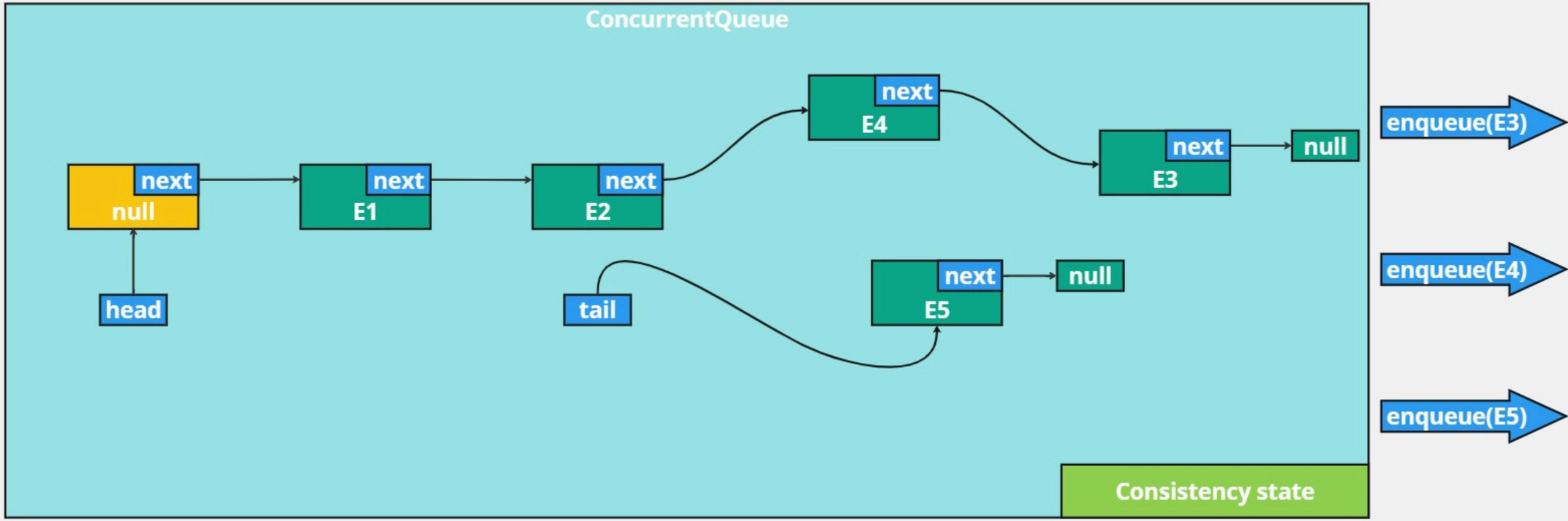
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

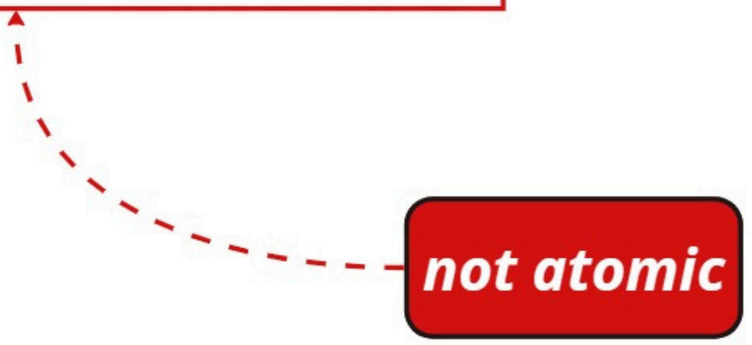
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```



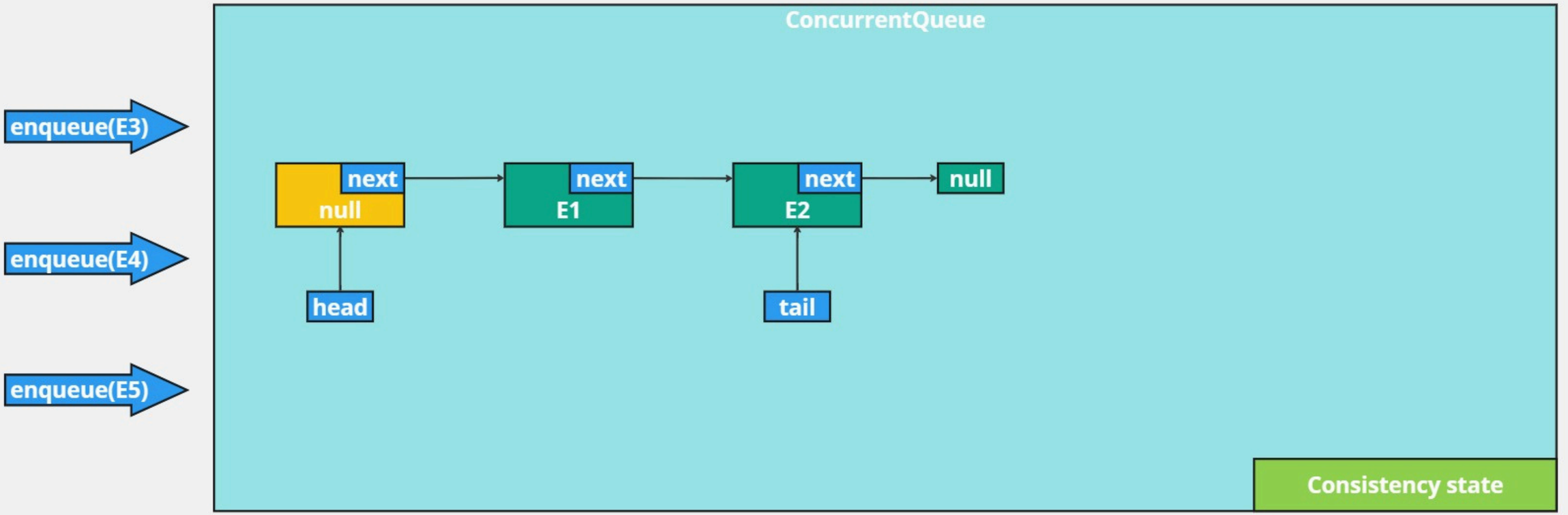
```java
private static final class Node<E> {
    1 usage
    private volatile E value;
    no usages
    private volatile Node<E> next;

    1 usage
    public Node() {

    }

    no usages
    public Node(final E value) {
        this.value = value;
    }
}
```

```java
private static final class Node<E> {
    1 usage
    private volatile E value;
    3 usages
    private final AtomicReference<Node<E>> next;

    1 usage
    public Node() {
        next = new AtomicReference<>();
    }

    1 usage
    public Node(final E value) {
        this.value = value;
        next = new AtomicReference<>();
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next == null) {
            tail.next = newNode;
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet( expectedValue: null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

atomic



enqueue(E3)

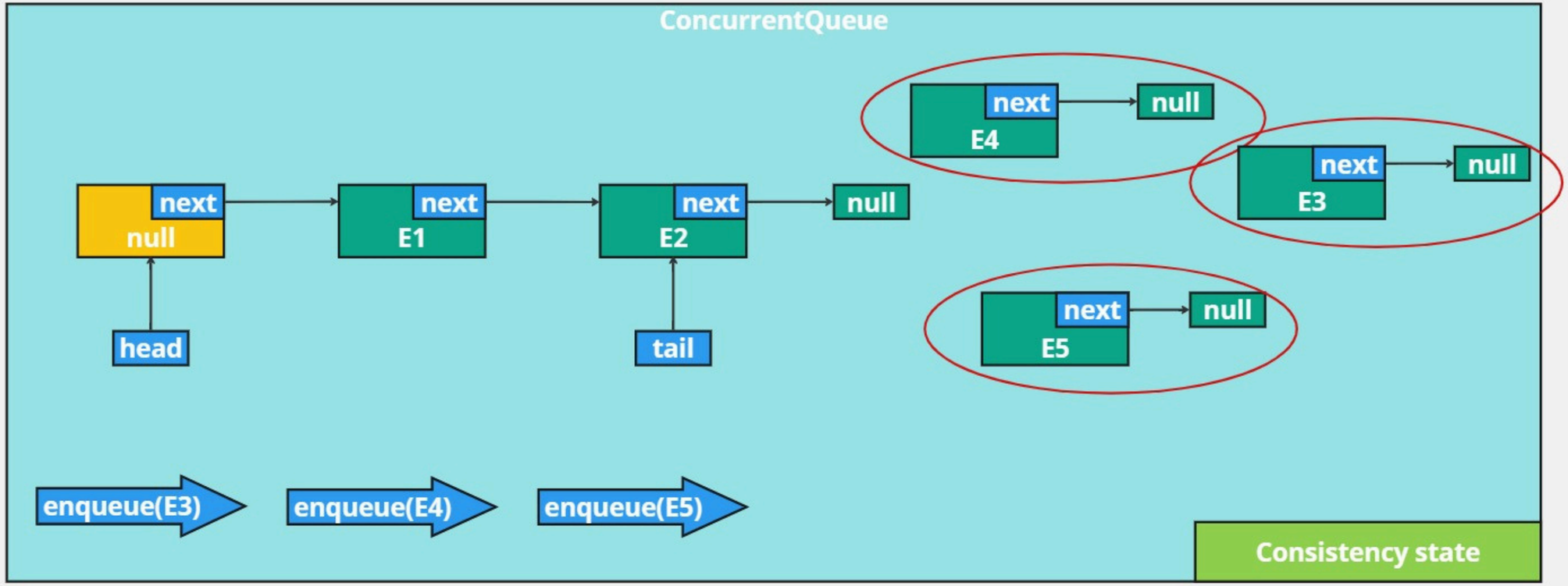
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

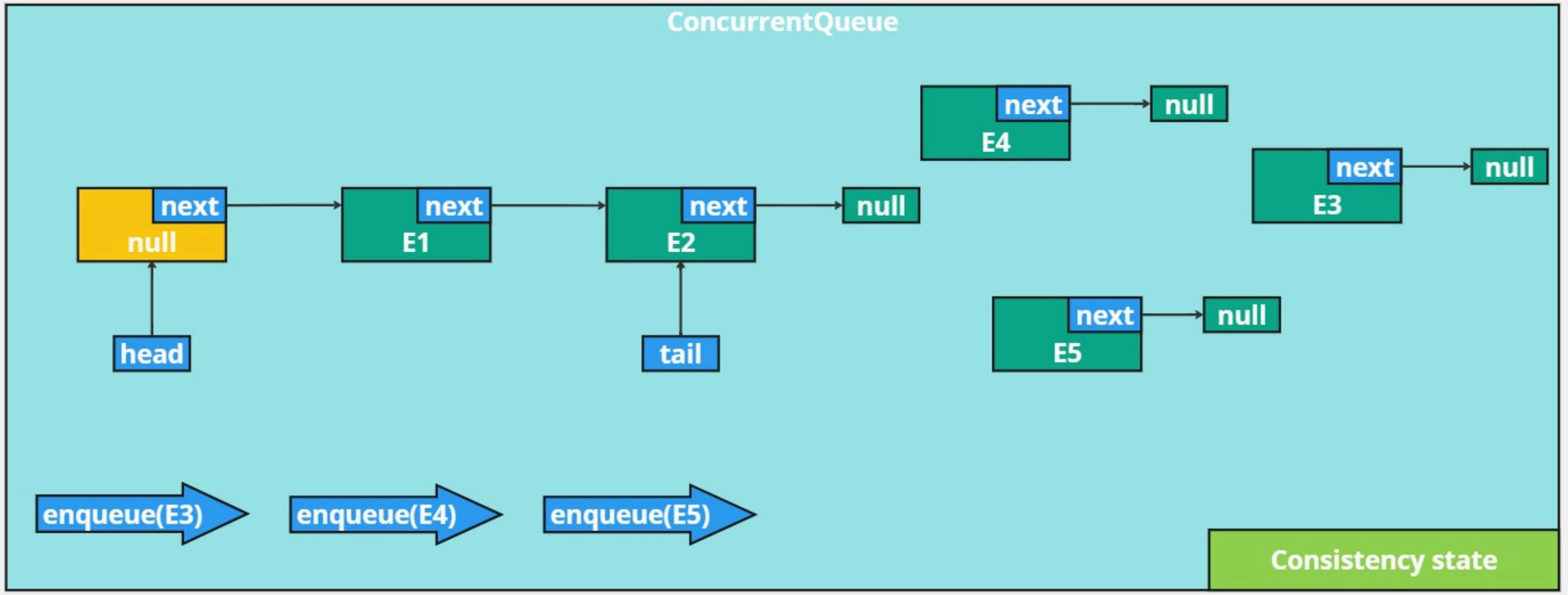
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

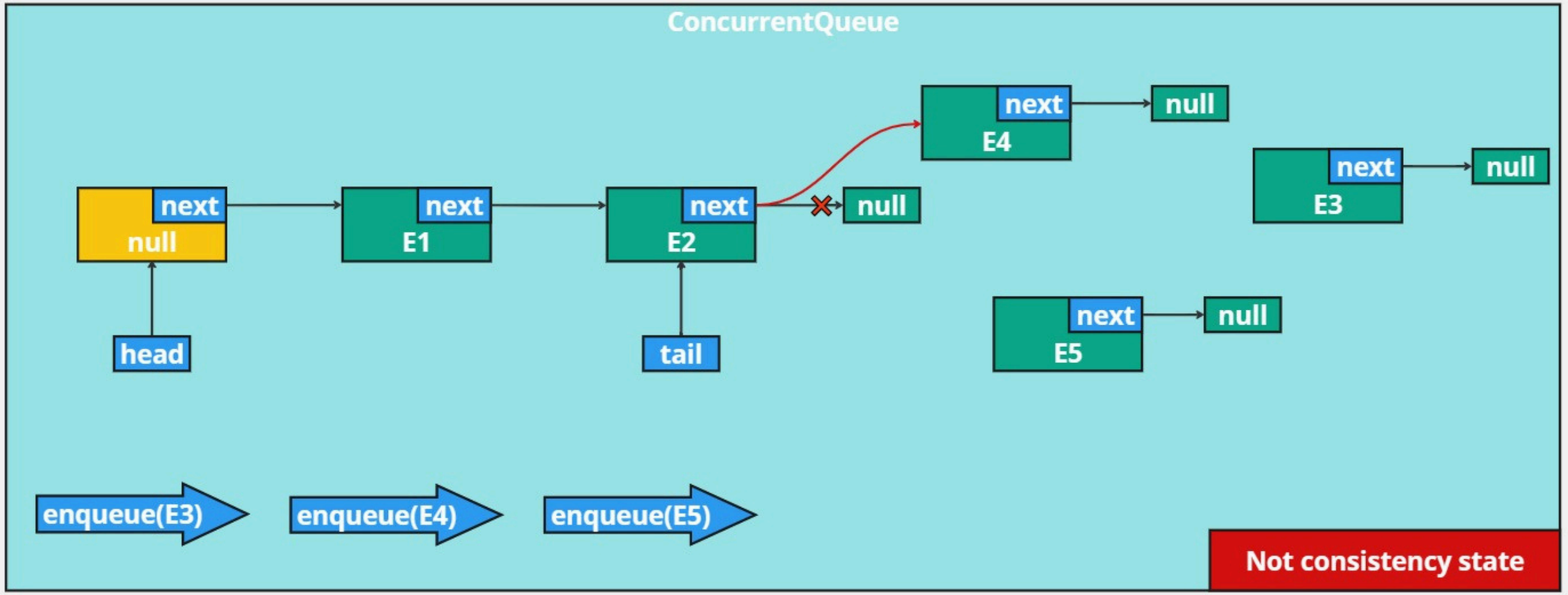
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

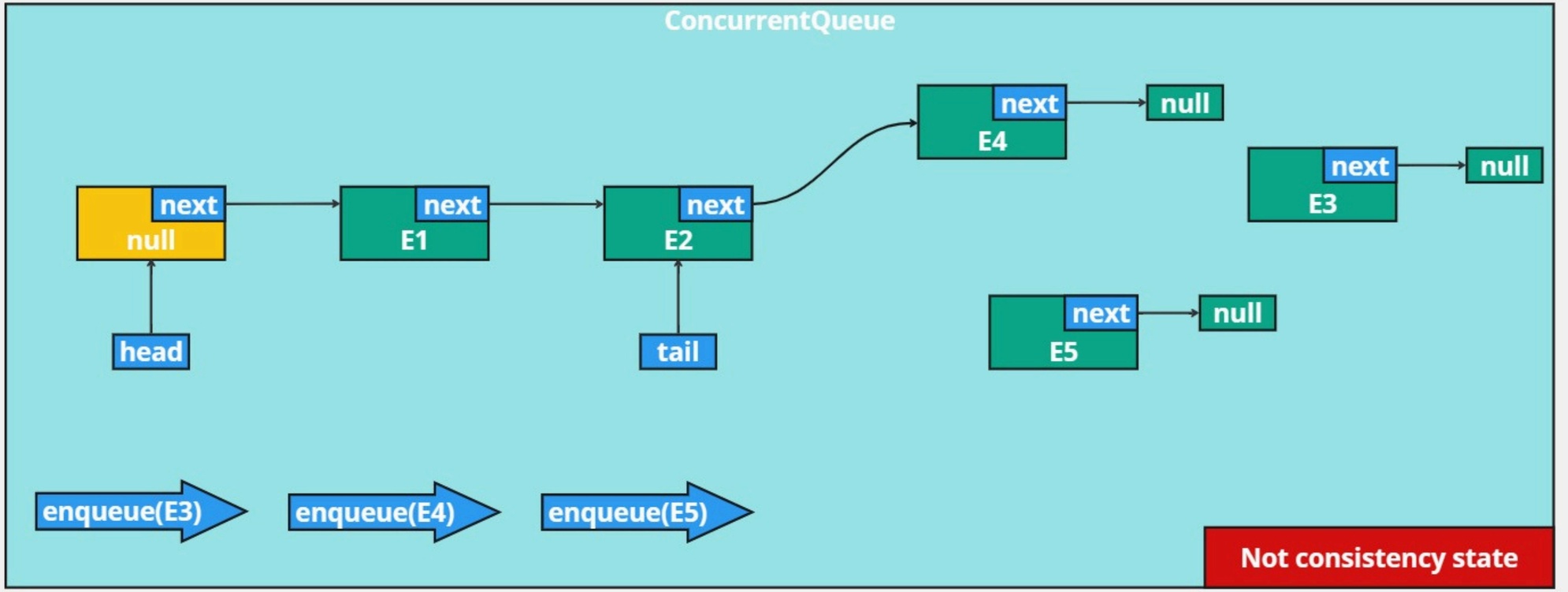
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

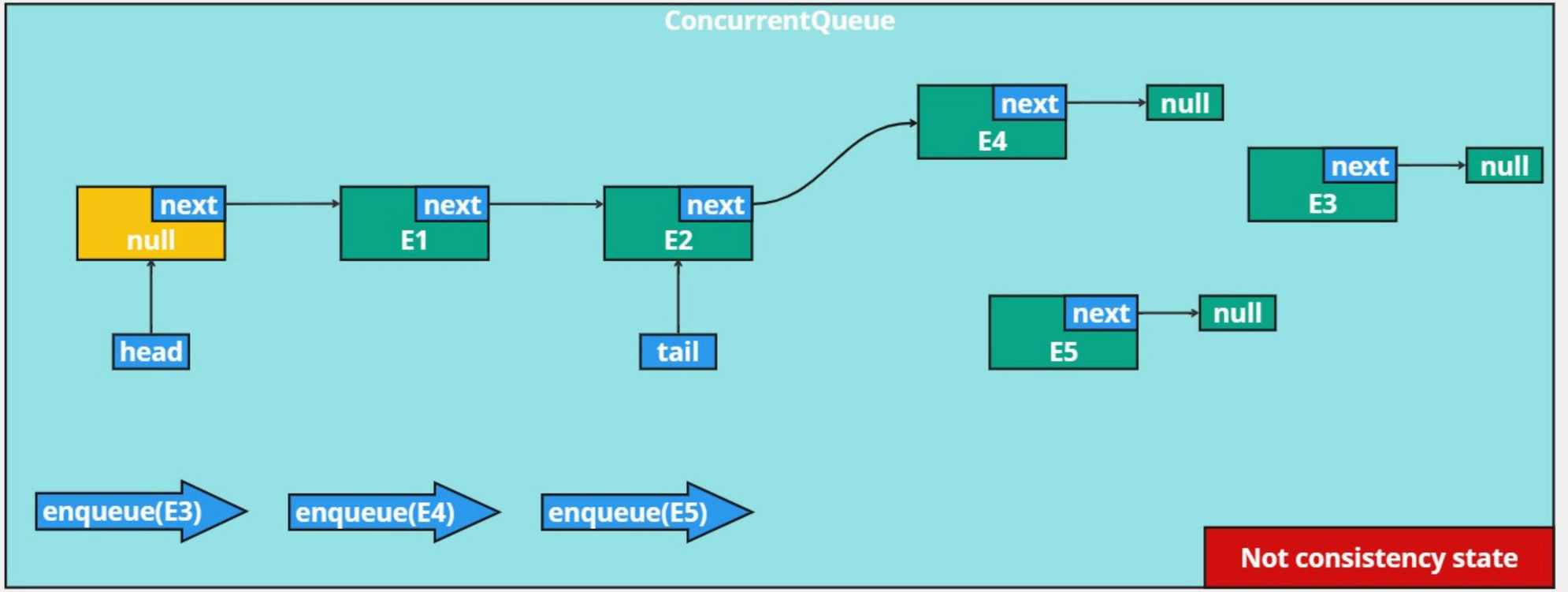
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

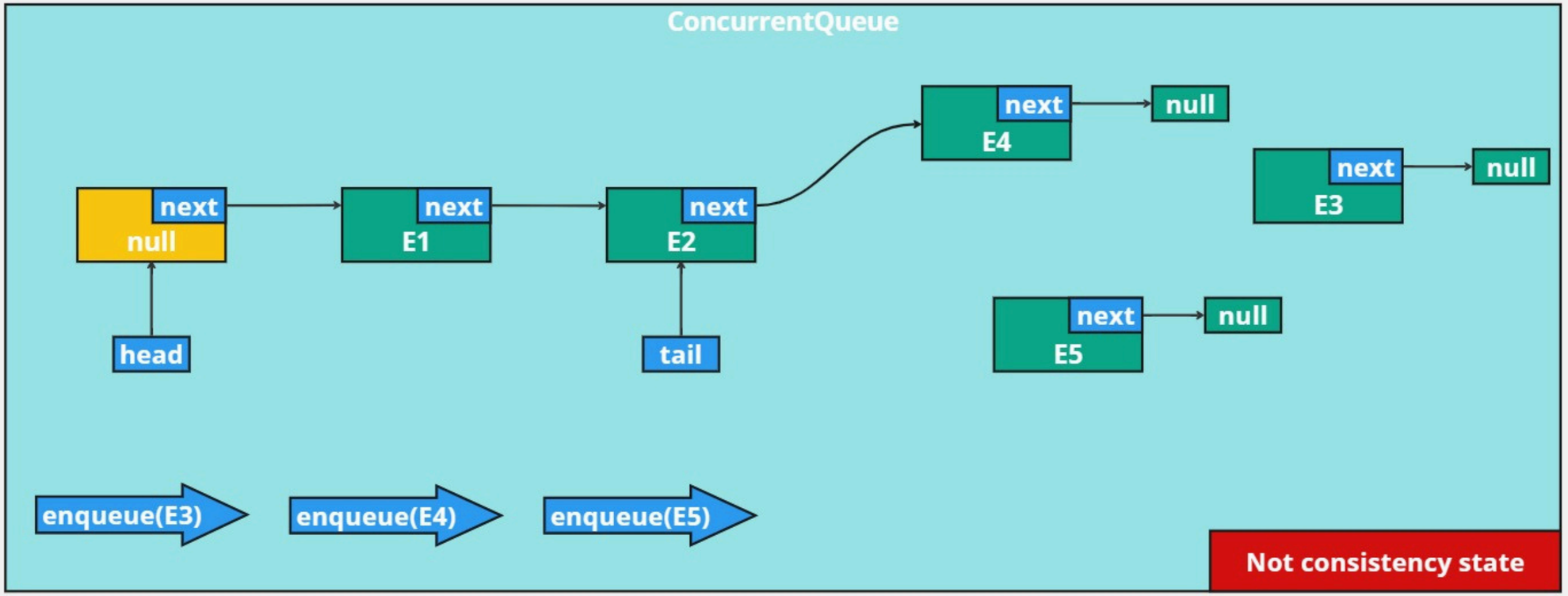
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

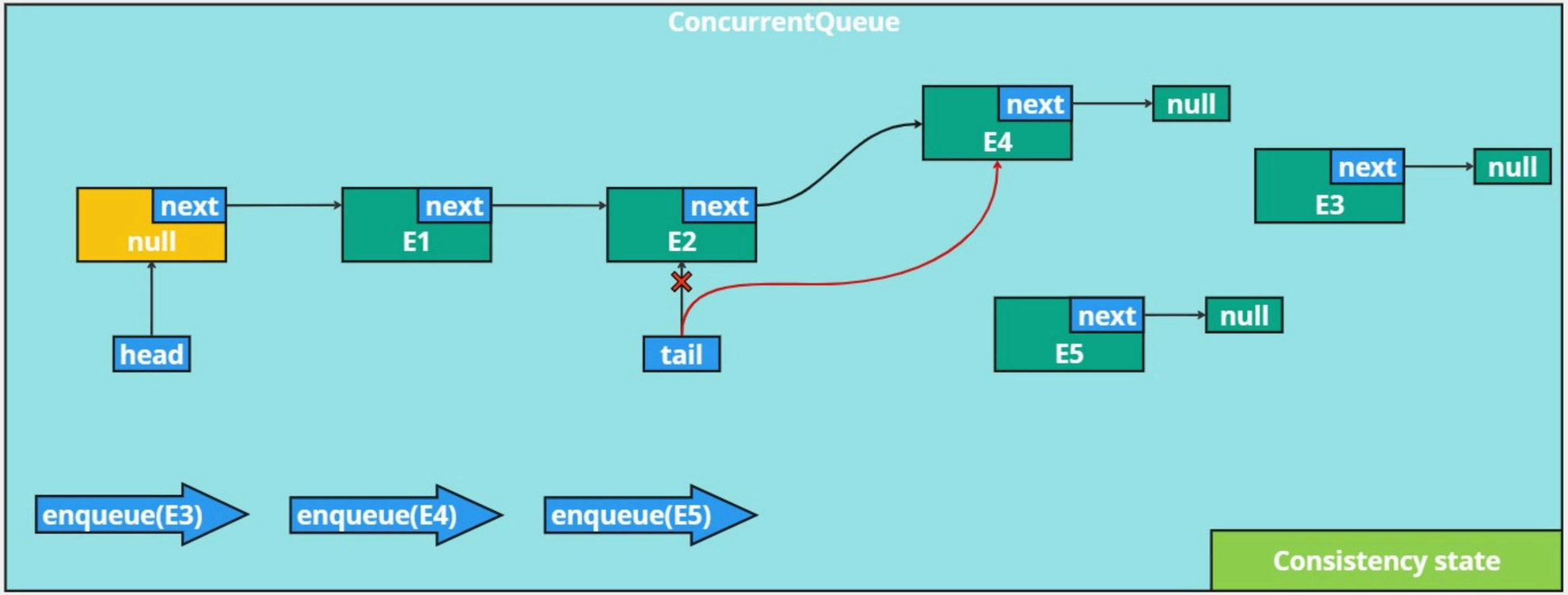
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

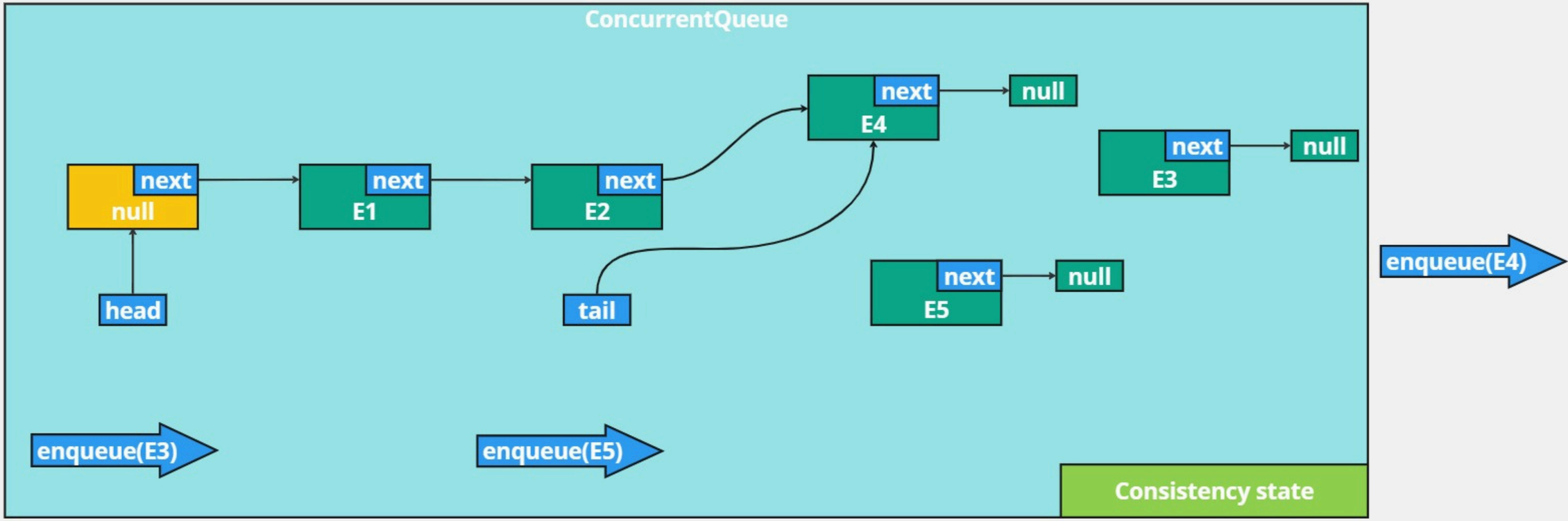
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

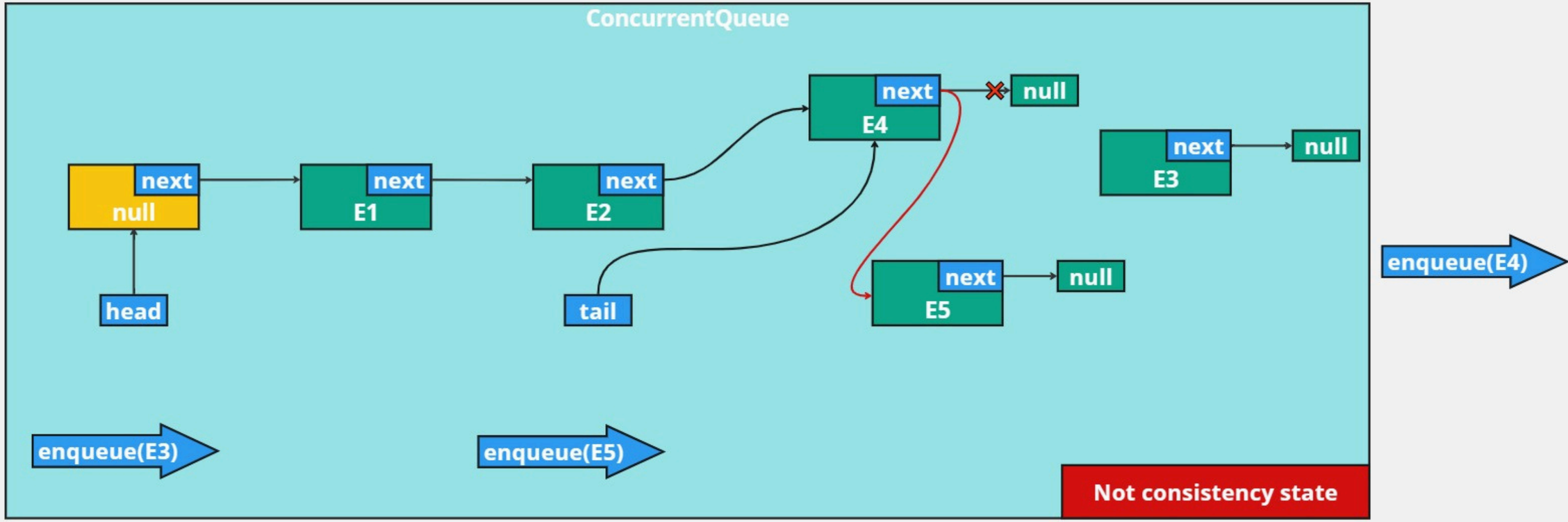
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

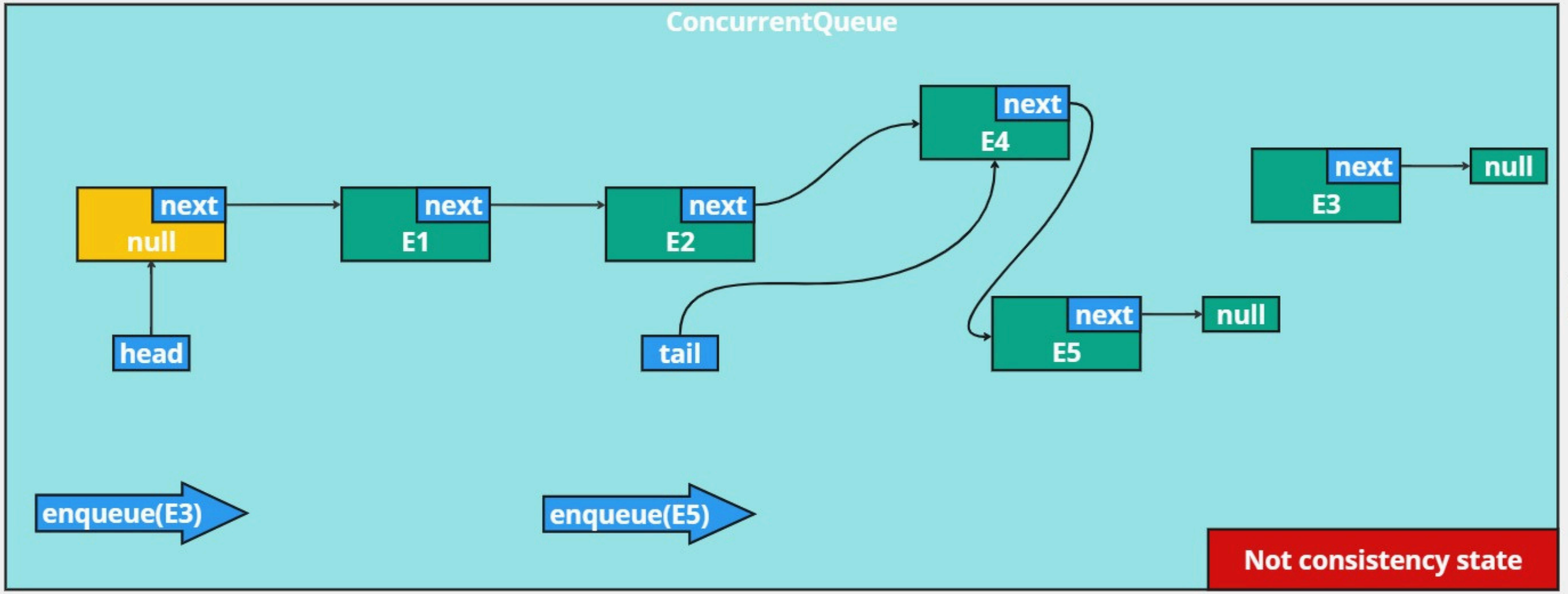
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

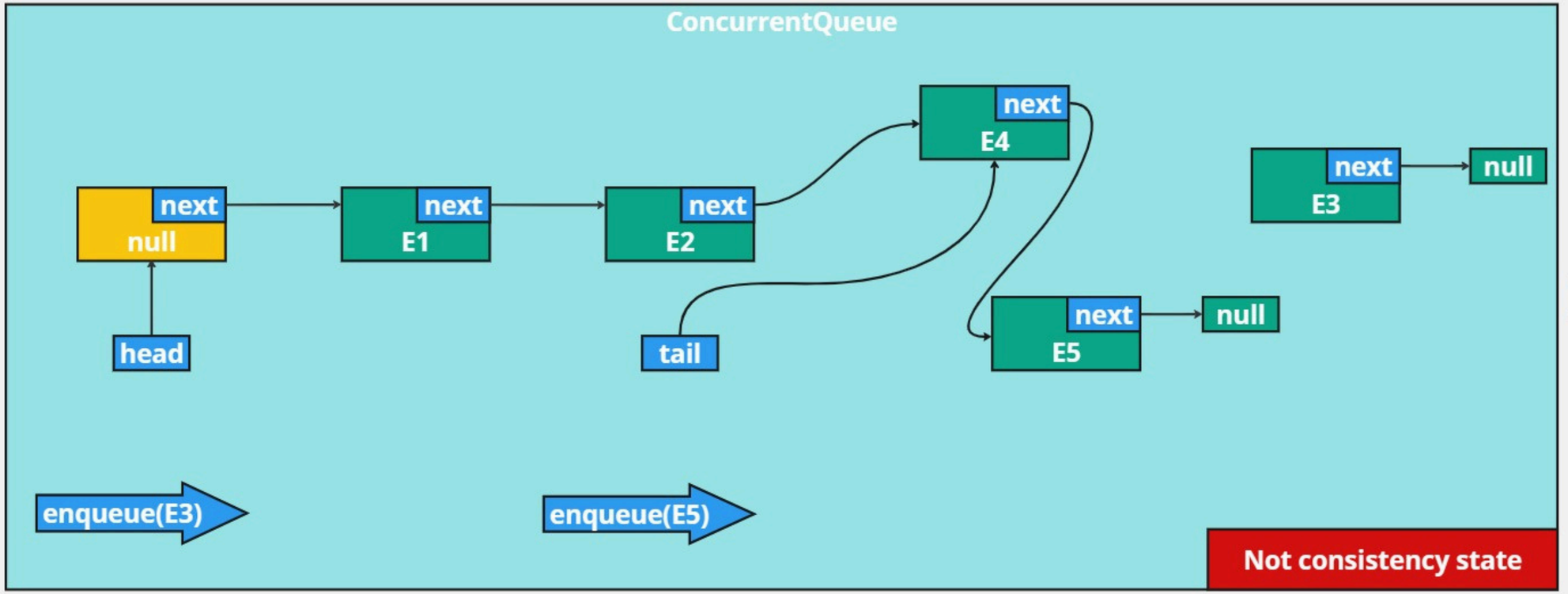
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

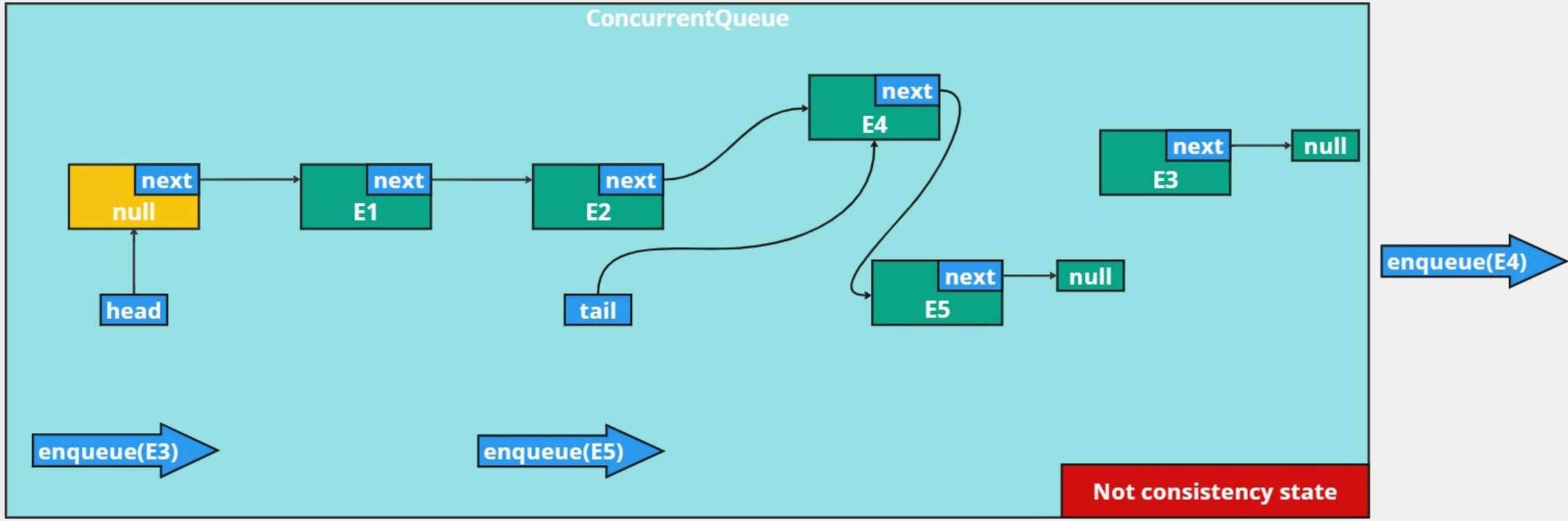
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

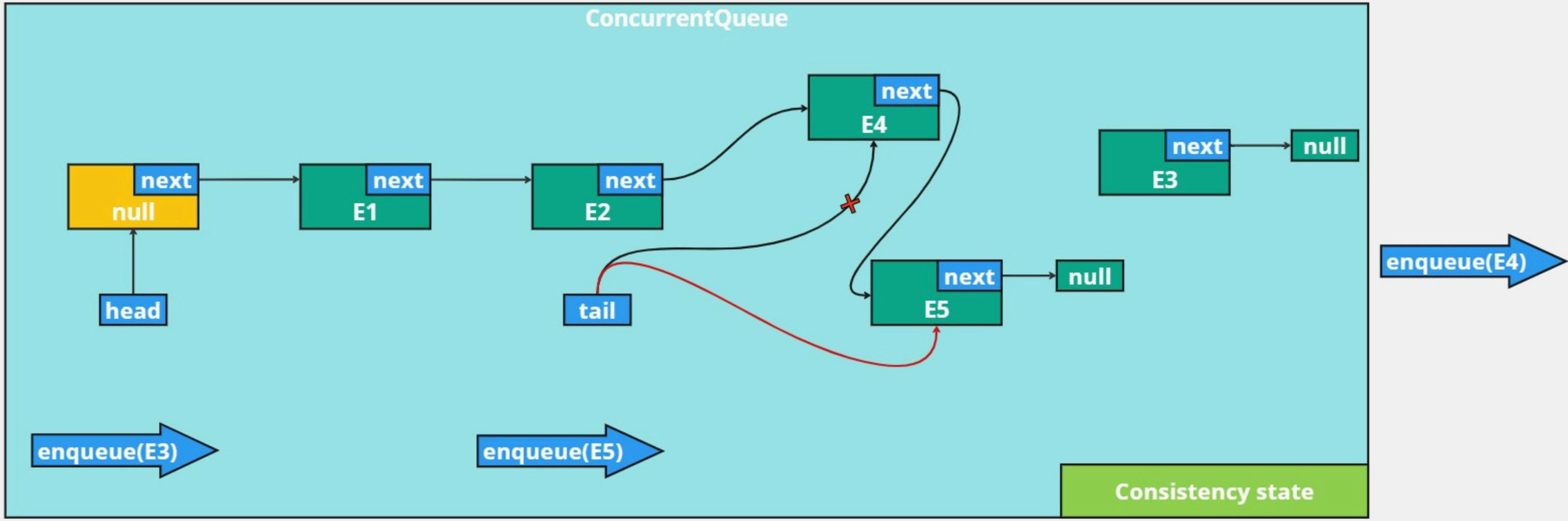
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

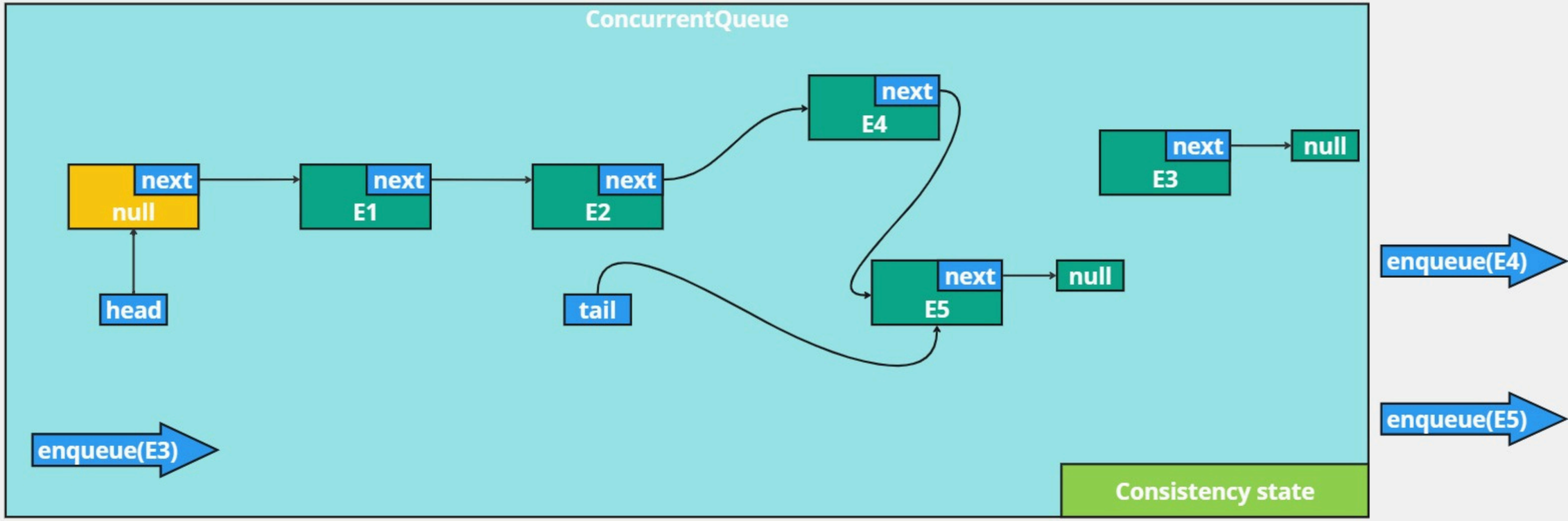
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

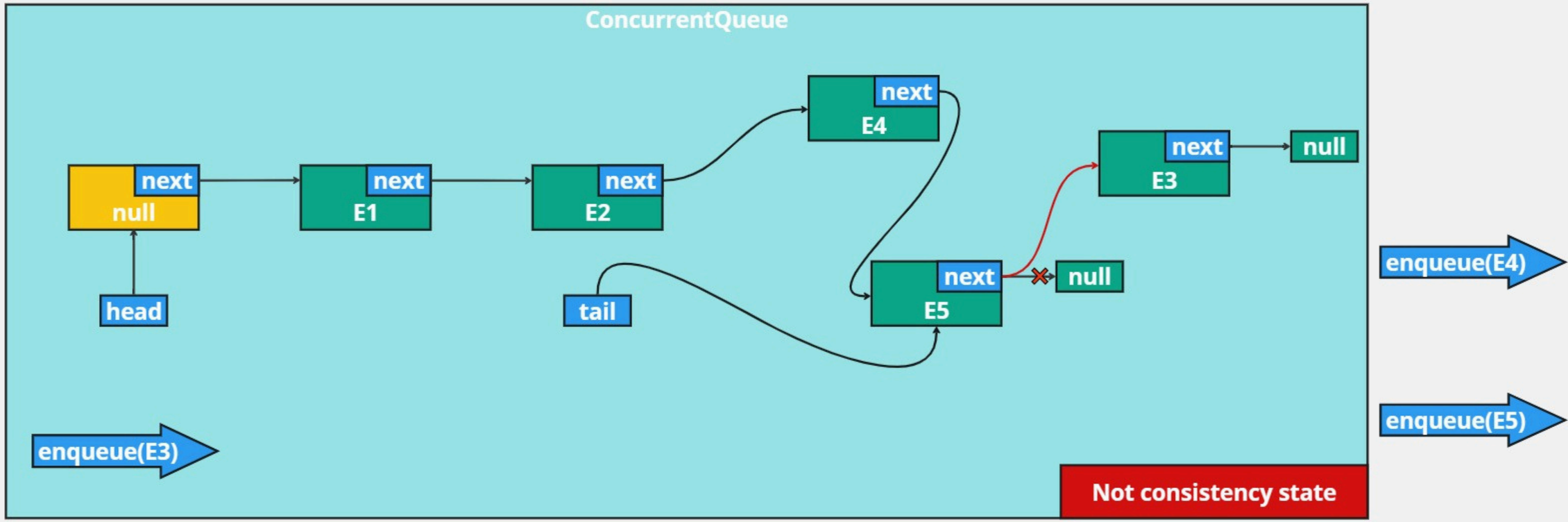
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

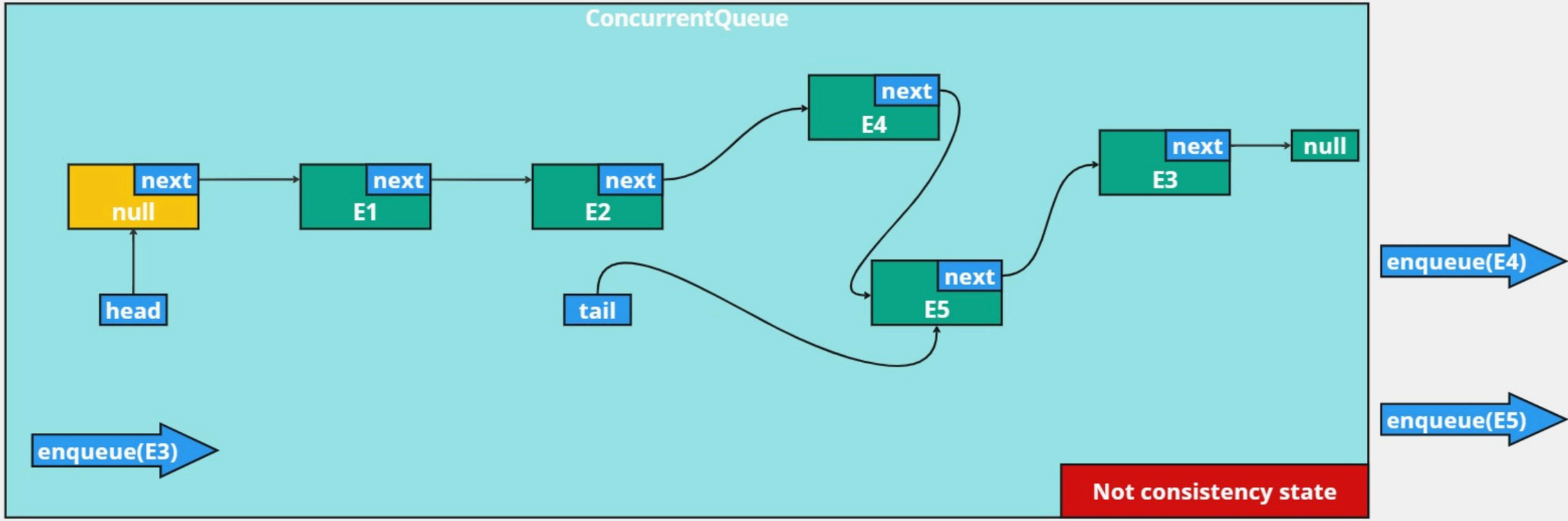
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

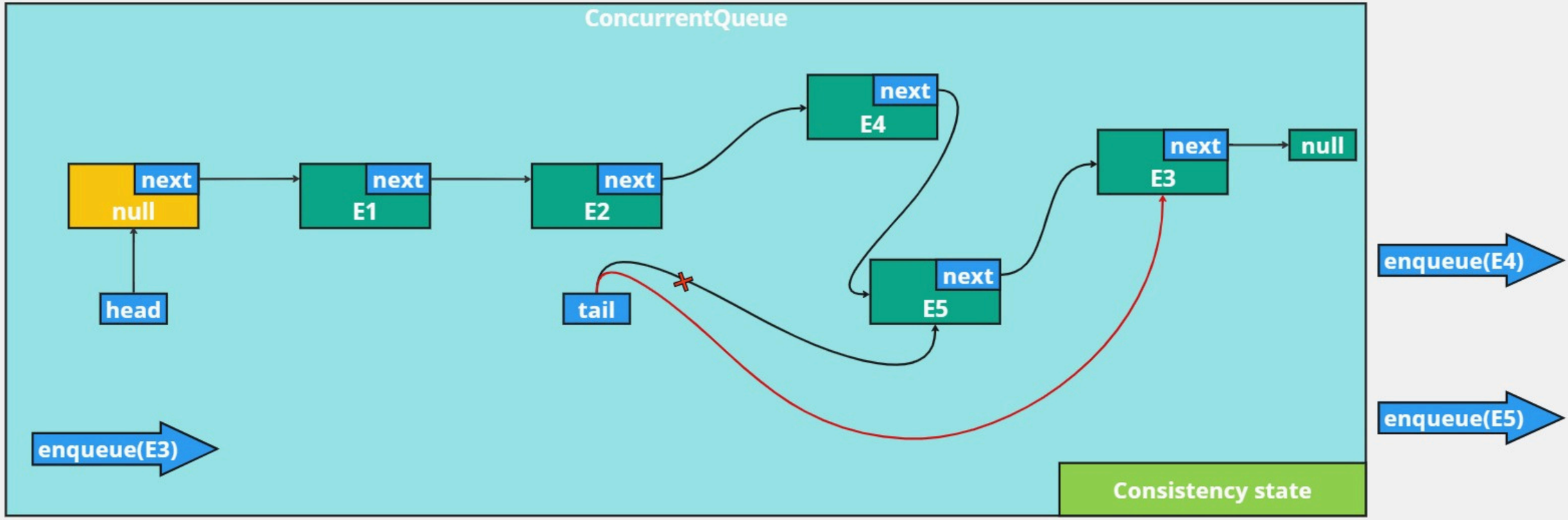
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

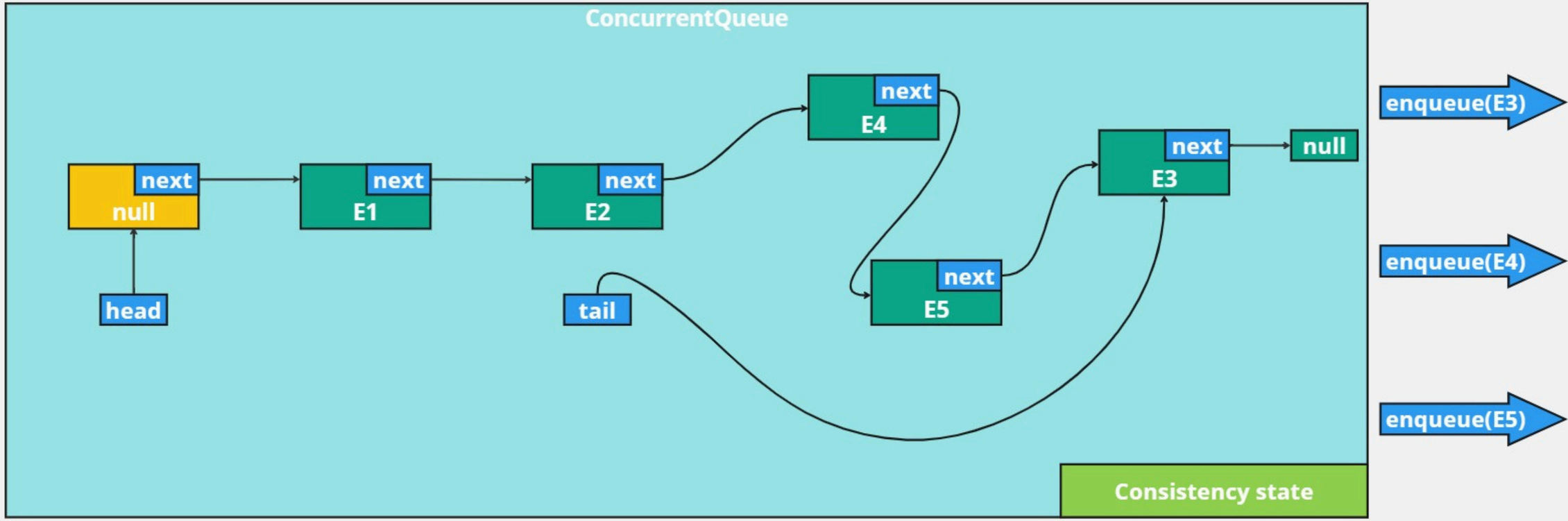
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

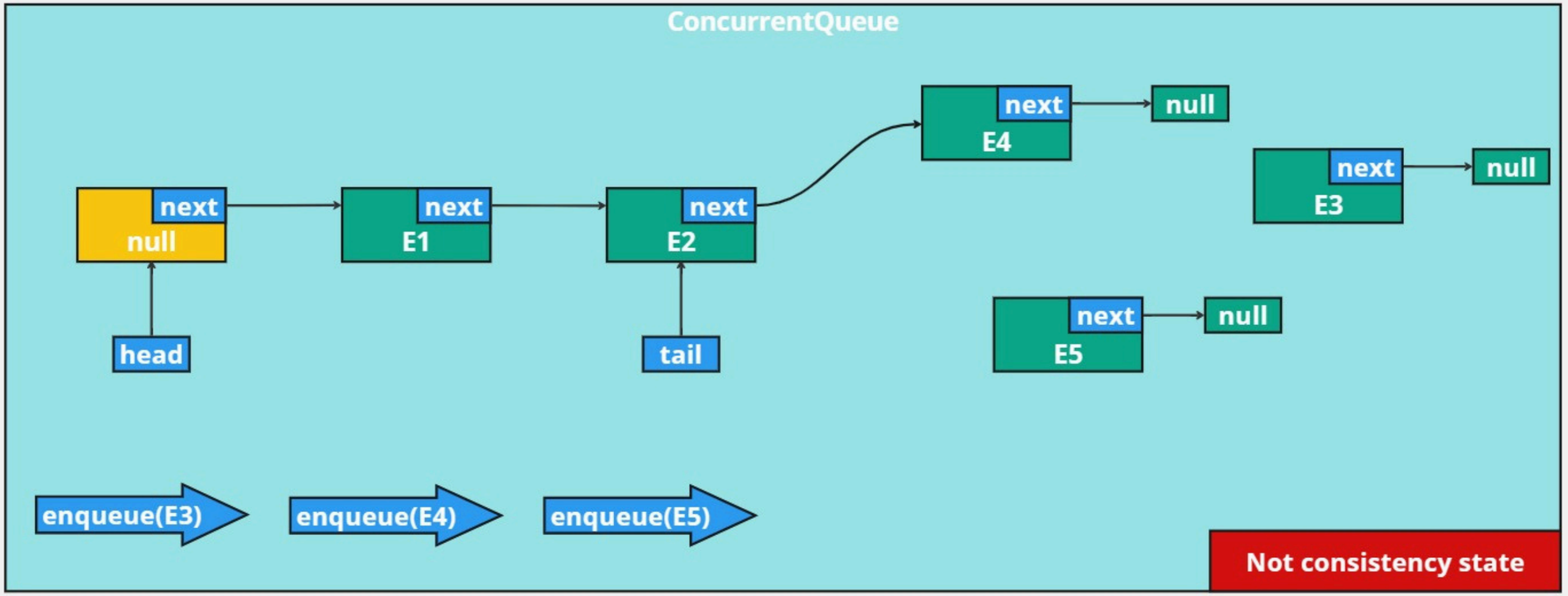
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```


```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

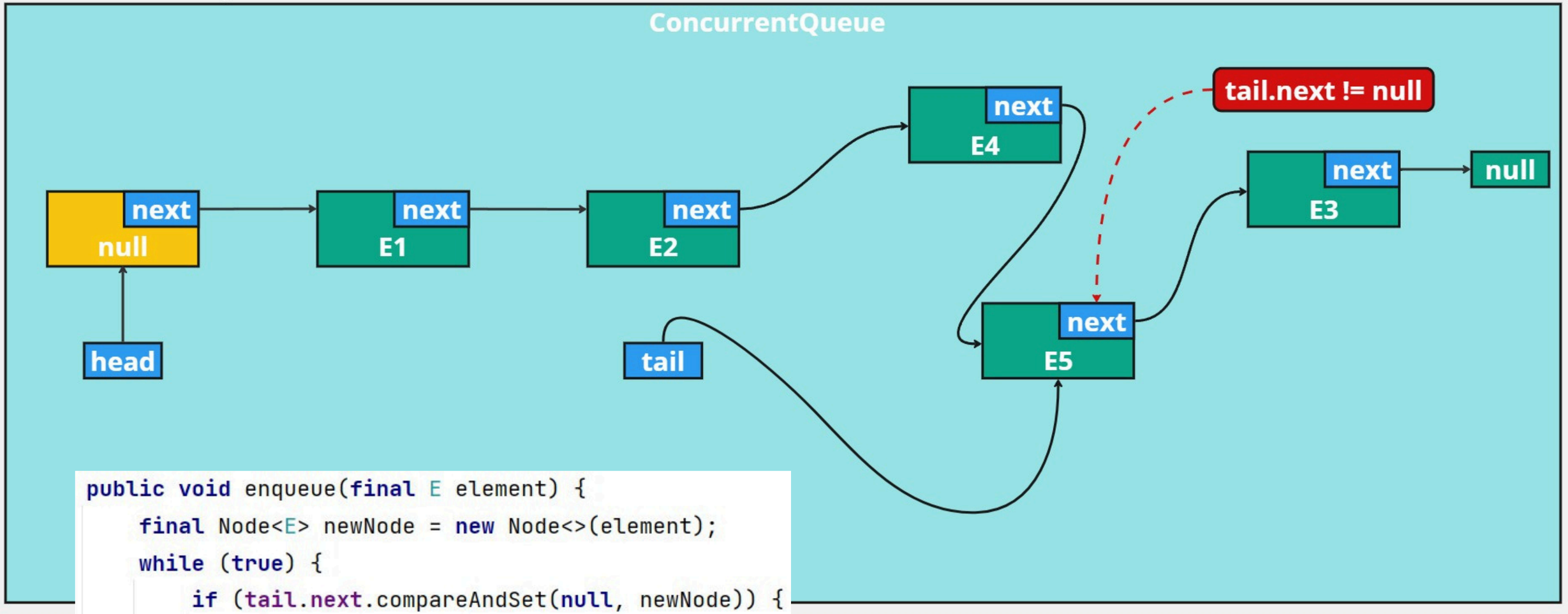
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

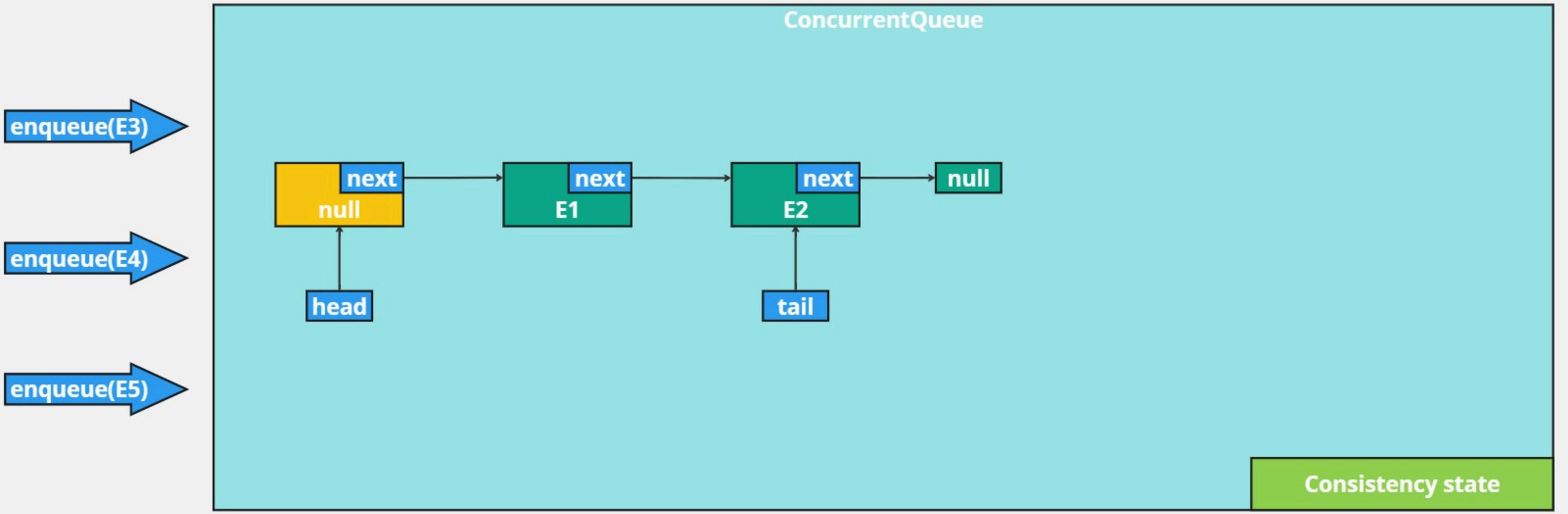
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        }
    }
    tail = newNode;
}
```

Not consistency state




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

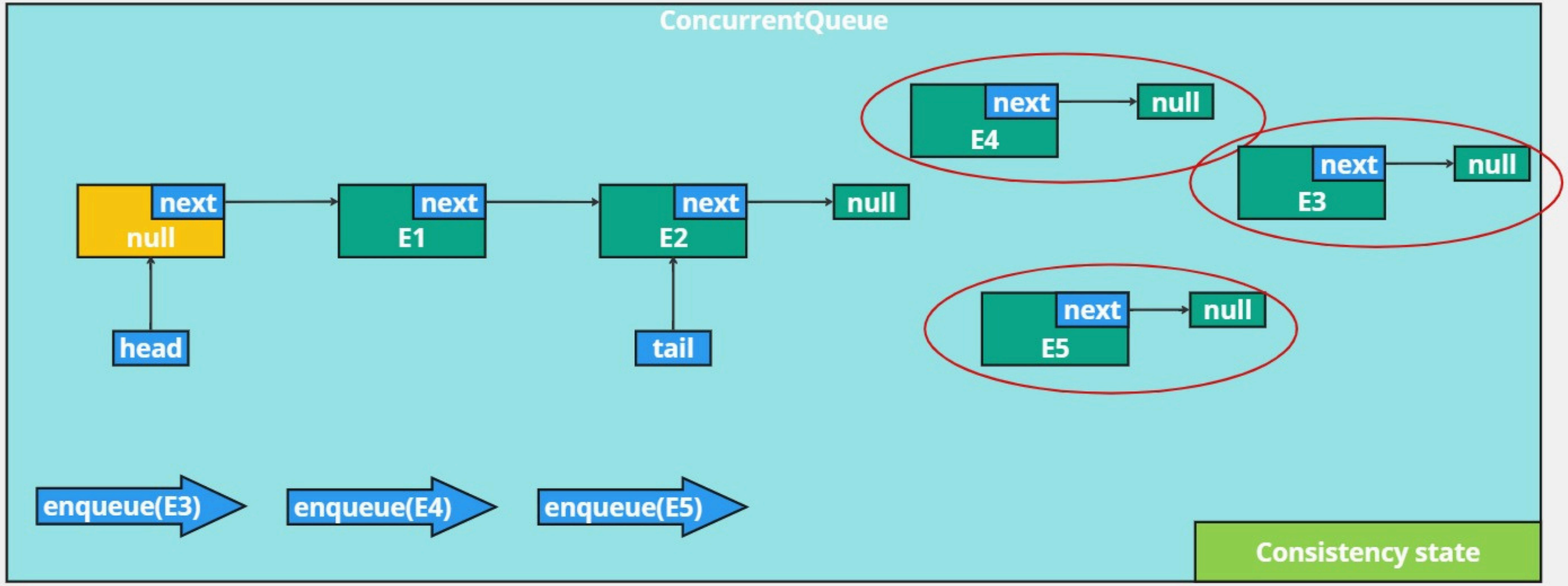
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

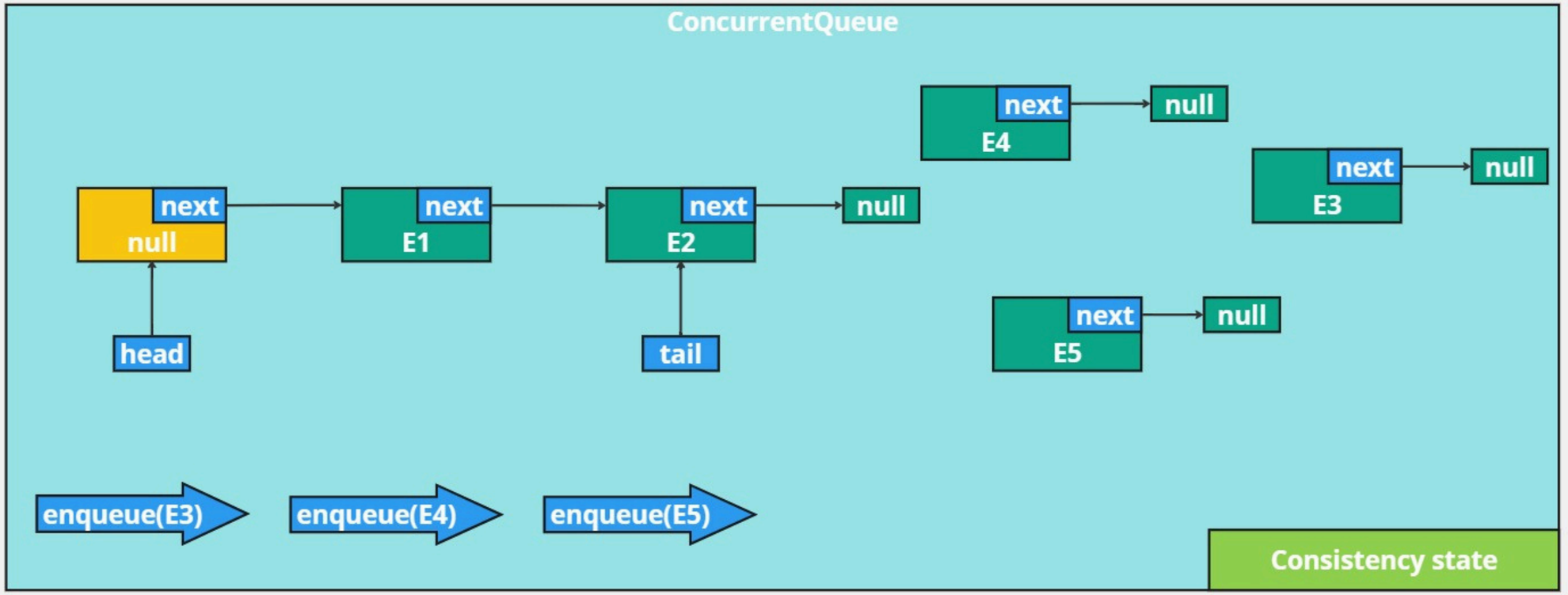
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

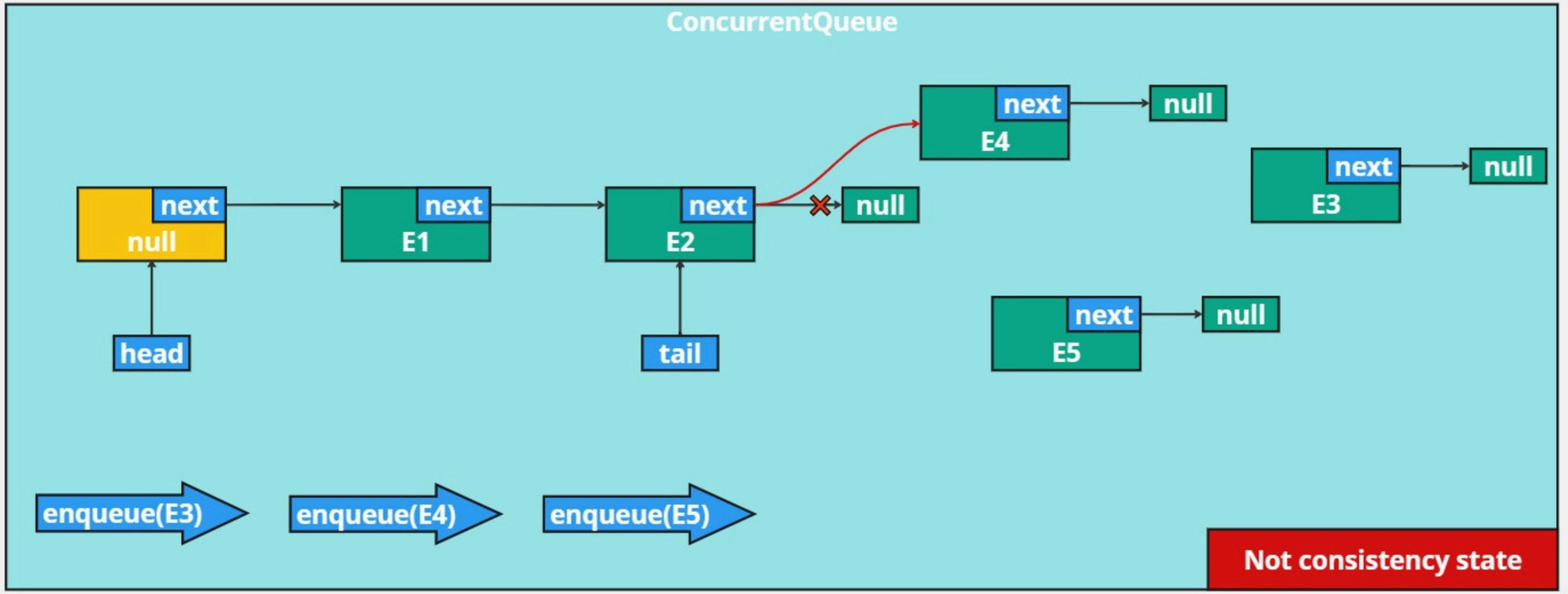
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

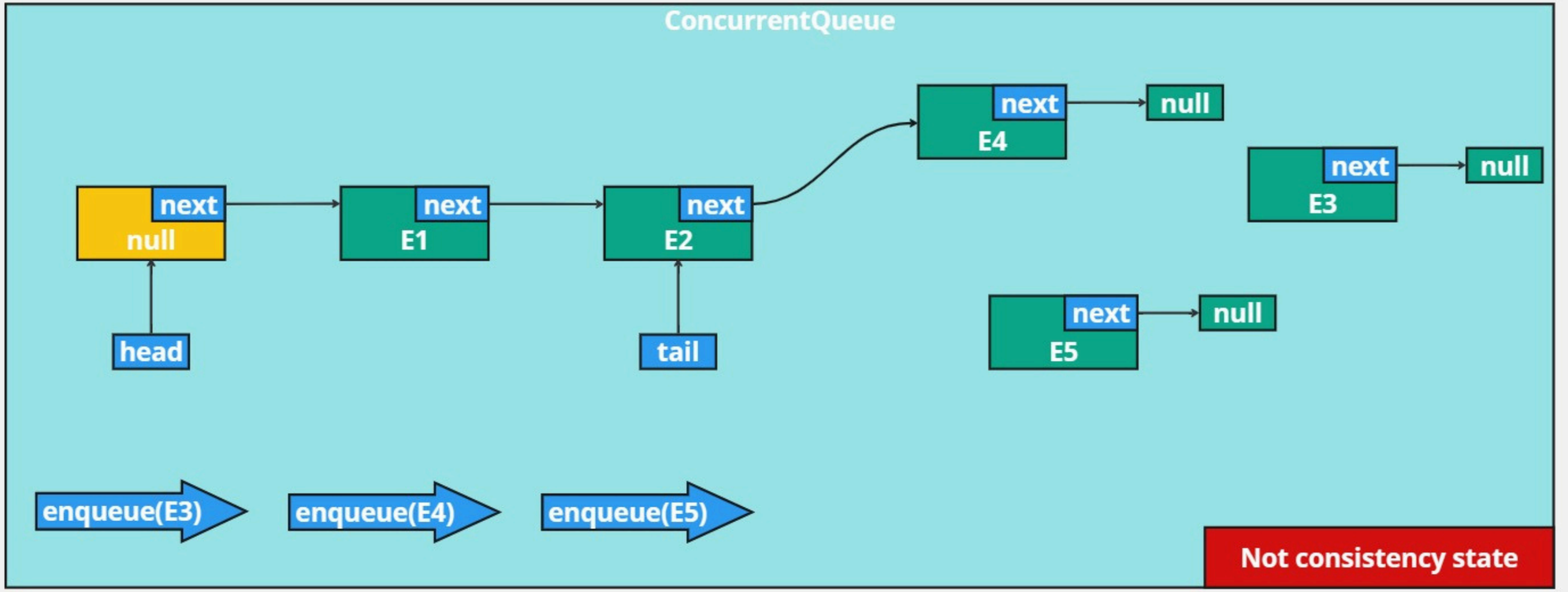
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

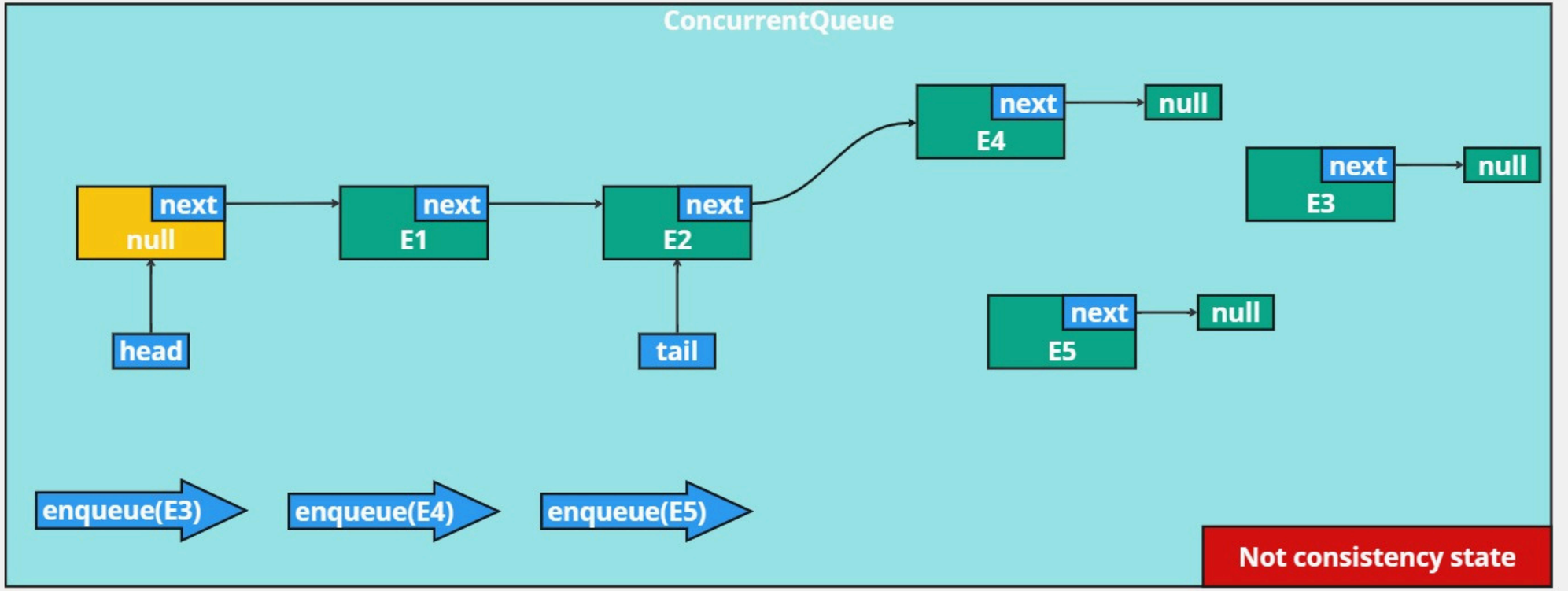
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

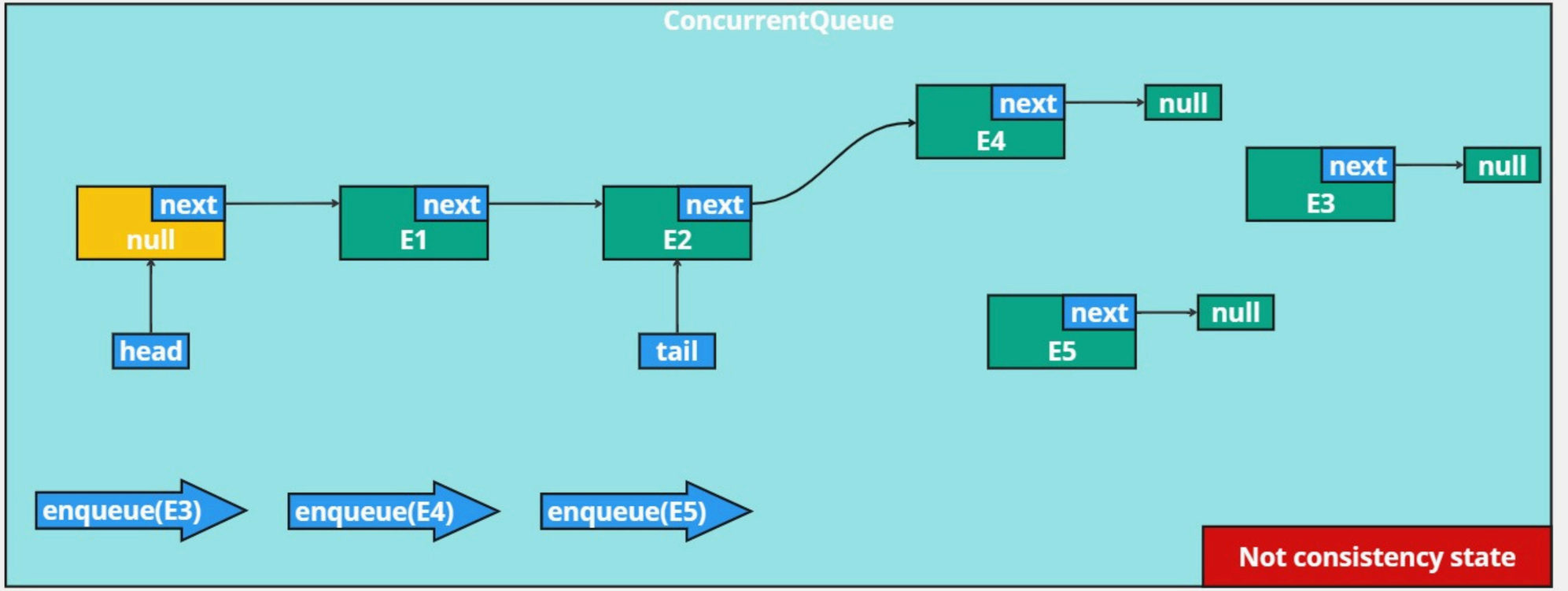
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

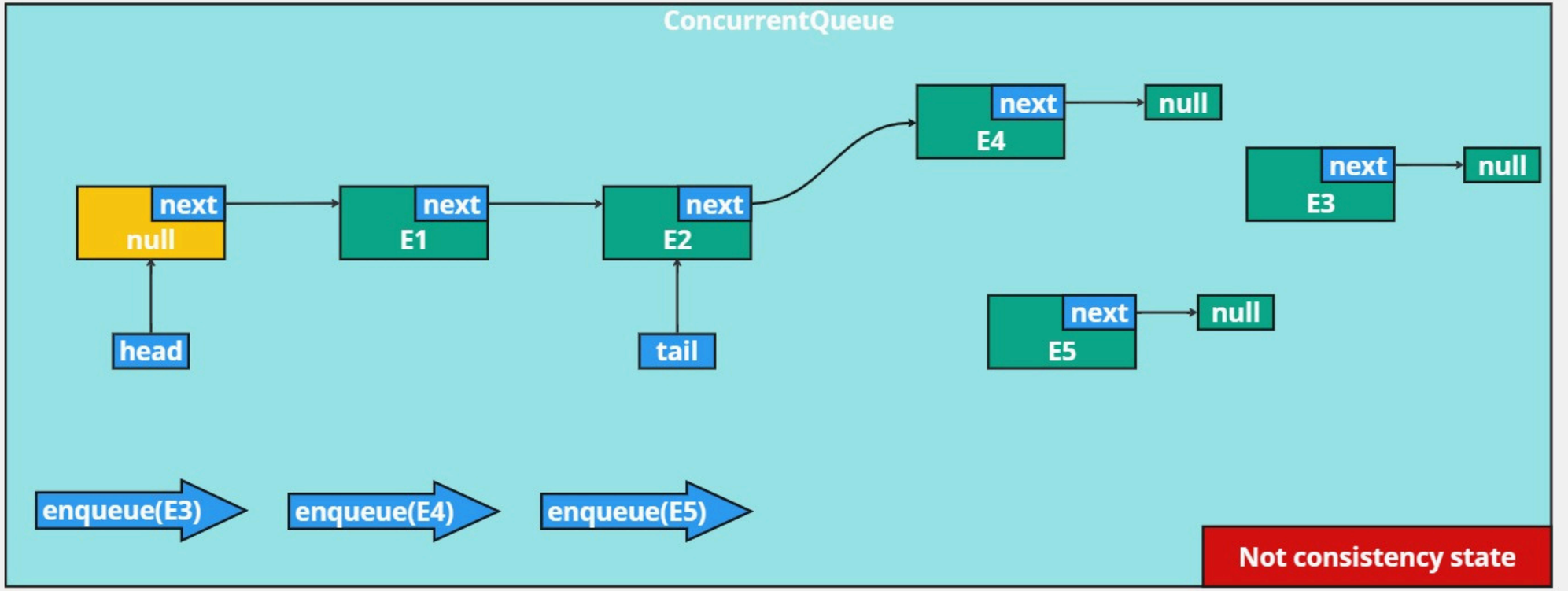
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

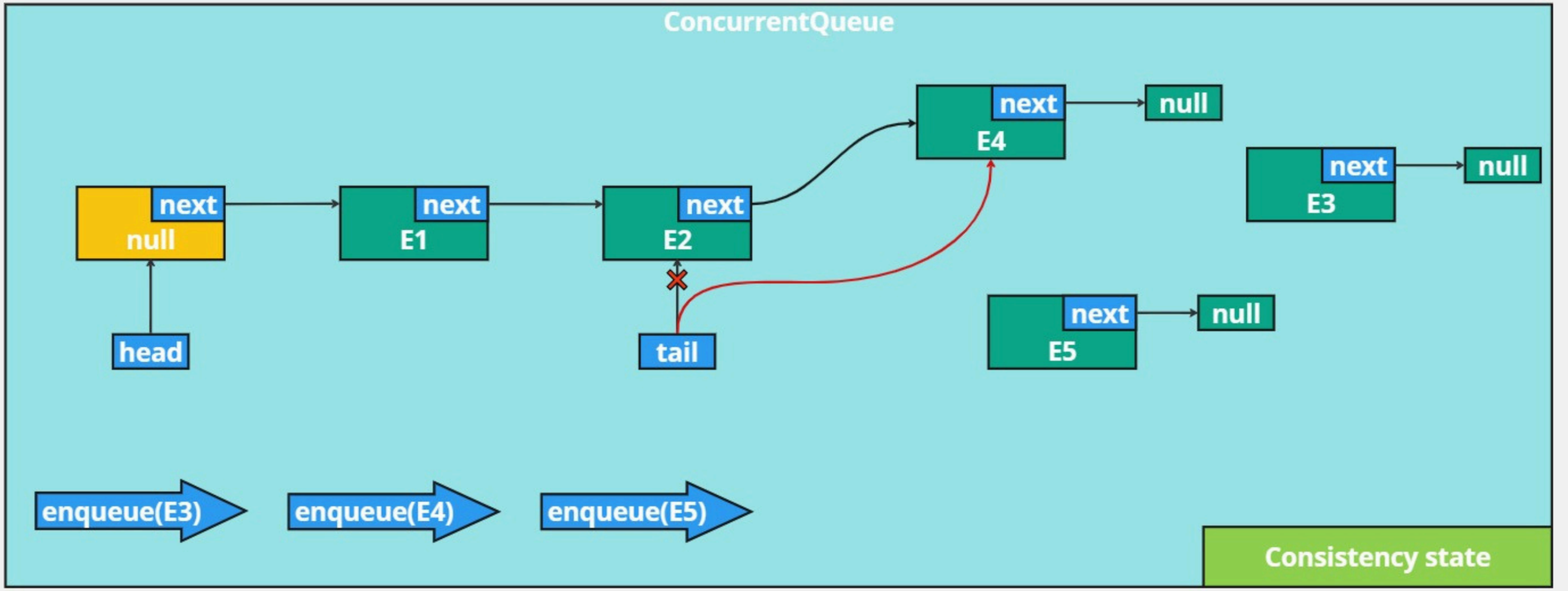
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

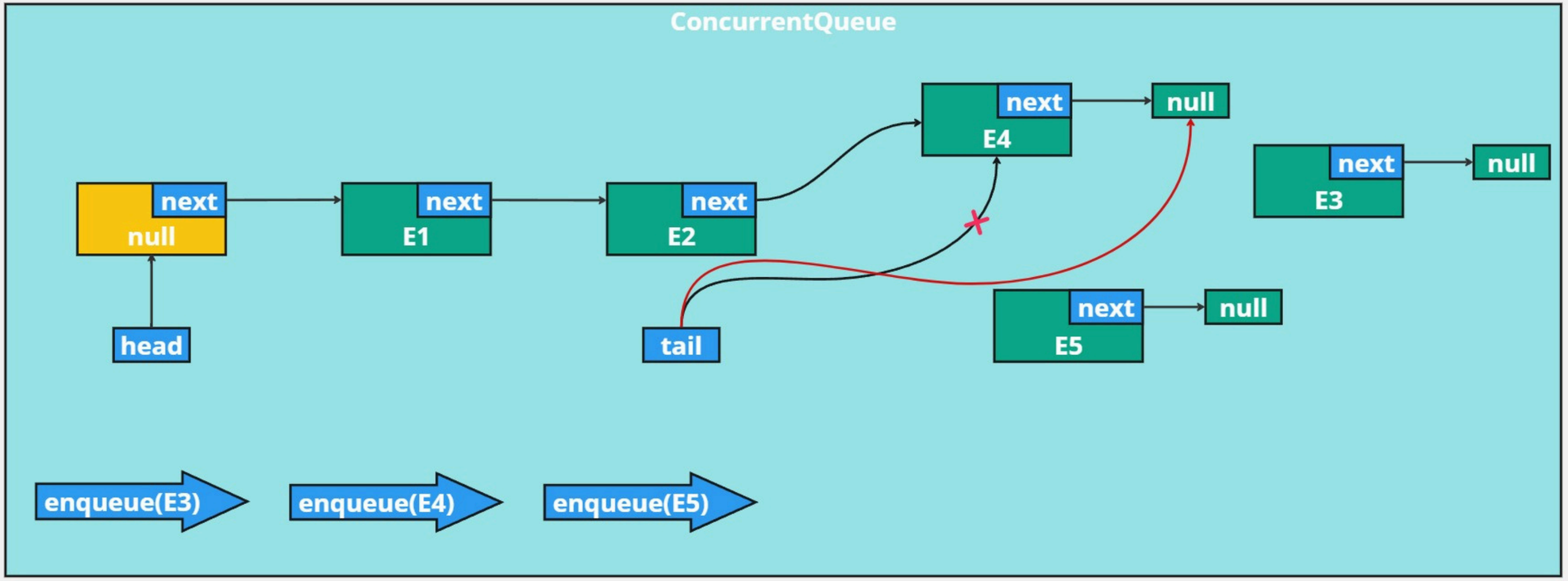
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

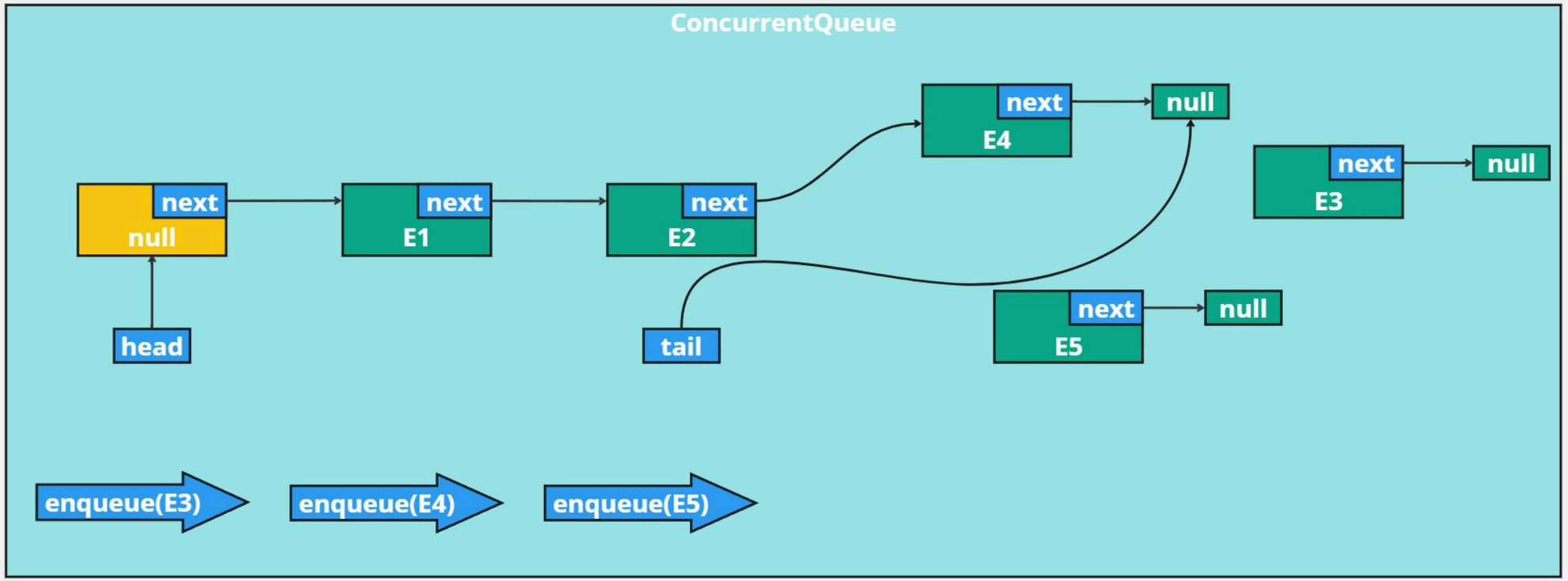
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

NullPointerException

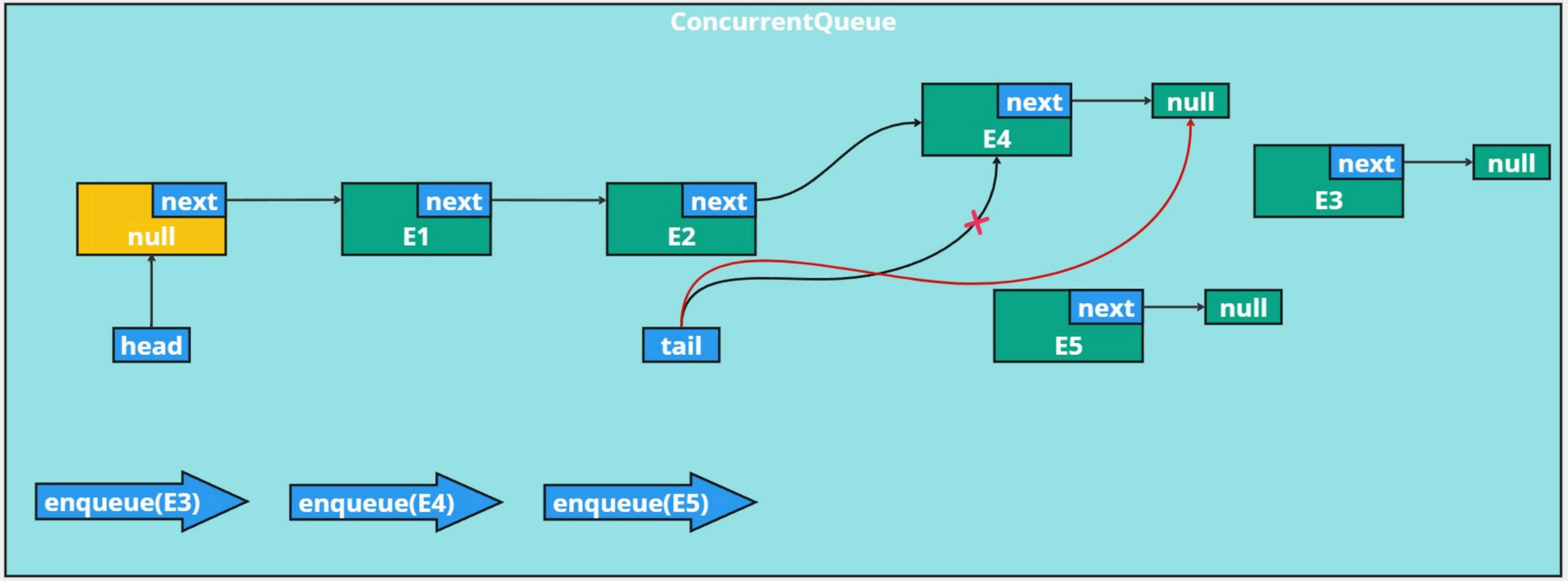
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

NullPointerException



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        if (tail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail = tail.next.get();
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail;
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            if (tail == previousTail) {
                tail = previousTail.next.get();
            }
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail;
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            if (tail == previousTail) {
                tail = previousTail.next.get();
            }
        }
    }
    tail = newNode;
}
```

not atomic

```
private volatile Node<E> tail;
```

```
public ConcurrentQueue() {
    final Node<E> dummyNode = new Node<>();
    head = dummyNode;
    tail = dummyNode;
}
```

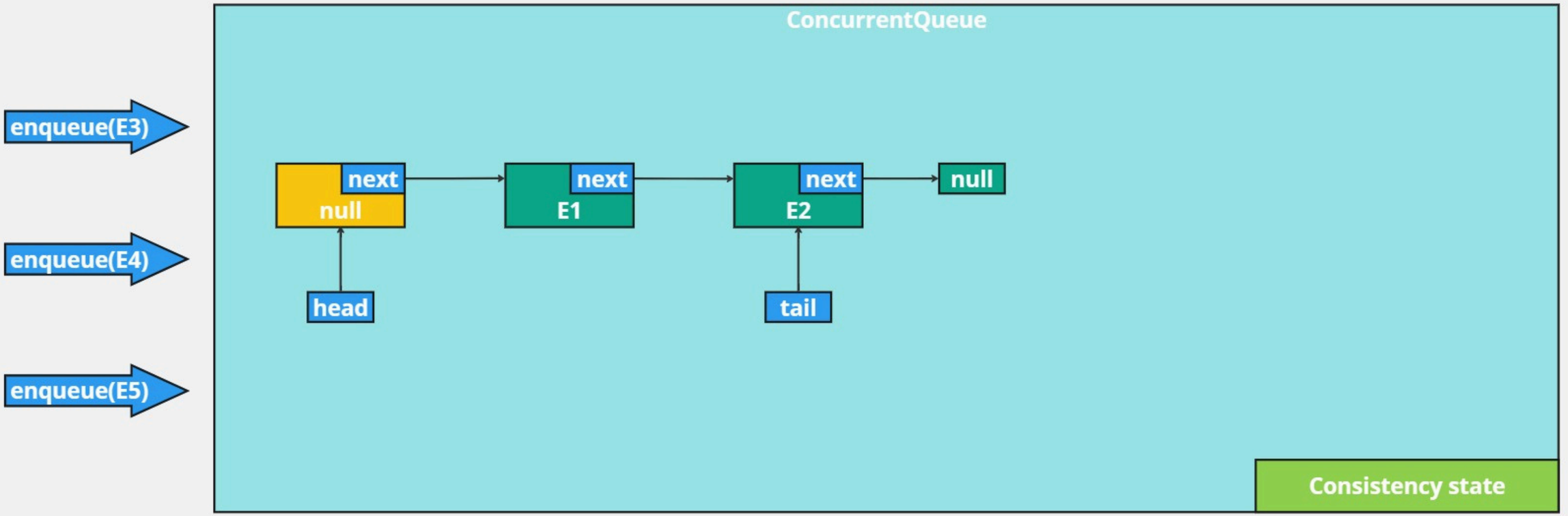
```
private final AtomicReference<Node<E>> tail;
```

```
public ConcurrentQueue() {
    final Node<E> dummyNode = new Node<>();
    head = dummyNode;
    tail = new AtomicReference<>(dummyNode);
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail;
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            if (tail == previousTail) {
                tail = previousTail.next.get();
            }
        }
    }
    tail = newNode;
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

atomic



enqueue(E3)

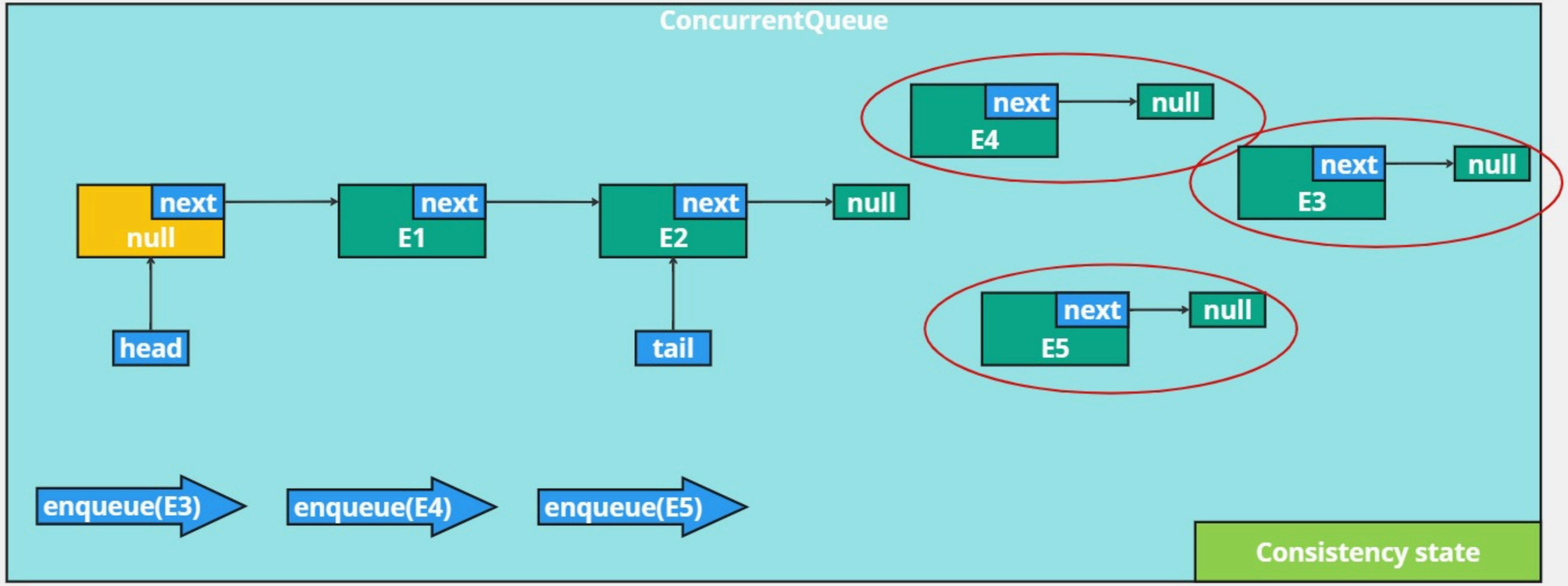
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

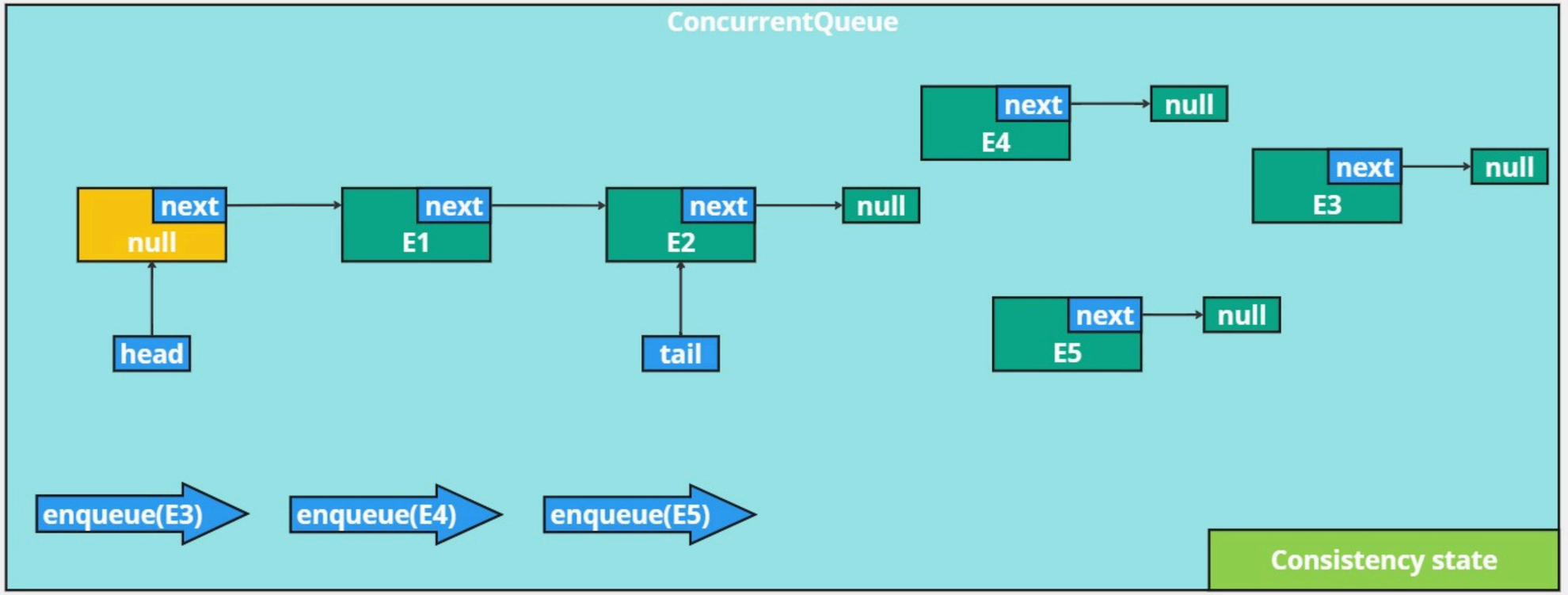
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

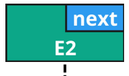
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

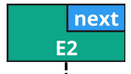


enqueue(E3)



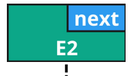

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

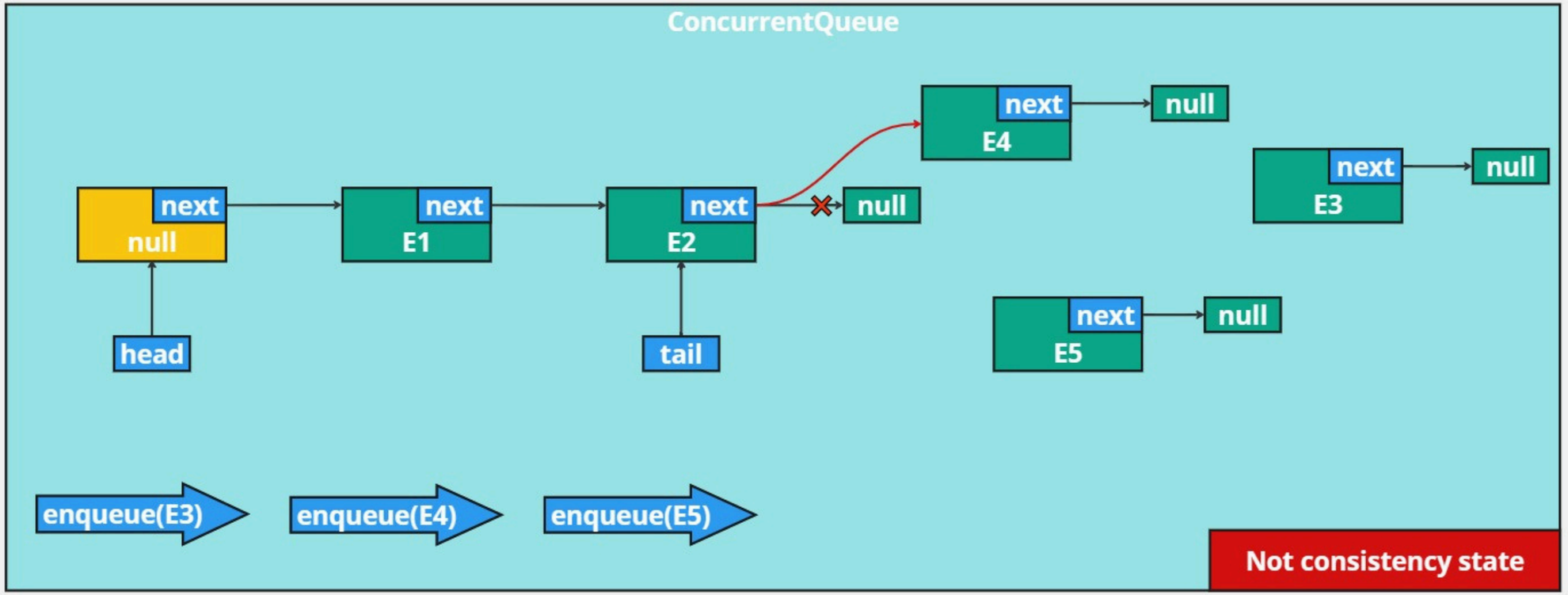



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

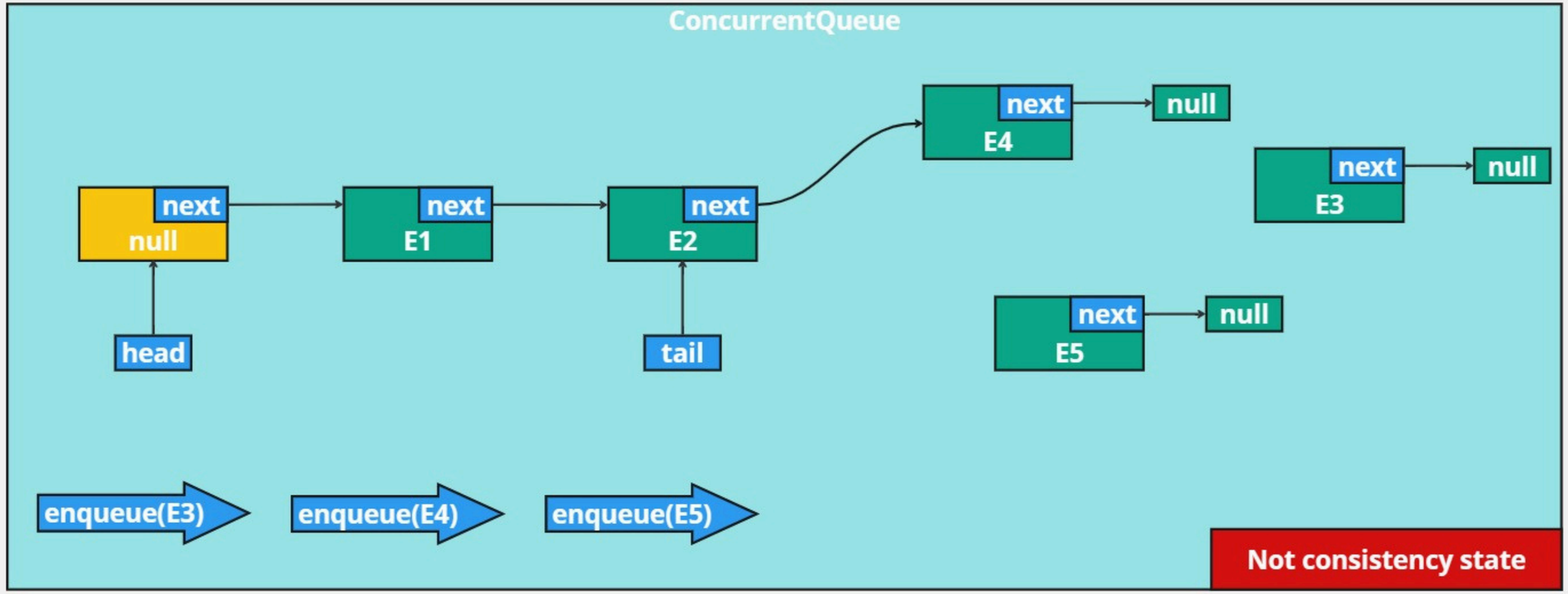
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

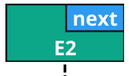
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

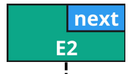


enqueue(E3)



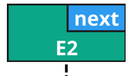

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

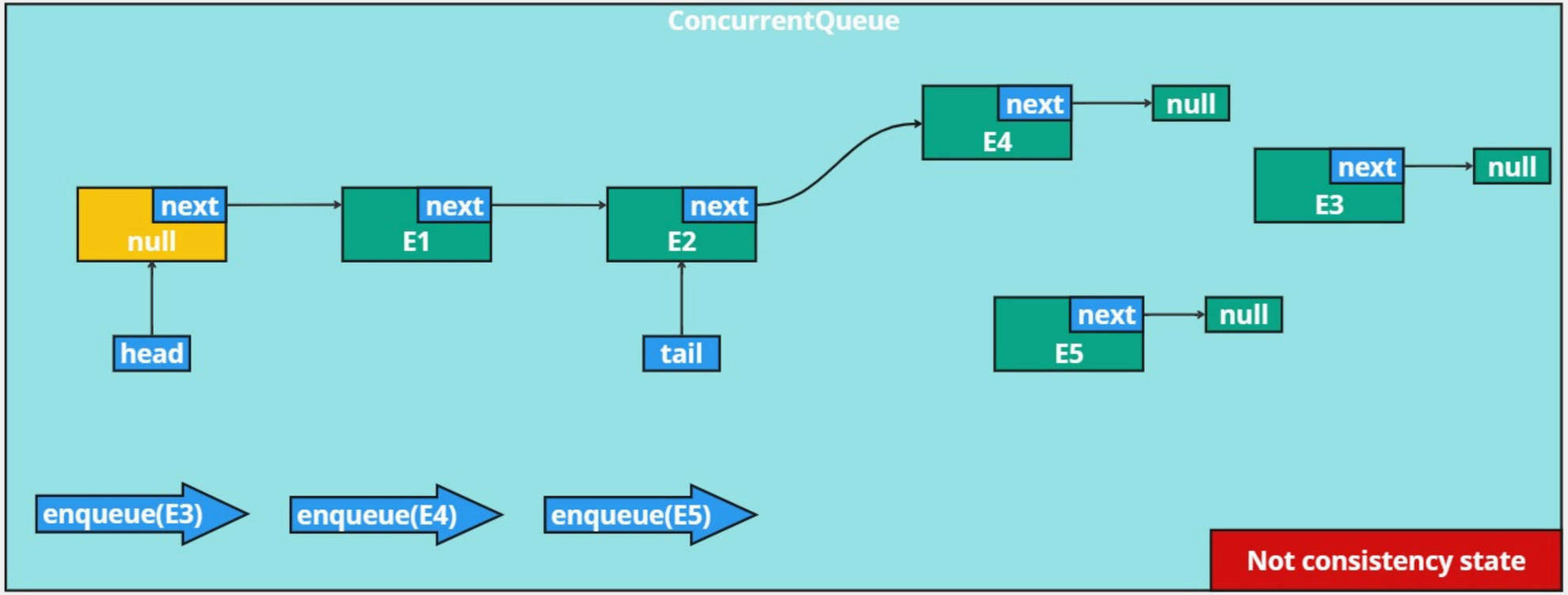



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

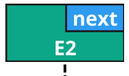
enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

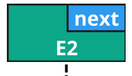



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

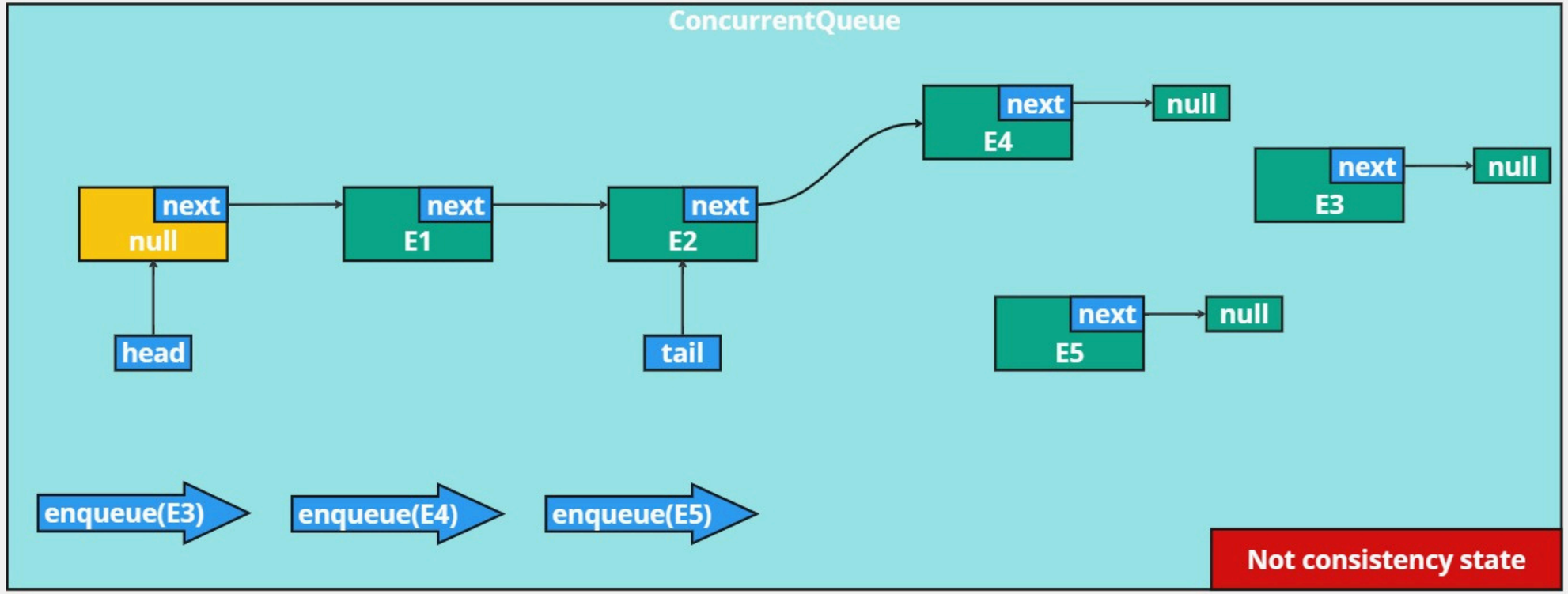
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

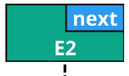
enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

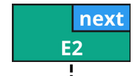



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

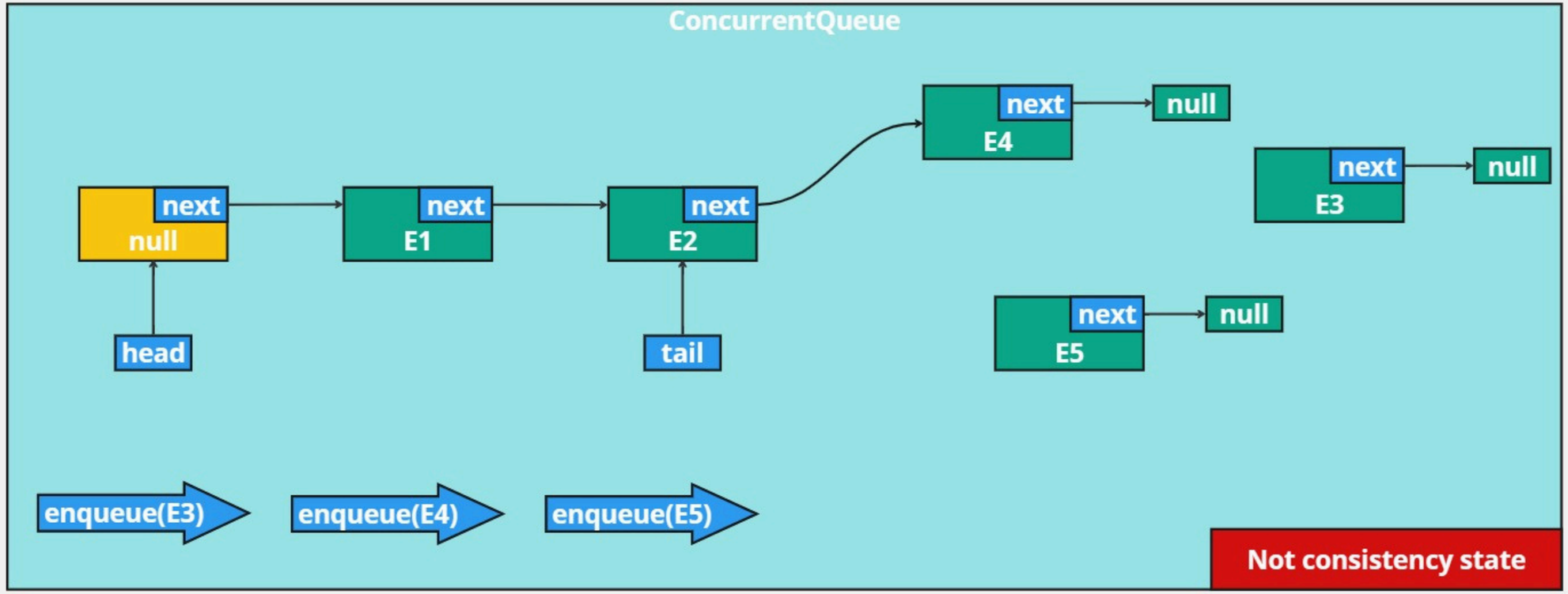
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

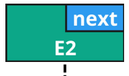
enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

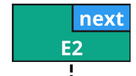



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

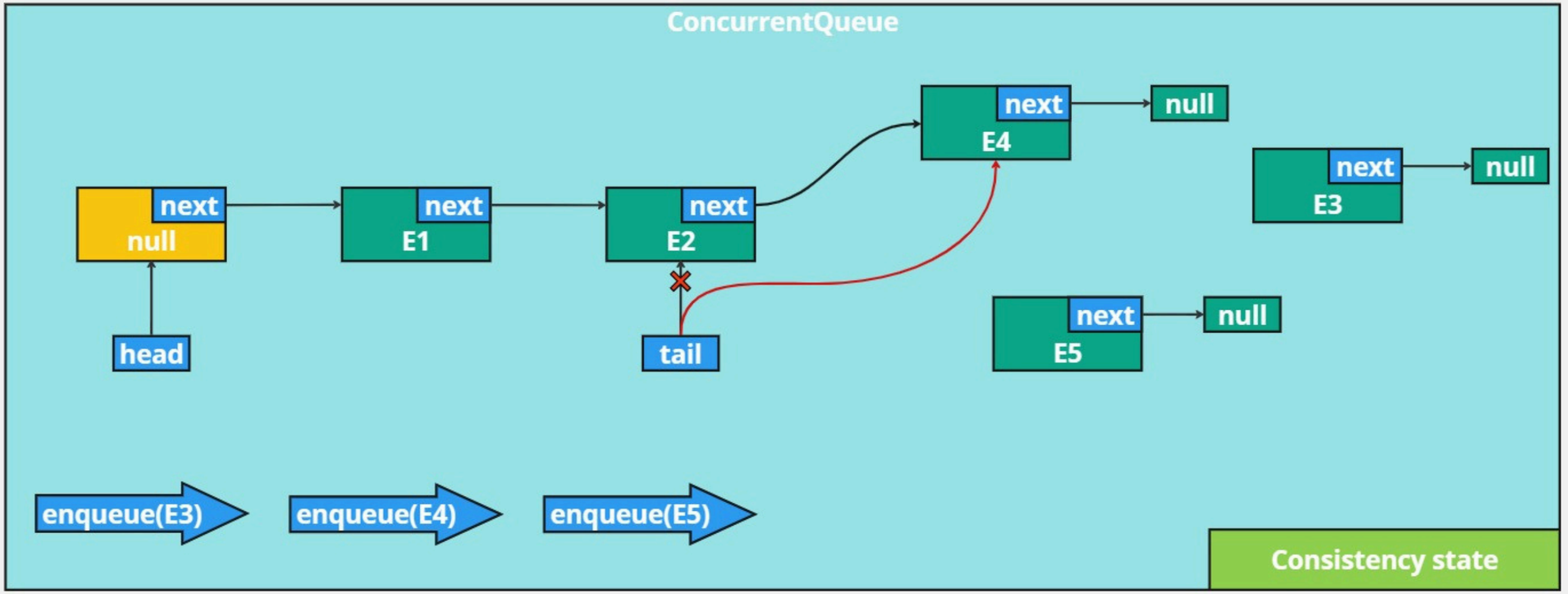
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

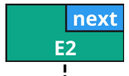
enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

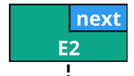



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

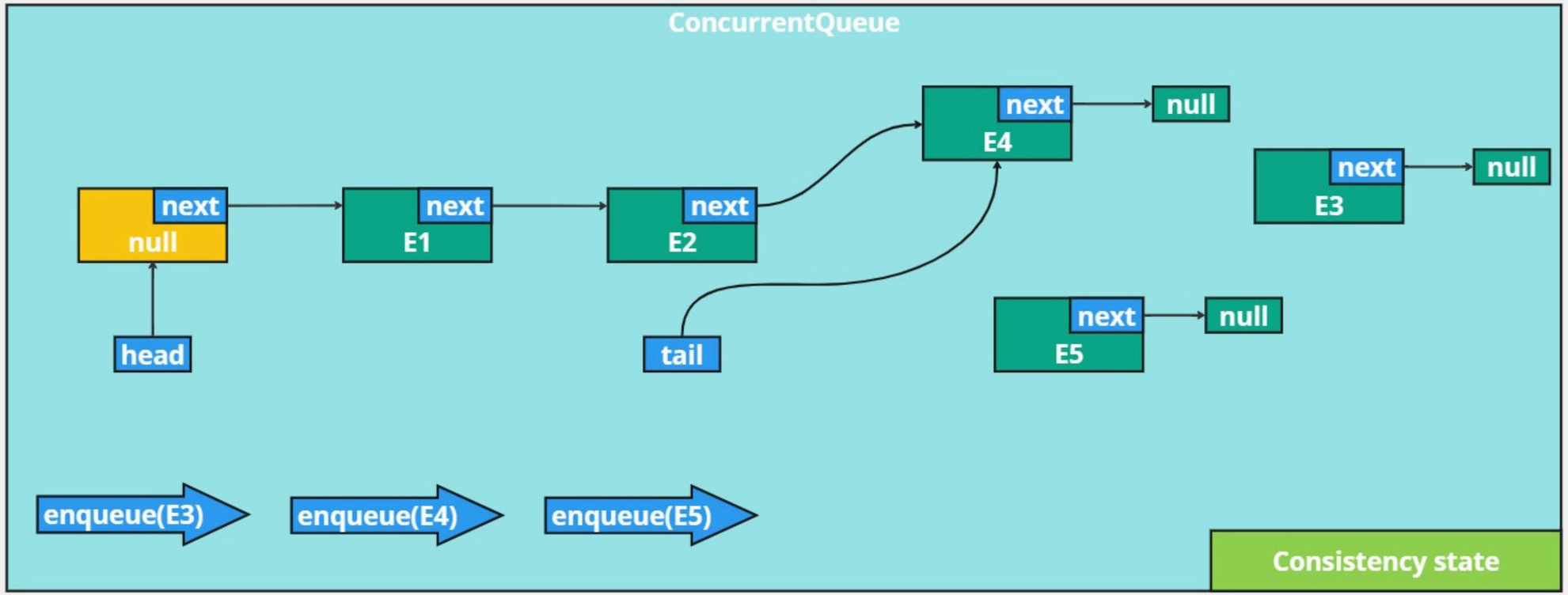
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



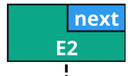
enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

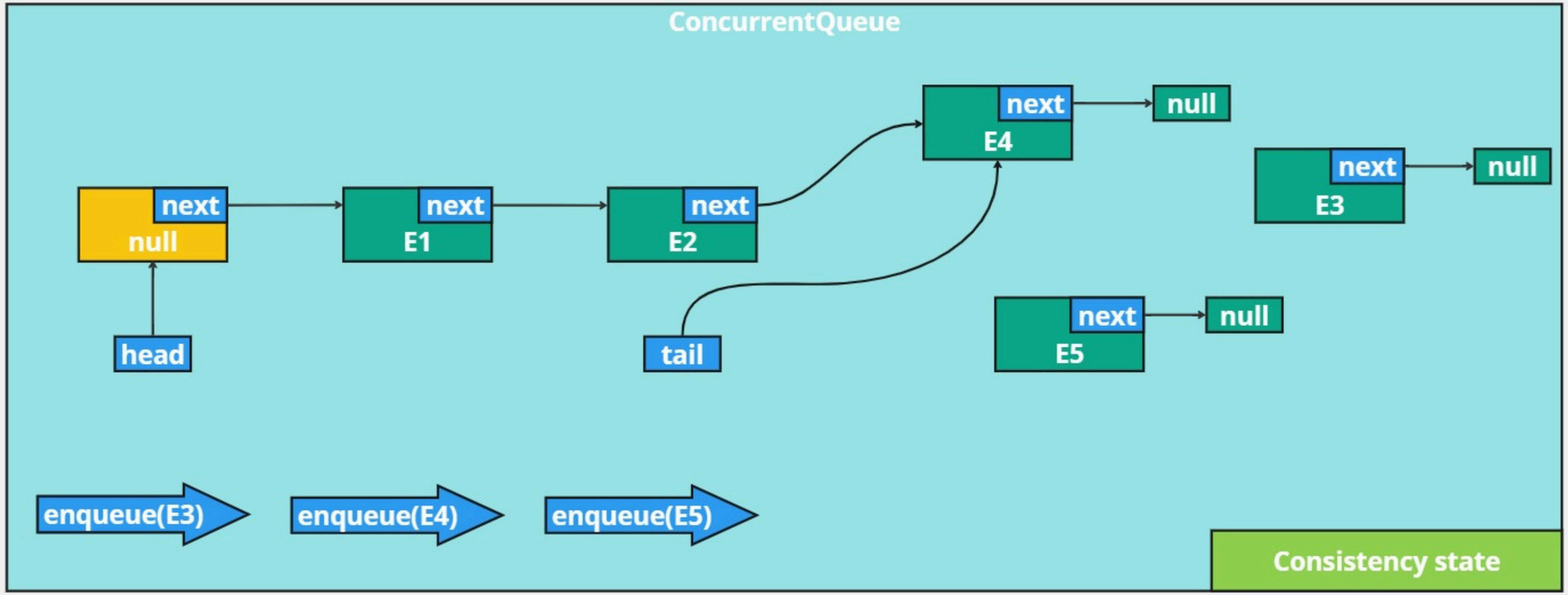
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

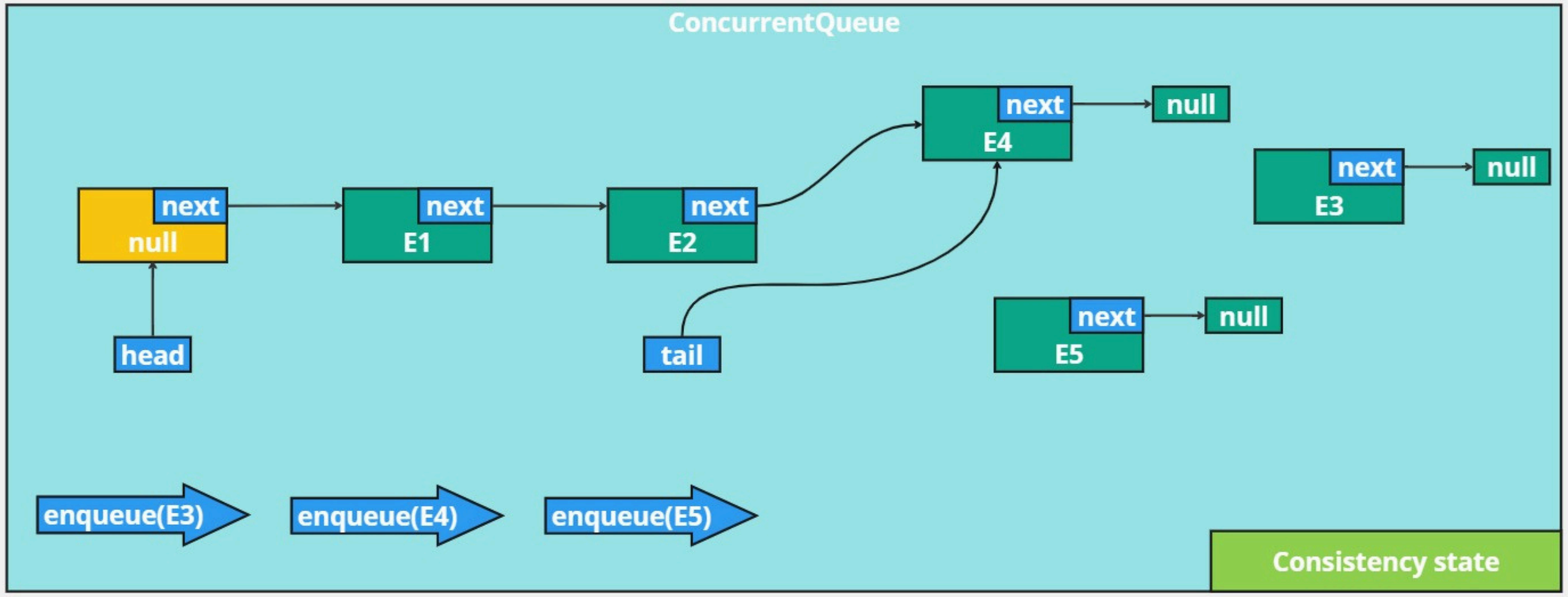
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

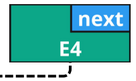
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

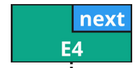



```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

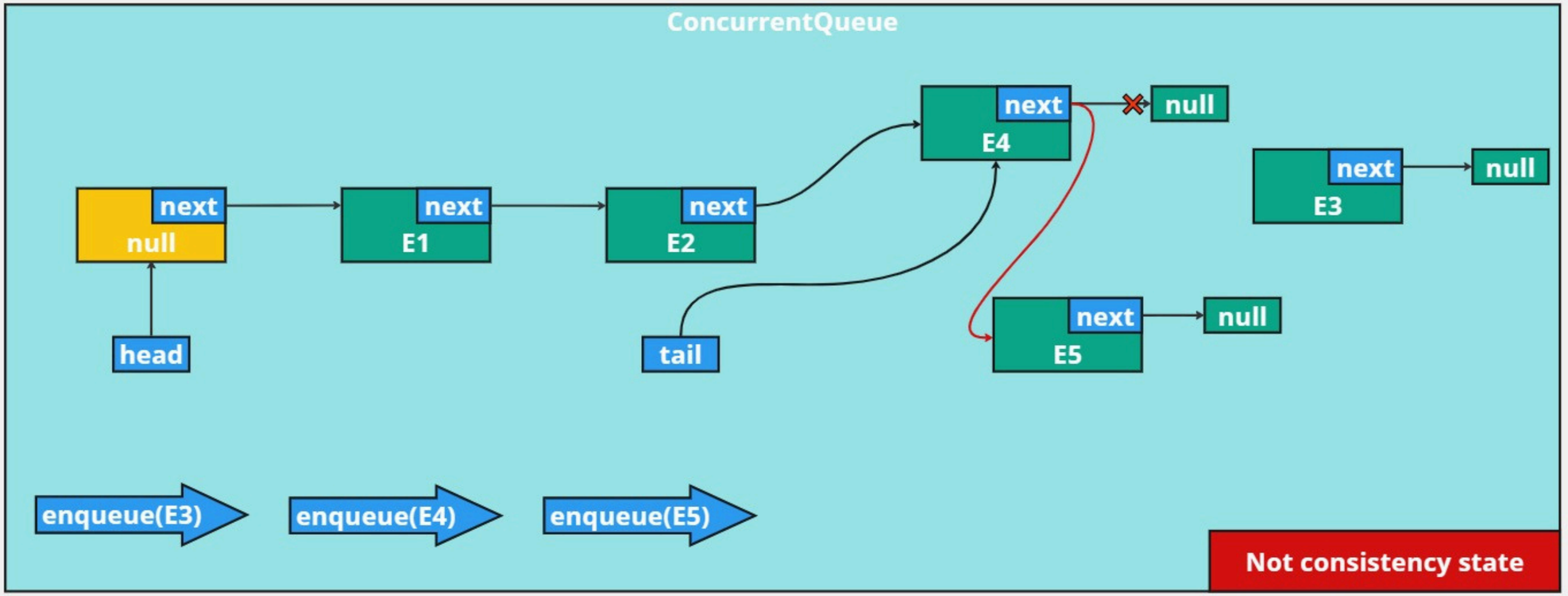
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

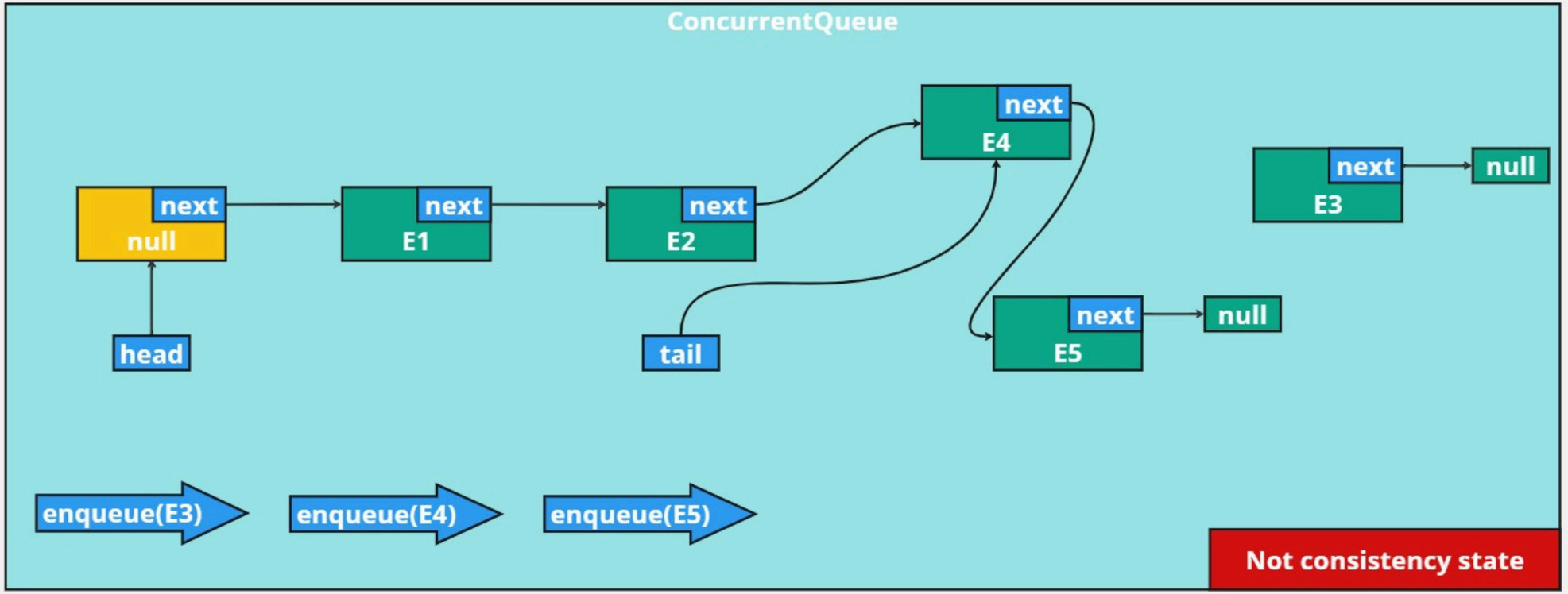
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

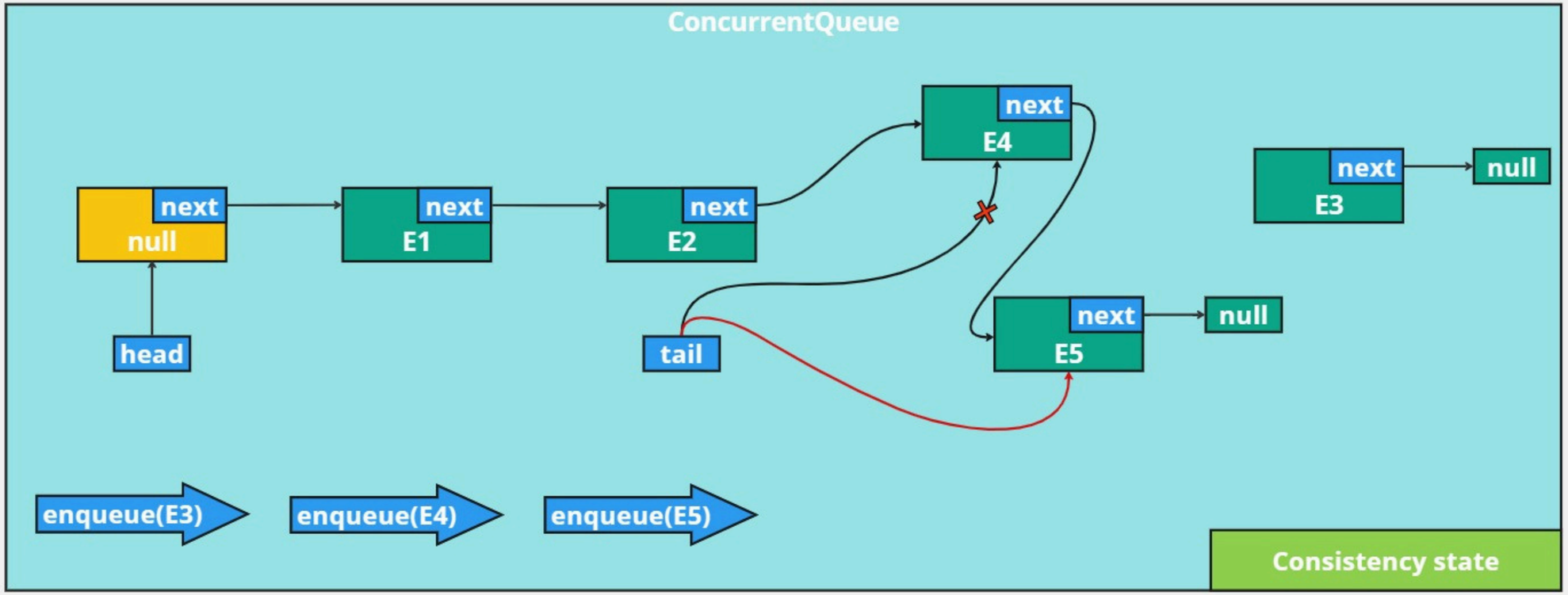
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

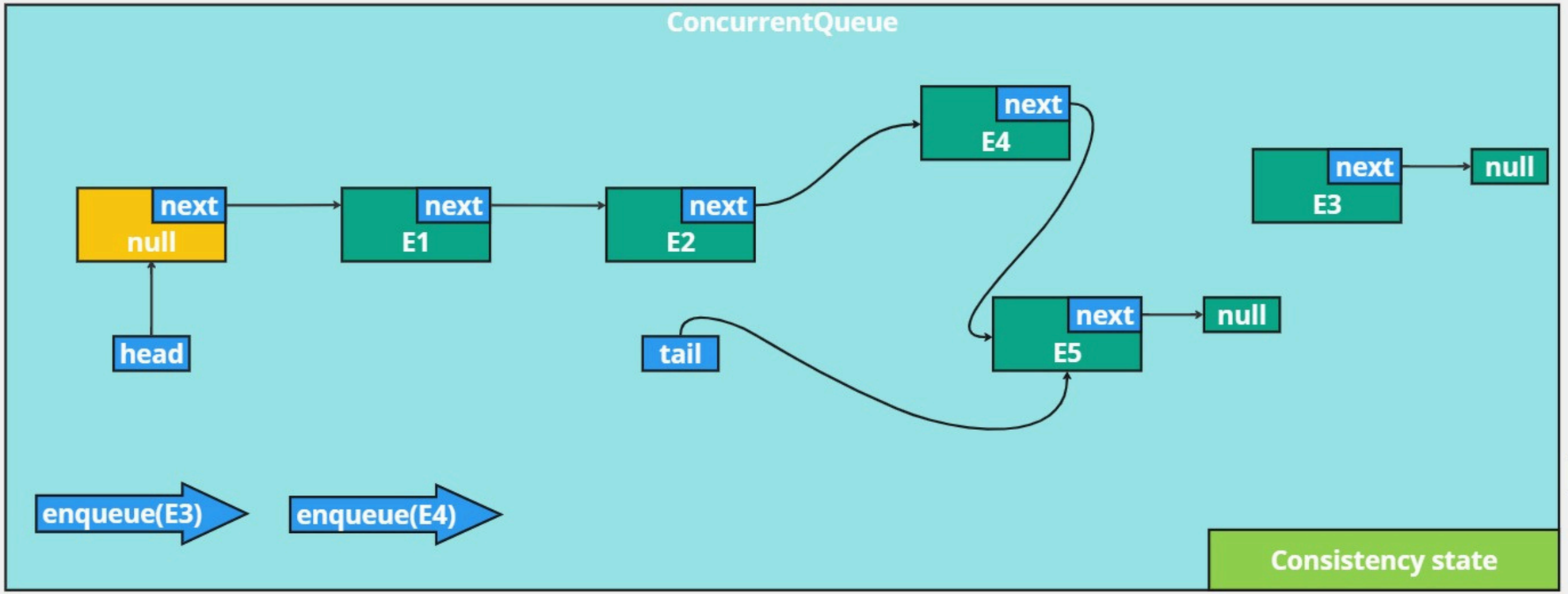
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

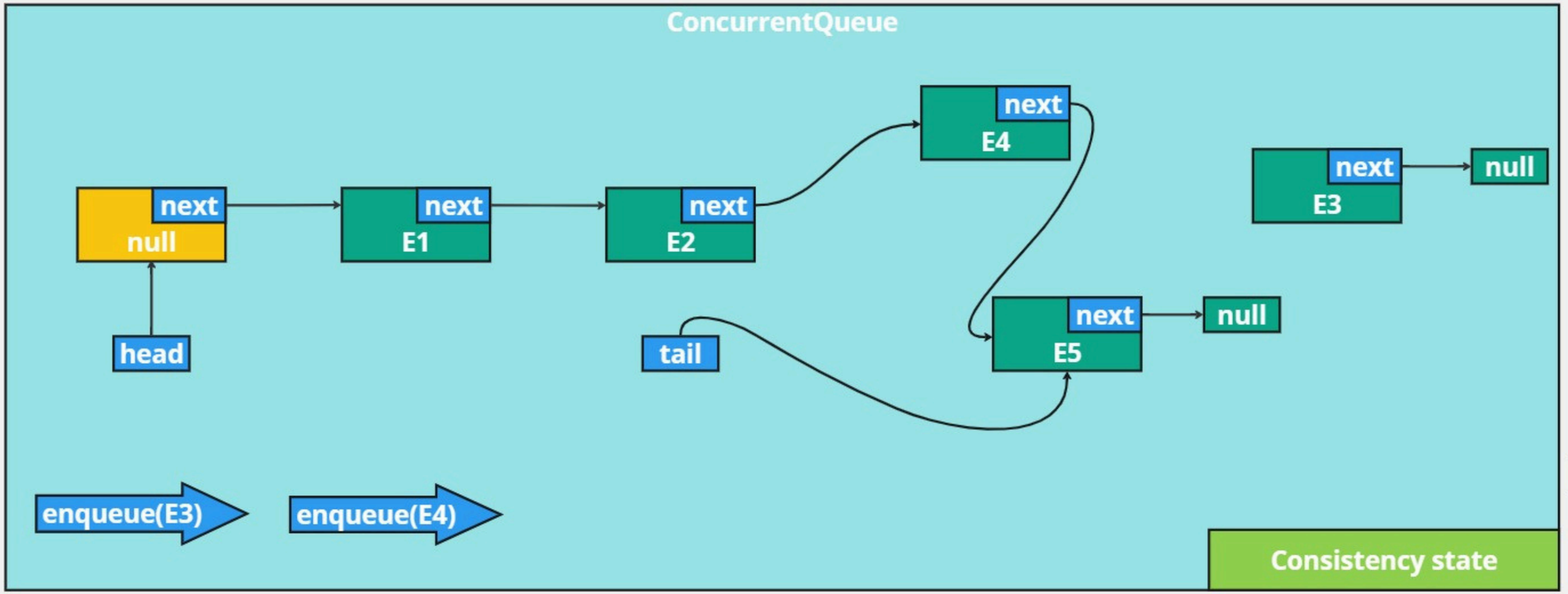
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

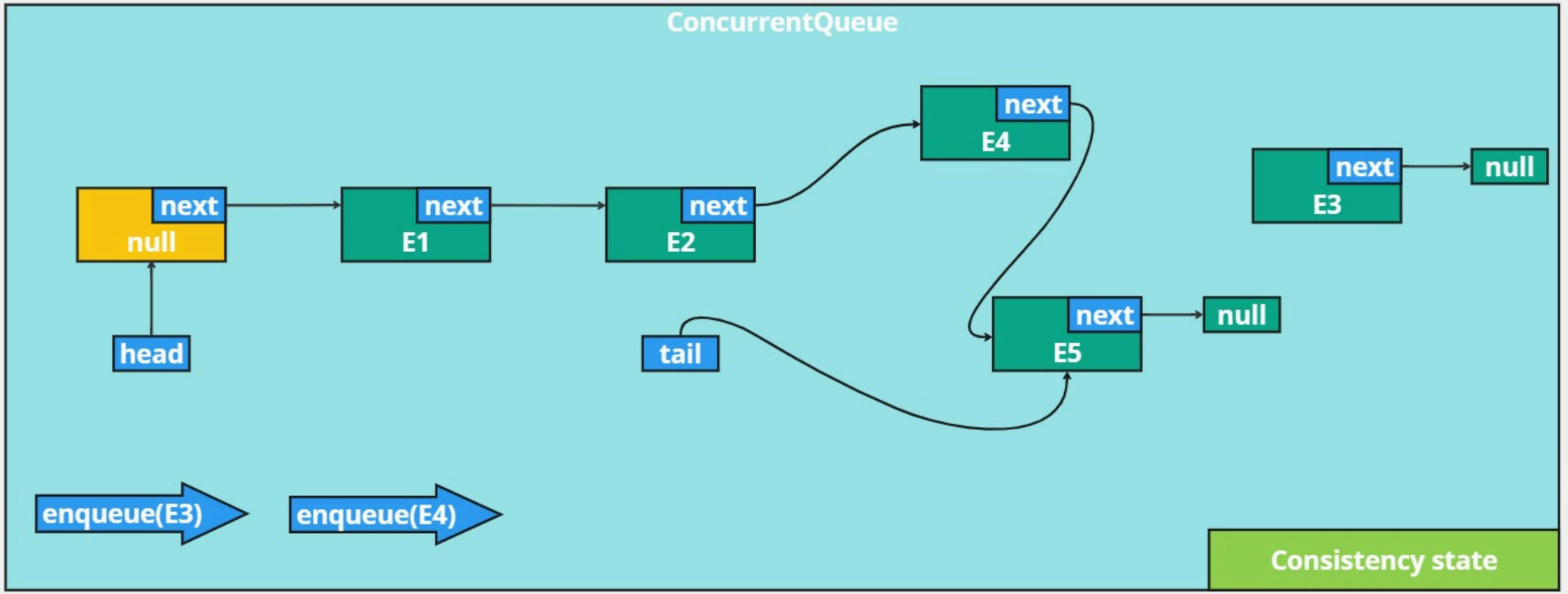
next E4

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

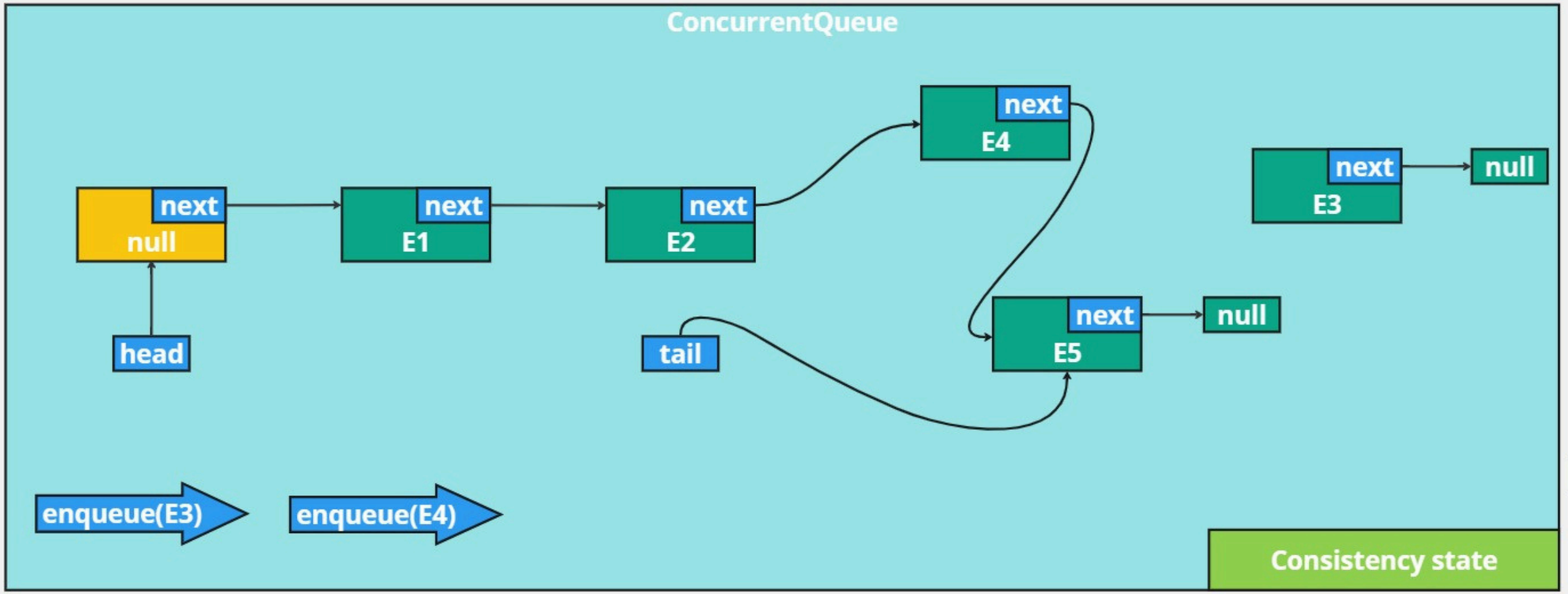
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

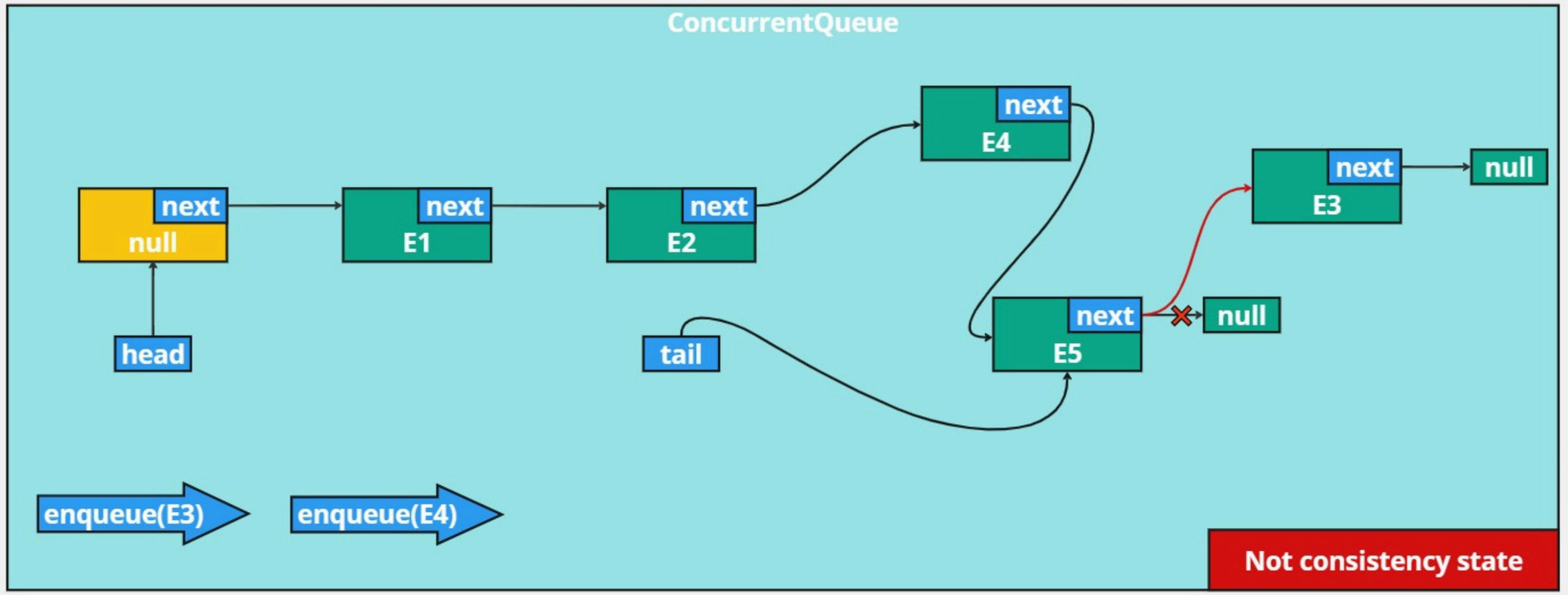
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

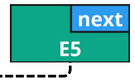
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)



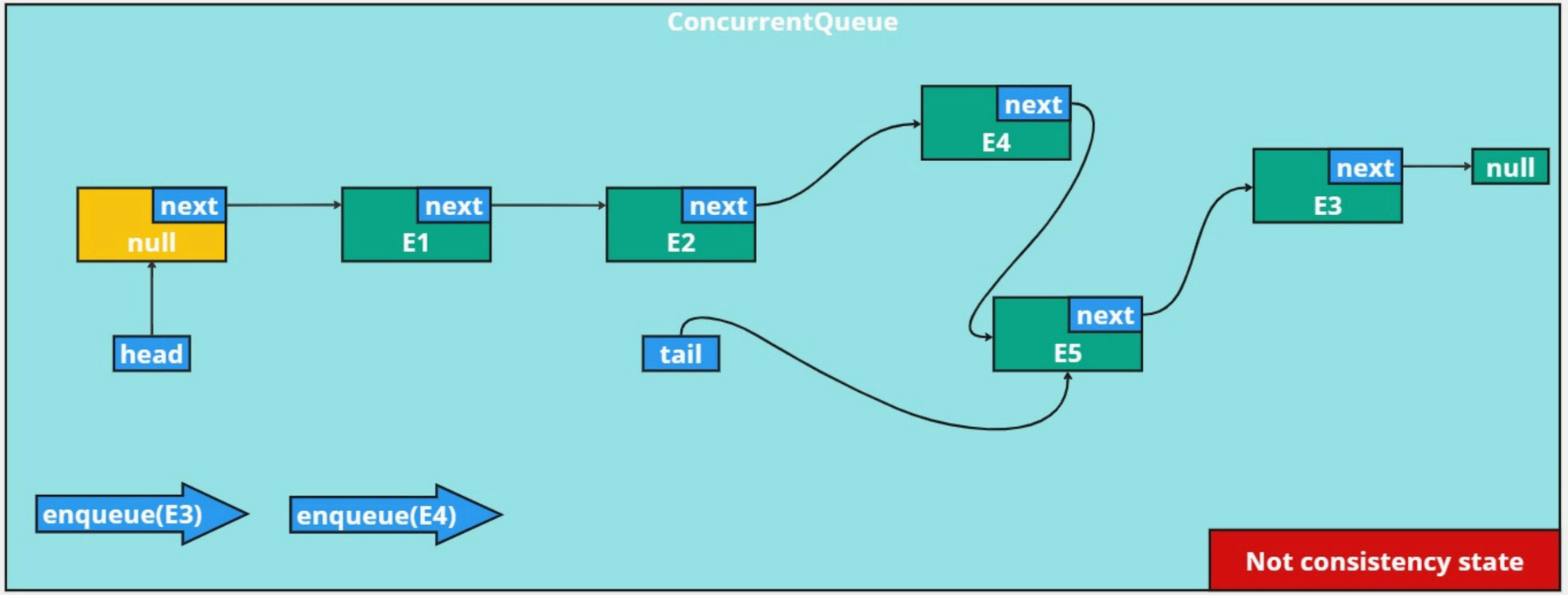

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

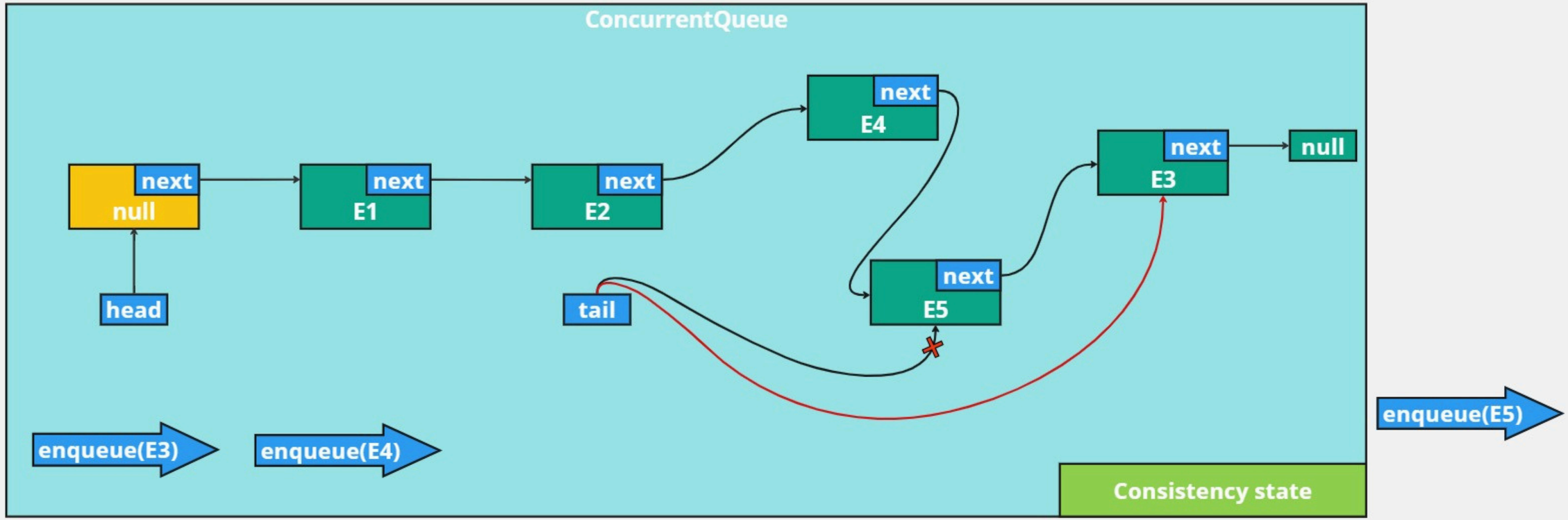
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

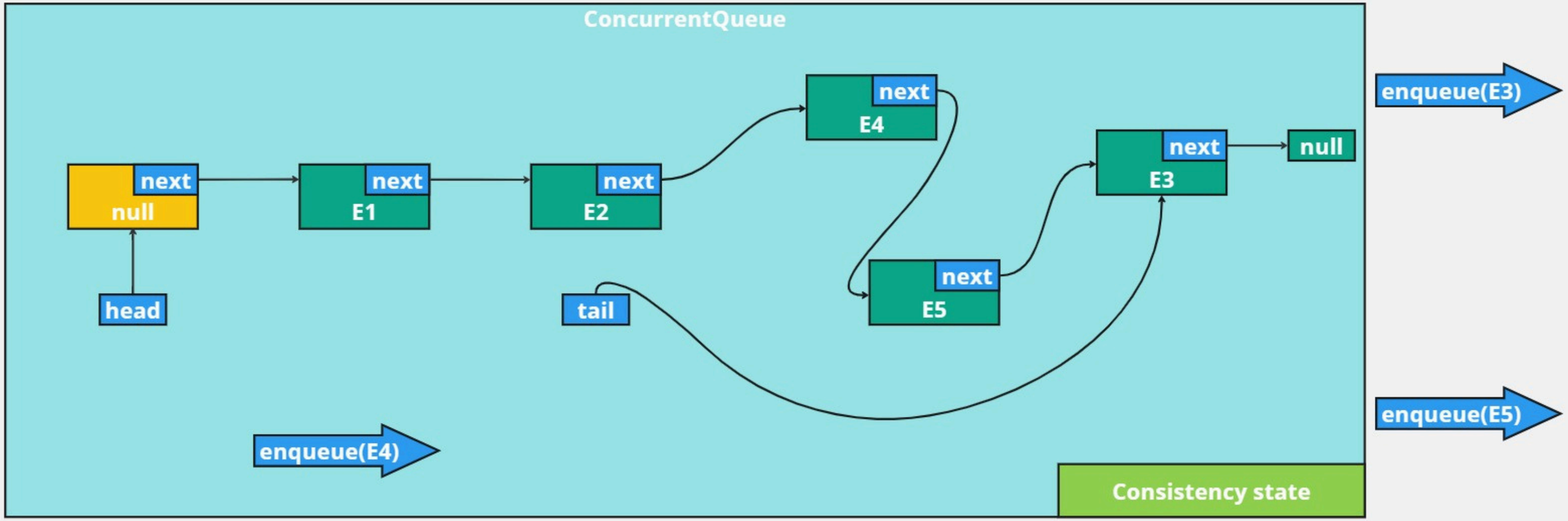
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

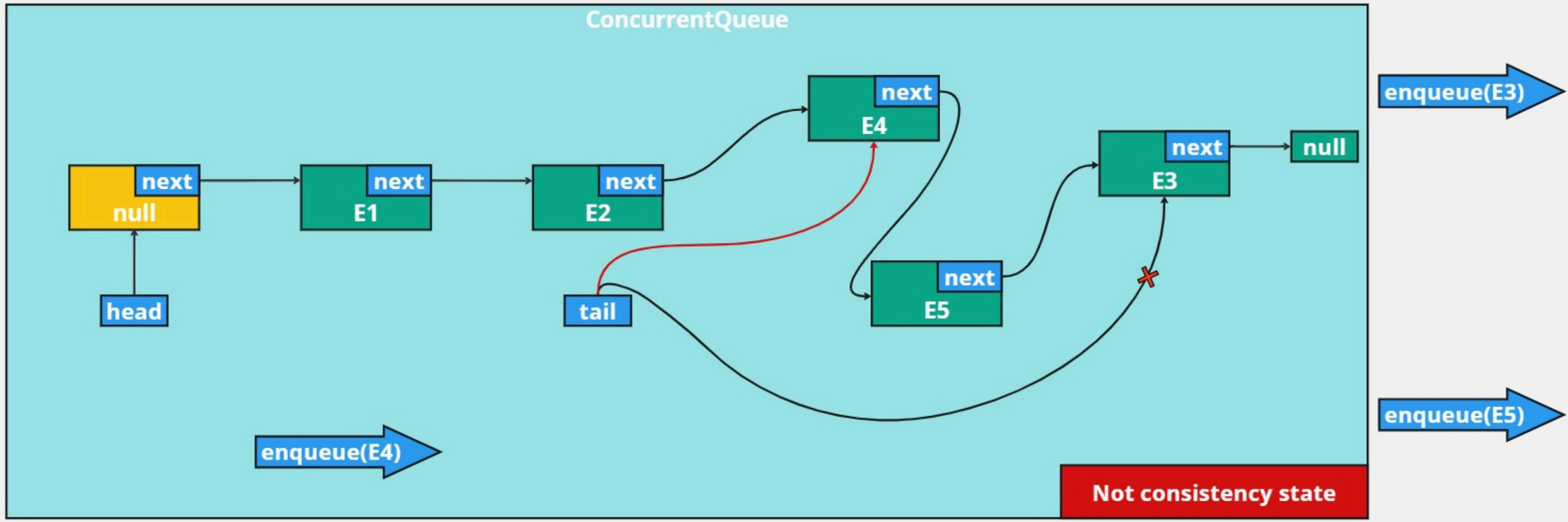
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

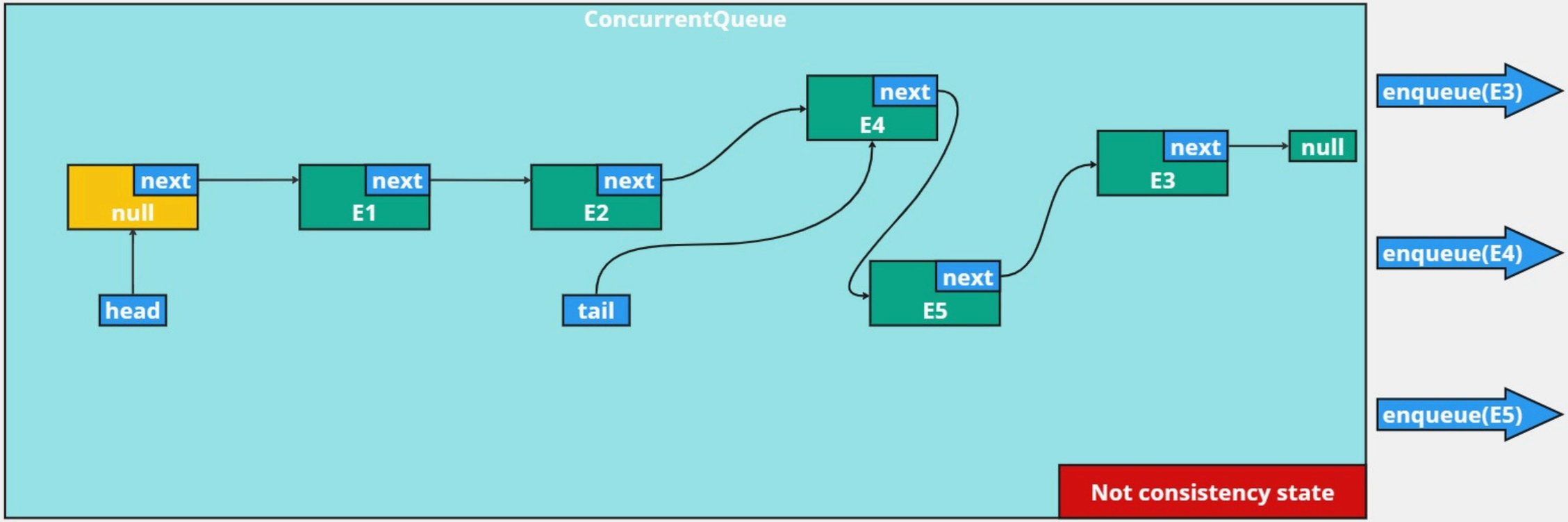
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            break;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
    tail.set(newNode);
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            if (tail.get() == previousTail) {
                tail.set(newNode);
            }
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

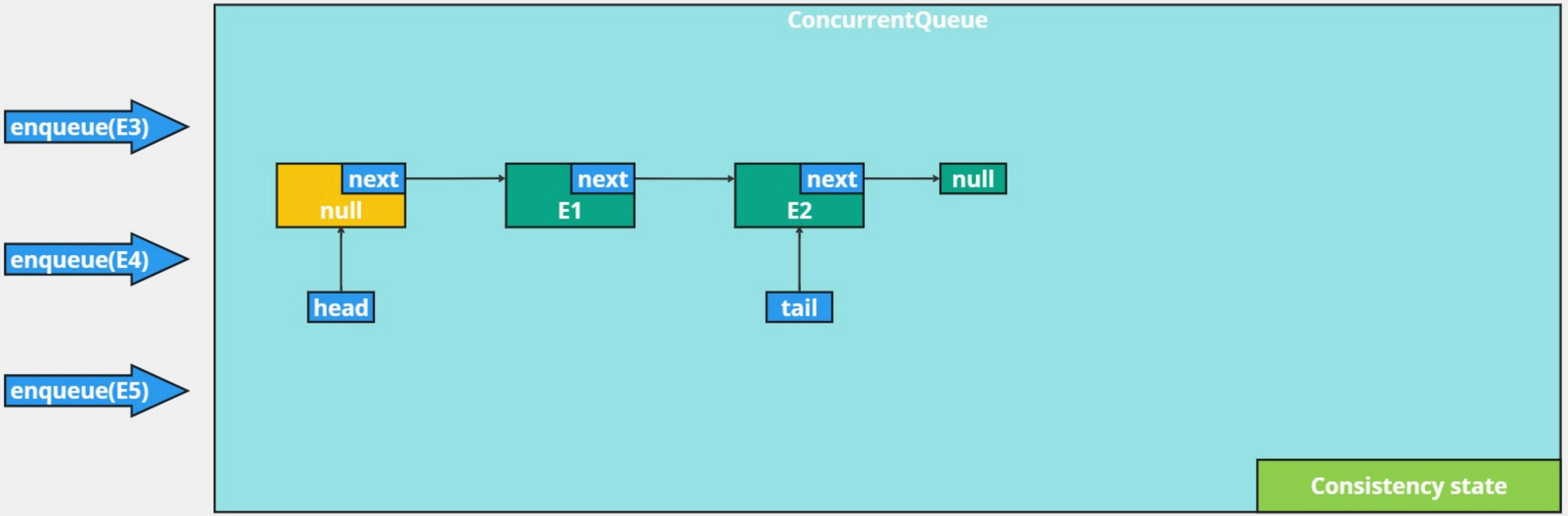
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            if (tail.get() == previousTail) {
                tail.set(newNode);
            }
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

not atomic

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            if (tail.get() == previousTail) {
                tail.set(newNode);
            }
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

atomic



enqueue(E3)

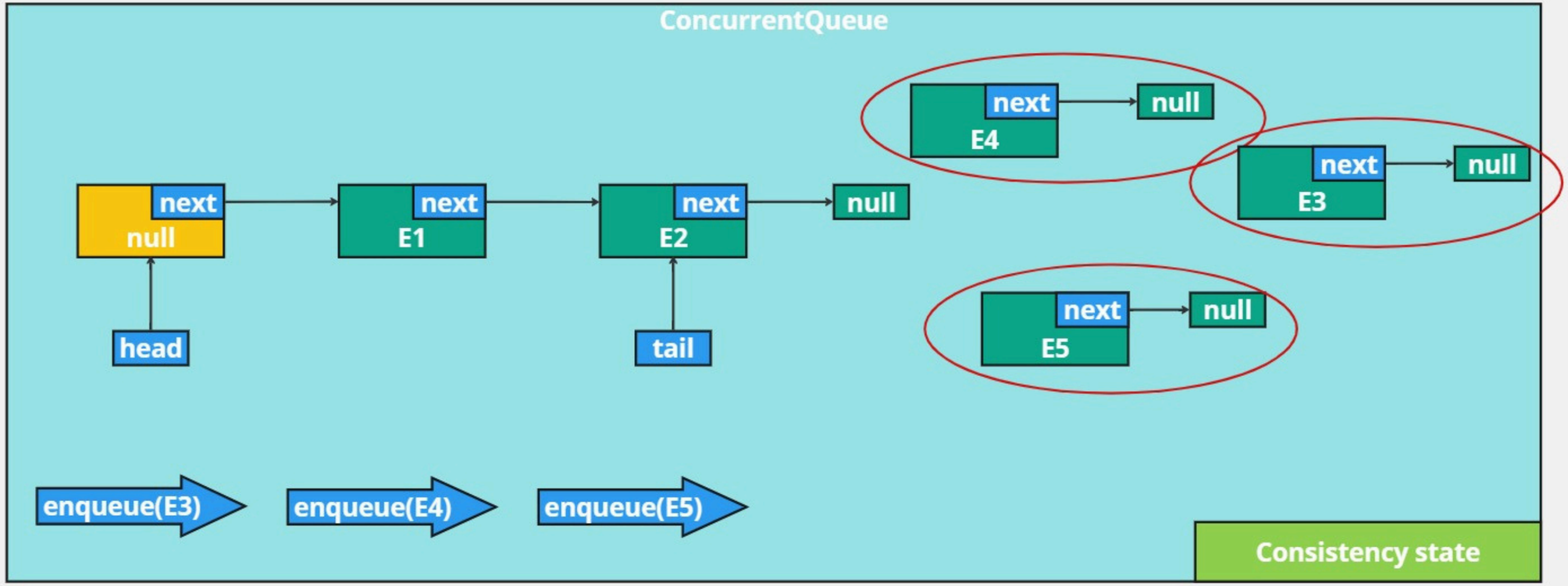
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

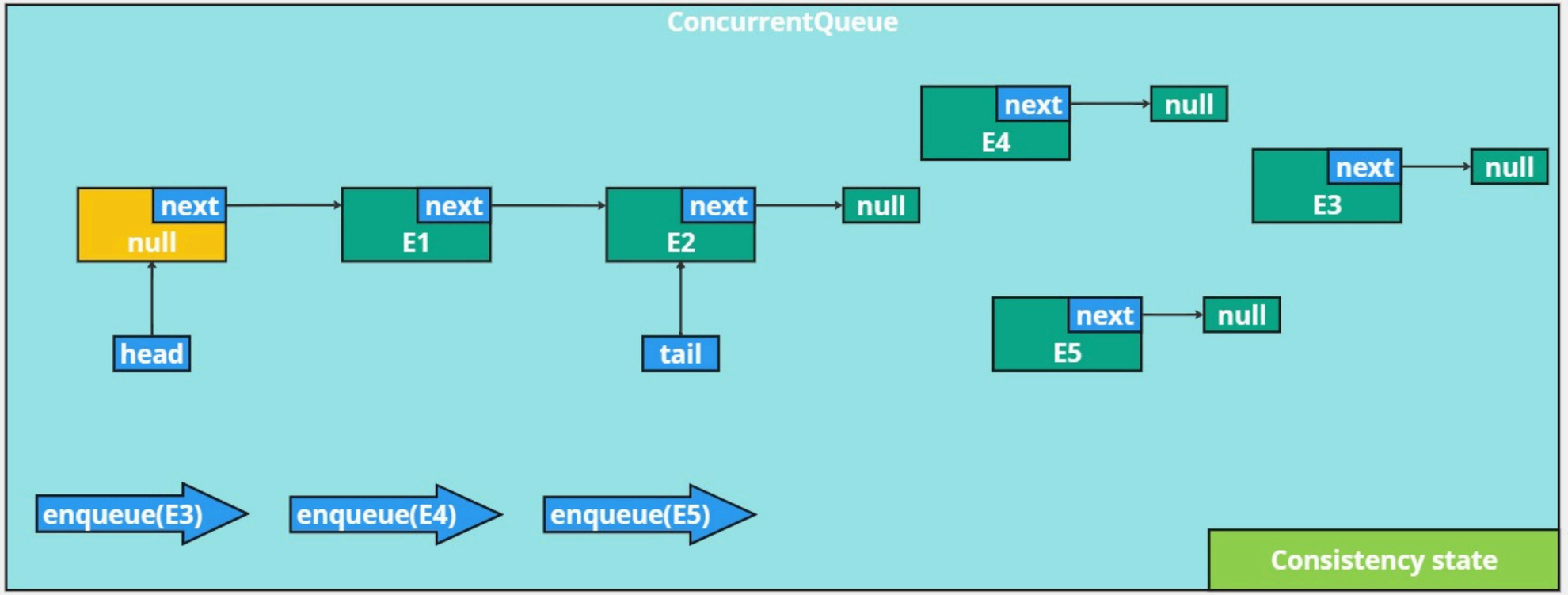
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

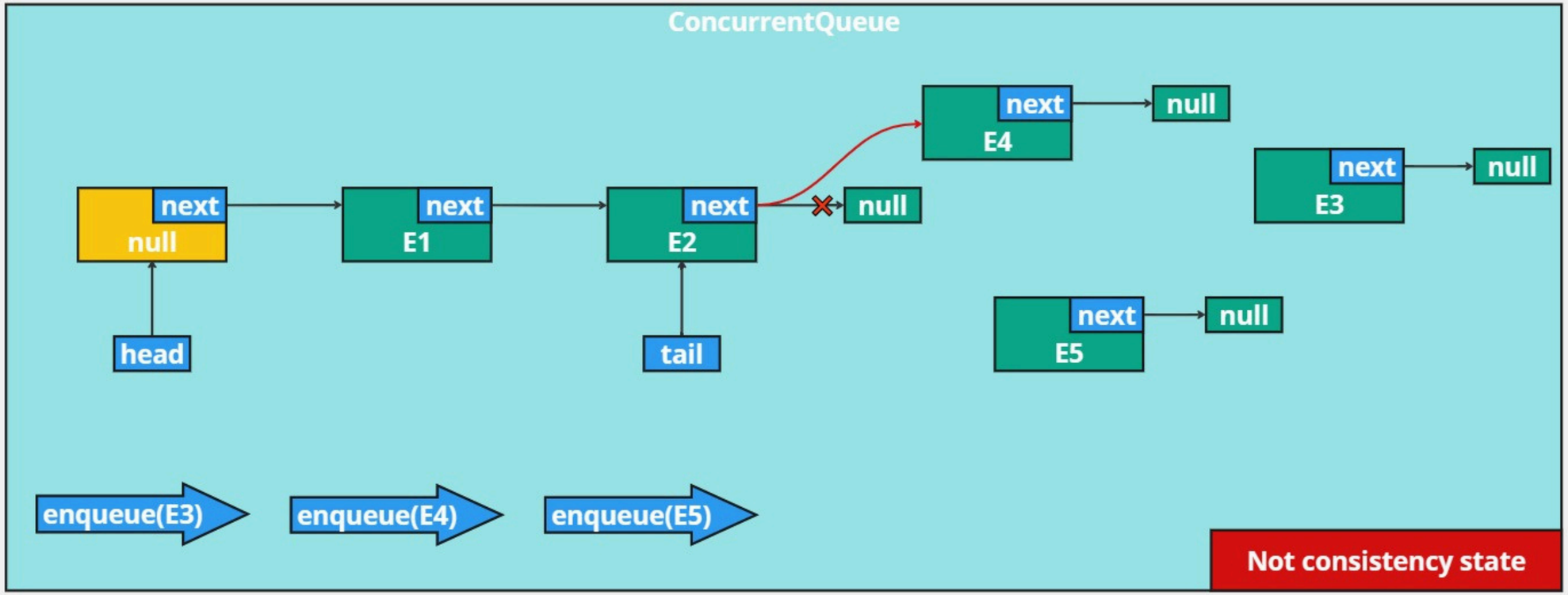
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

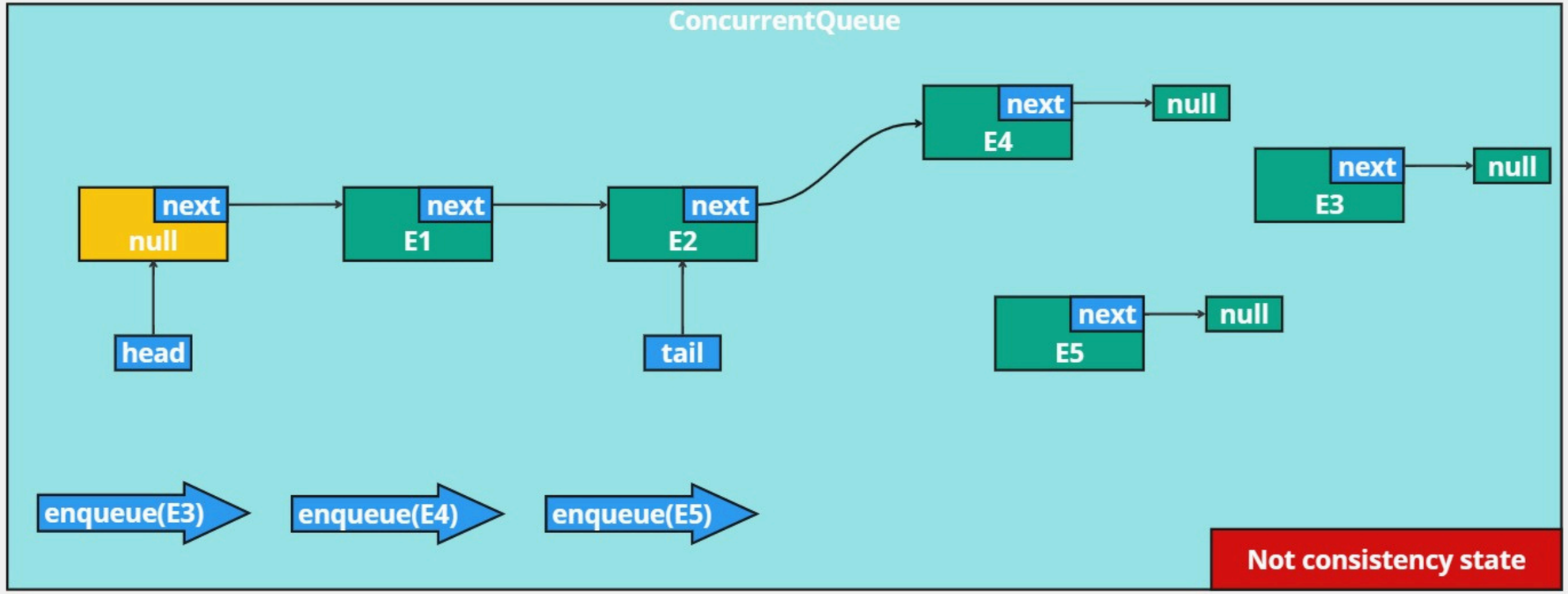
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

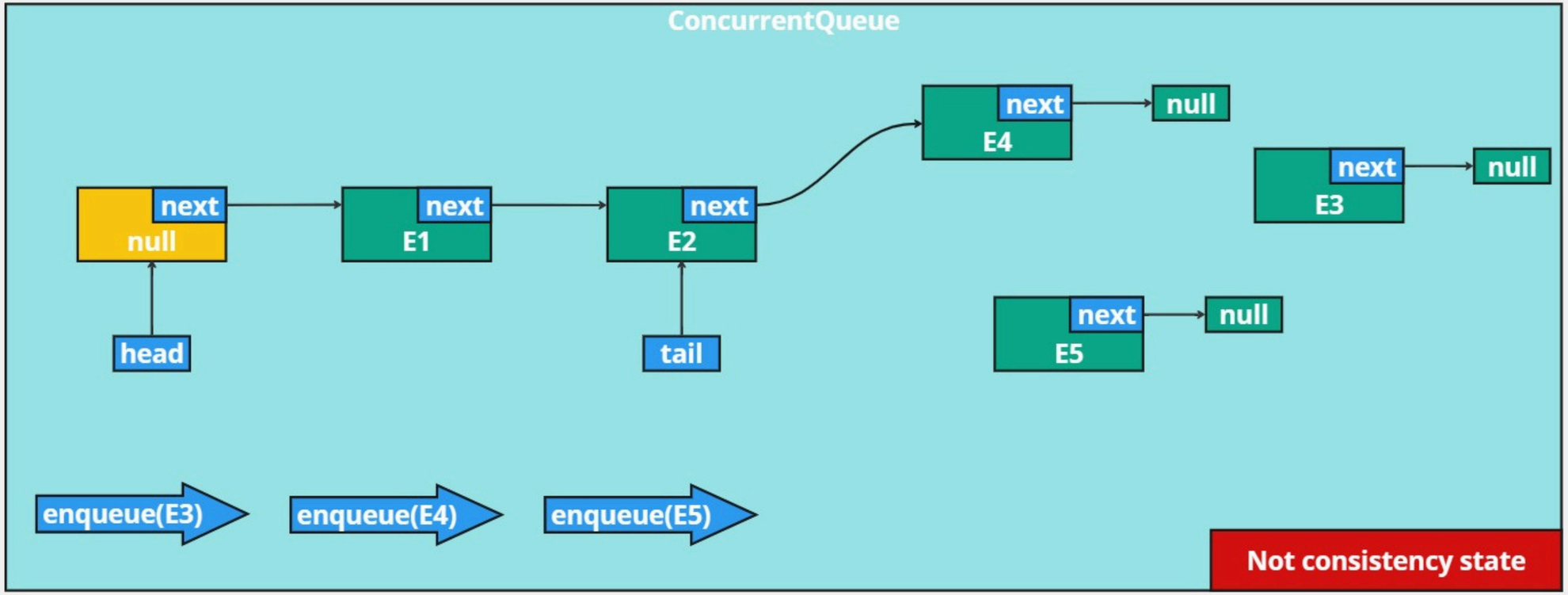
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

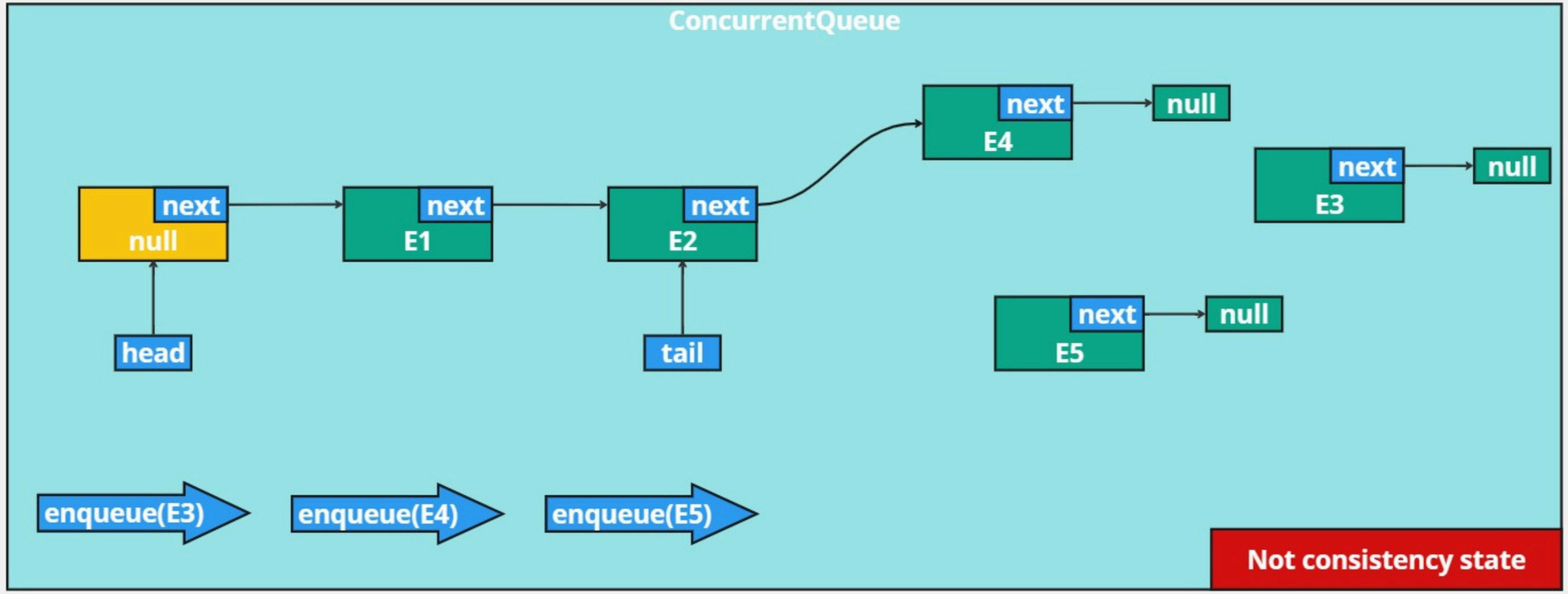
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

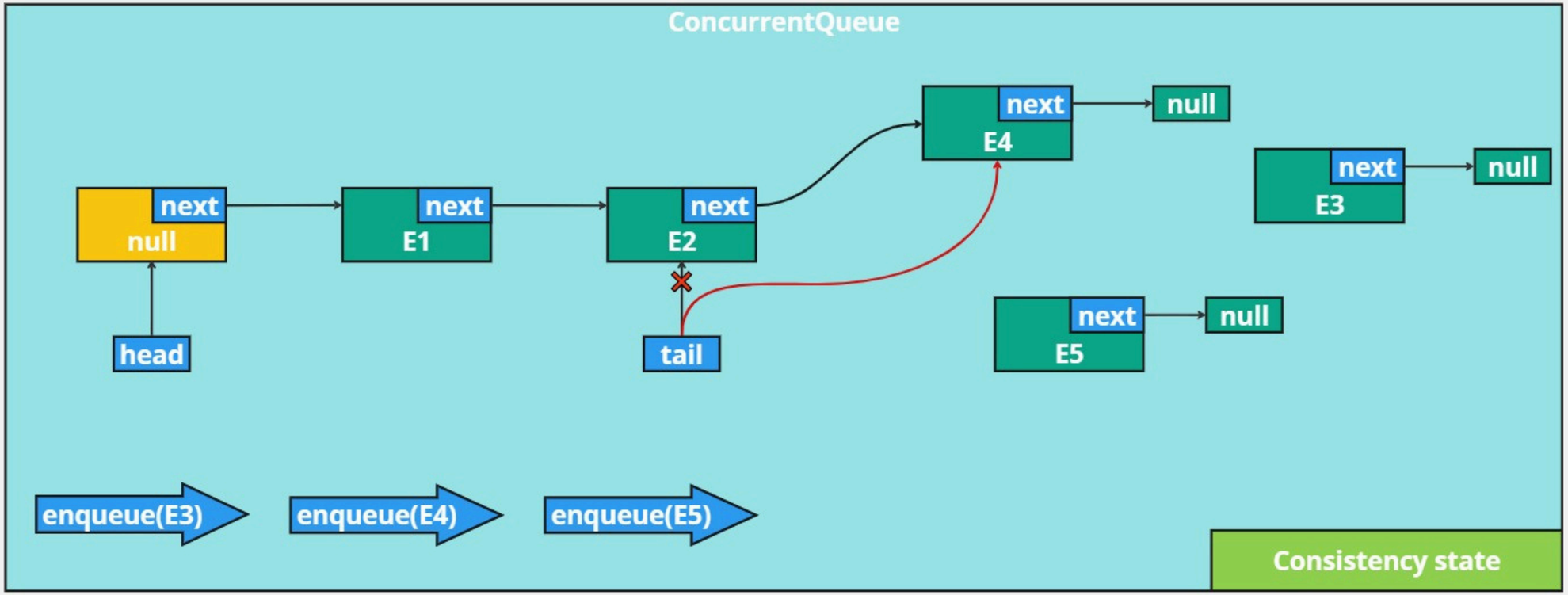
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

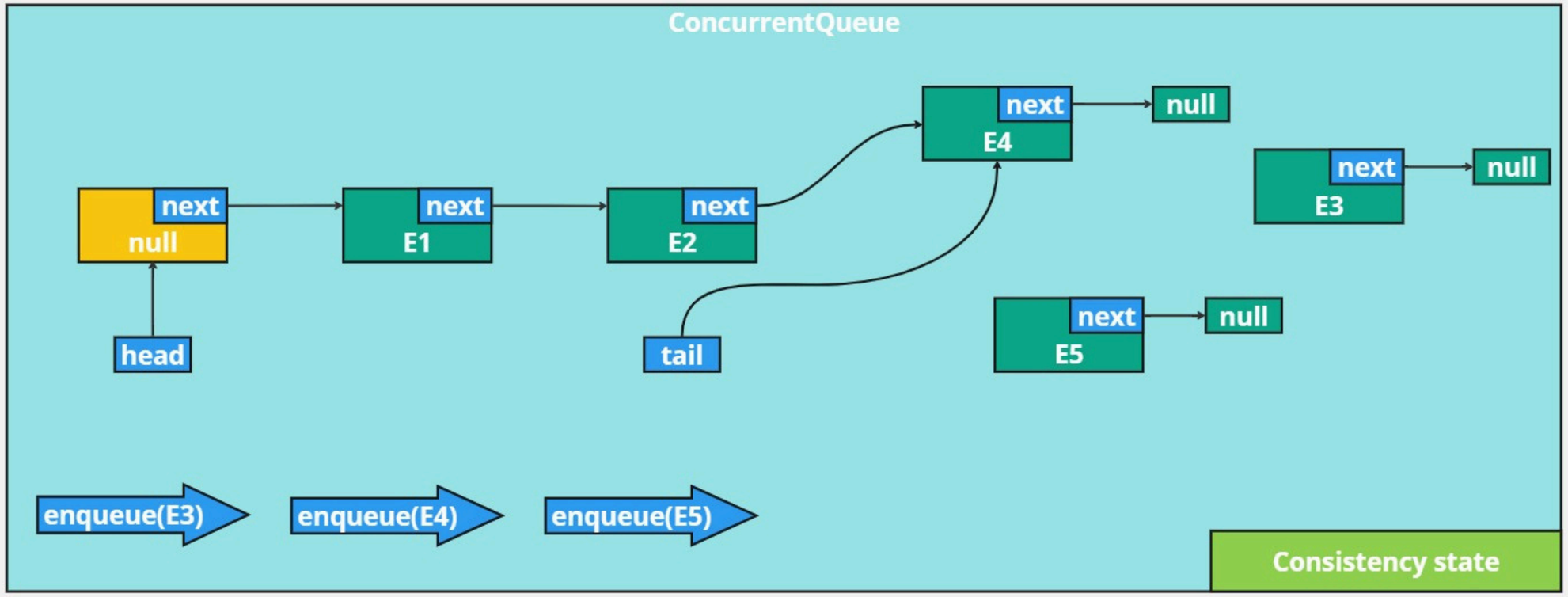
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

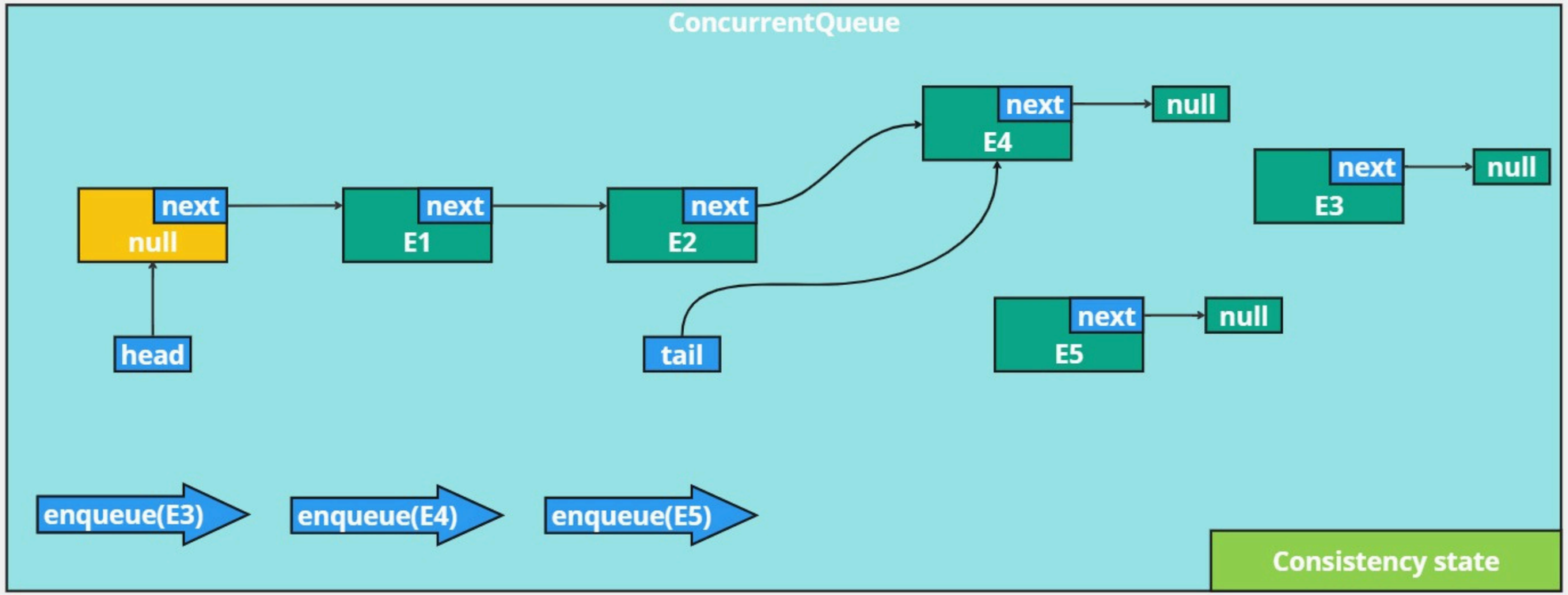
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

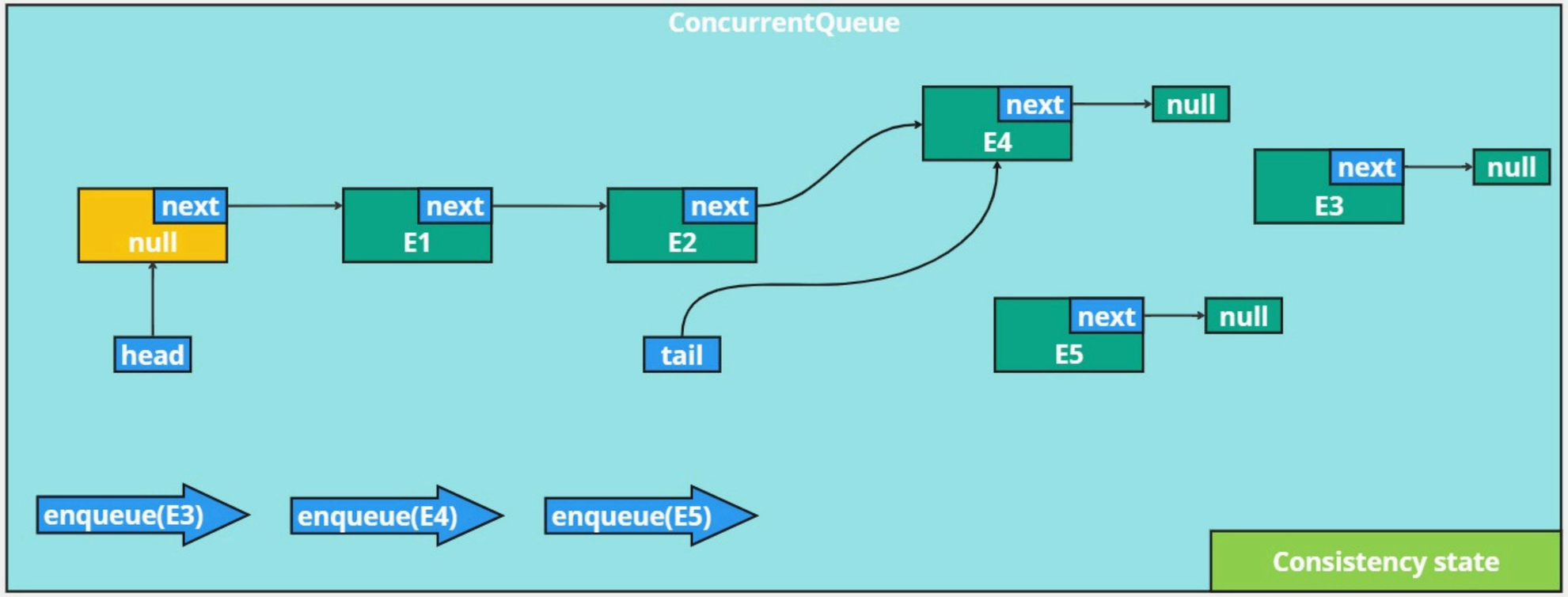
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

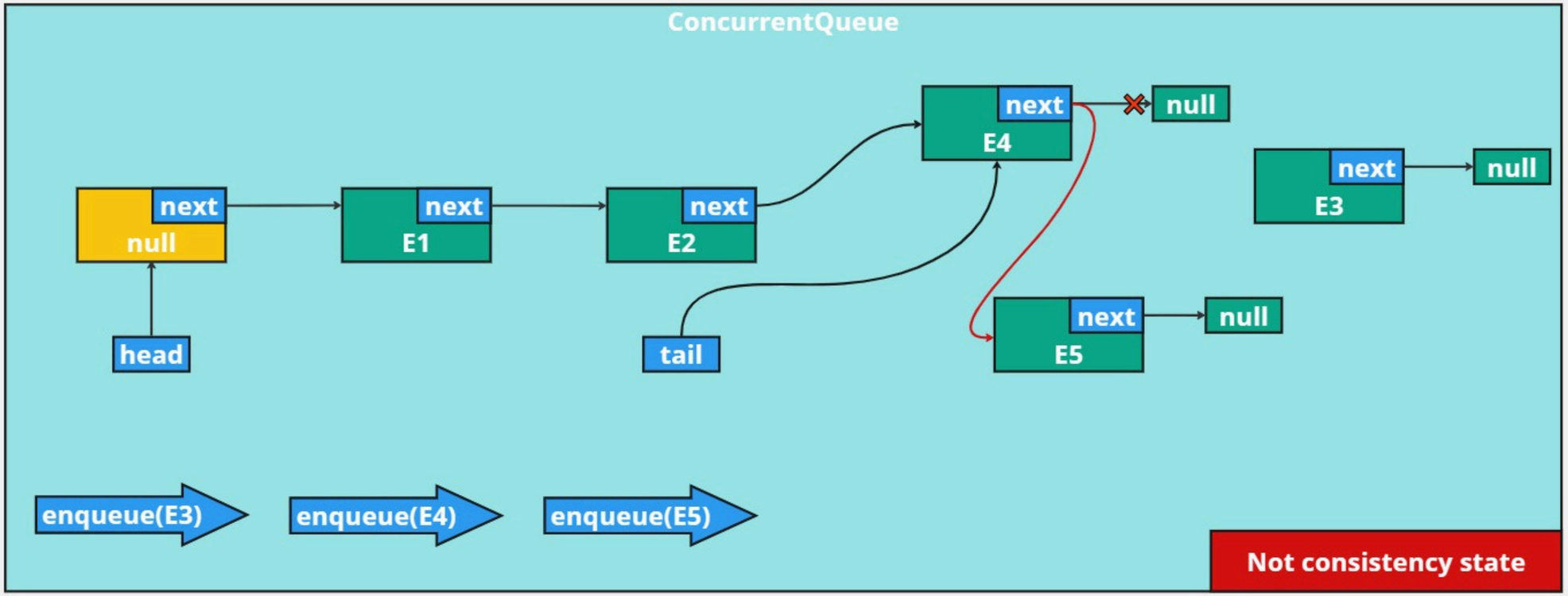
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

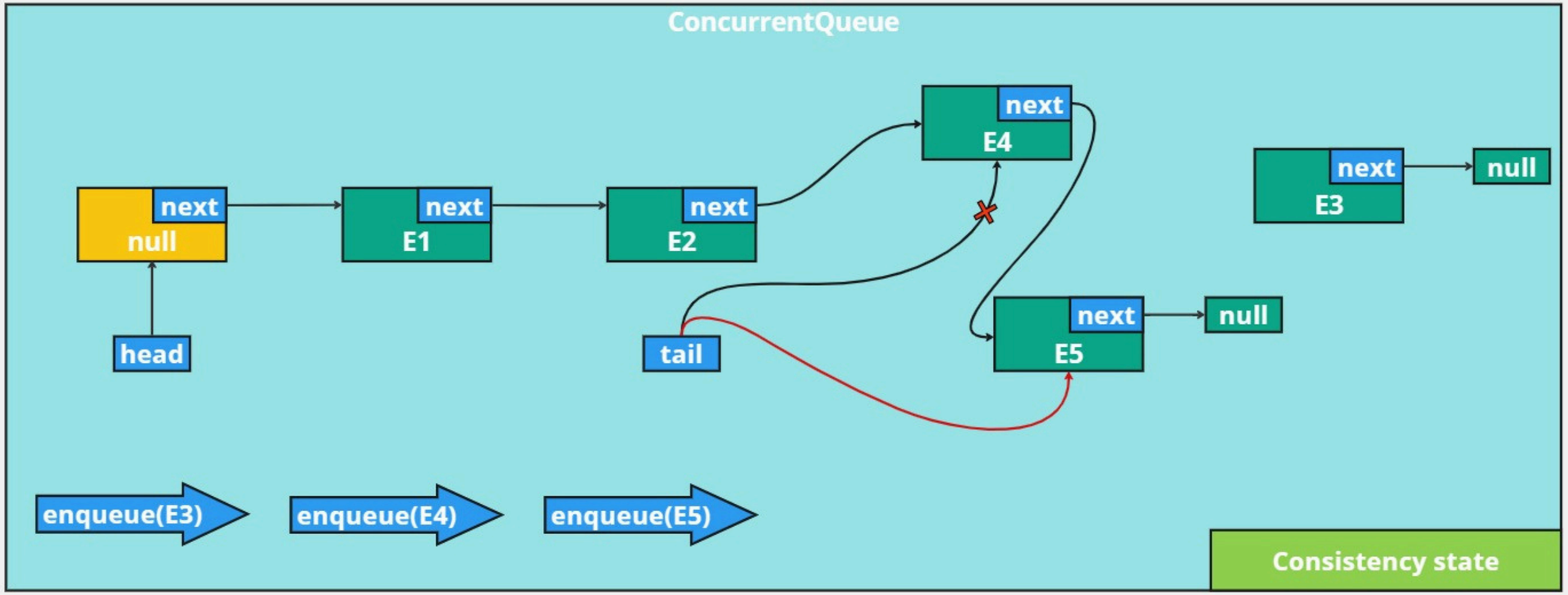
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

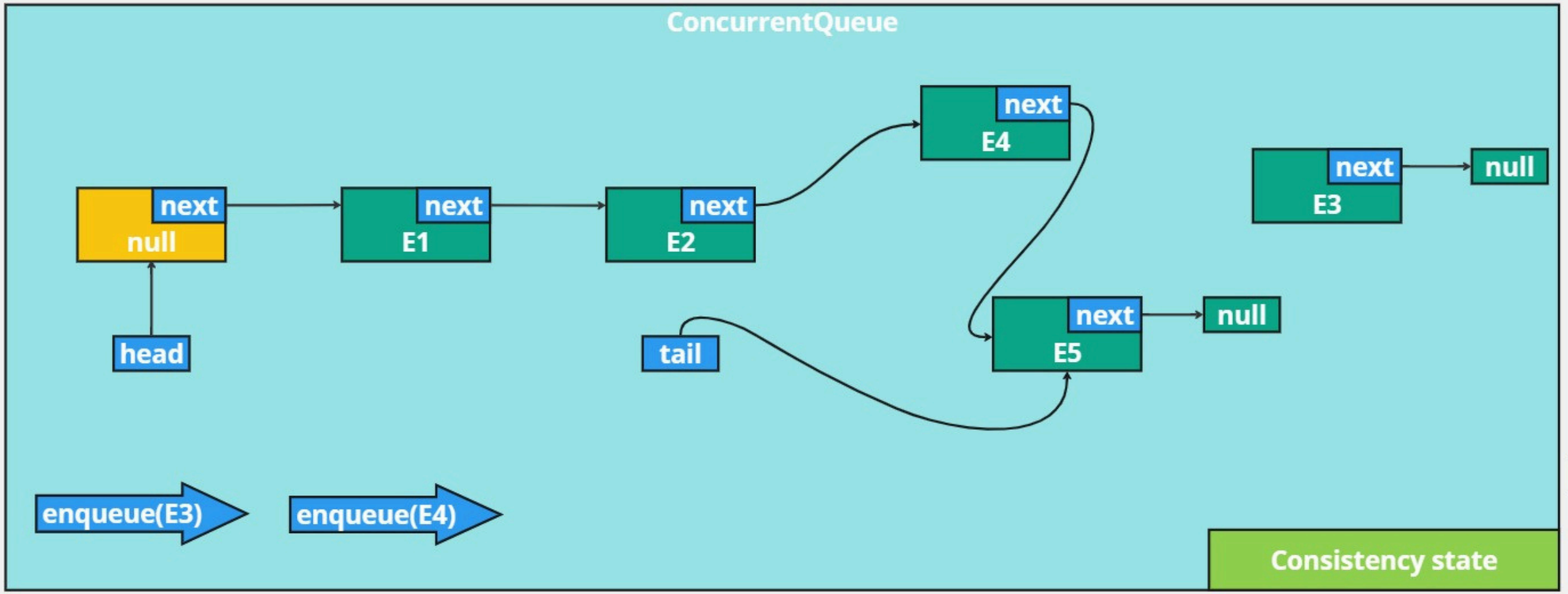
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

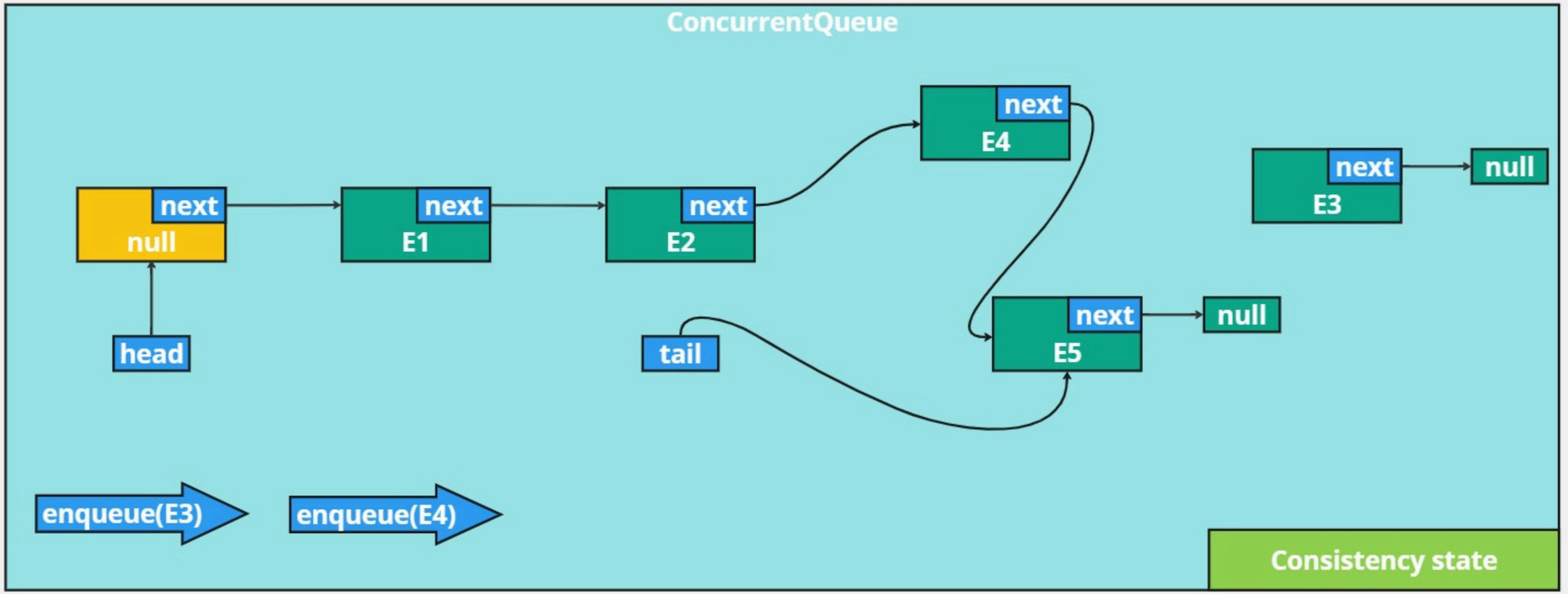
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

next E4

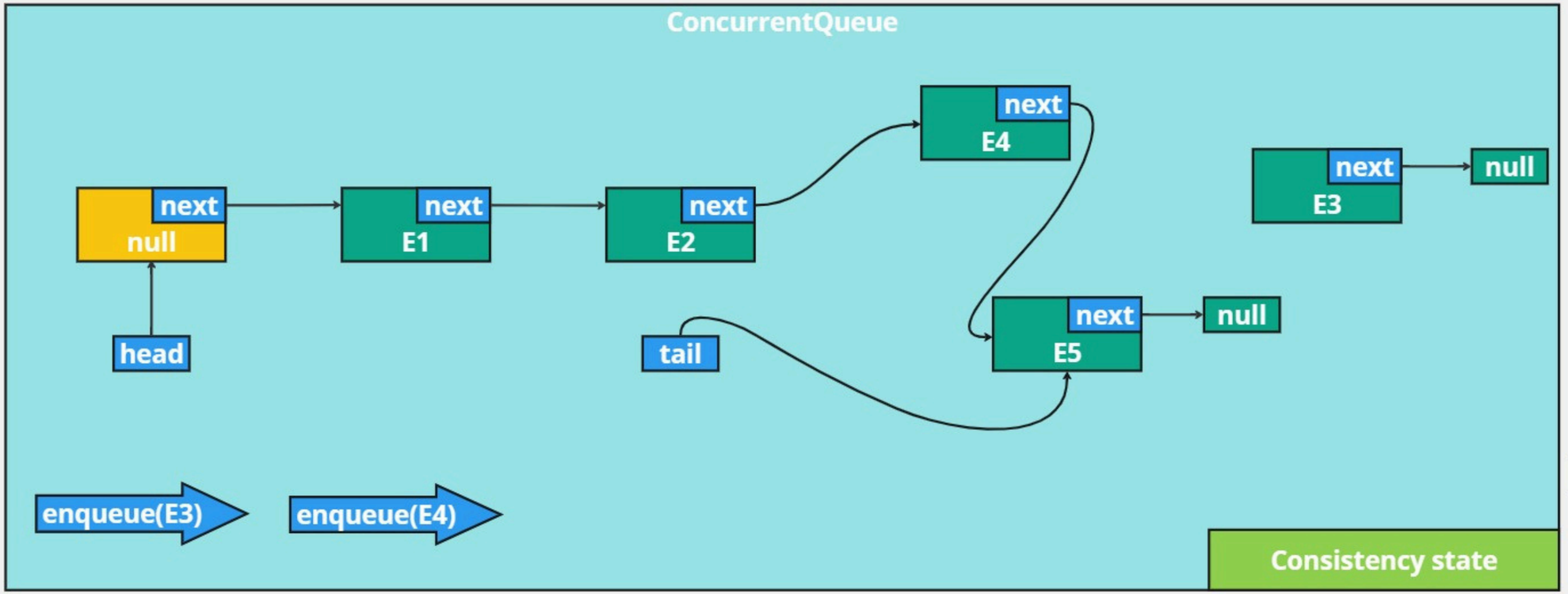
enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

next E2

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

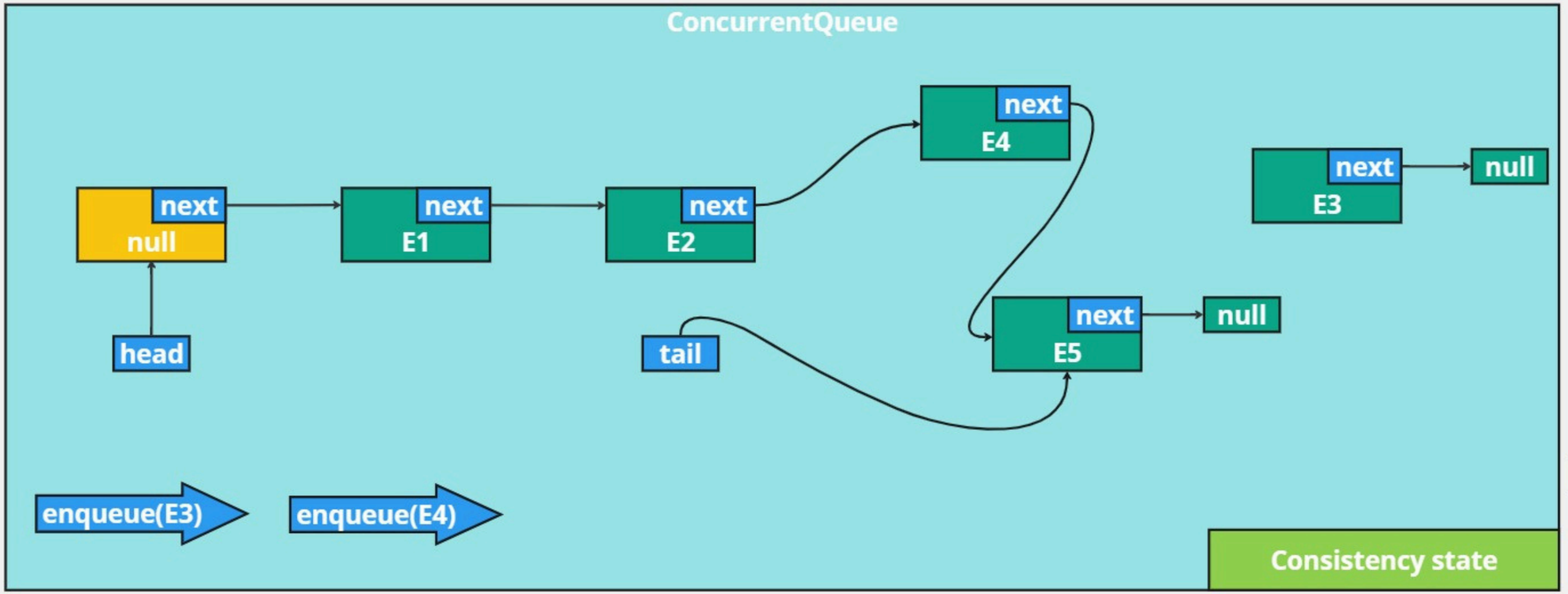
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

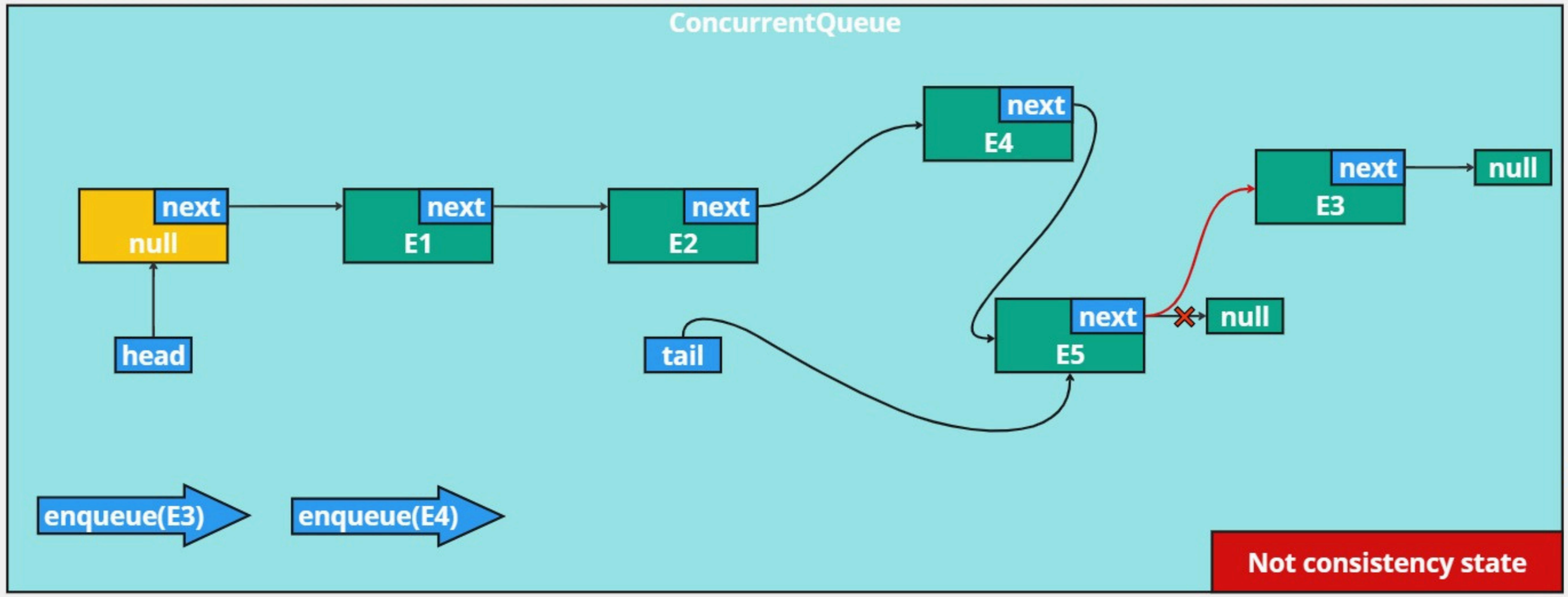
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

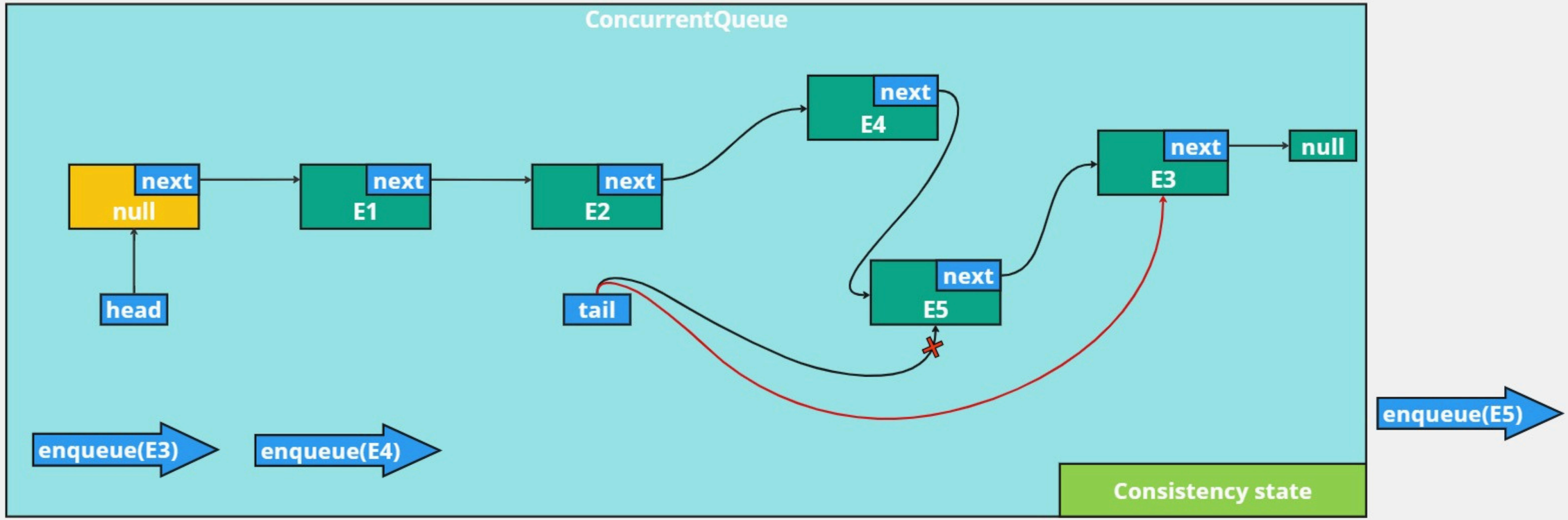
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

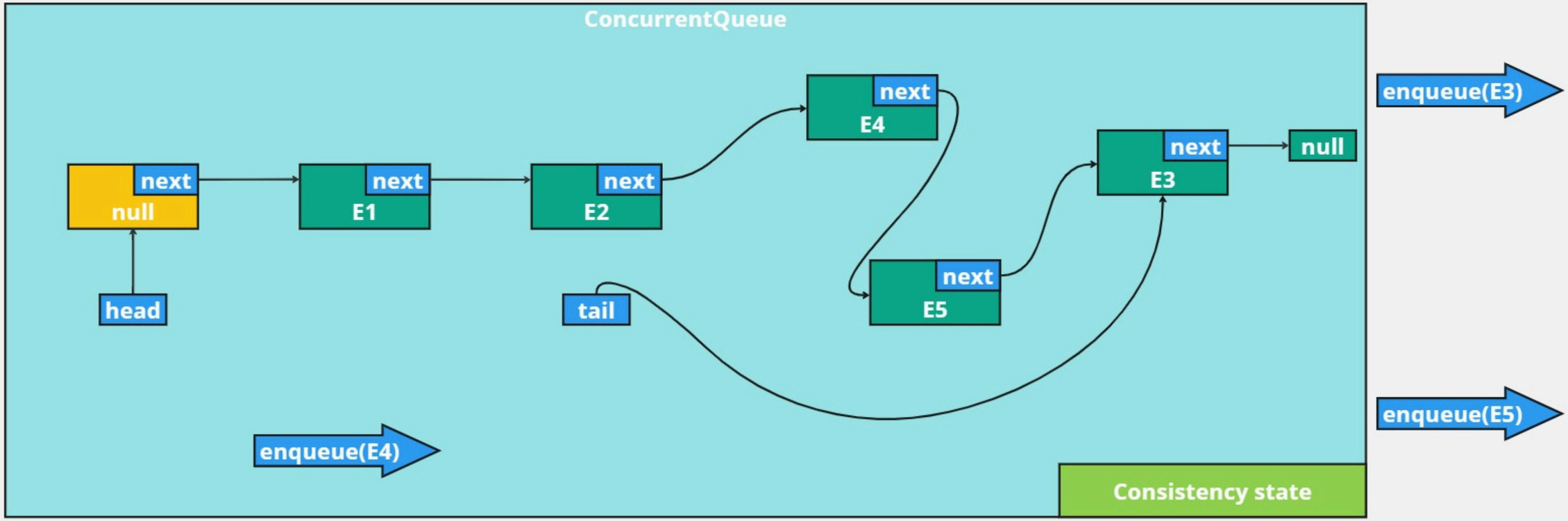
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

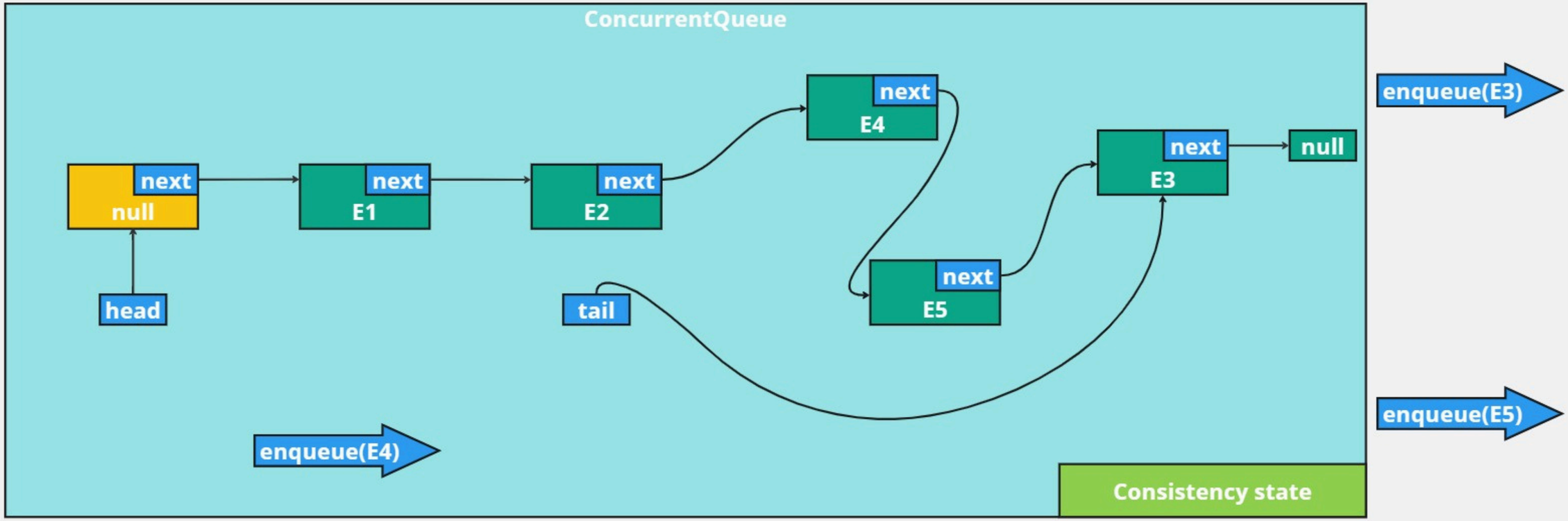
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

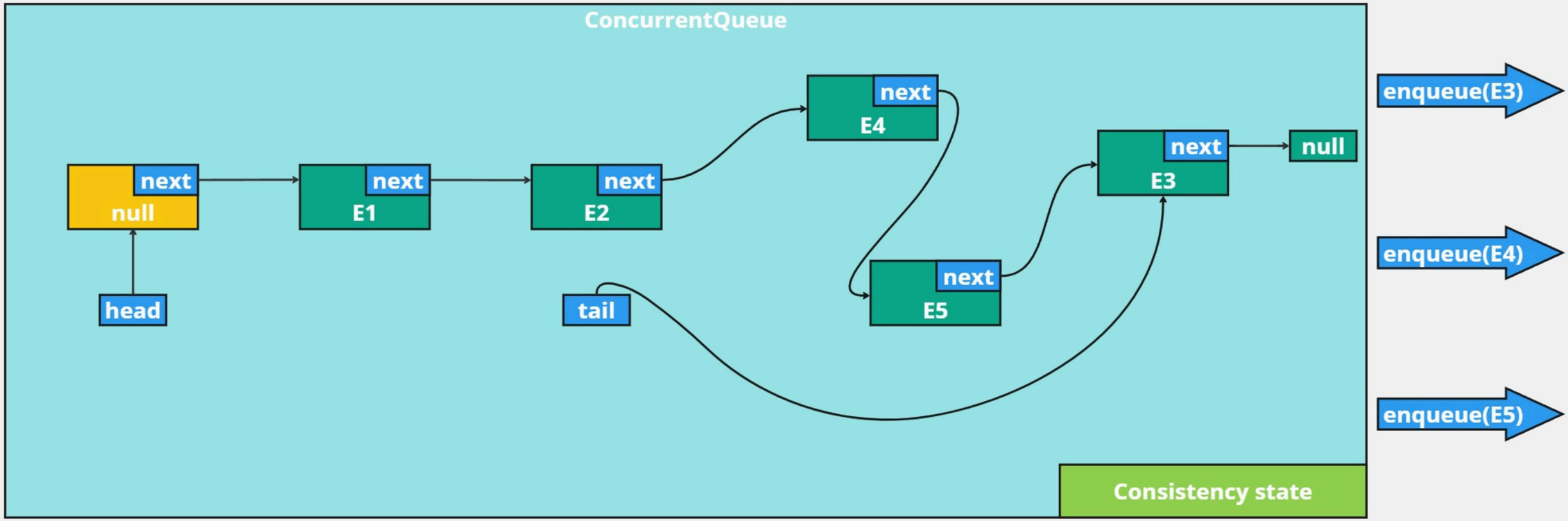
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E4)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

enqueue(E5)

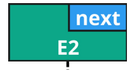
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



enqueue(E3)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

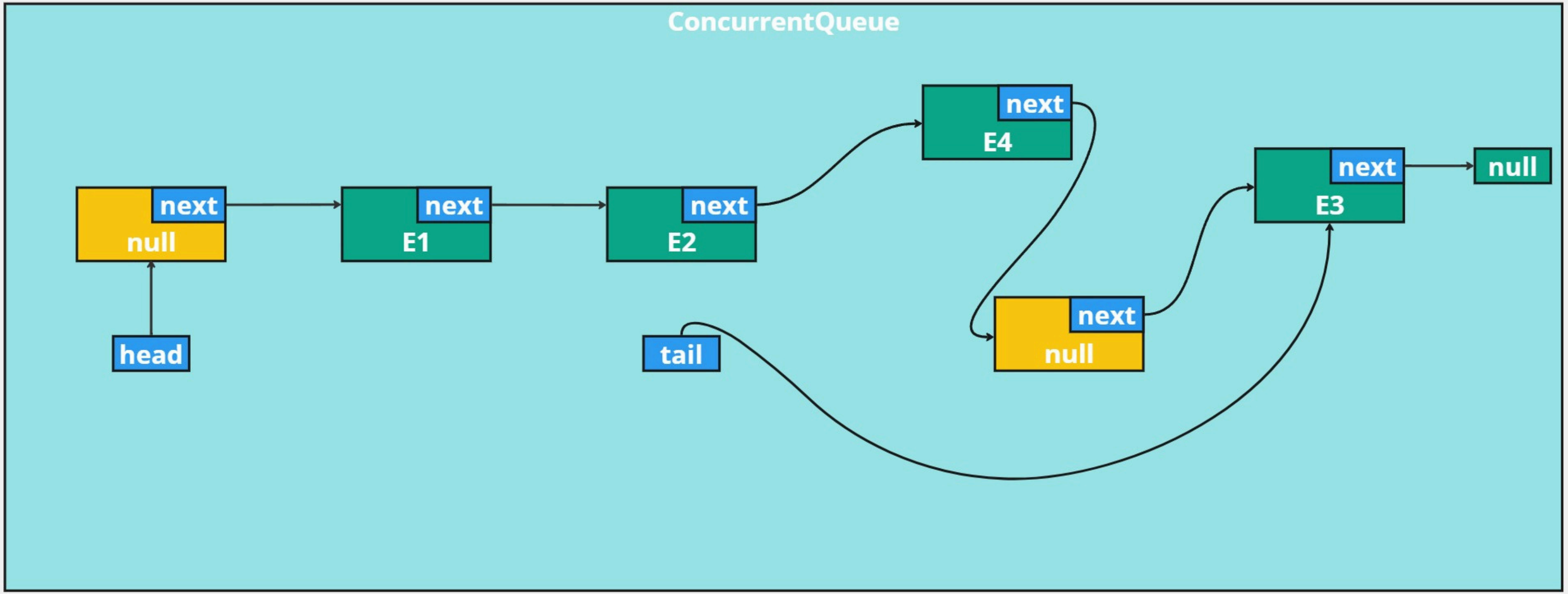
enqueue(E4)




```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

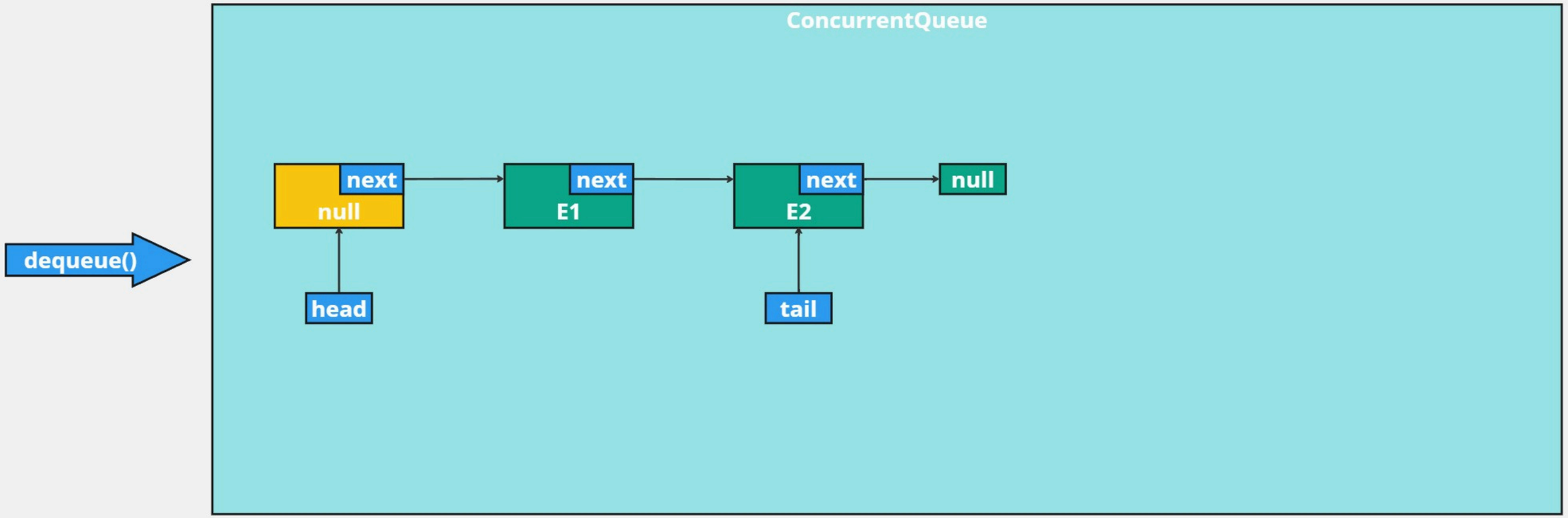
enqueue(E5)

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

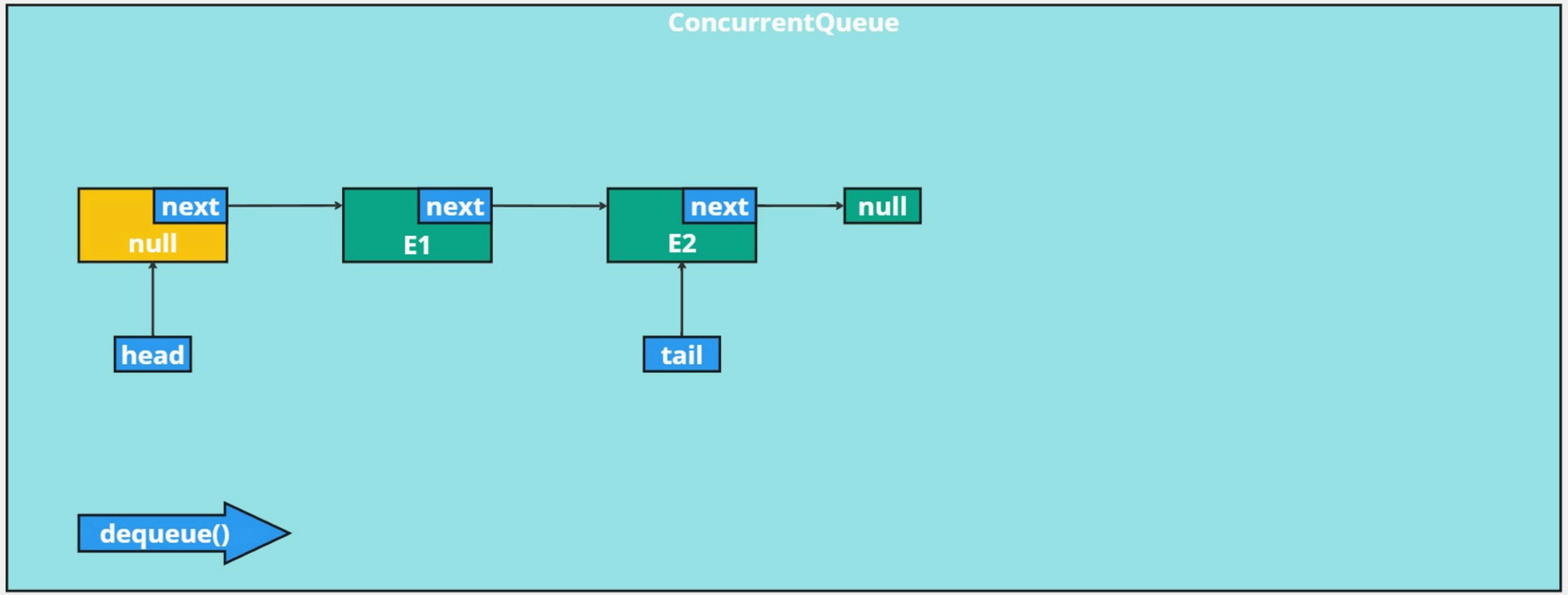

ConcurrentQueue
next
null
next
E1
next
E2
next
E4
next
null
next
E3
null
head
tail

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

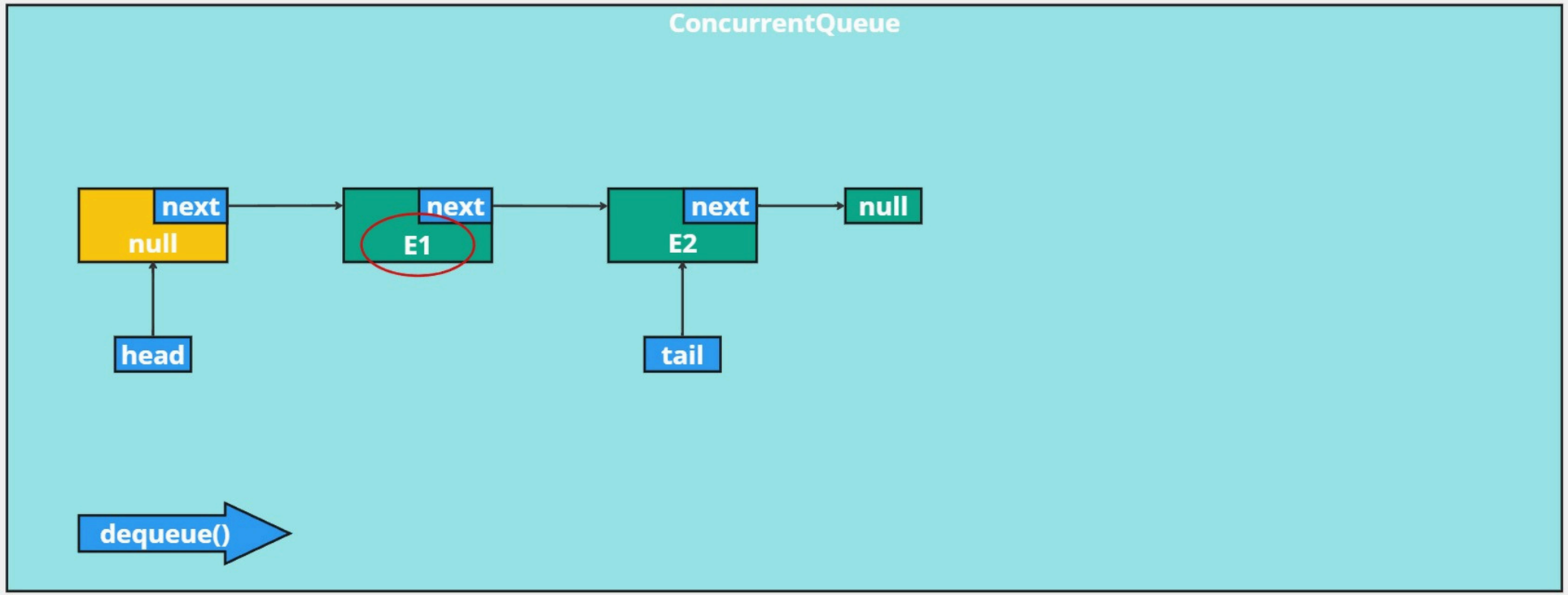
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(requireNonNull(element));
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



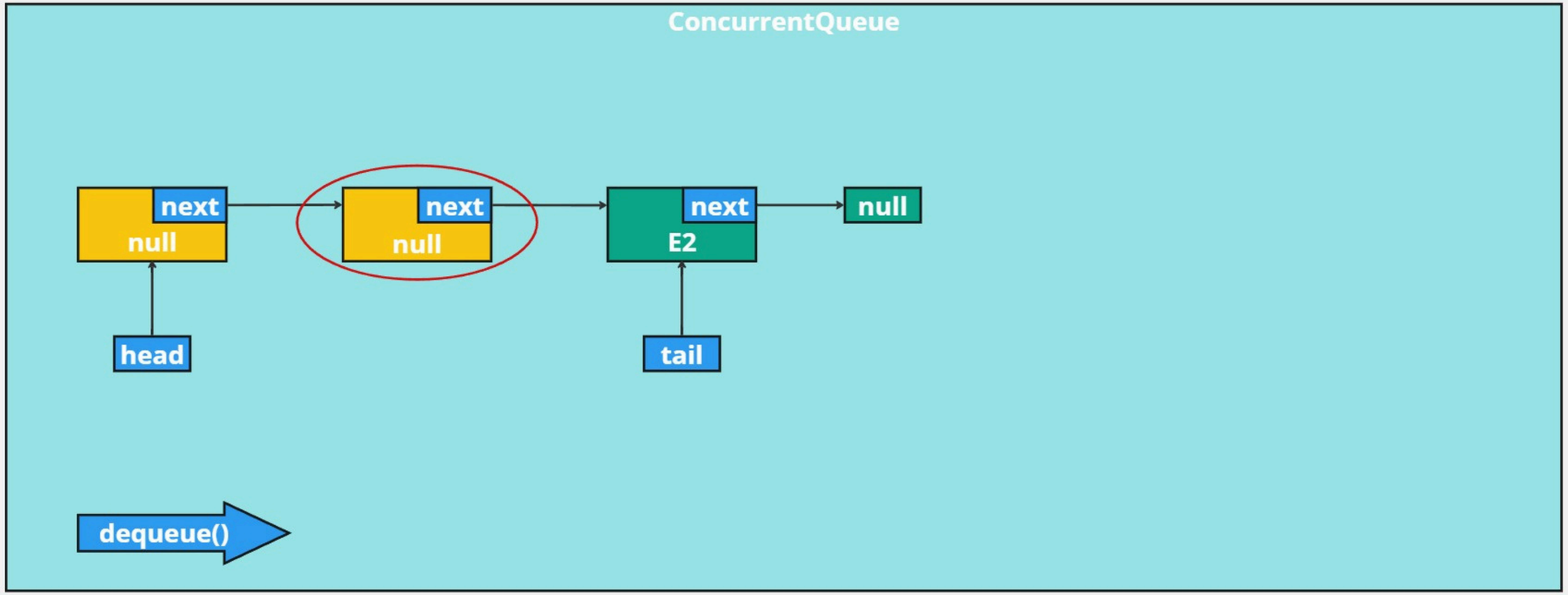

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



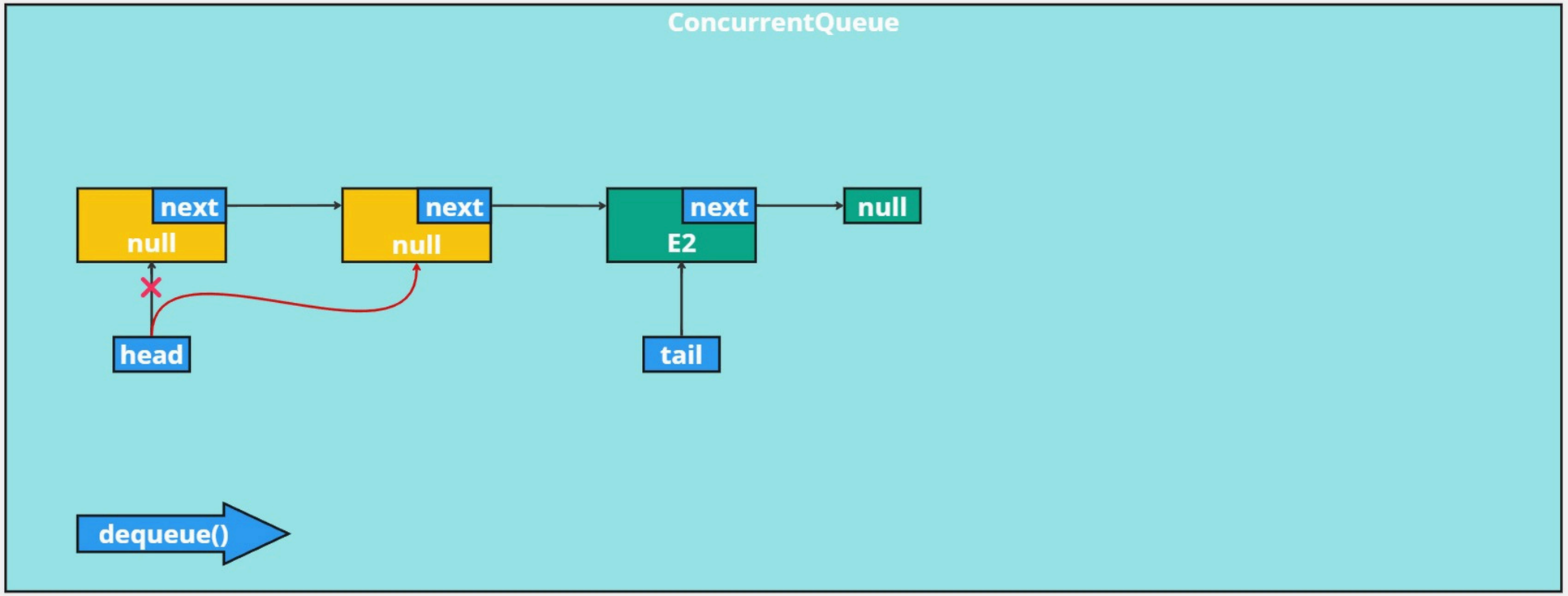

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



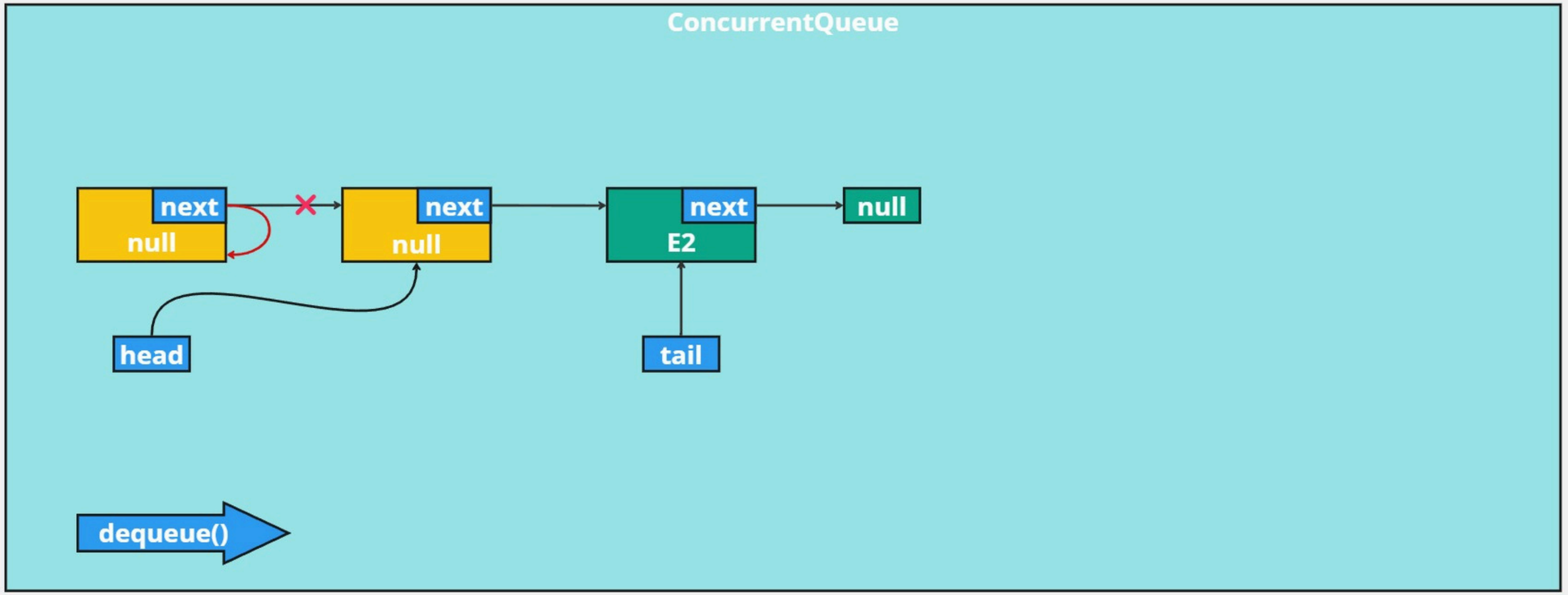

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



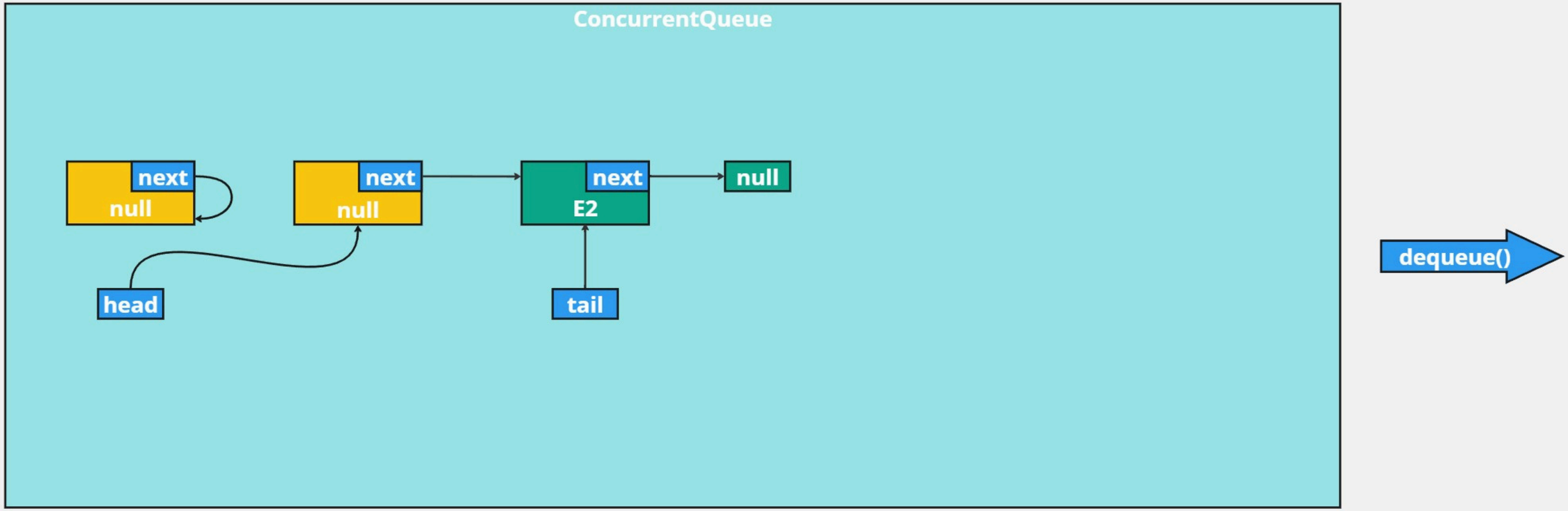

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



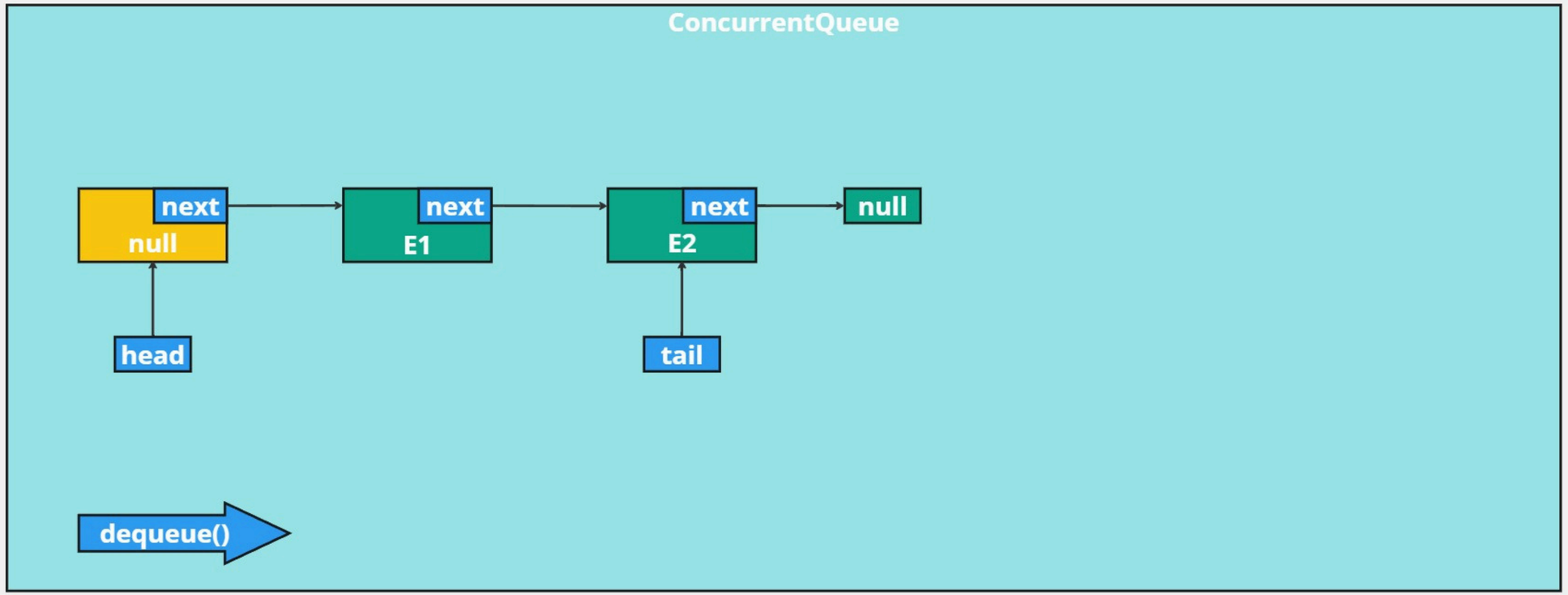

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



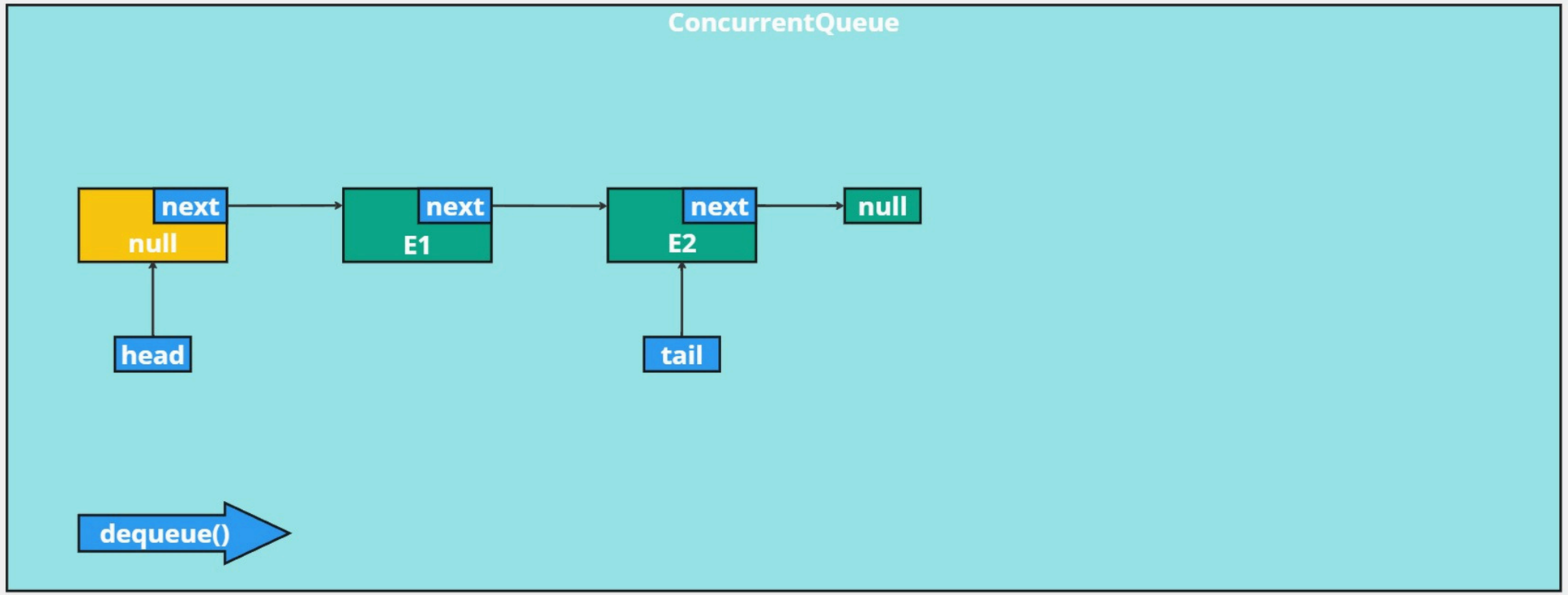

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



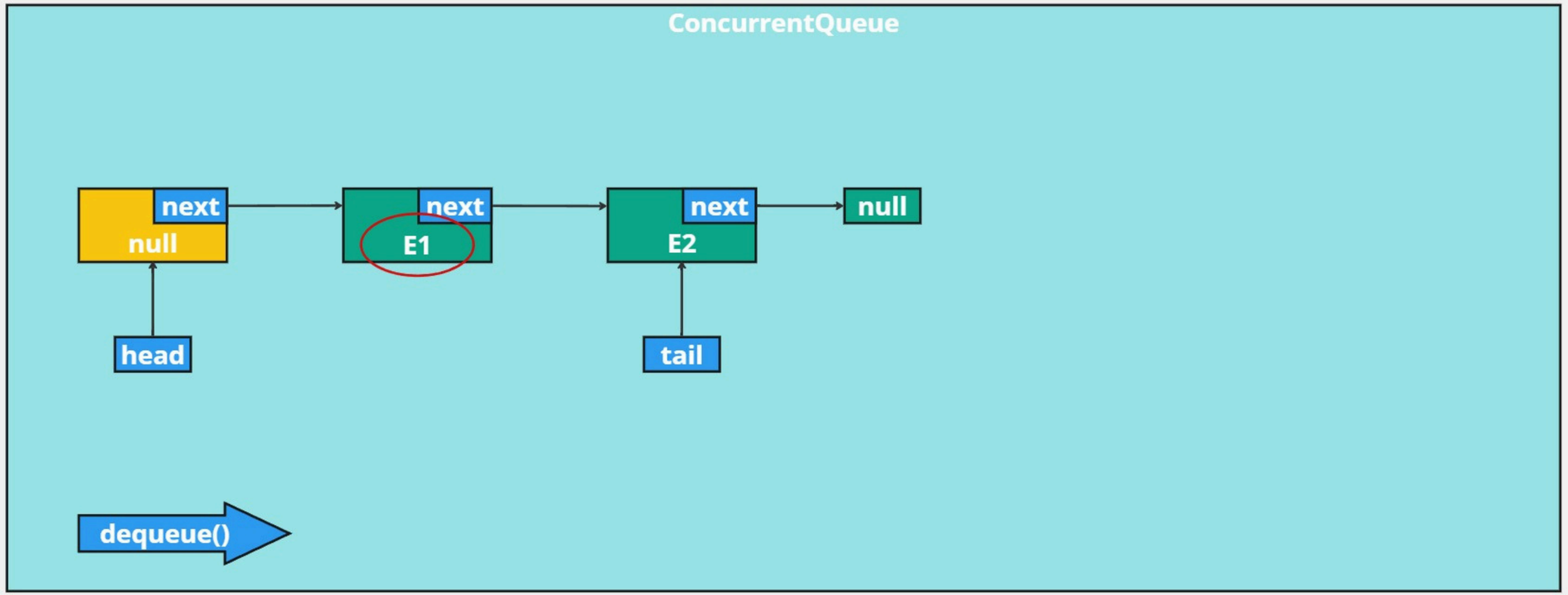

```
public Optional<E> dequeue() {
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



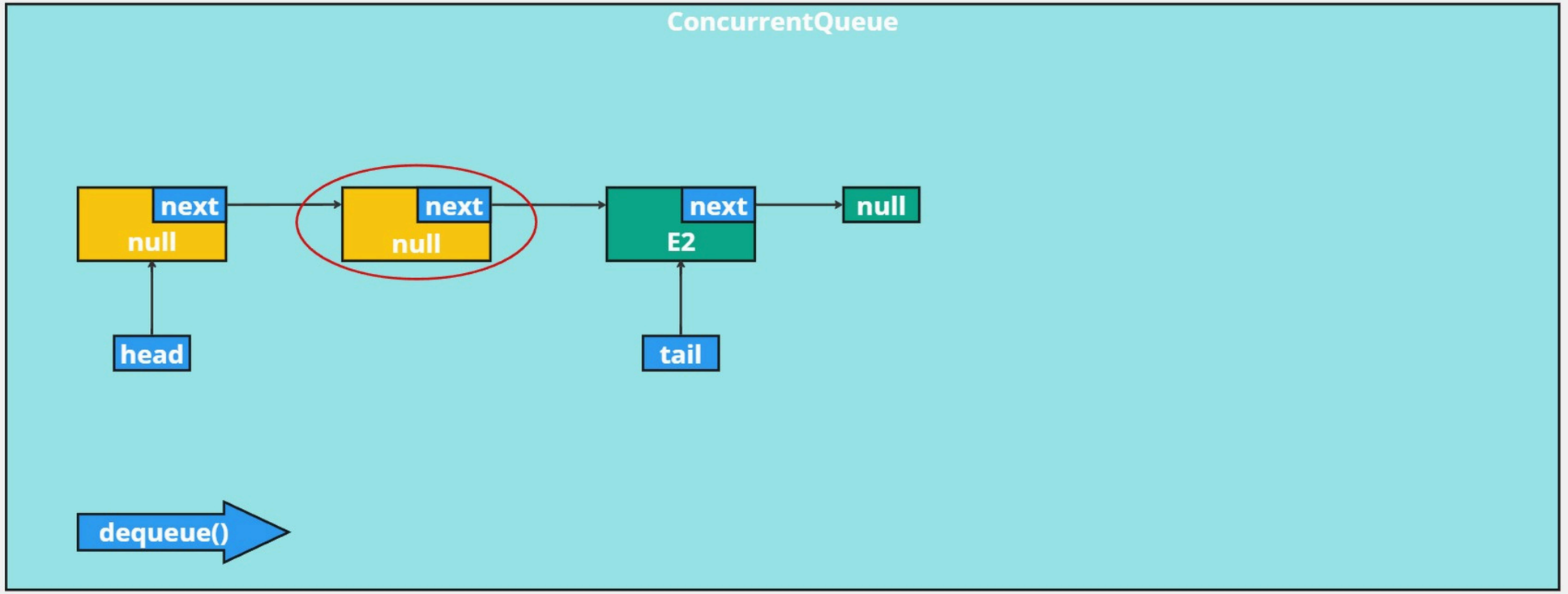

```
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    //check if queue is empty
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



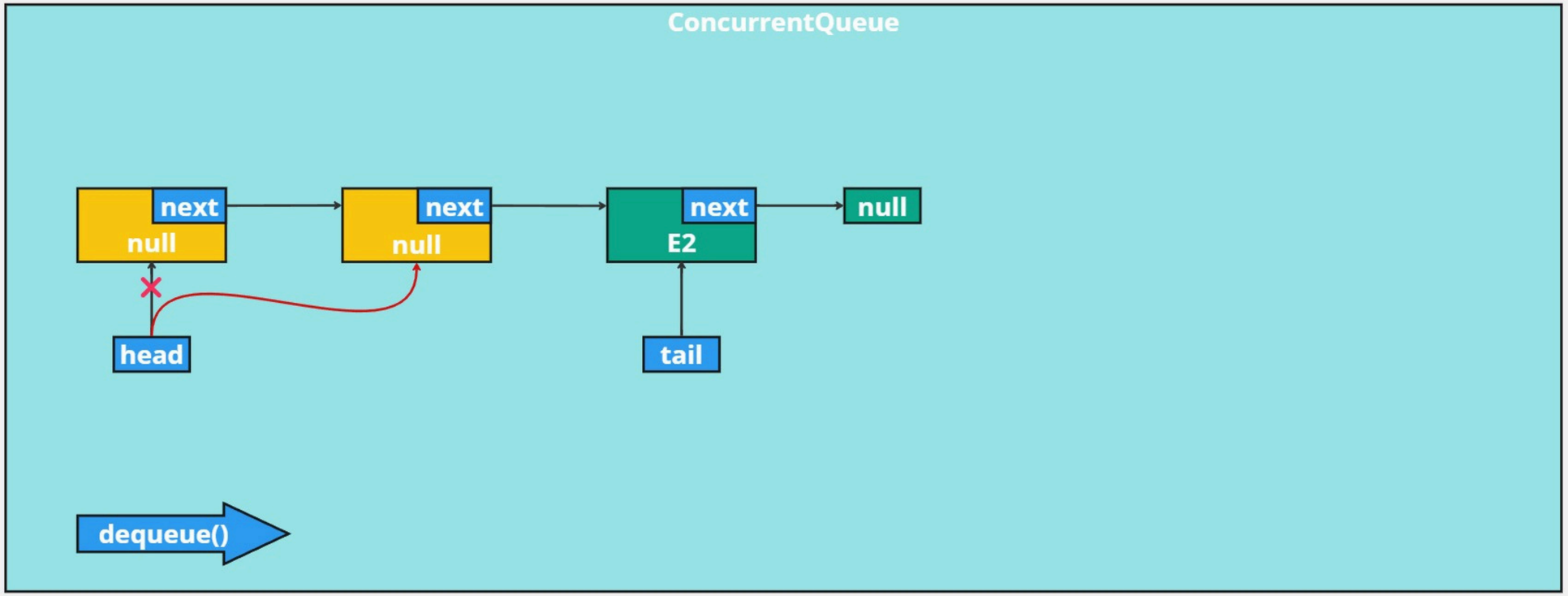

```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    //extract first node's value
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



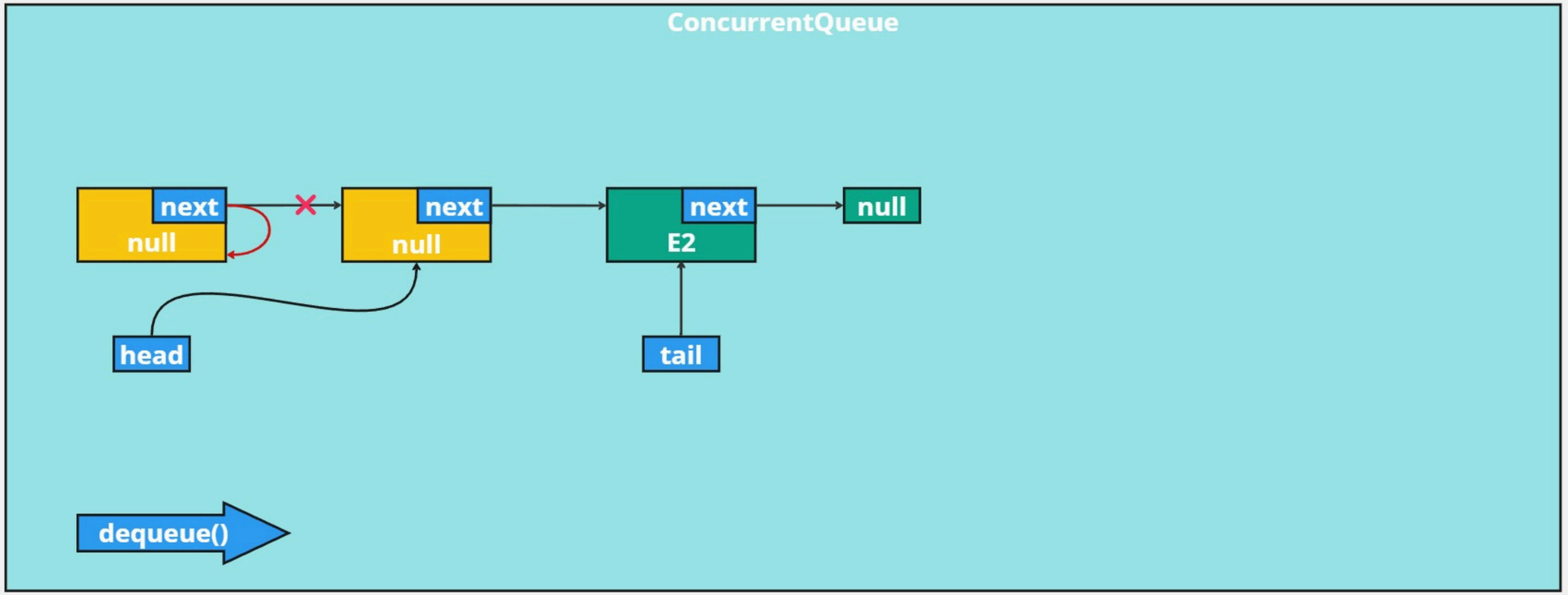

```
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    //first node's value should be assigned null
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



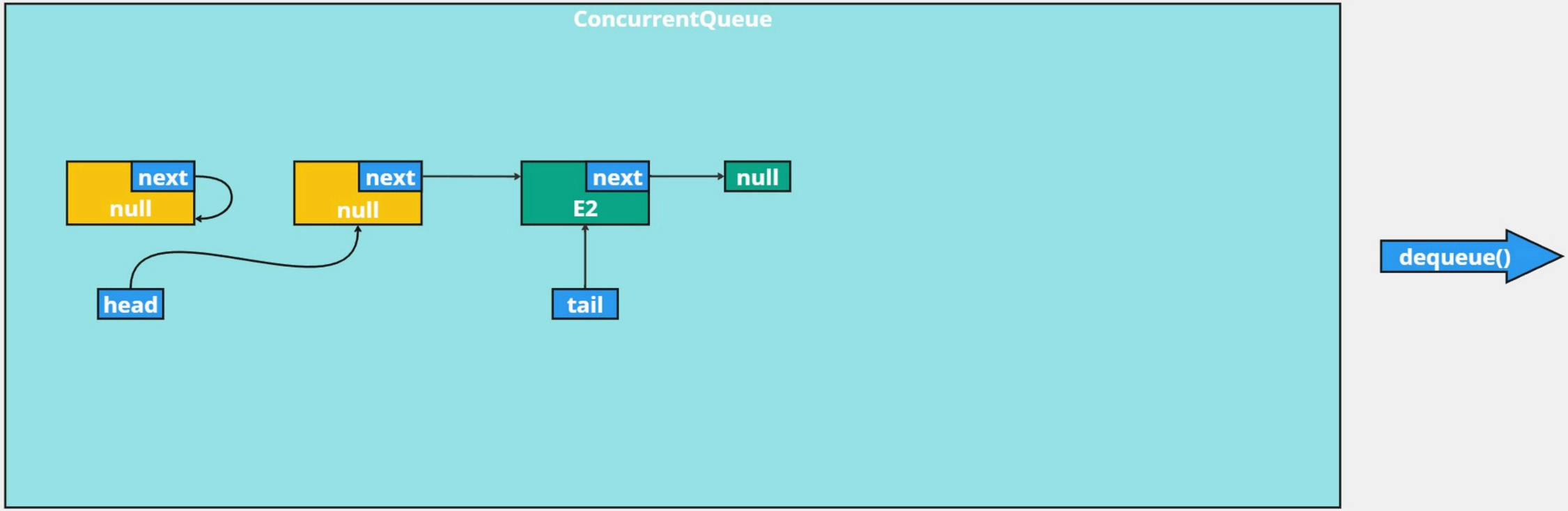

```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    //head should be assigned first node
    //previous head's next should be assigned previous head
    //return element
}
```



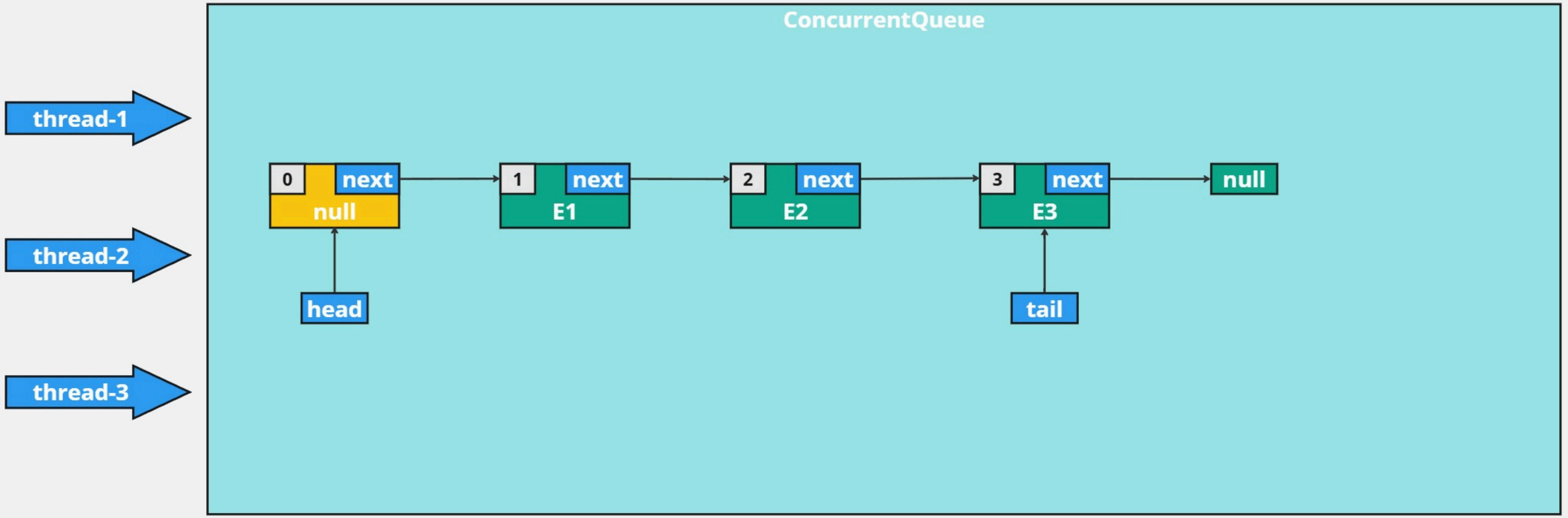

```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    //previous head's next should be assigned previous head
    //return element
}
```




```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    //return element
}
```




```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

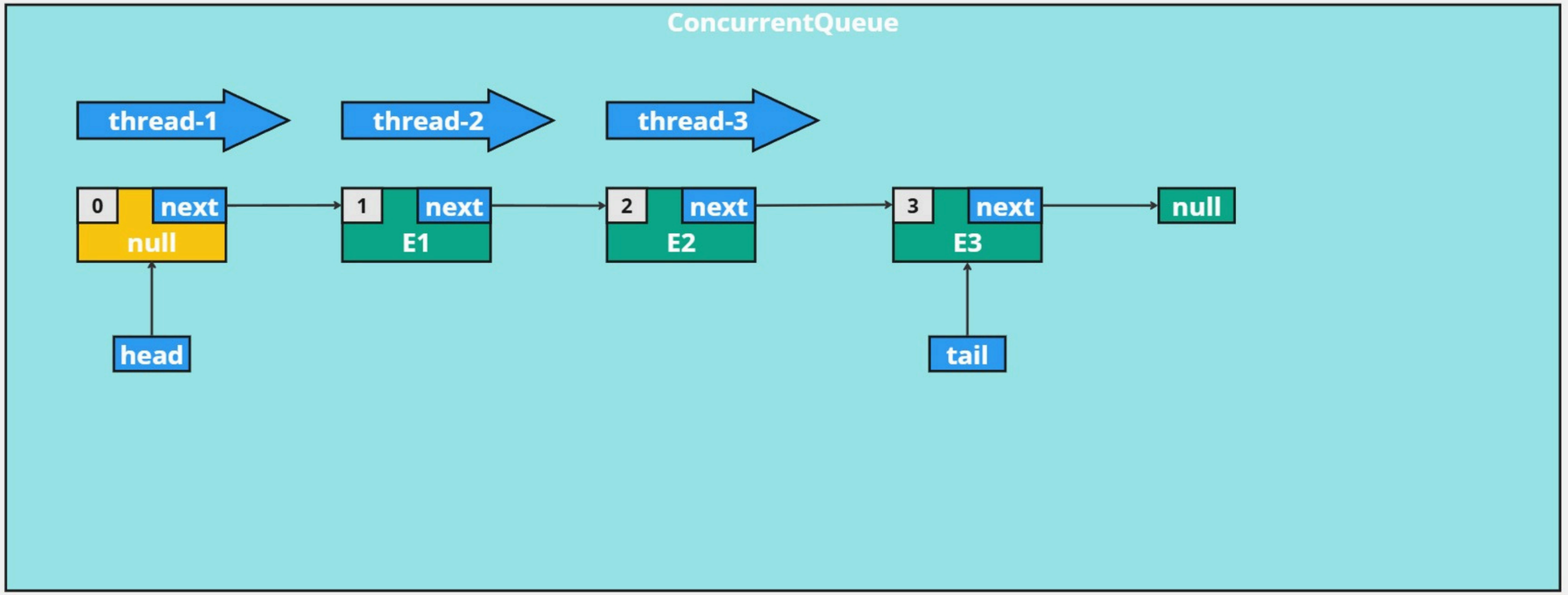
```
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

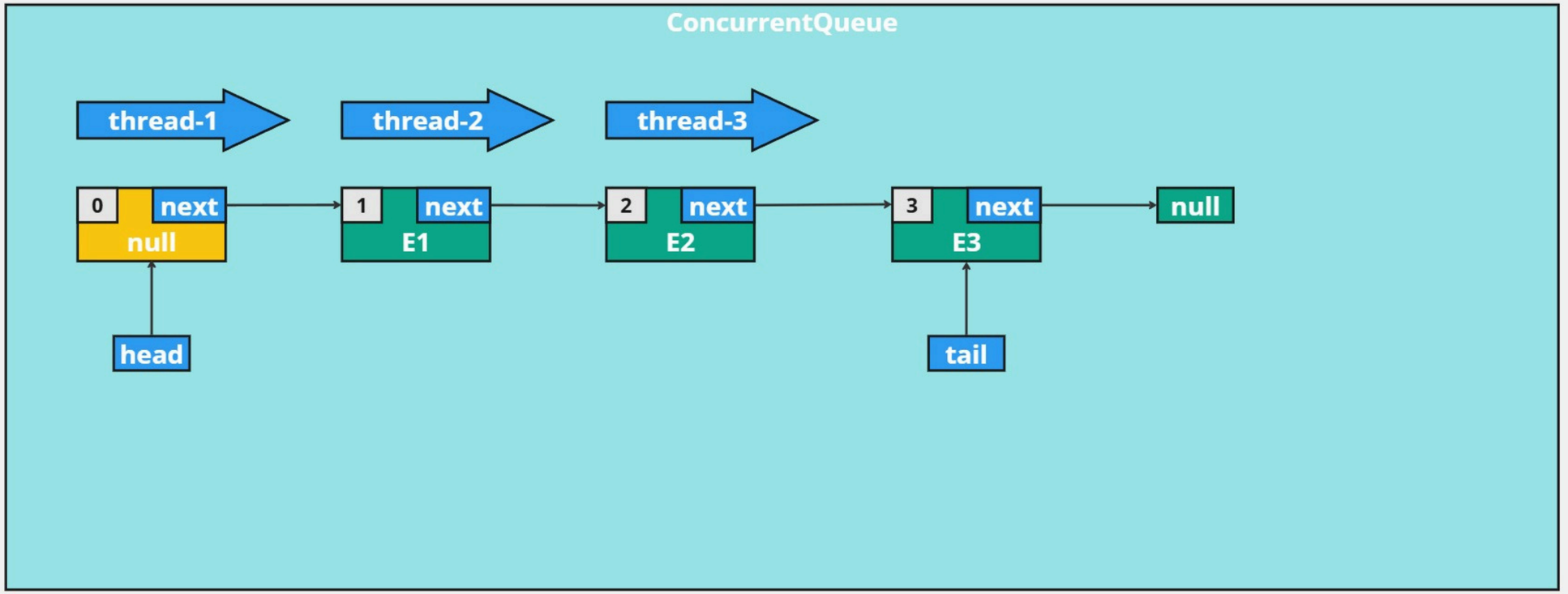
```
public Optional<E> dequeue() {                                    0
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                    0
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                    0
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

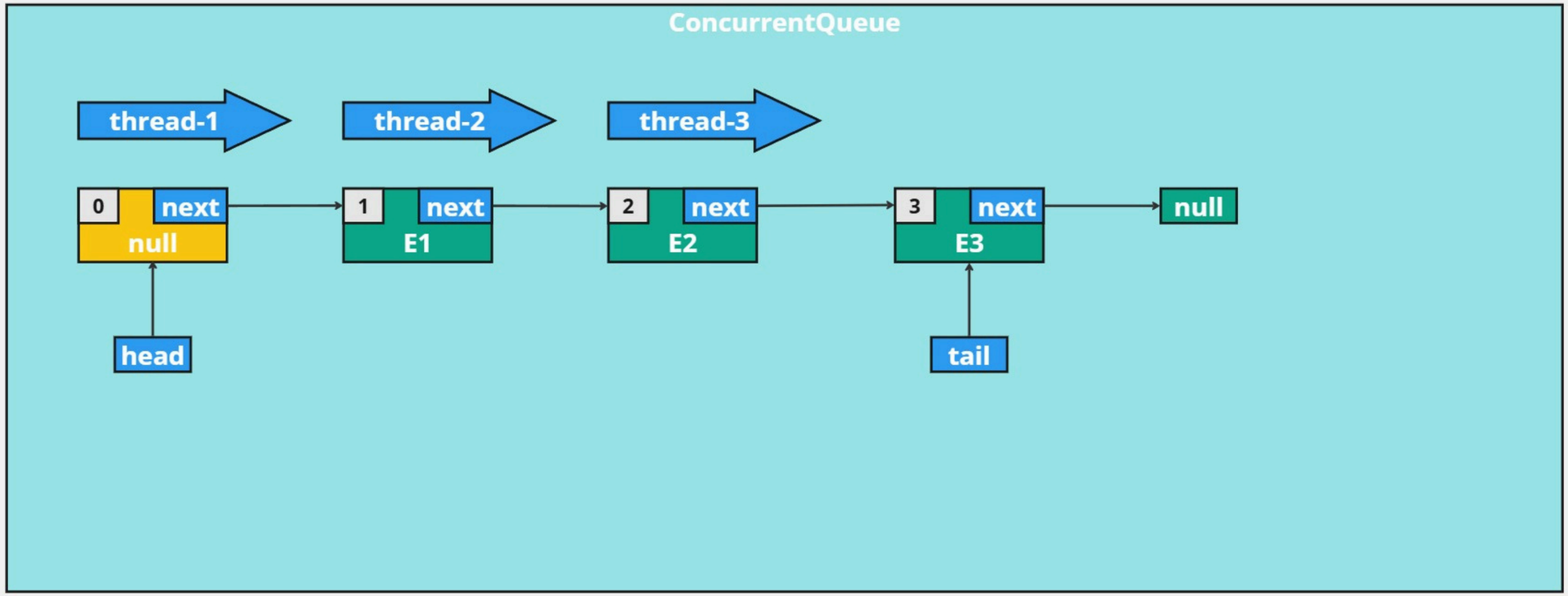
```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

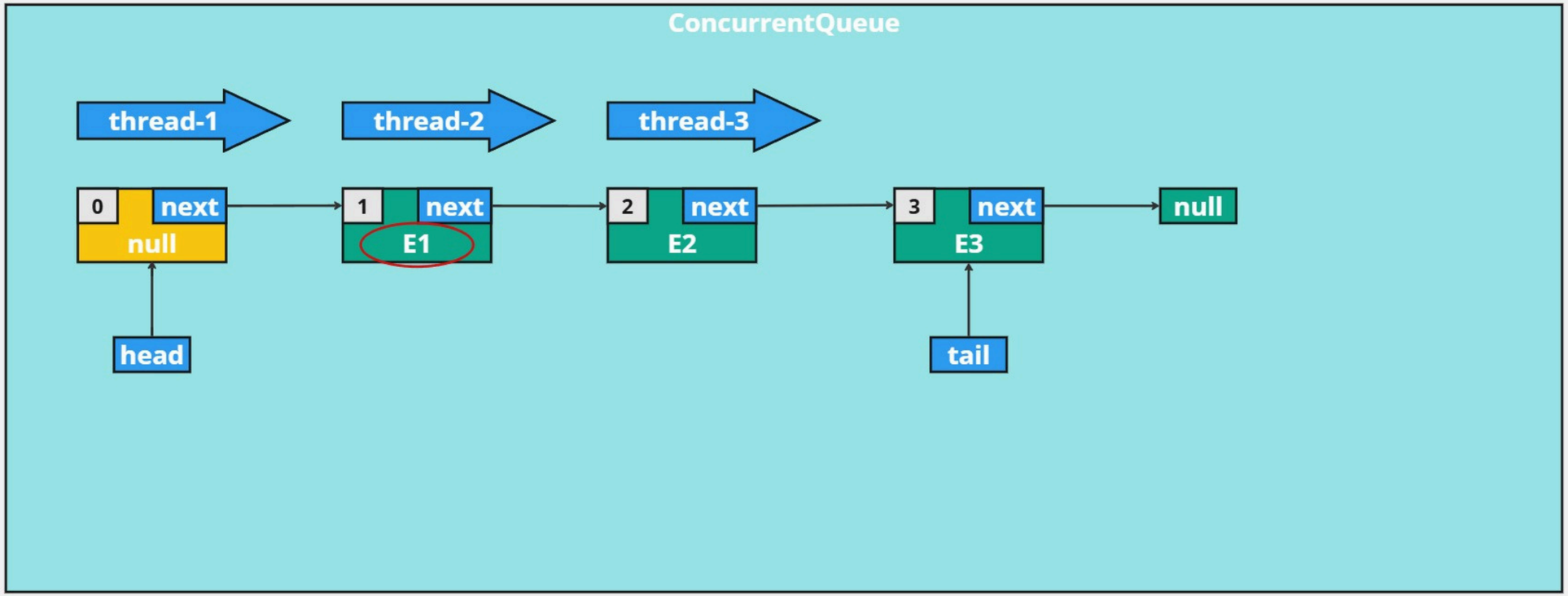
```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

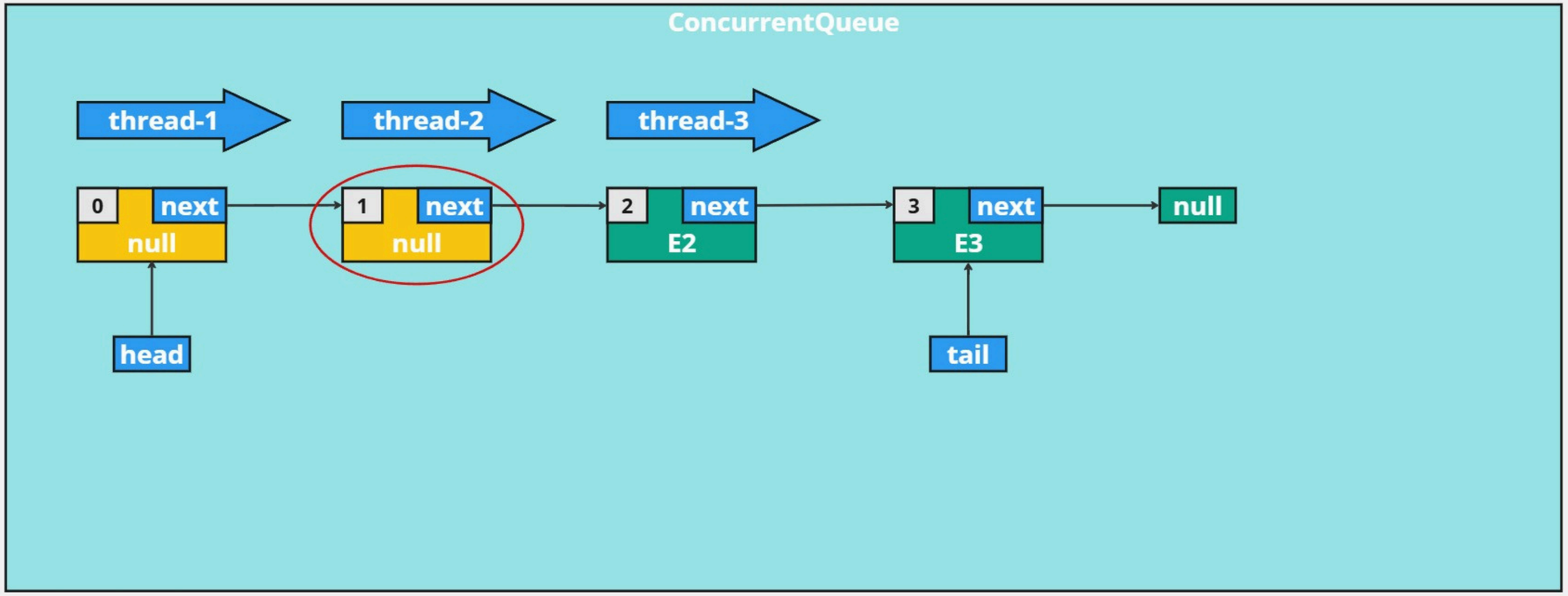
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

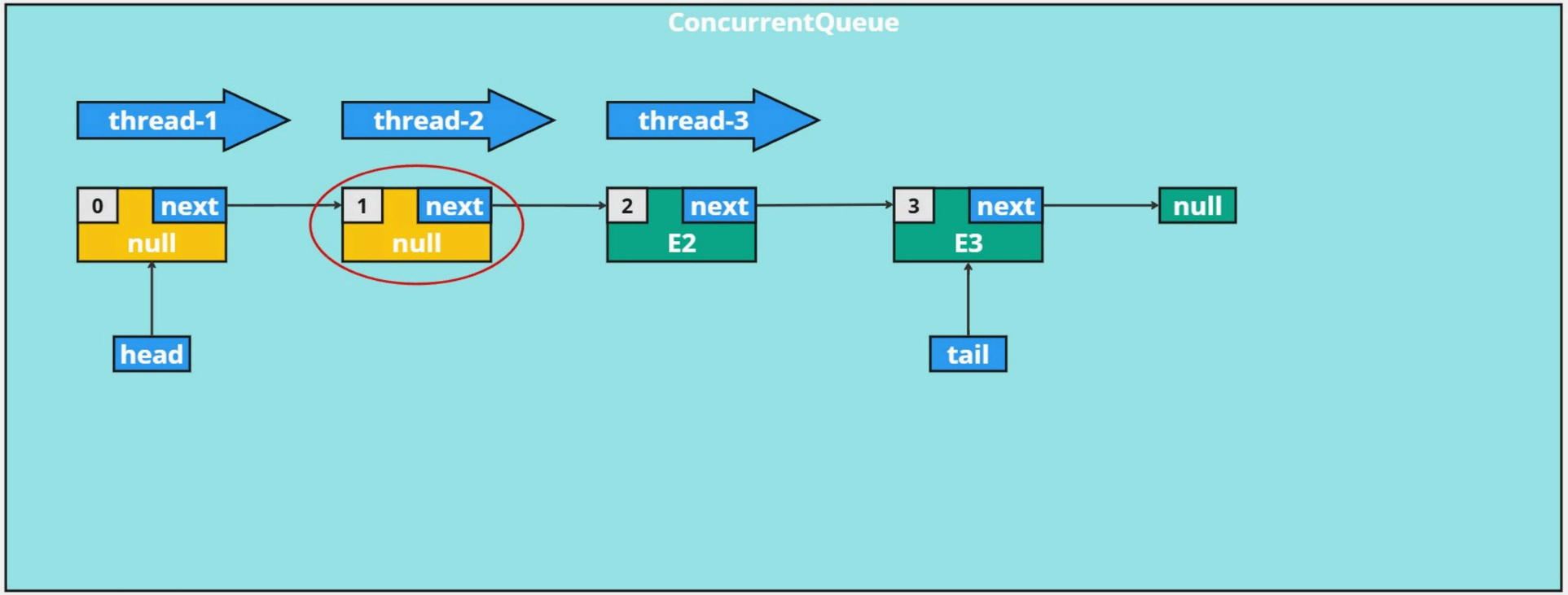
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

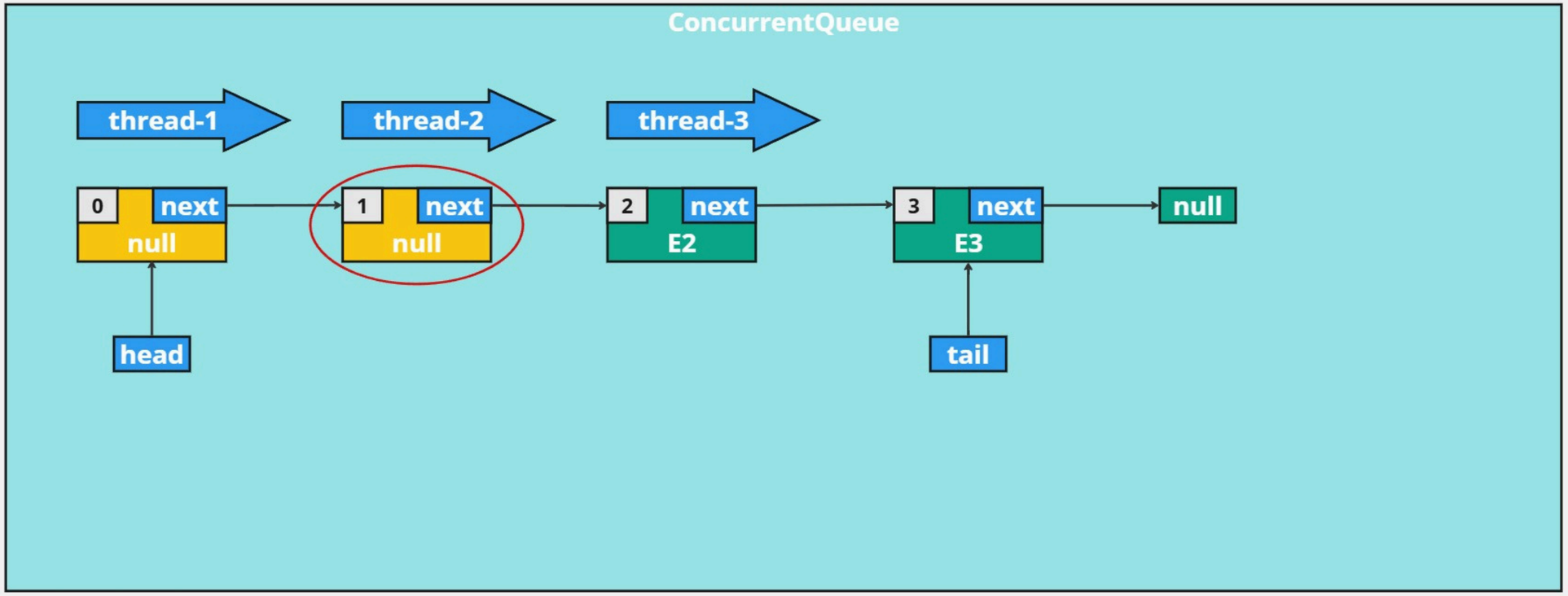
```
public Optional<E> dequeue() {                          0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                          0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                          0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

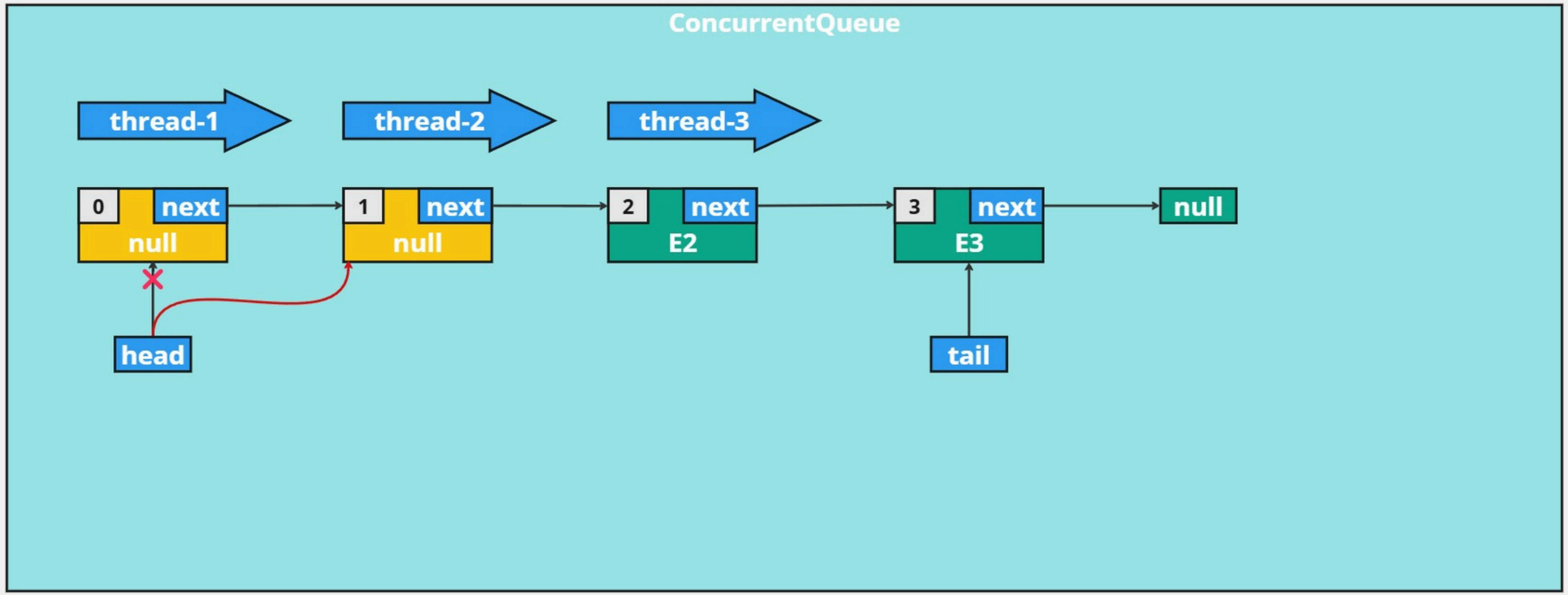
```
public Optional<E> dequeue() {                          0
    final Node<E> previousHead = head;                  1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                          0
    final Node<E> previousHead = head;                  1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                          0
    final Node<E> previousHead = head;                  1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                   E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

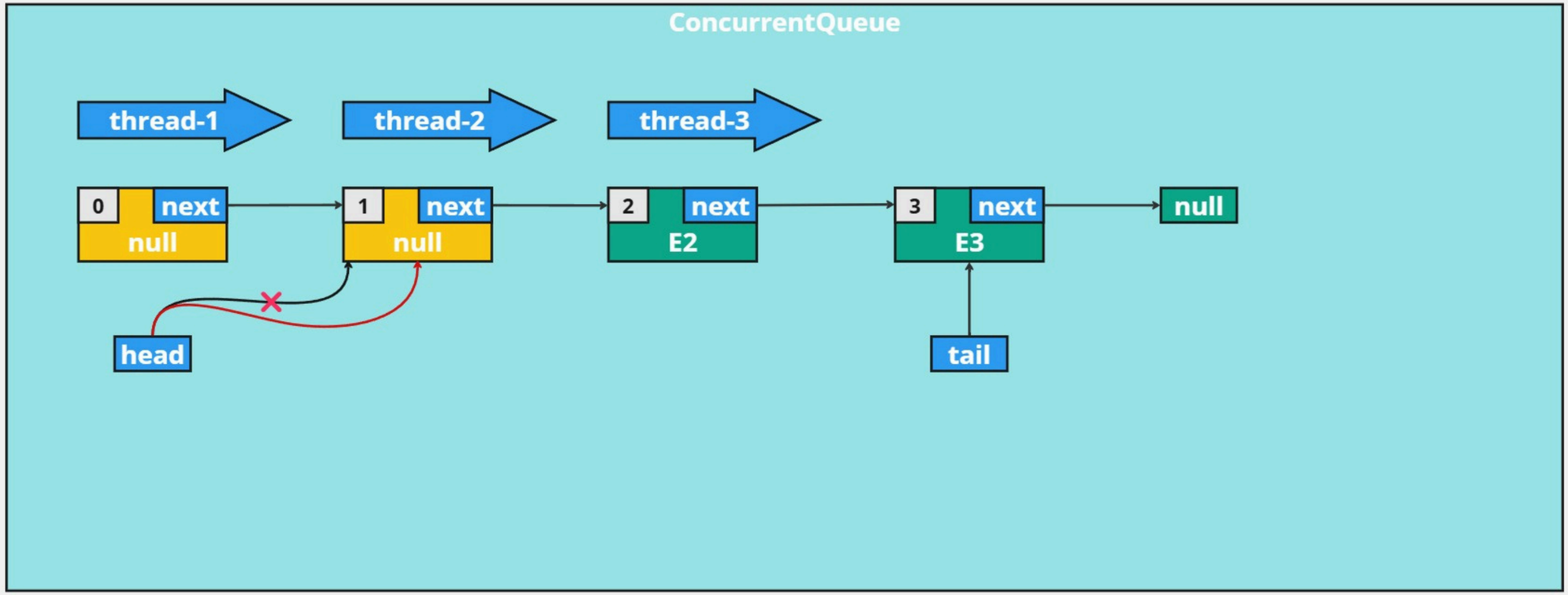
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

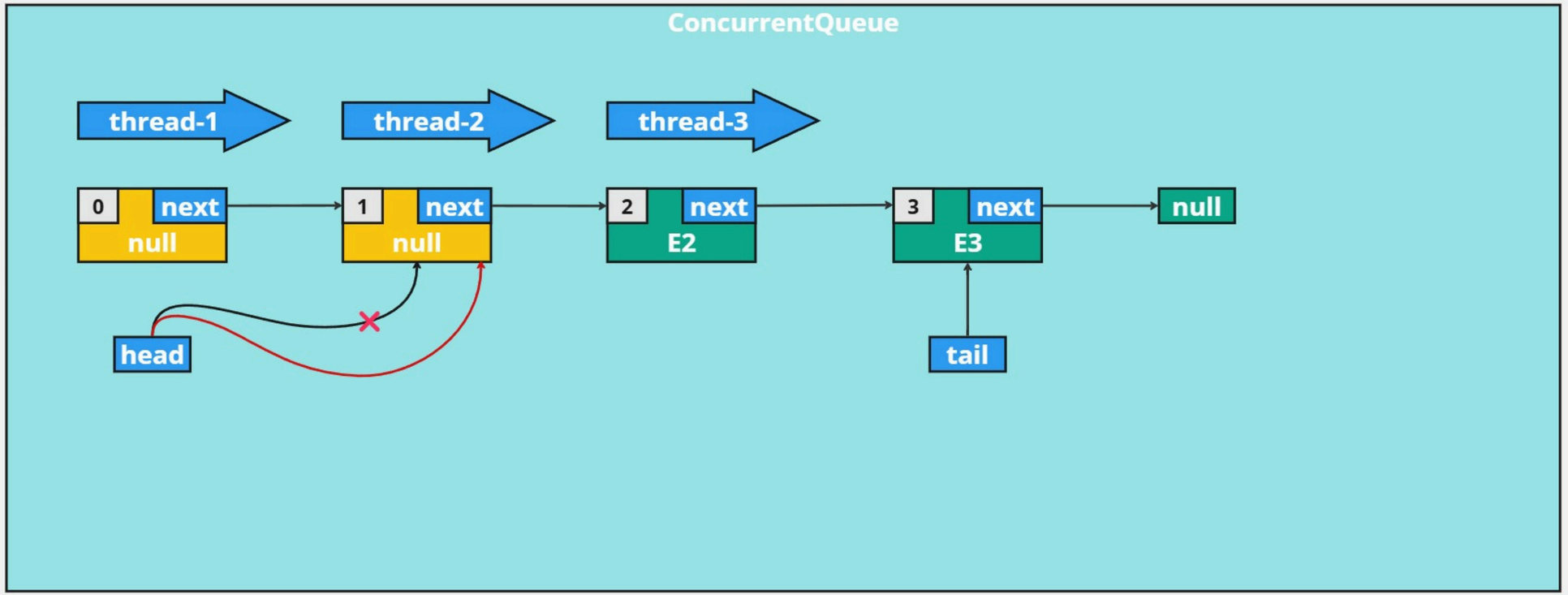
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

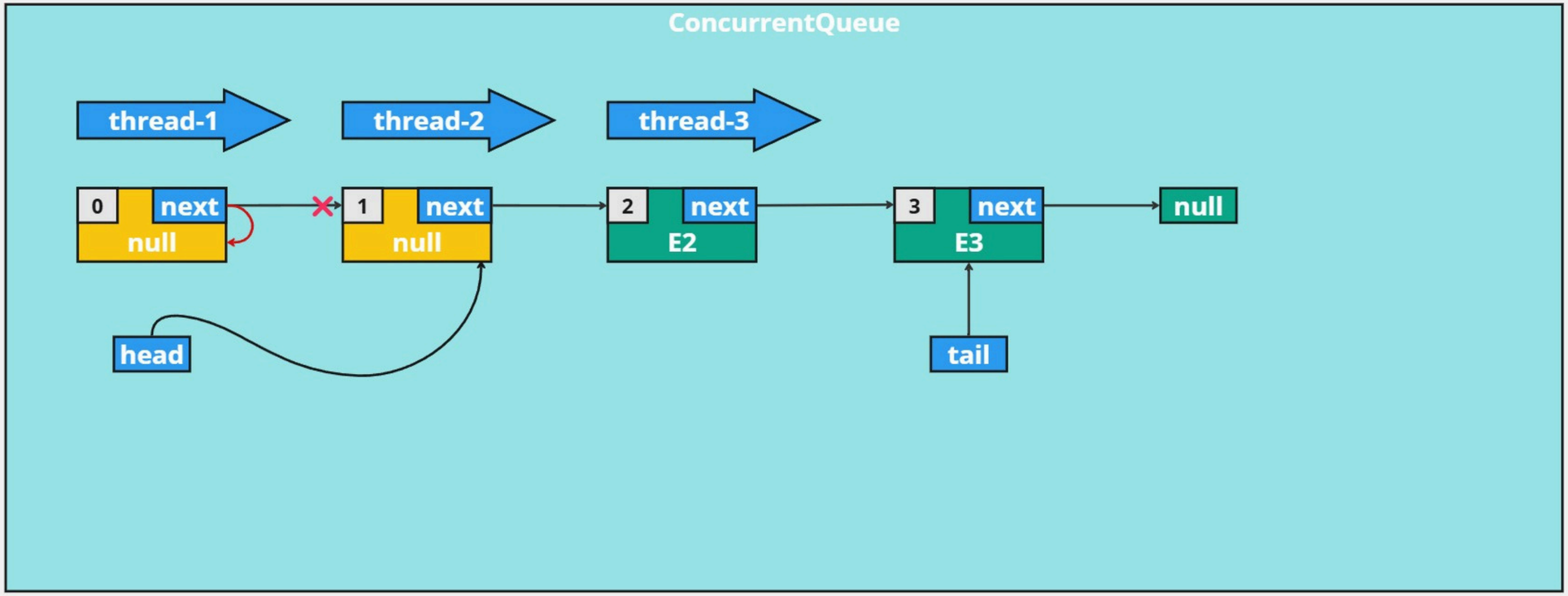
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

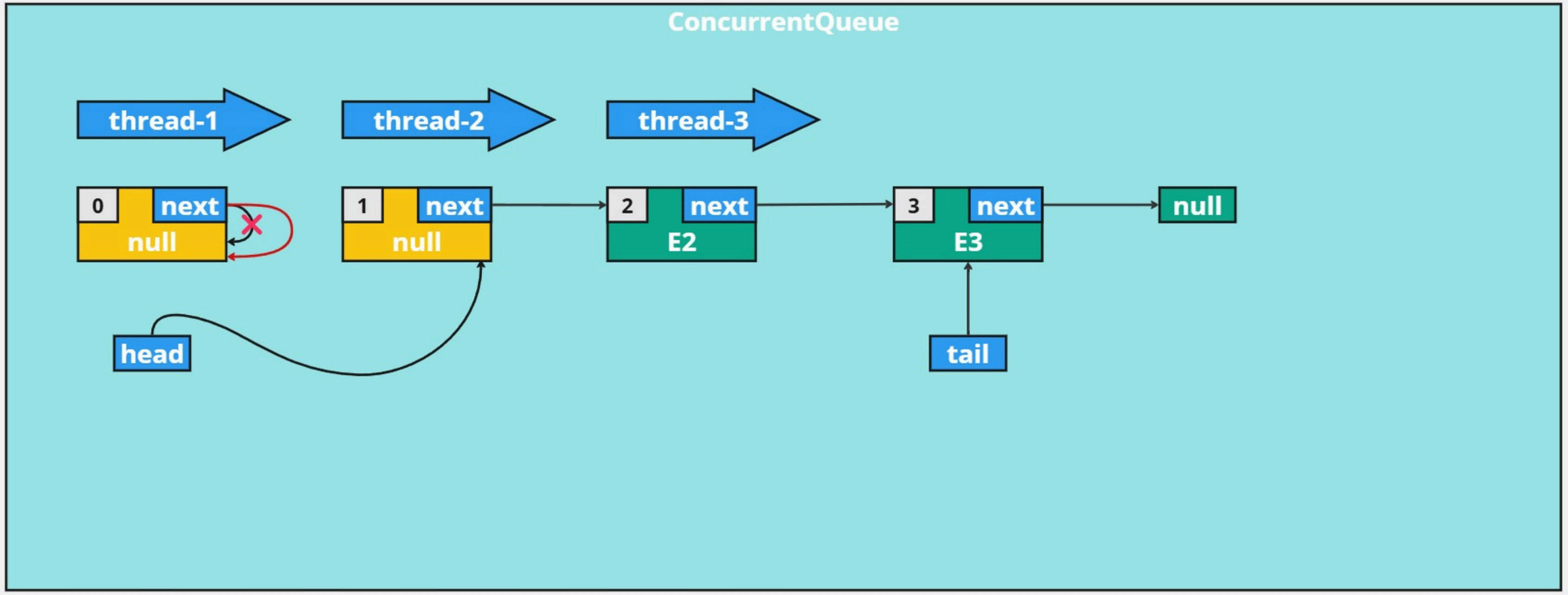
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

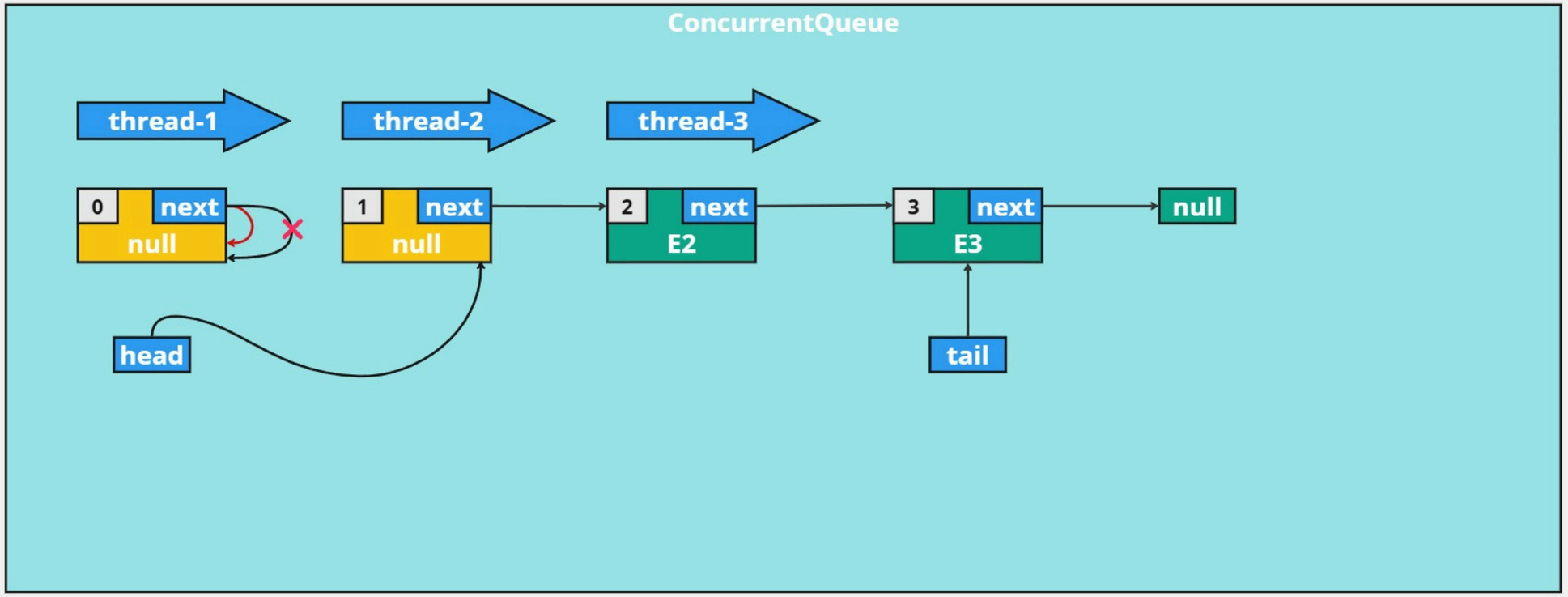
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

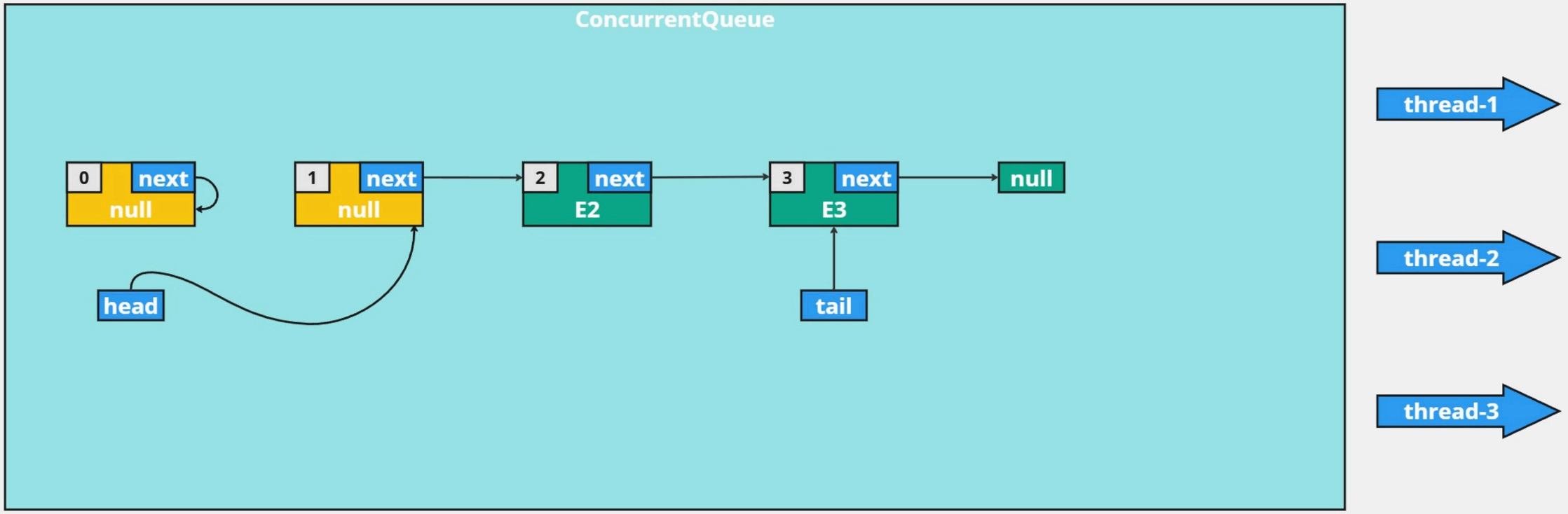
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                      1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```



thread-1

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

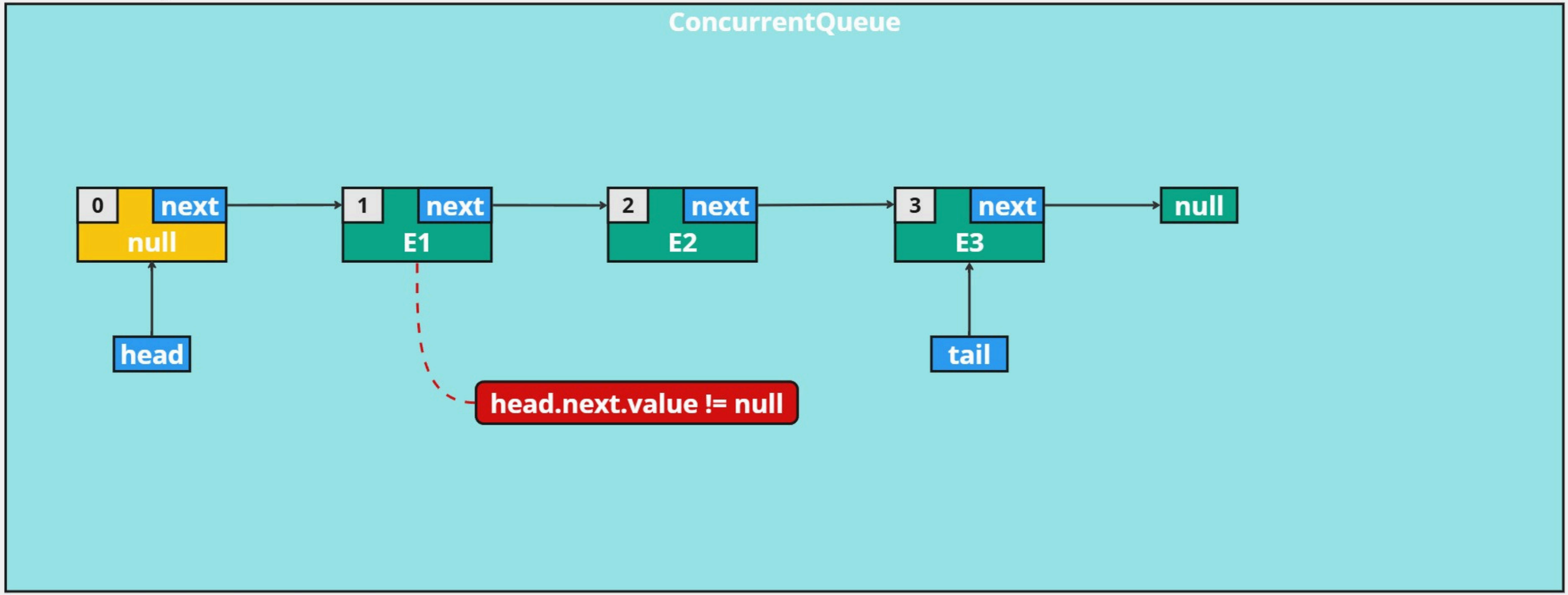
thread-2

```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

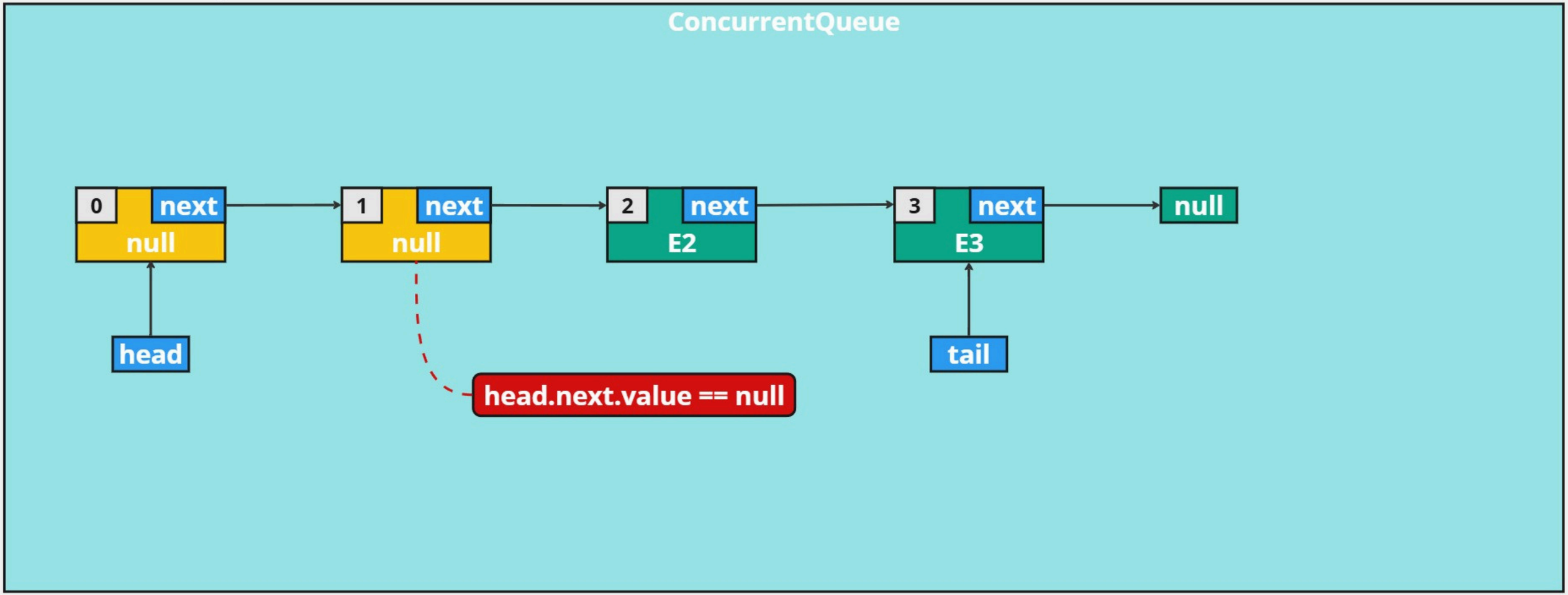
thread-3

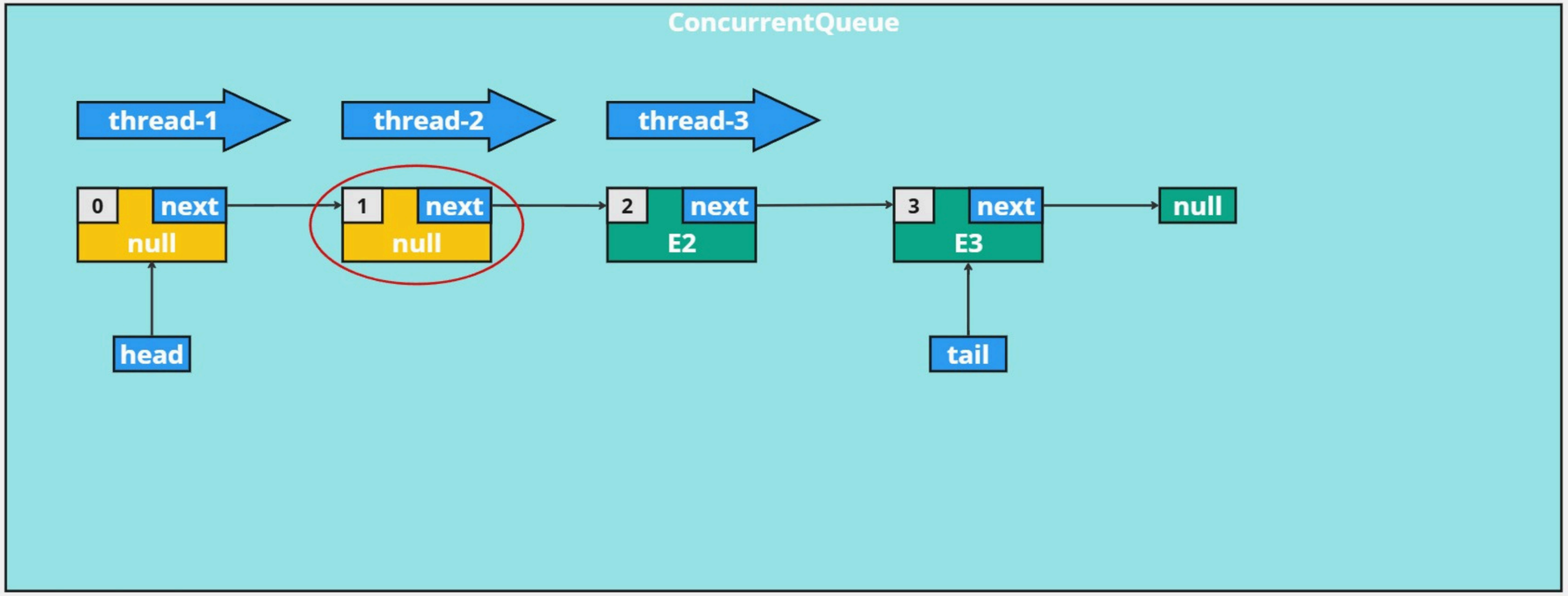
```
public Optional<E> dequeue() {                                  0
    final Node<E> previousHead = head;                          1
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

Consistency state



Not consistency state





thread-1

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-2

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;          -->  head.next.value == null
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

thread-3

```
public Optional<E> dequeue() {                                  0
1   final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }                                                           E1
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

```java
public Optional<E> dequeue() {
    final Node<E> previousHead = head;
    final Node<E> nextHead = previousHead.next.get();
    if (previousHead == tail.get()) {
        return empty();
    }
    final E element = nextHead.value;
    nextHead.value = null;
    head = nextHead;
    previousHead.next.set(previousHead);
    return Optional.of(element);
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value;
        if (element != null) {
            nextHead.value = null;
            head = nextHead;
            previousHead.next.set(previousHead);
            return Optional.of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value;
        if (element != null) {
            nextHead.value = null;
            head = nextHead;
            previousHead.next.set(previousHead);
            return Optional.of(element);
        }
    }
}
```

not atomic

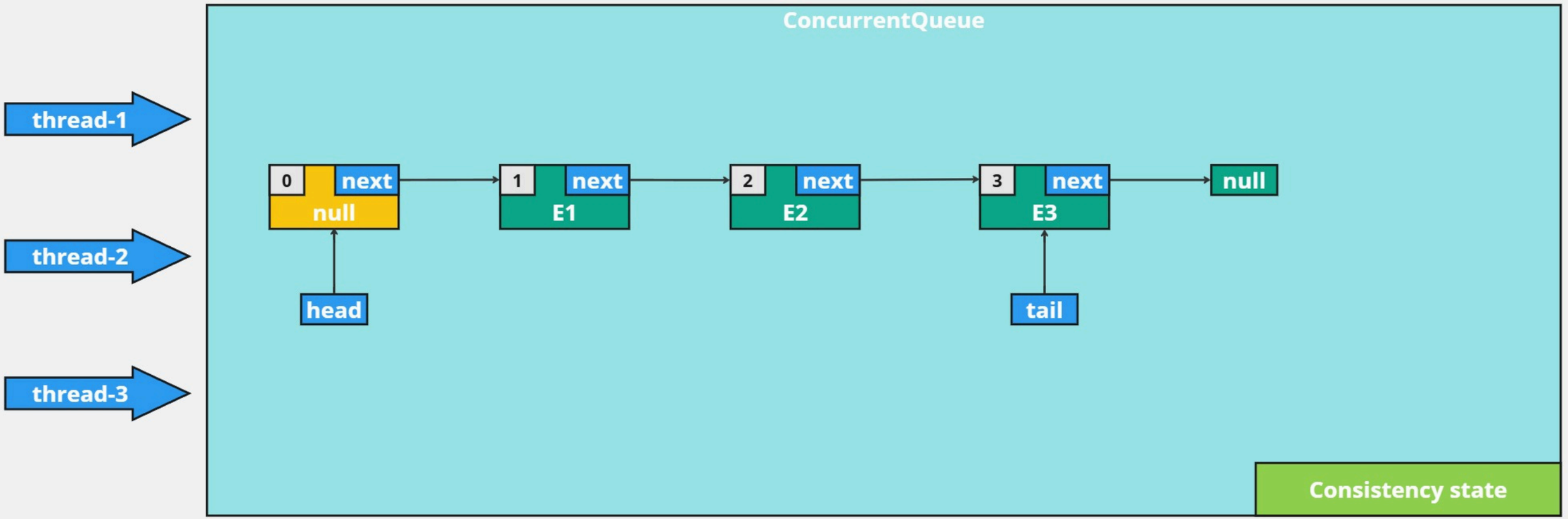
```
private static final class Node<E> {
    1 usage
    private volatile E value;
    3 usages
    private final AtomicReference<Node<E>> next;

    1 usage
    public Node() {
        next = new AtomicReference<>();
    }

    1 usage
    public Node(final E value) {
        this.value = value;
        next = new AtomicReference<>();
    }
}
```

```
private static final class Node<E> {
    4 usages
    private final AtomicReference<E> value;
    6 usages
    private final AtomicReference<Node<E>> next;

    1 usage  Vlad Zuev
    public Node() {
        value = new AtomicReference<>();
        next = new AtomicReference<>();
    }

    1 usage  Vlad Zuev
    public Node(final E value) {
        this.value = new AtomicReference<>(value);
        next = new AtomicReference<>();
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value;
        if (element != null) {
            nextHead.value = null;
            head = nextHead;
            previousHead.next.set(previousHead);
            return Optional.of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

atomic



thread-1

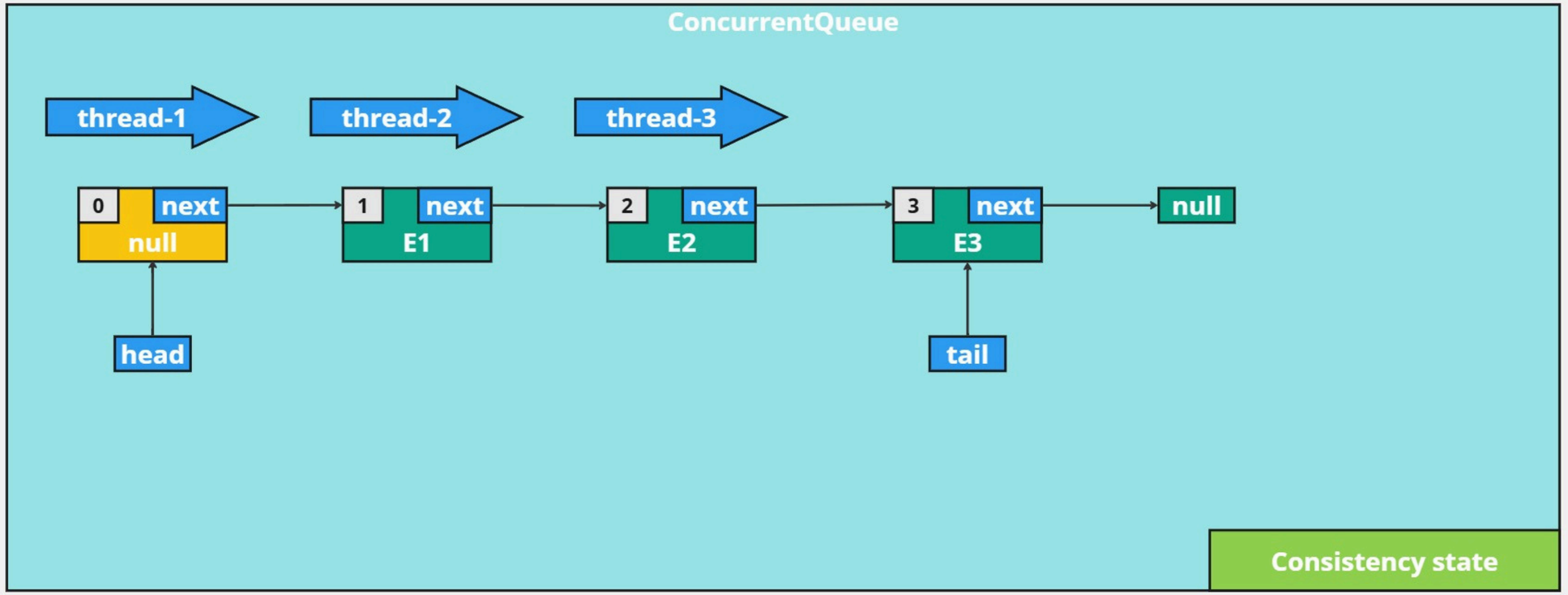
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

thread-2

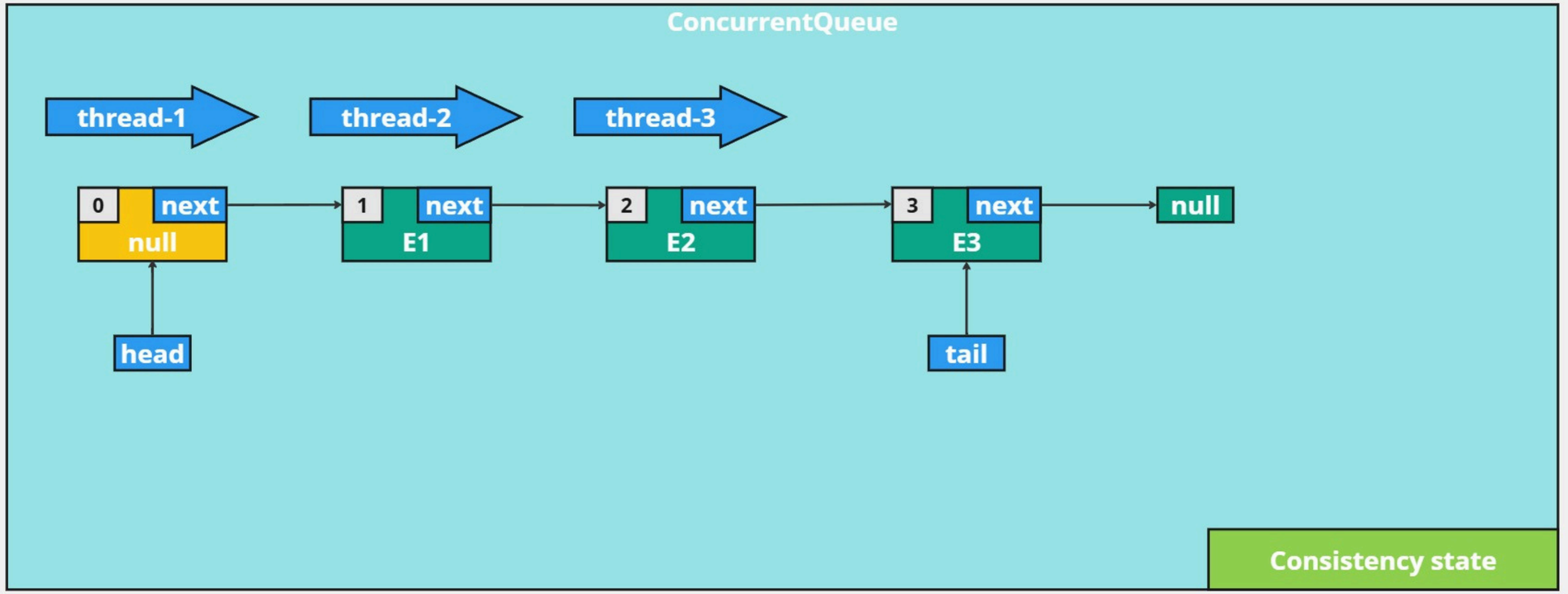
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

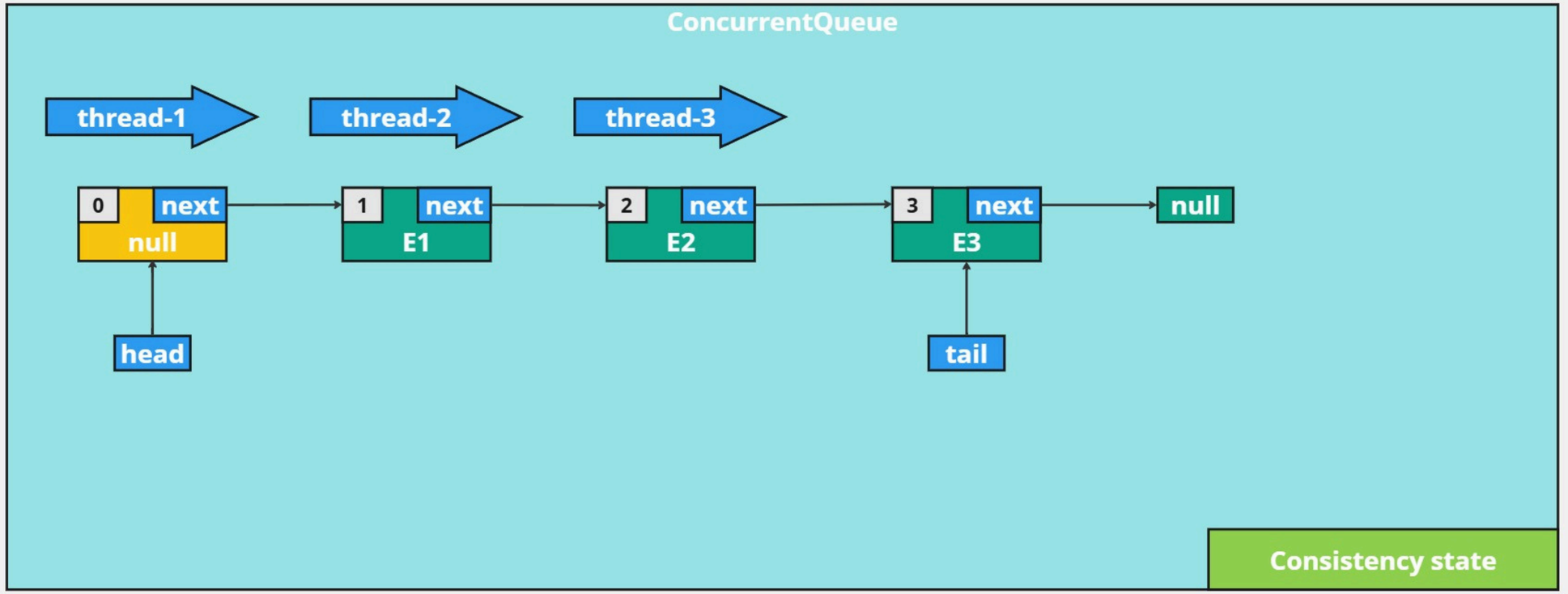
1

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1



thread-1

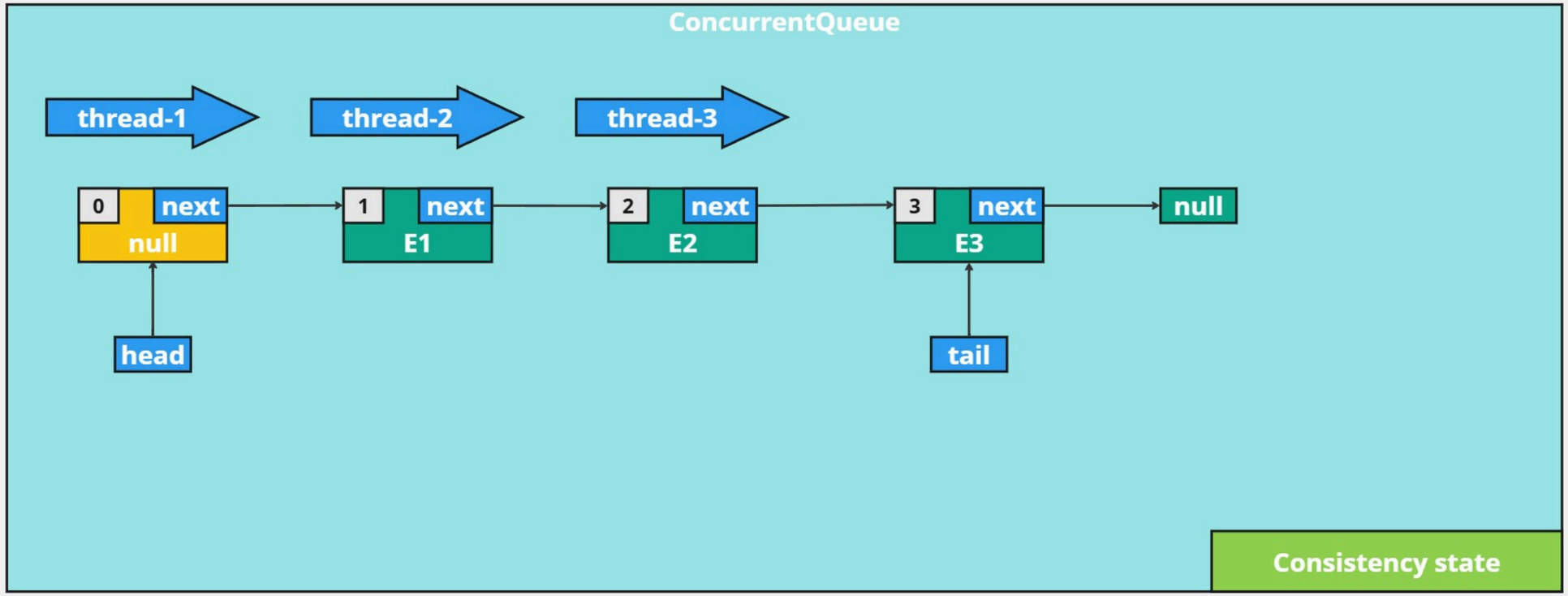
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

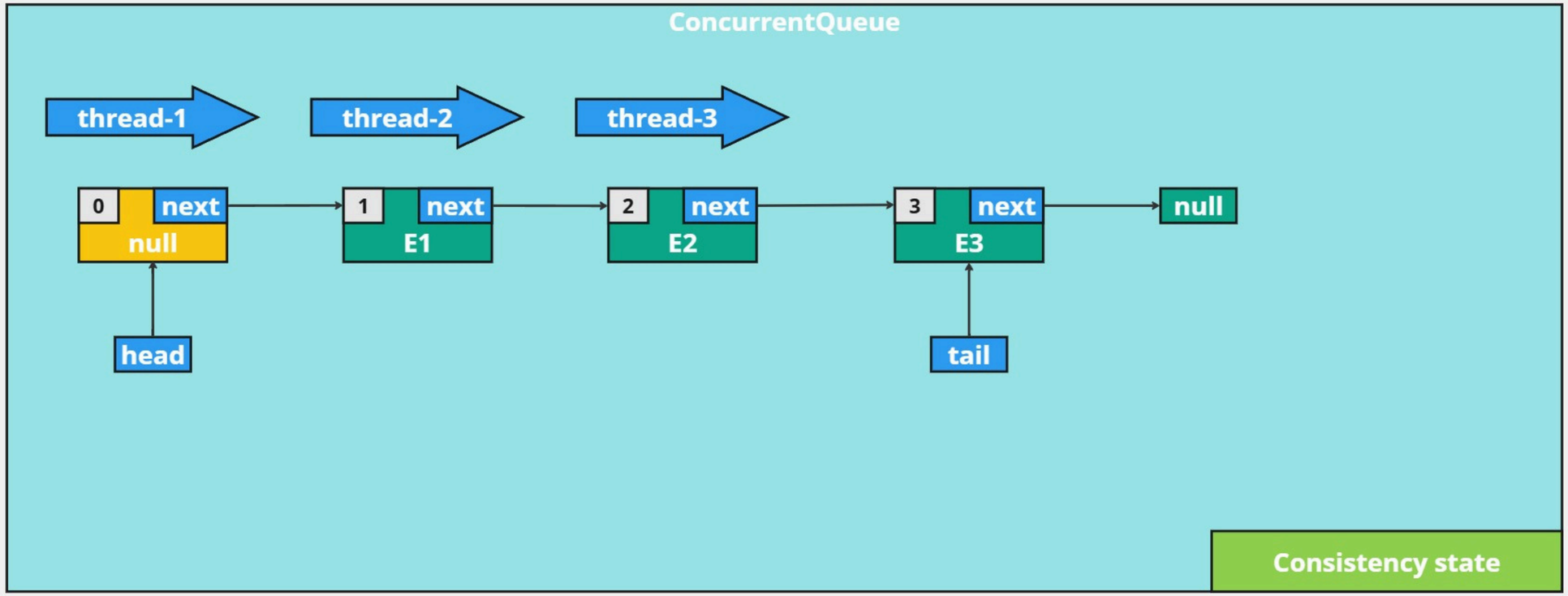
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

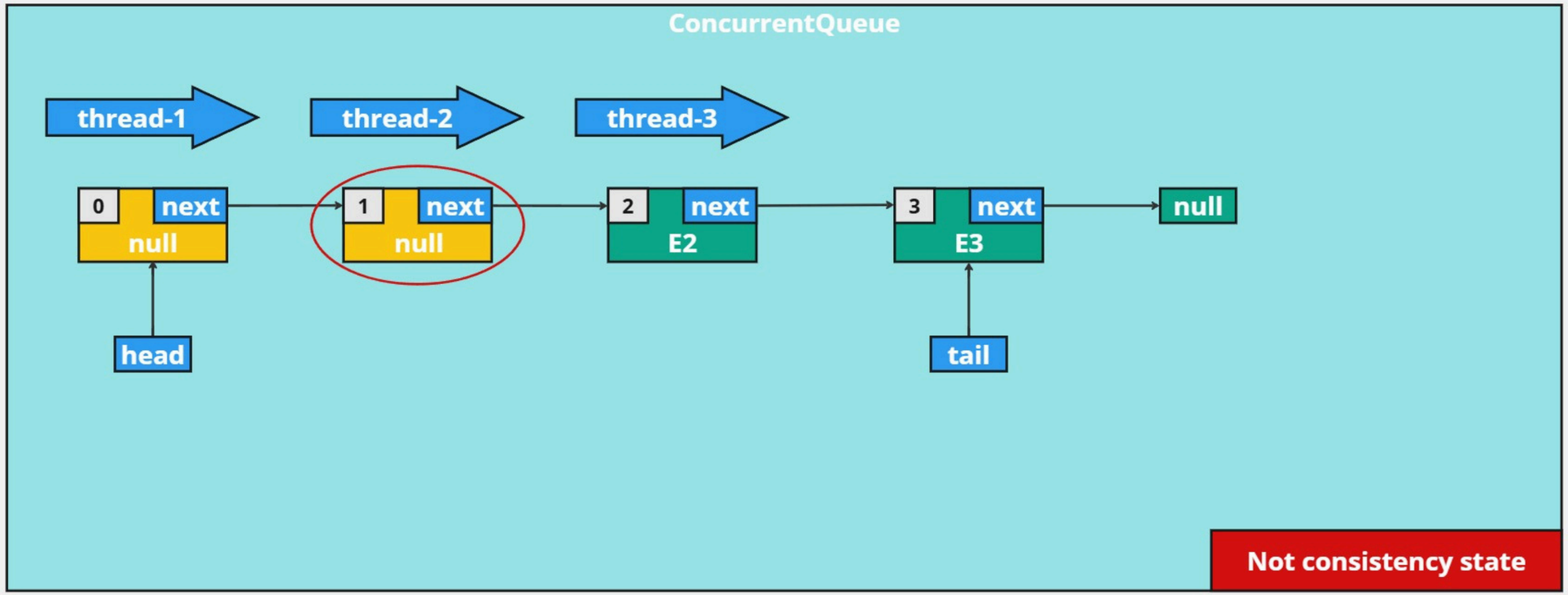
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0 1 E1

thread-2

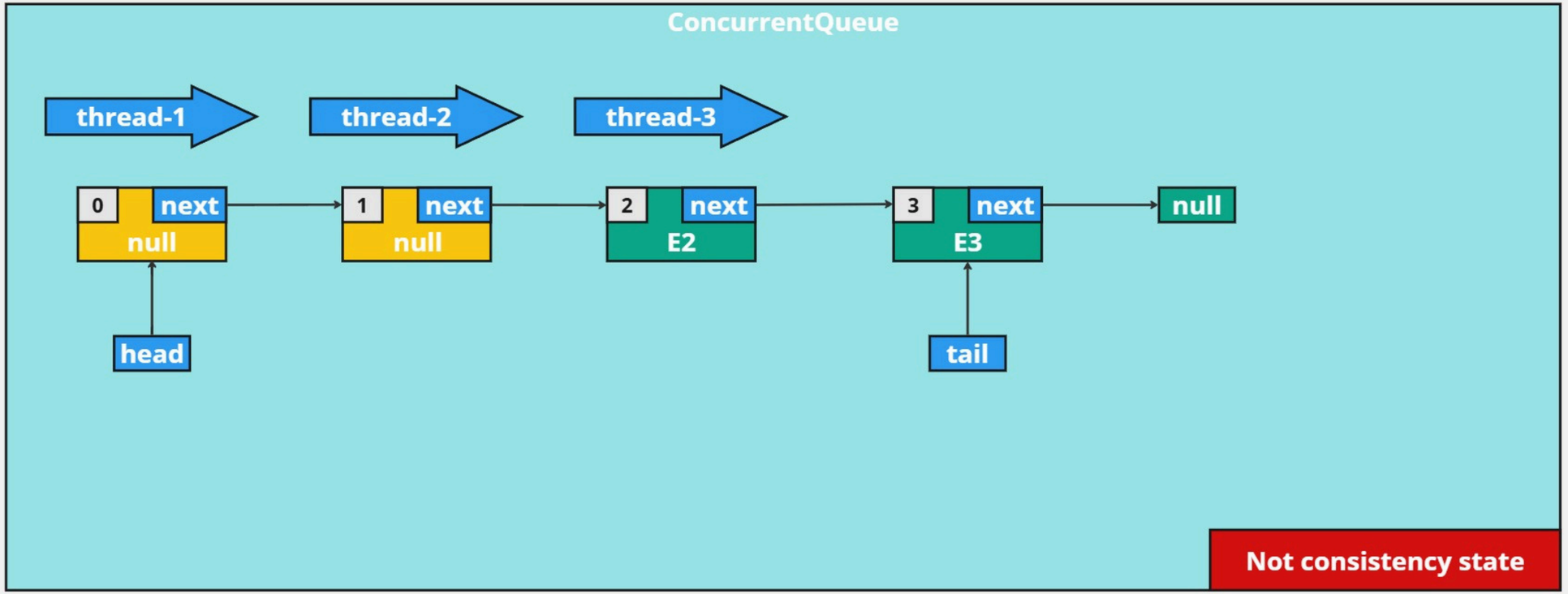
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0 1 E1

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0 1 E1



thread-1

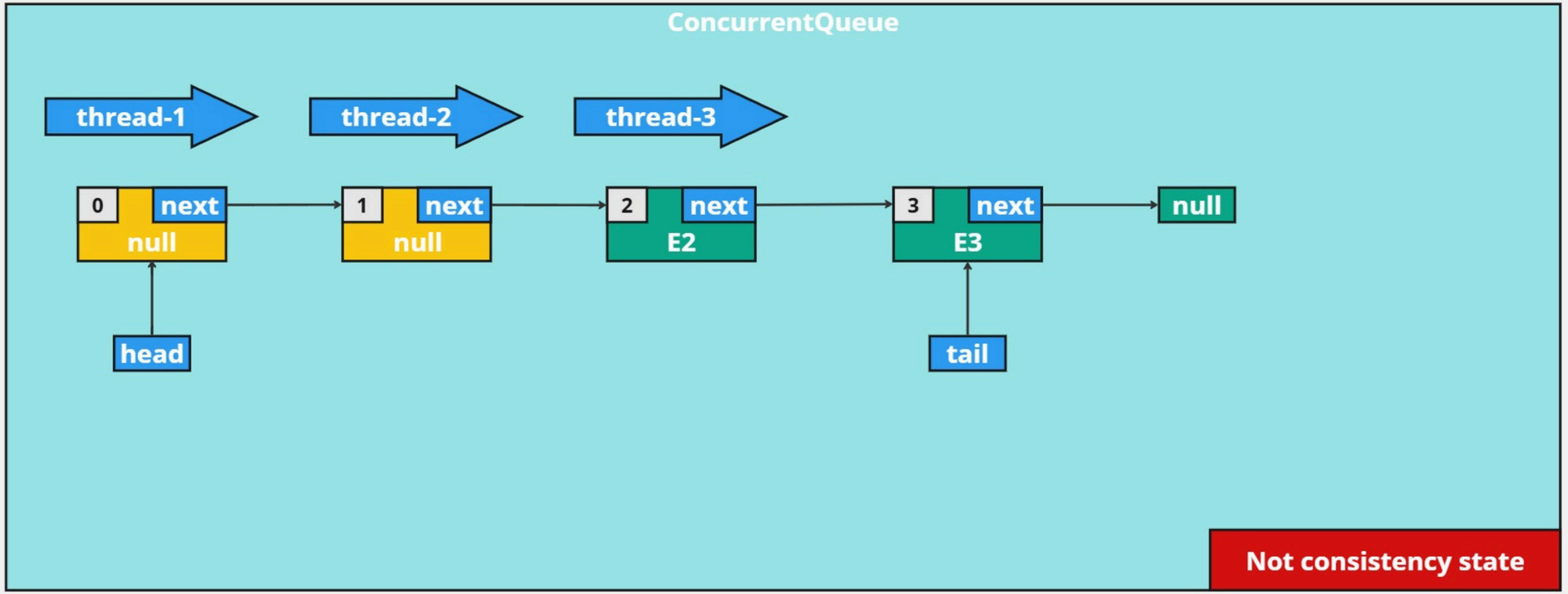
```
public Optional<E> dequeue() {
    while (true) {                                                          0
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
1       if (previousHead == tail.get()) {
            return empty();
        }                                                                   E1
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {                                                          0
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
1       if (previousHead == tail.get()) {
            return empty();
        }                                                                   E1
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {                                                          0
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
1       if (previousHead == tail.get()) {
            return empty();
        }                                                                   E1
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

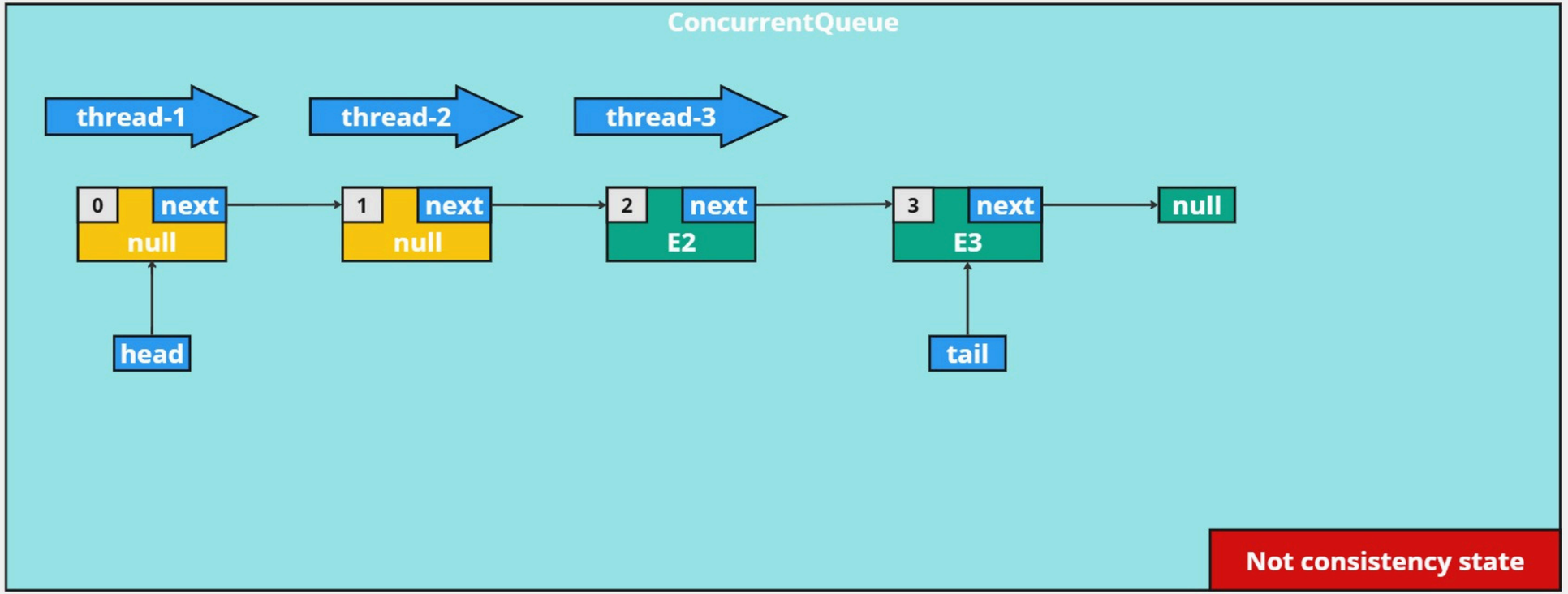
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

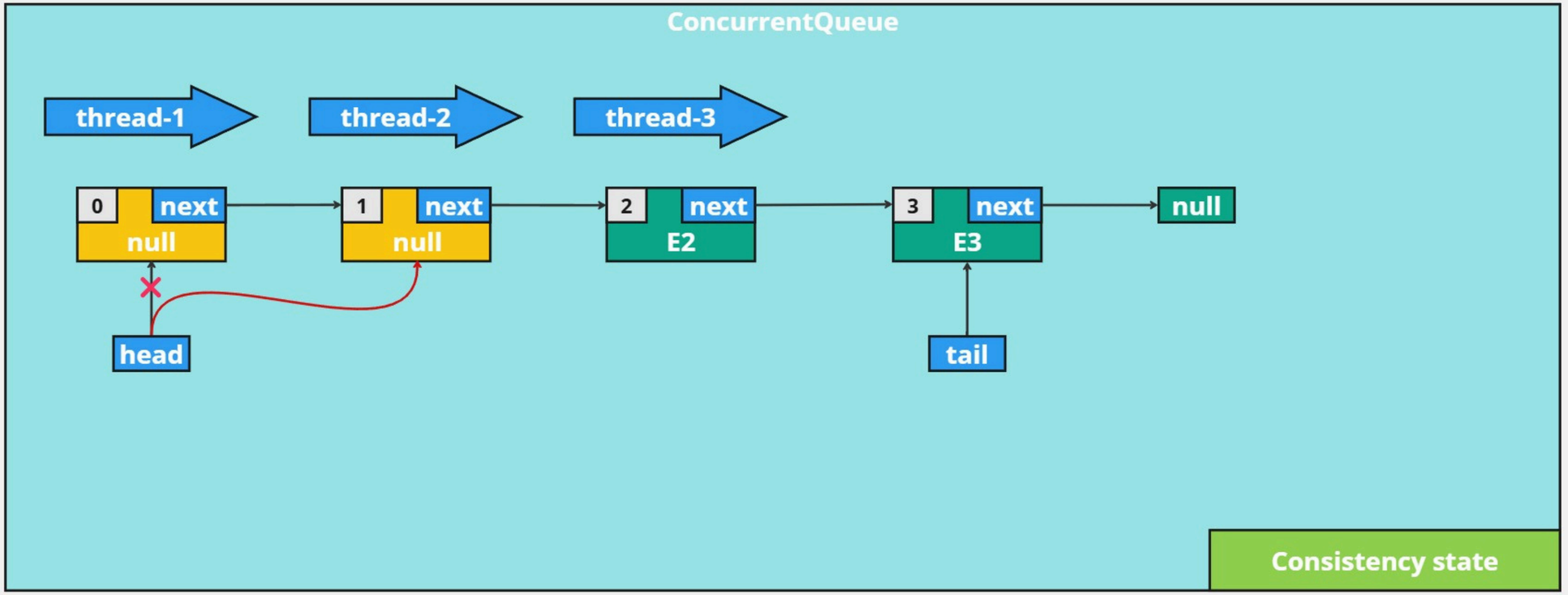
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

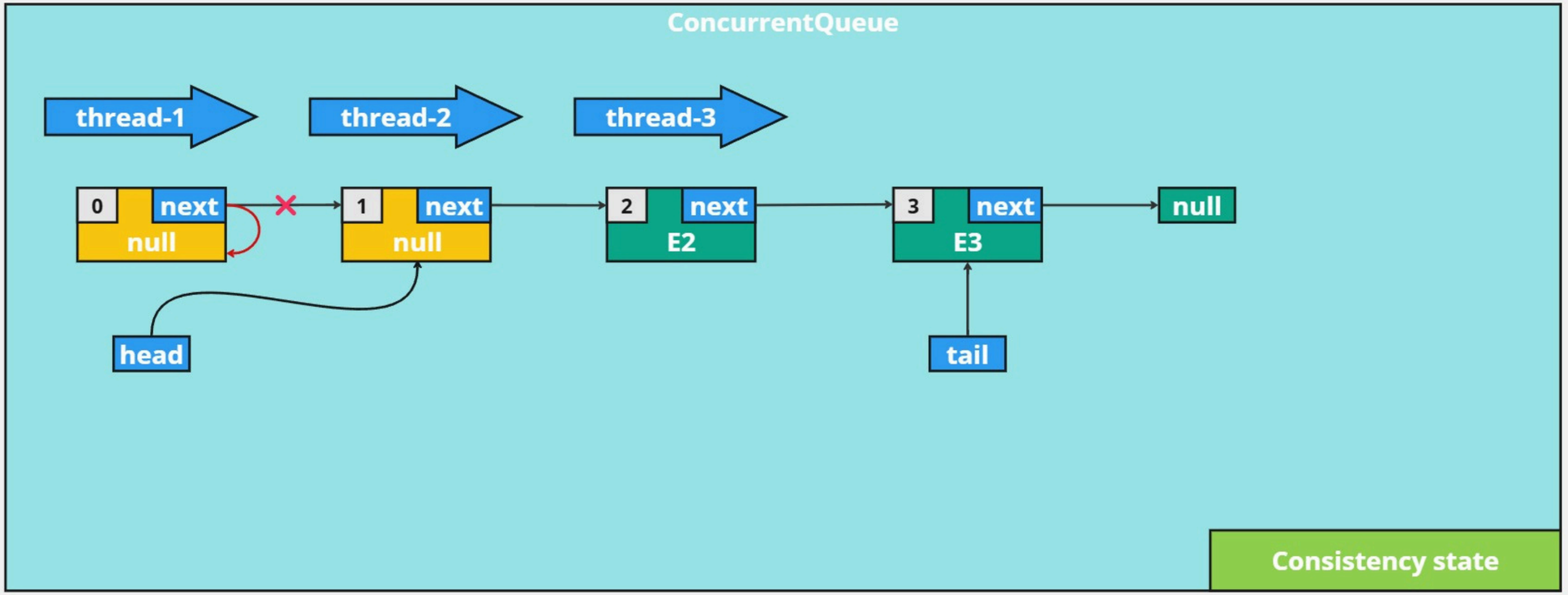
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

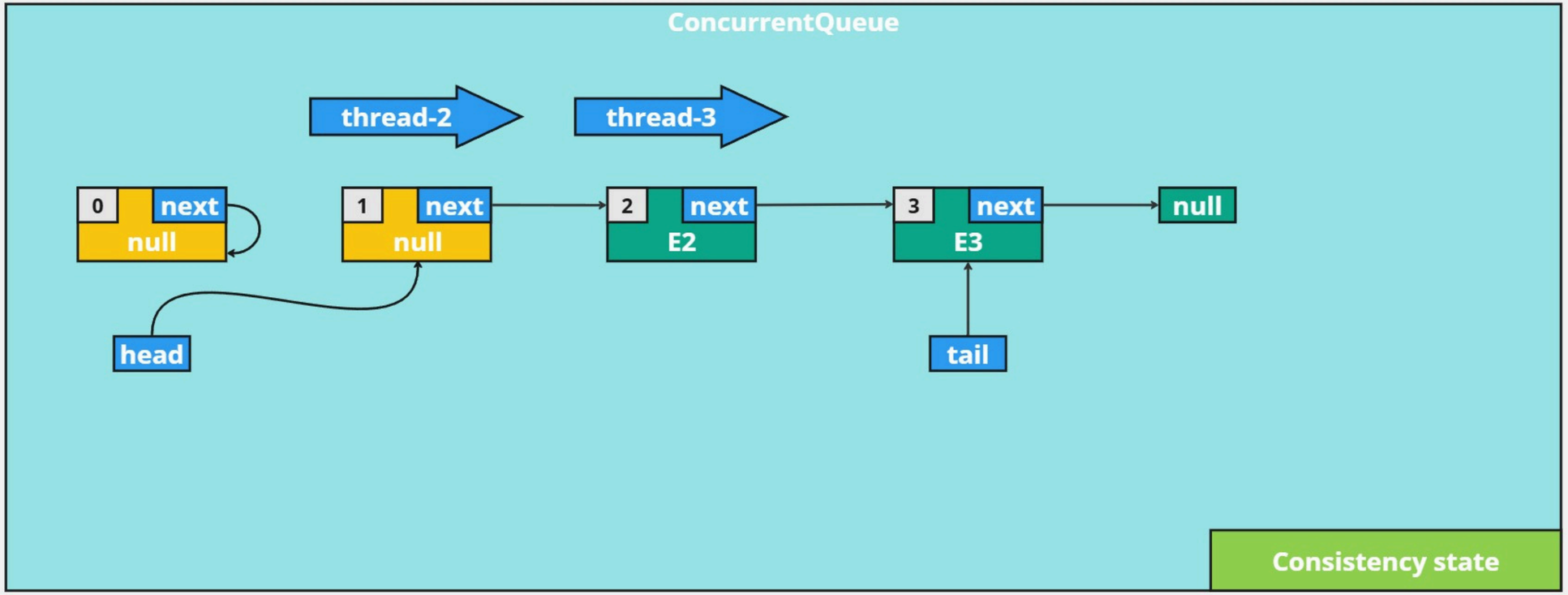
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

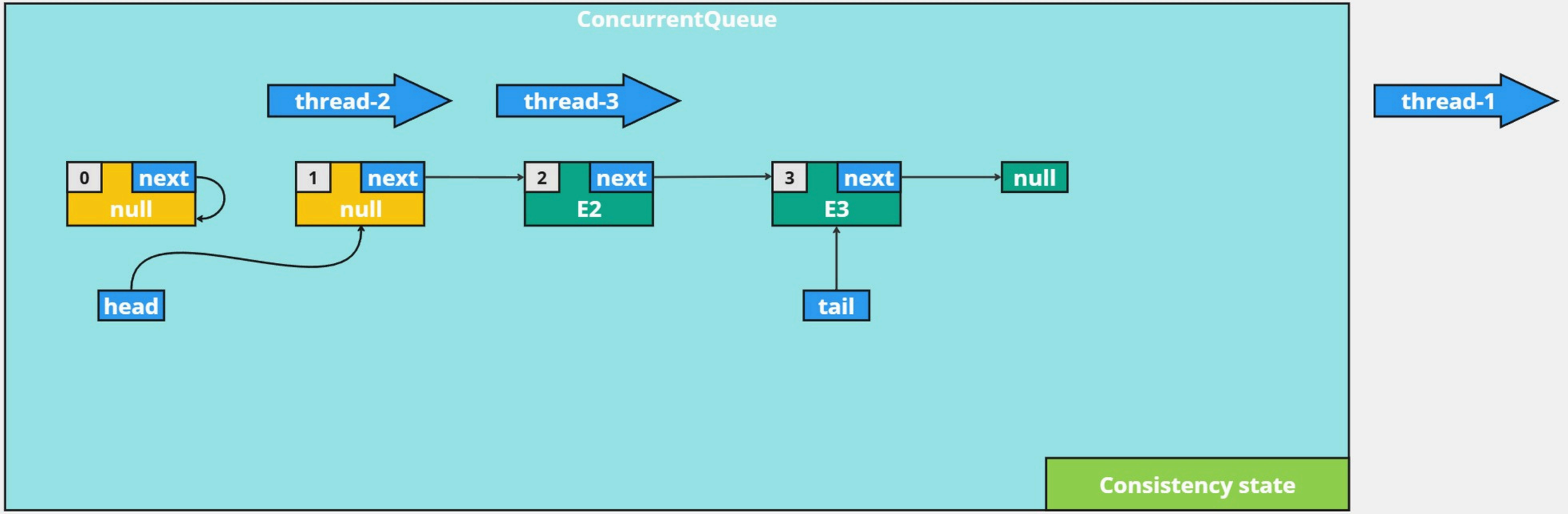
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

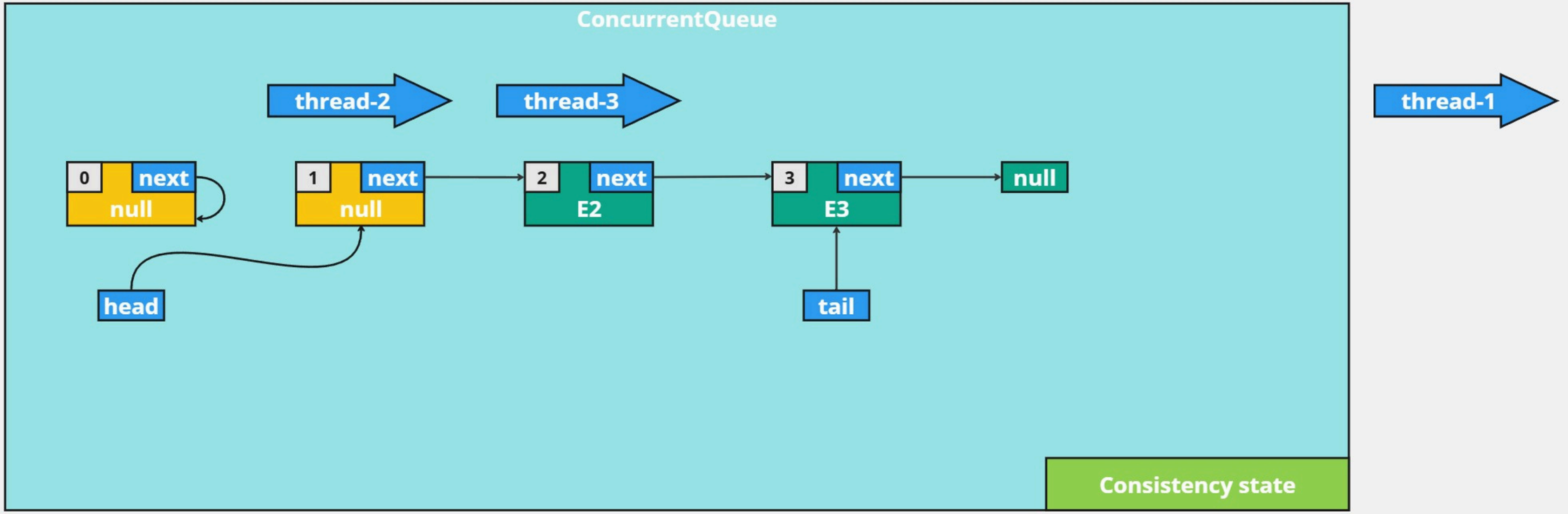
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1

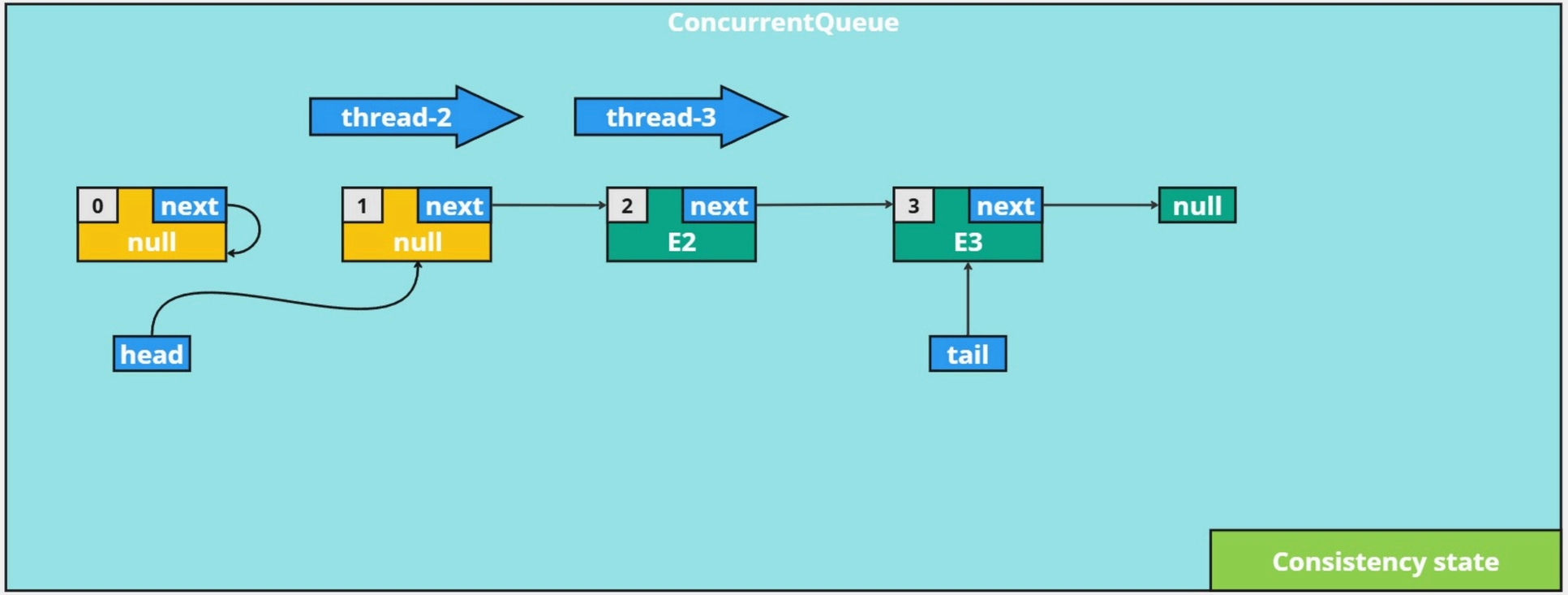
2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1

2



thread-1

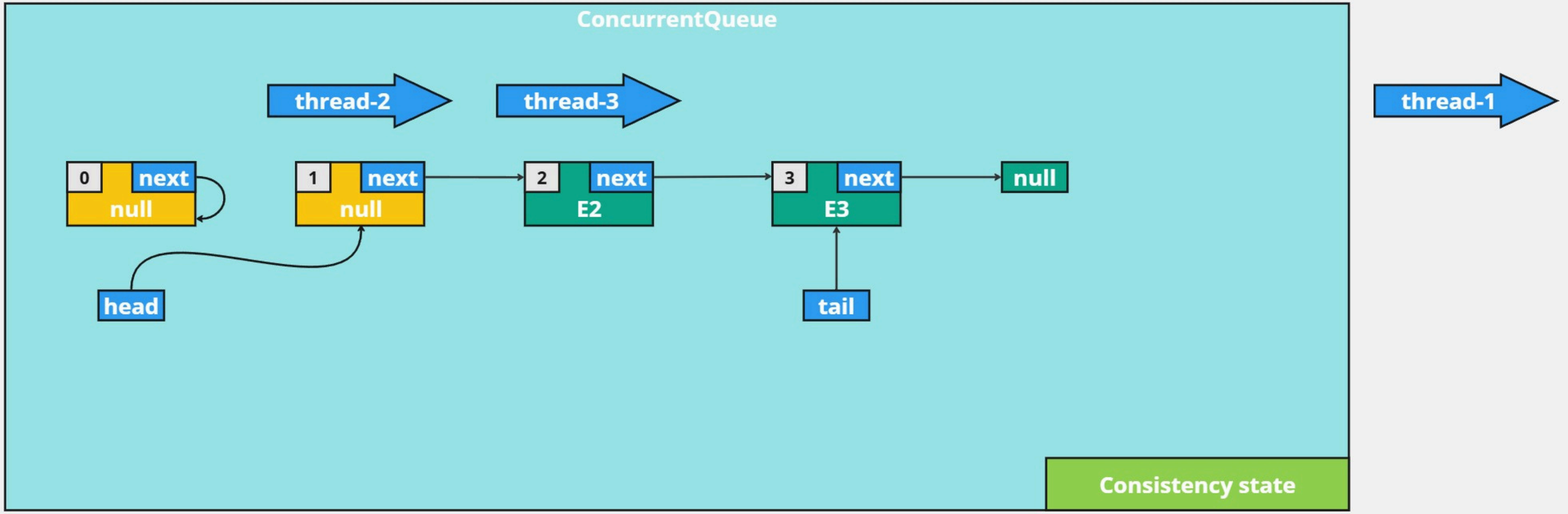
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 — `previousHead = head;`

2 — `previousHead == tail.get()`

E2 — `element = nextHead.value.get();`

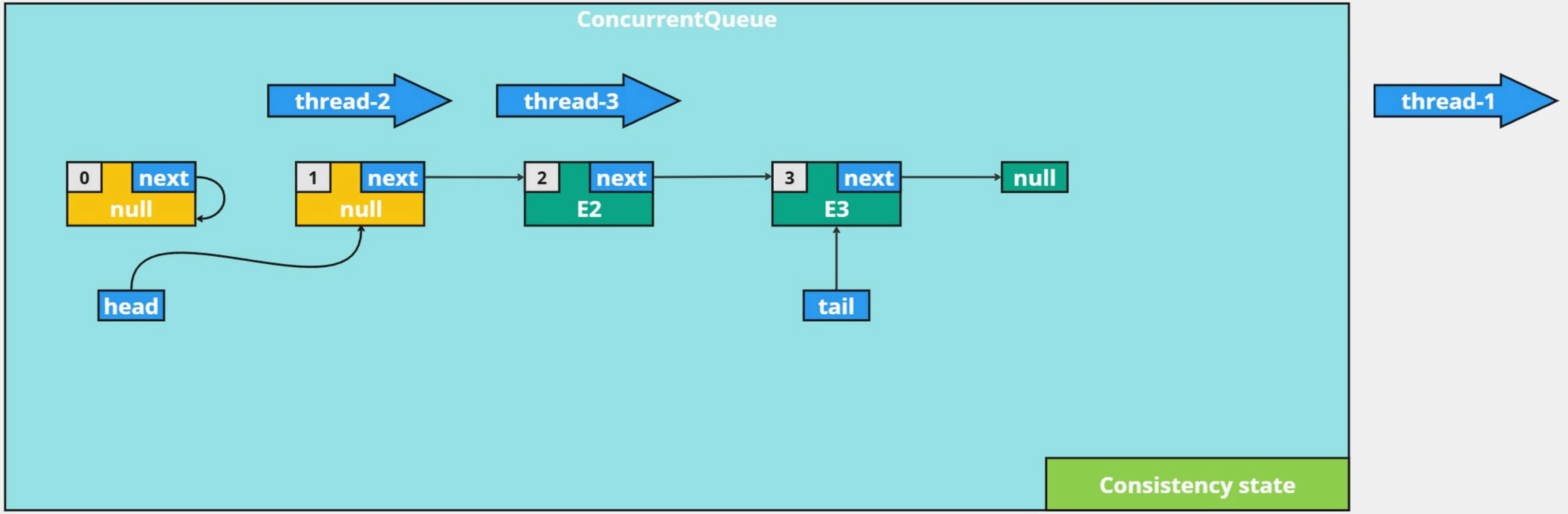
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 — `previousHead = head;`

2 — `previousHead == tail.get()`

E2 — `element = nextHead.value.get();`



thread-1

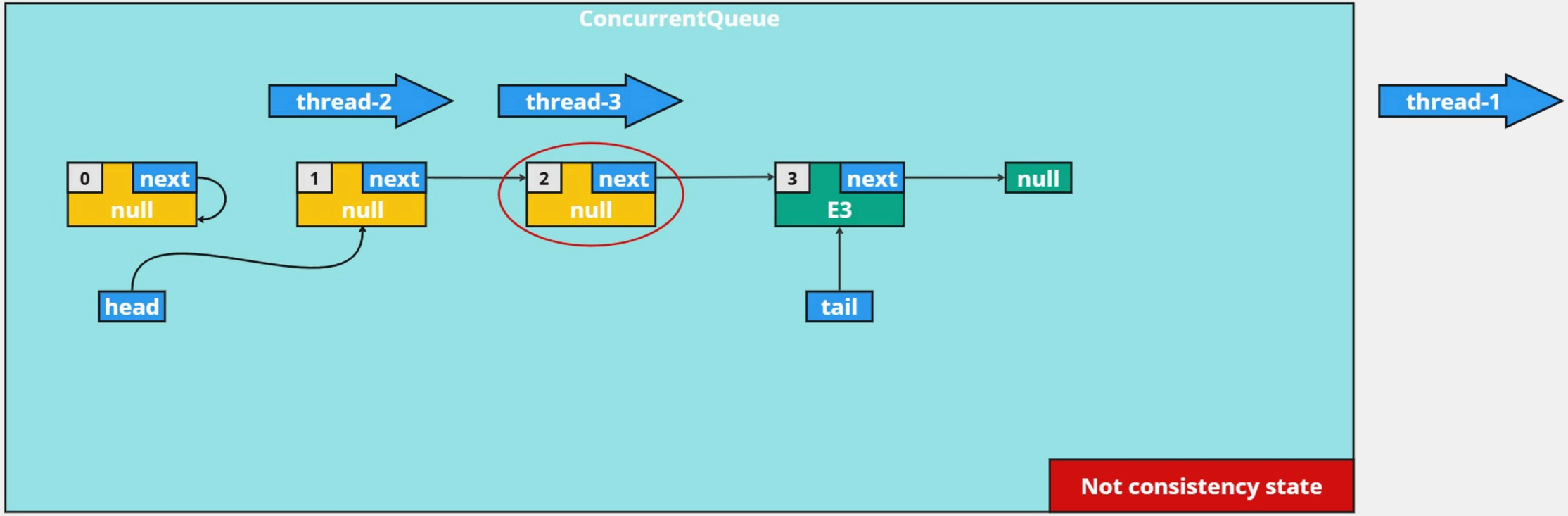
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 1
        final Node<E> nextHead = previousHead.next.get();     // 2
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E2
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 1
        final Node<E> nextHead = previousHead.next.get();     // 2
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E2
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

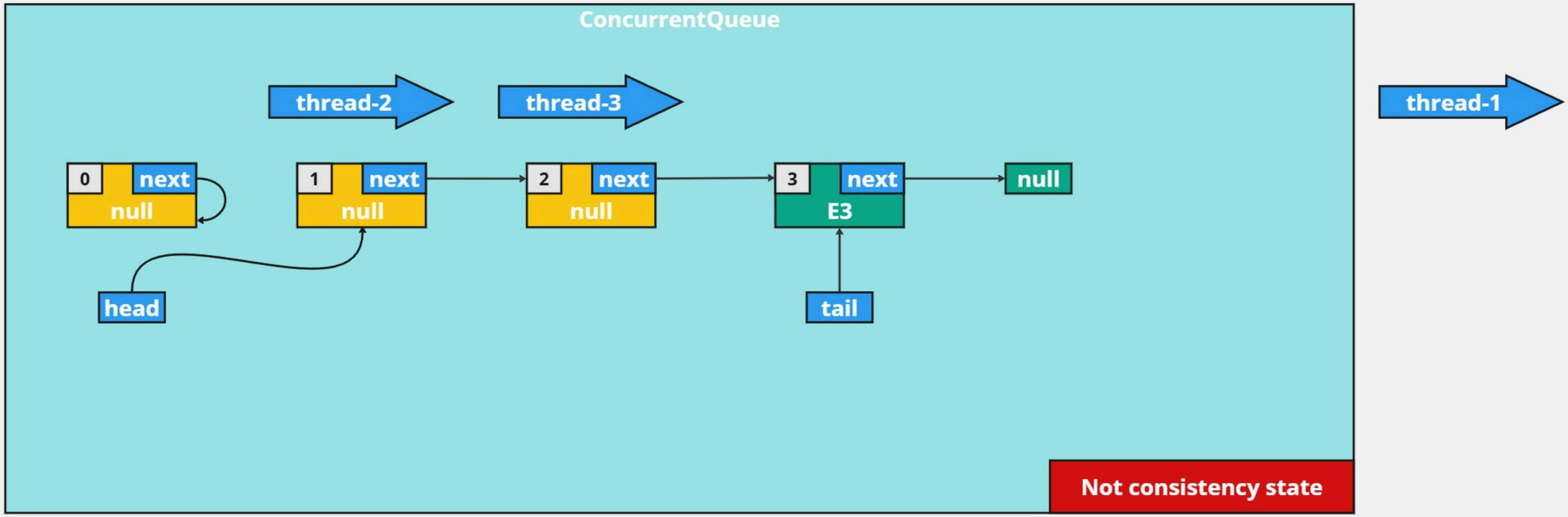
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2



thread-1

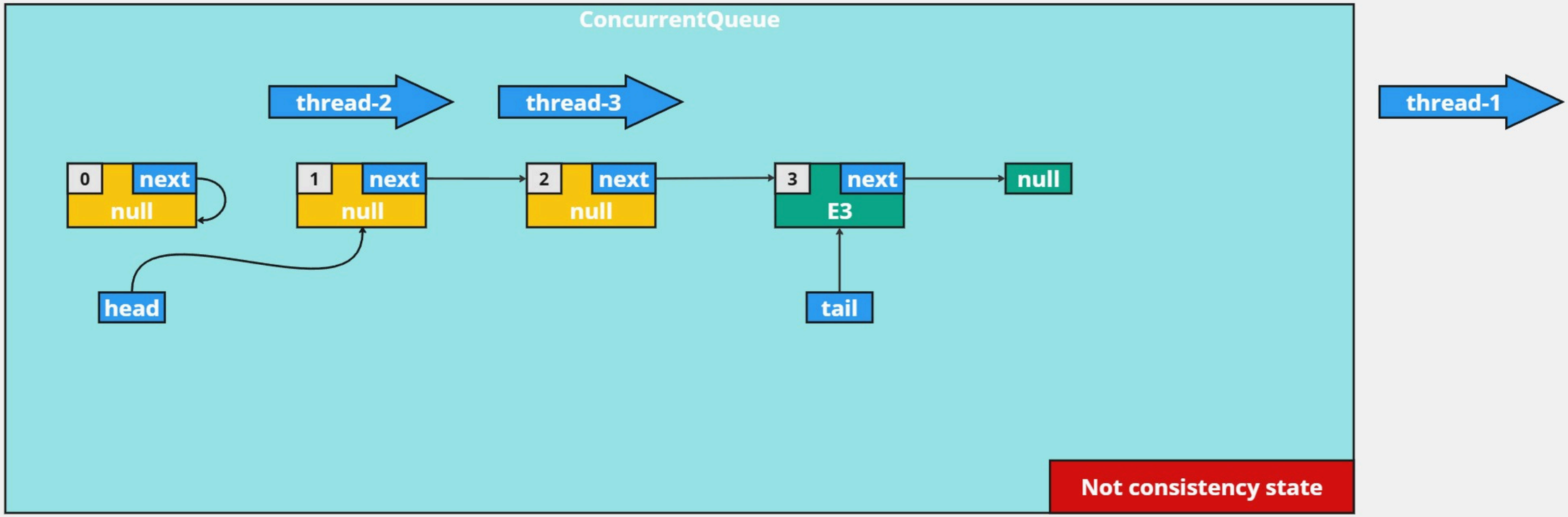
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 1
        final Node<E> nextHead = previousHead.next.get();     // 2
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E2
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 1
        final Node<E> nextHead = previousHead.next.get();     // 2
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E2
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

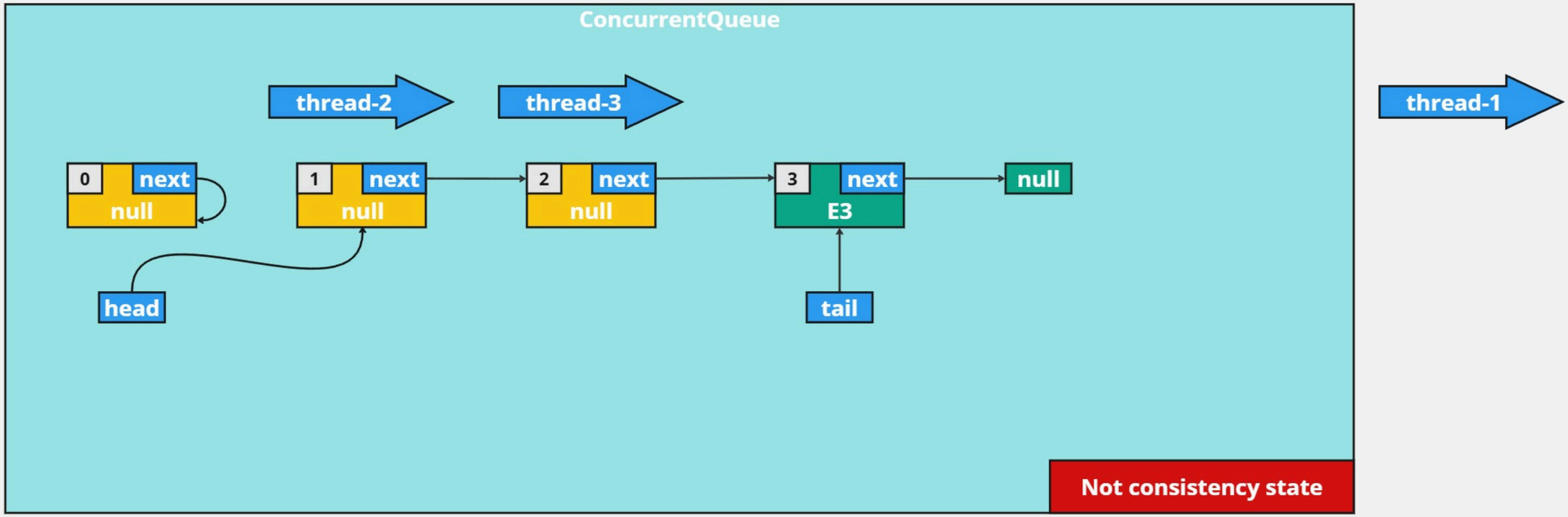
1

2

E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

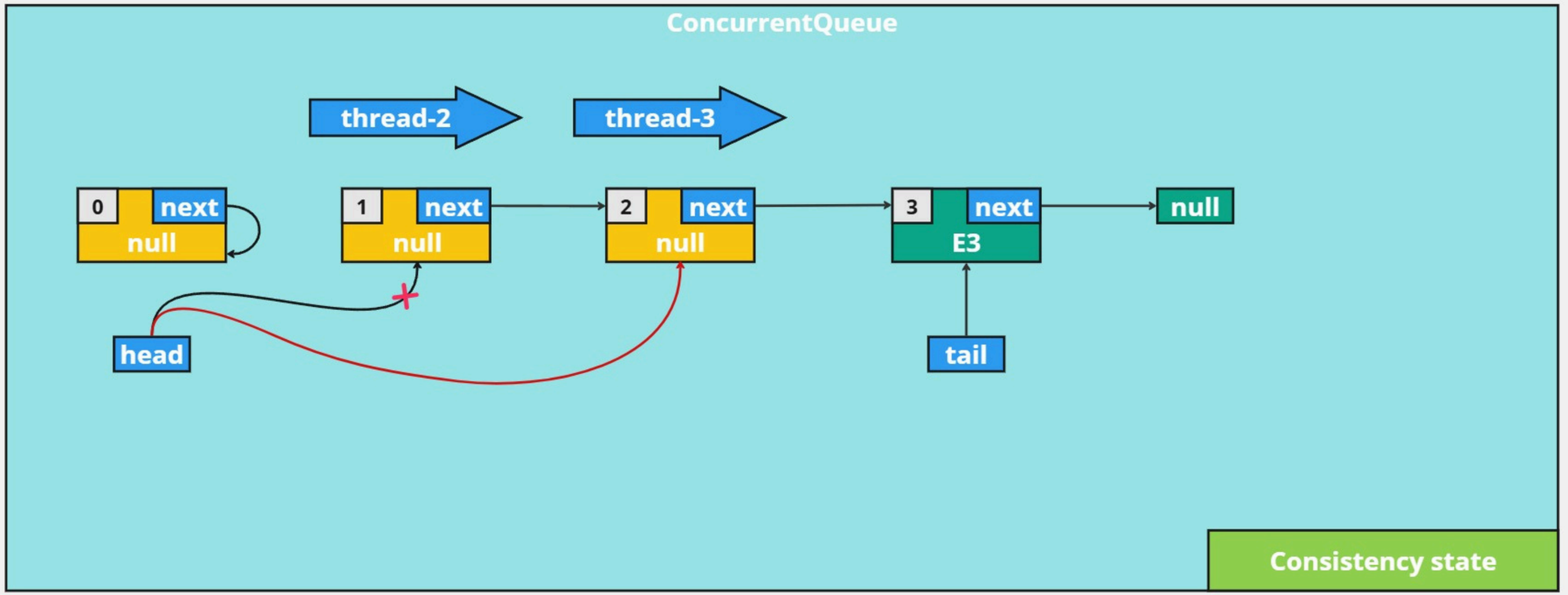
1

2

E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

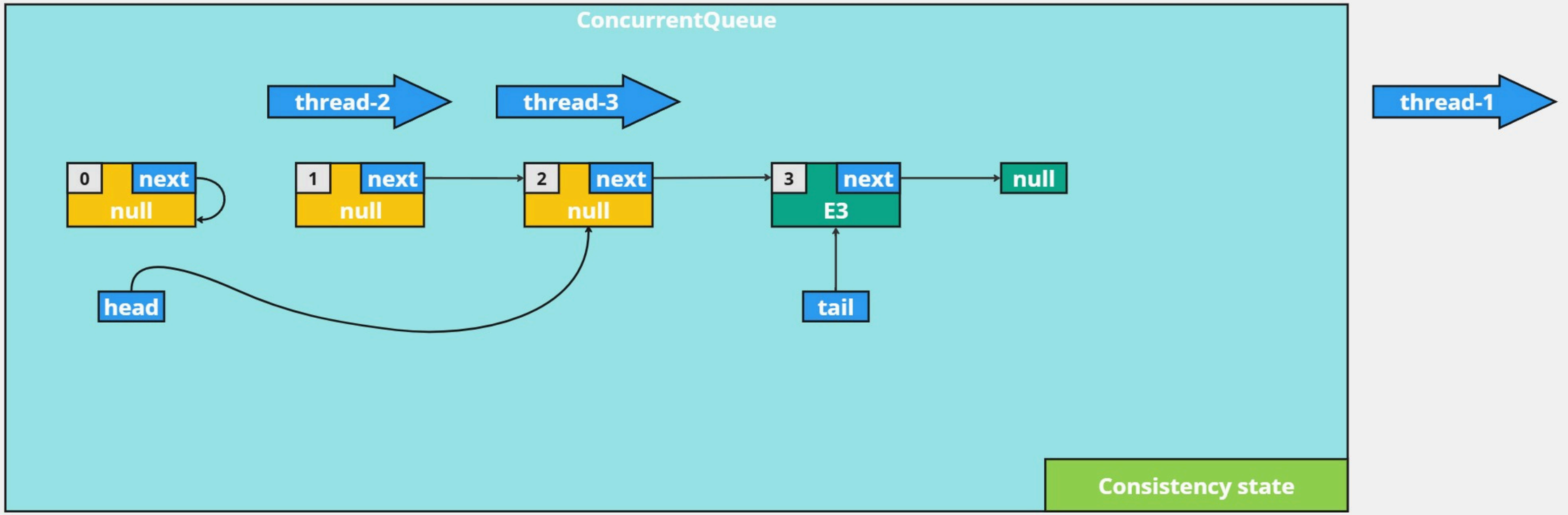
1

2

E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

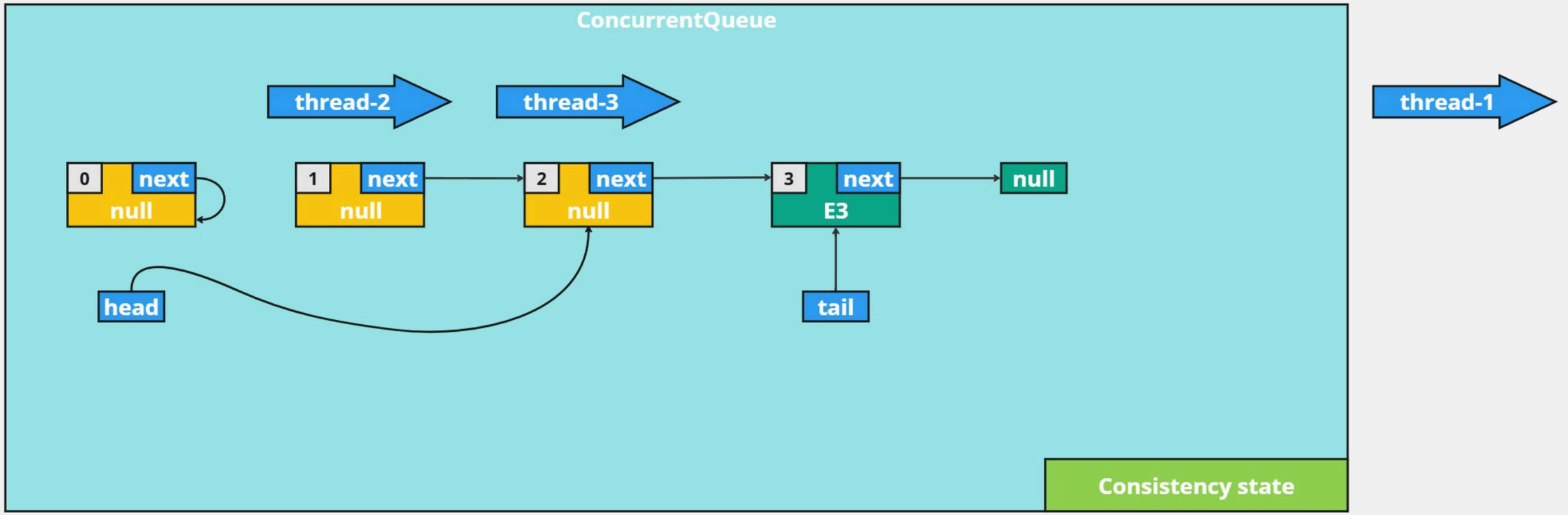
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

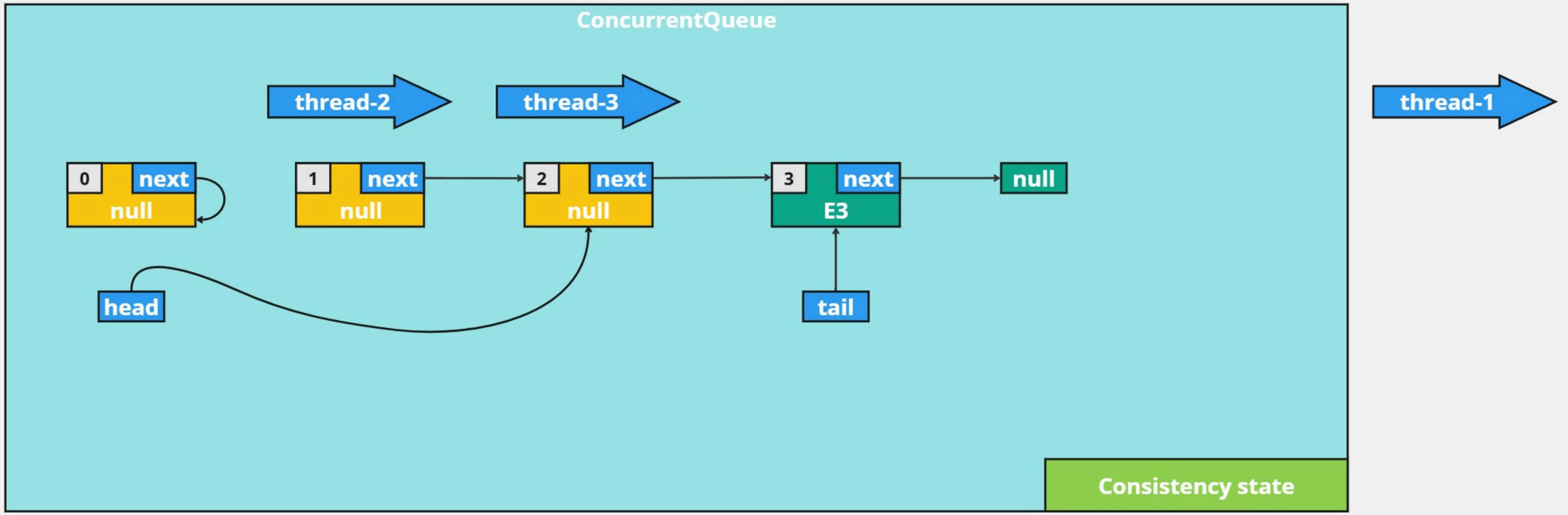
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

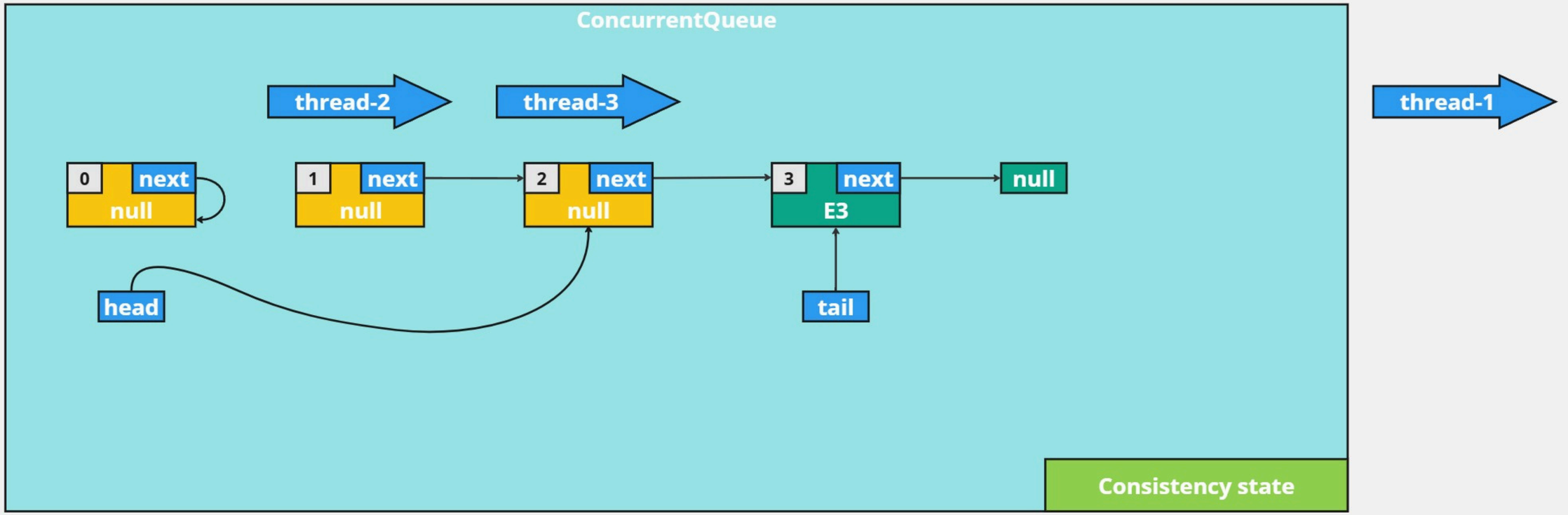
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

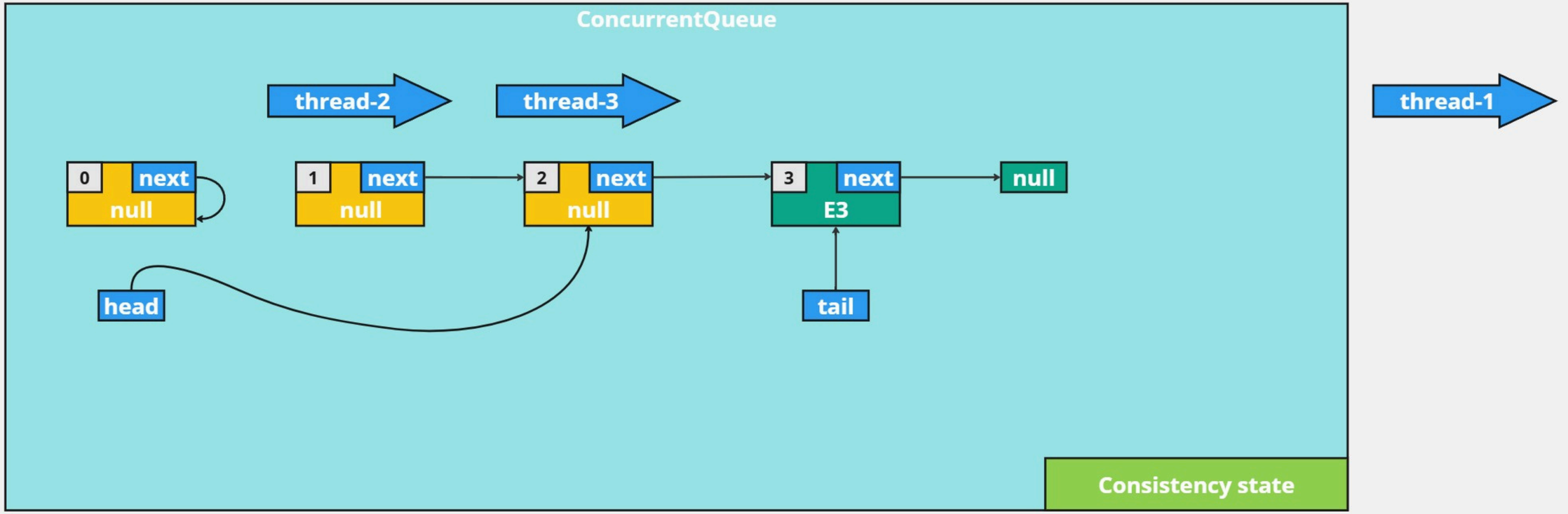
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3 E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

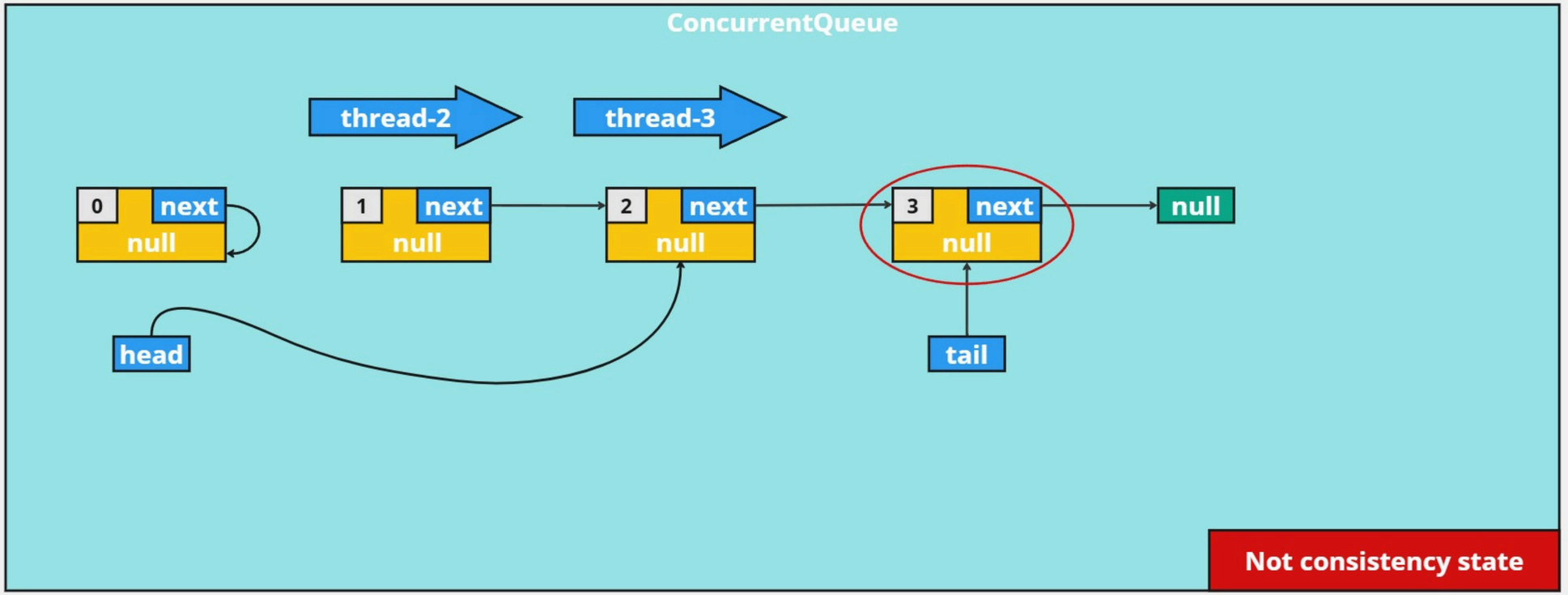
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3 E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

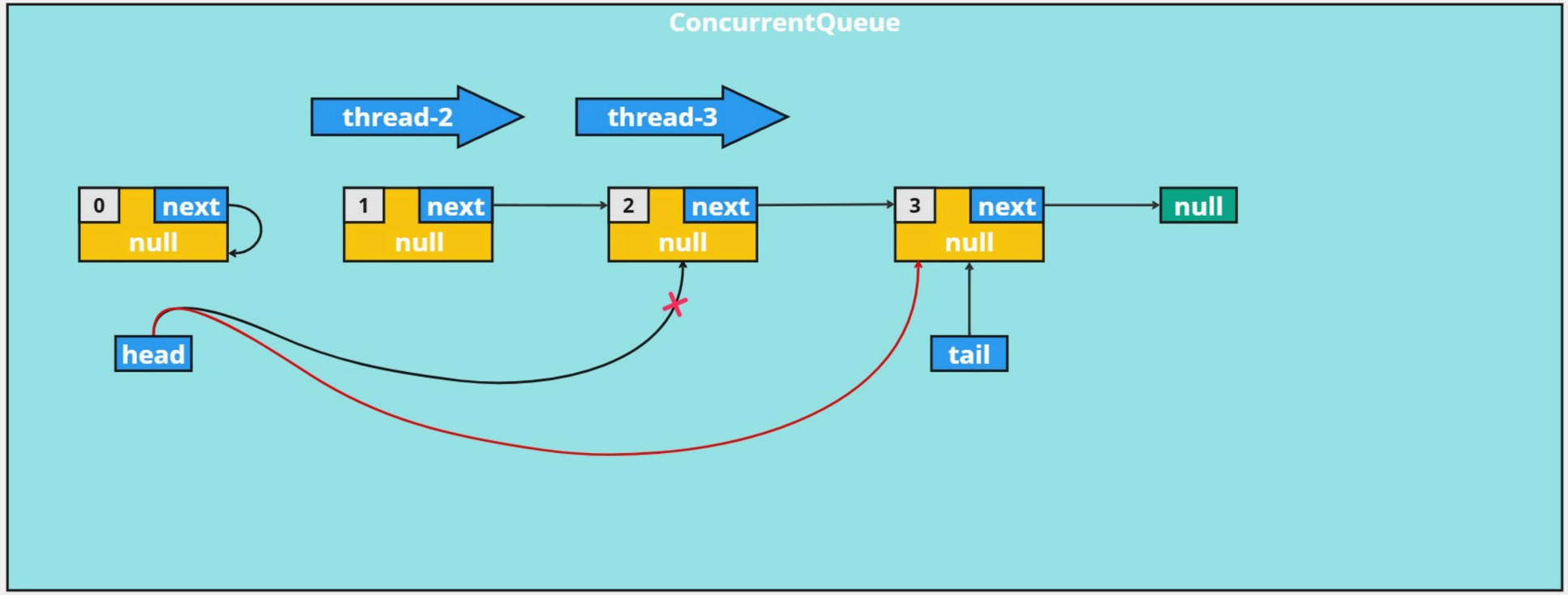
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3 E3



thread-1

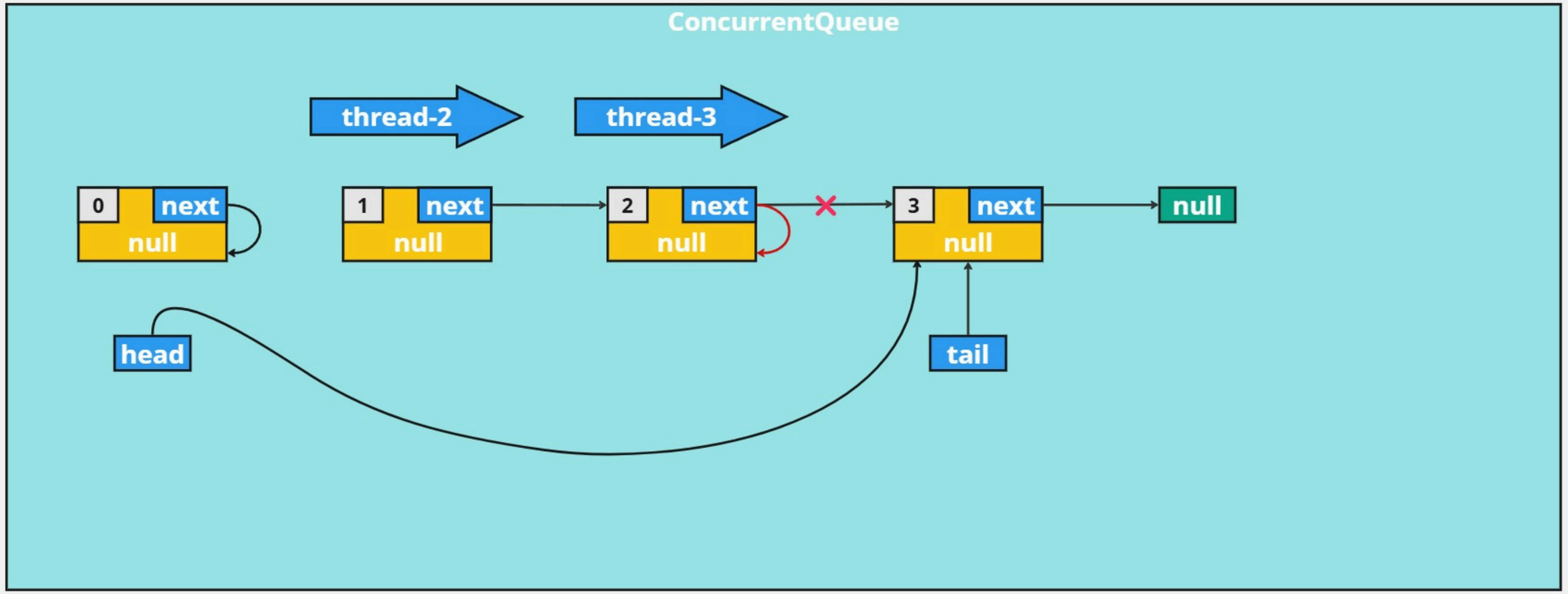
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 1
        final Node<E> nextHead = previousHead.next.get();     // 2
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E2
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;                    // 2
        final Node<E> nextHead = previousHead.next.get();     // 3
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();               // E3
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

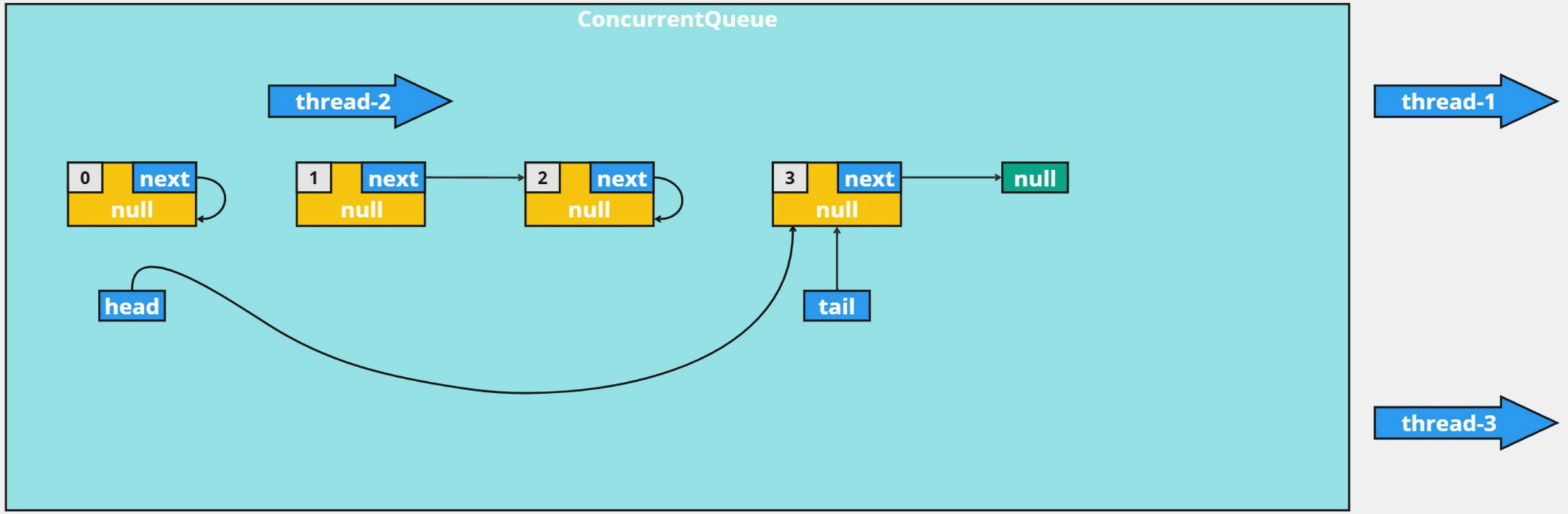
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3 E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

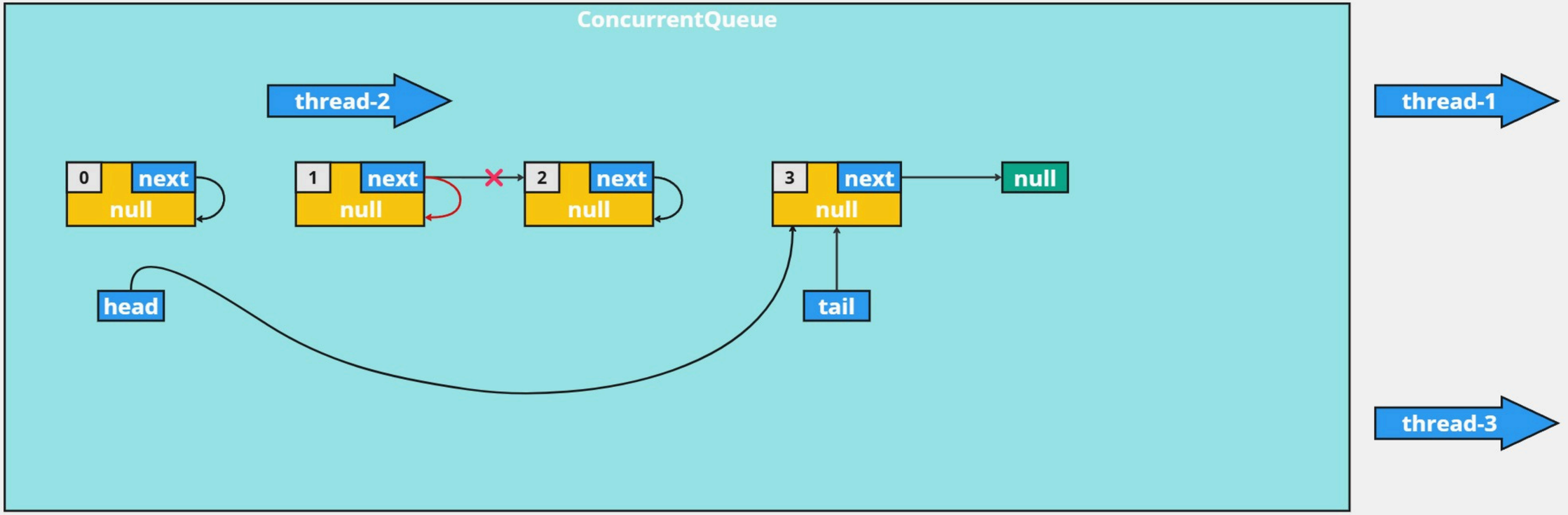
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

2 3 E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

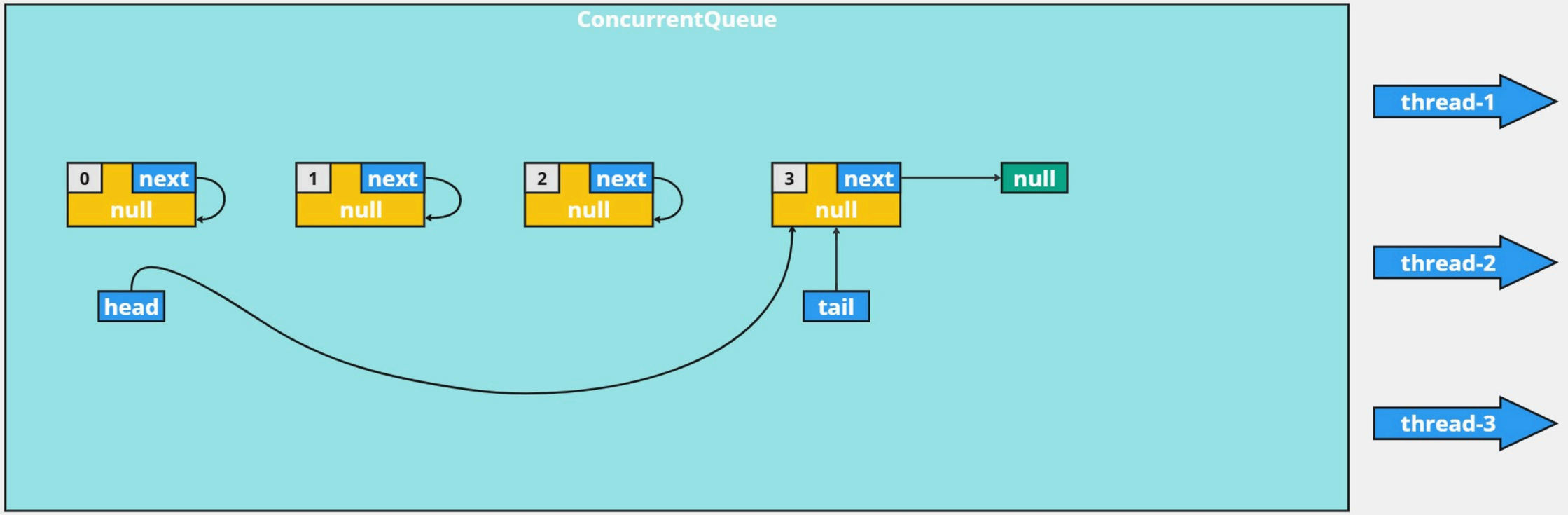
thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

1 2 E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

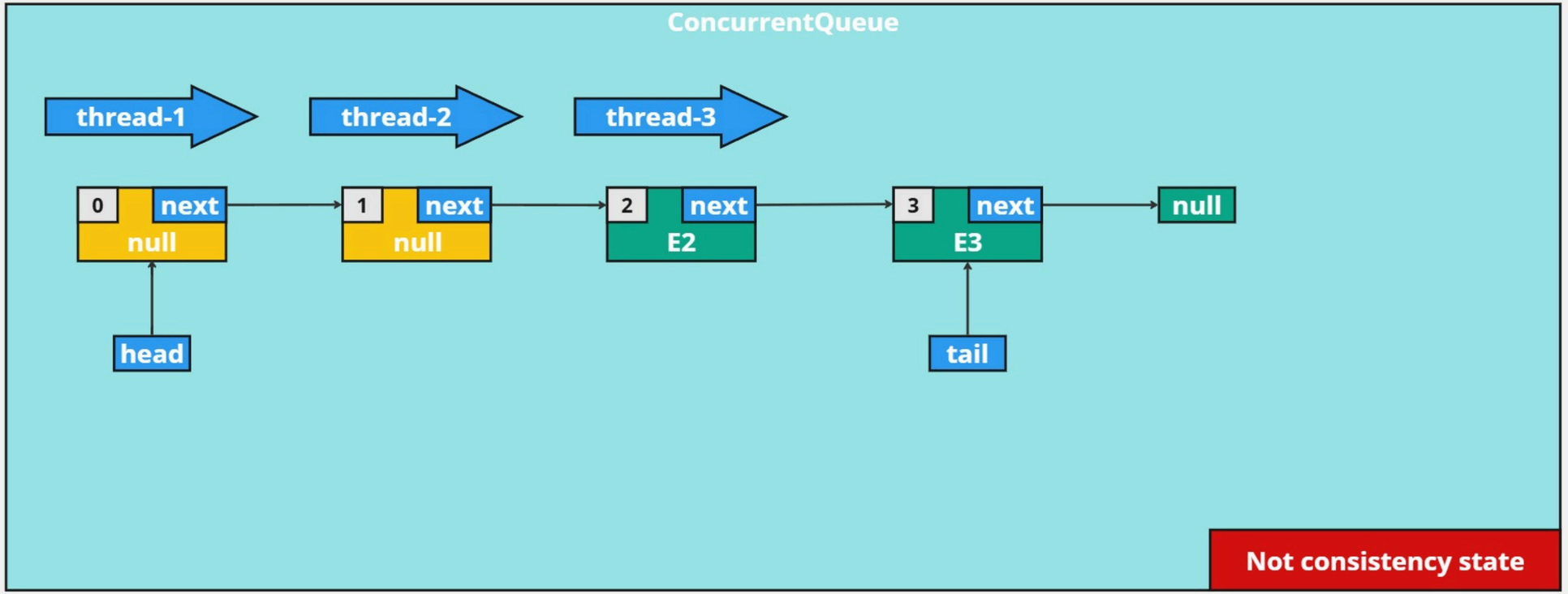
1

2

E2

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

0

1

E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```


```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```
private volatile Node<E> head;
```

```
public ConcurrentQueue() {
    final Node<E> dummyNode = new Node<>();
    head = dummyNode;
    tail = new AtomicReference<>(dummyNode);
}
```

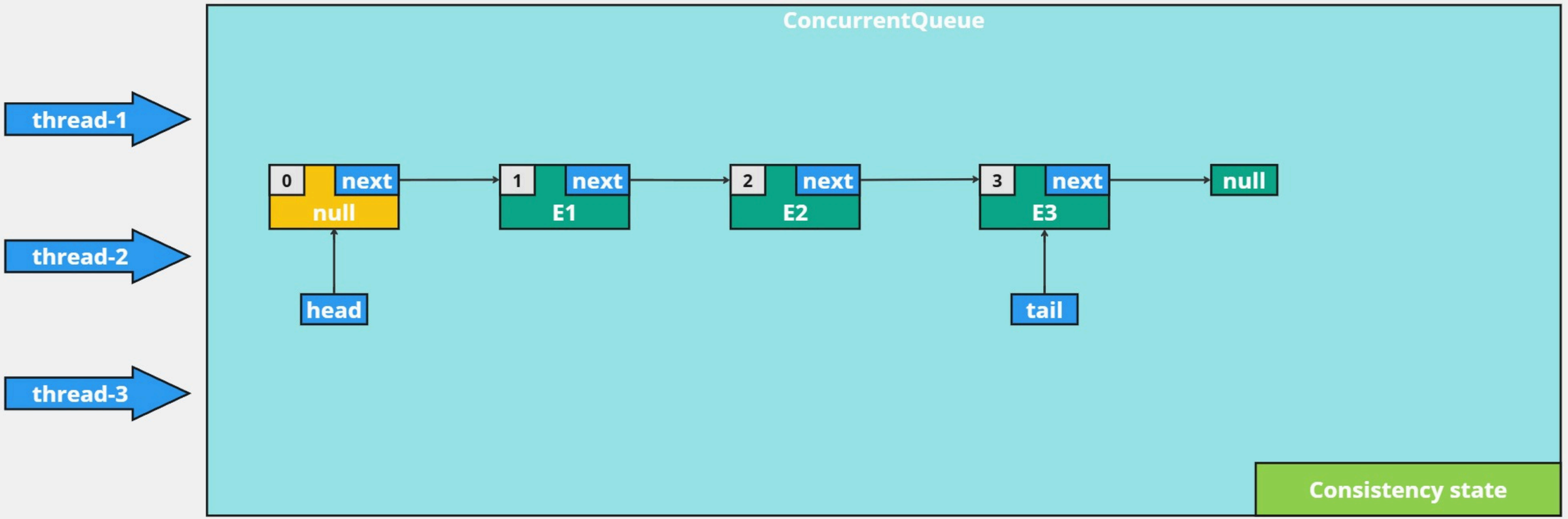
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head;
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            head = nextHead;
            previousHead.next.set(previousHead);
            return of(element);
        }
    }
}
```

```
private final AtomicReference<Node<E>> head;
```

```
public ConcurrentQueue() {
    final Node<E> dummyNode = new Node<>();
    head = new AtomicReference<>(dummyNode);
    tail = new AtomicReference<>(dummyNode);
}
```

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

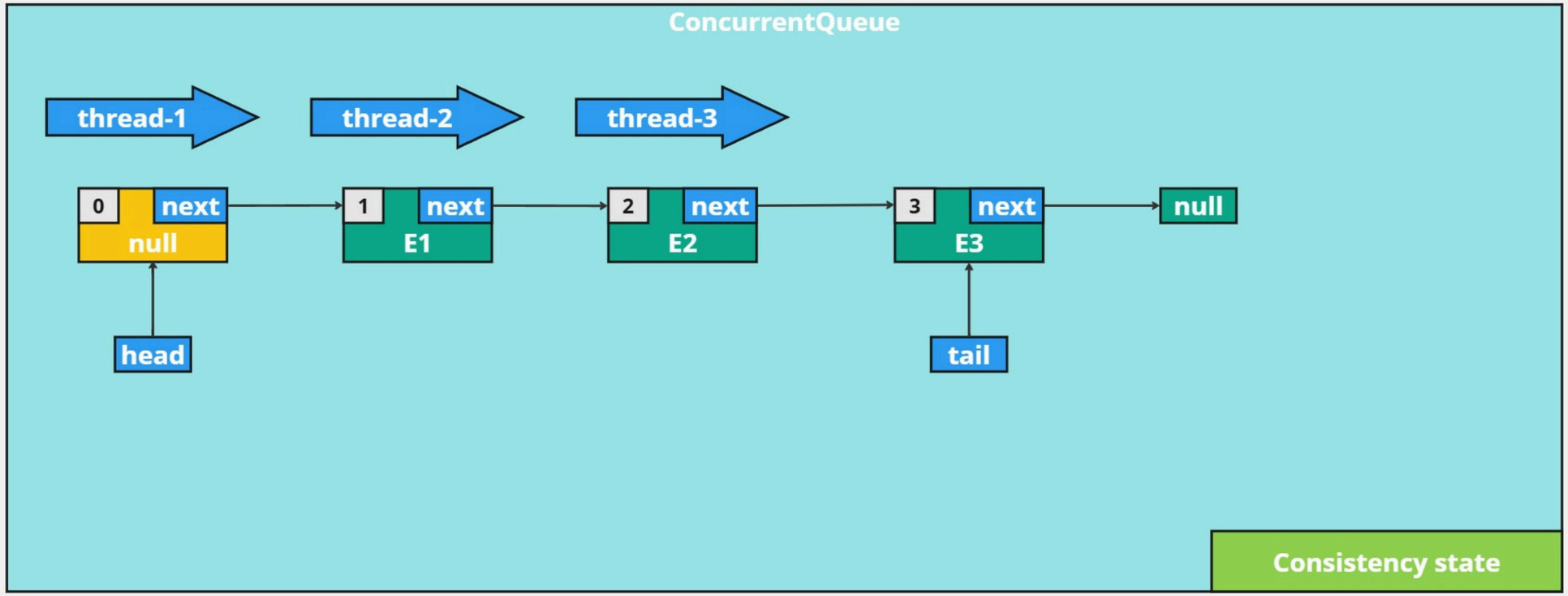
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

thread-2

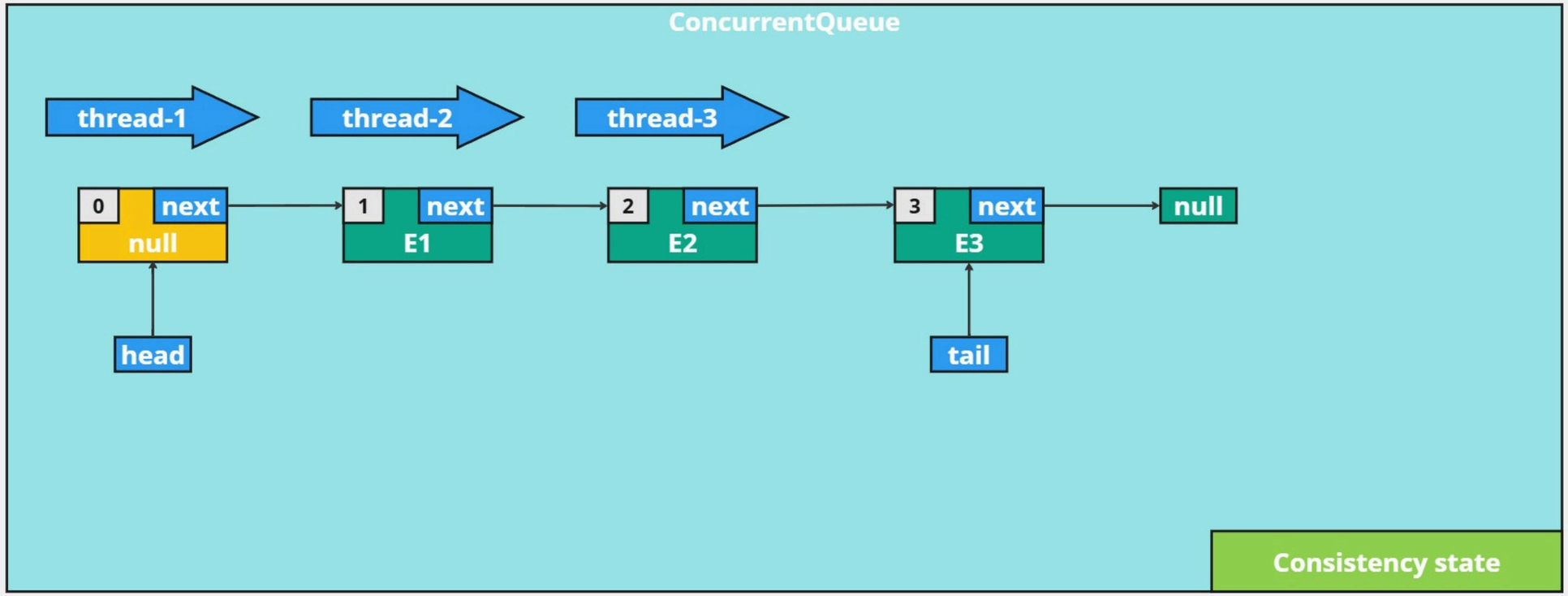
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

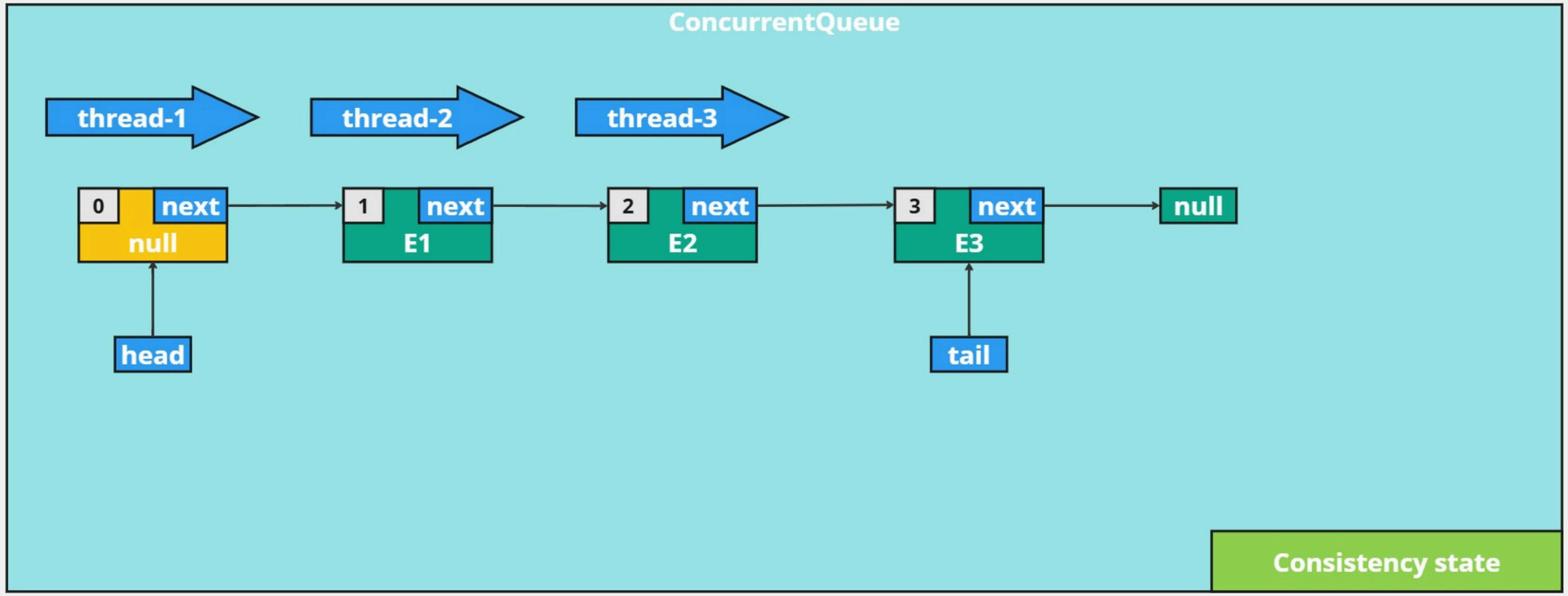
1

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

1



thread-1

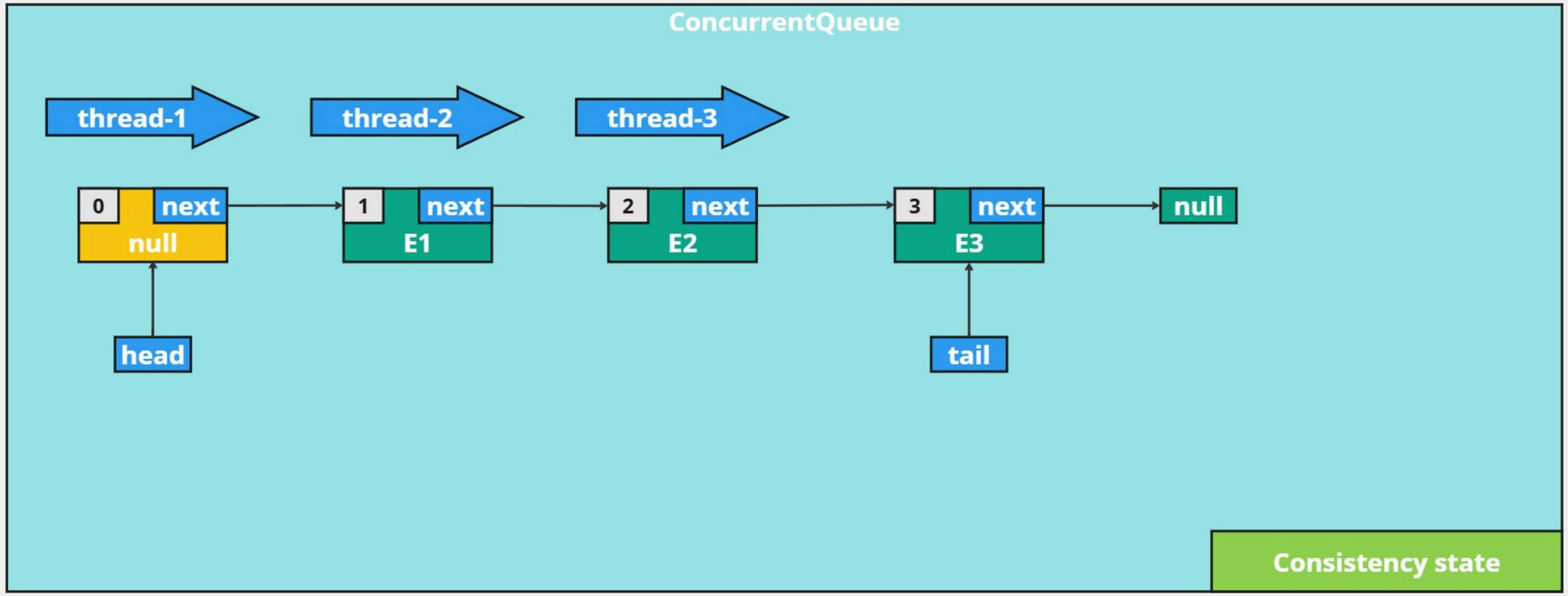
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

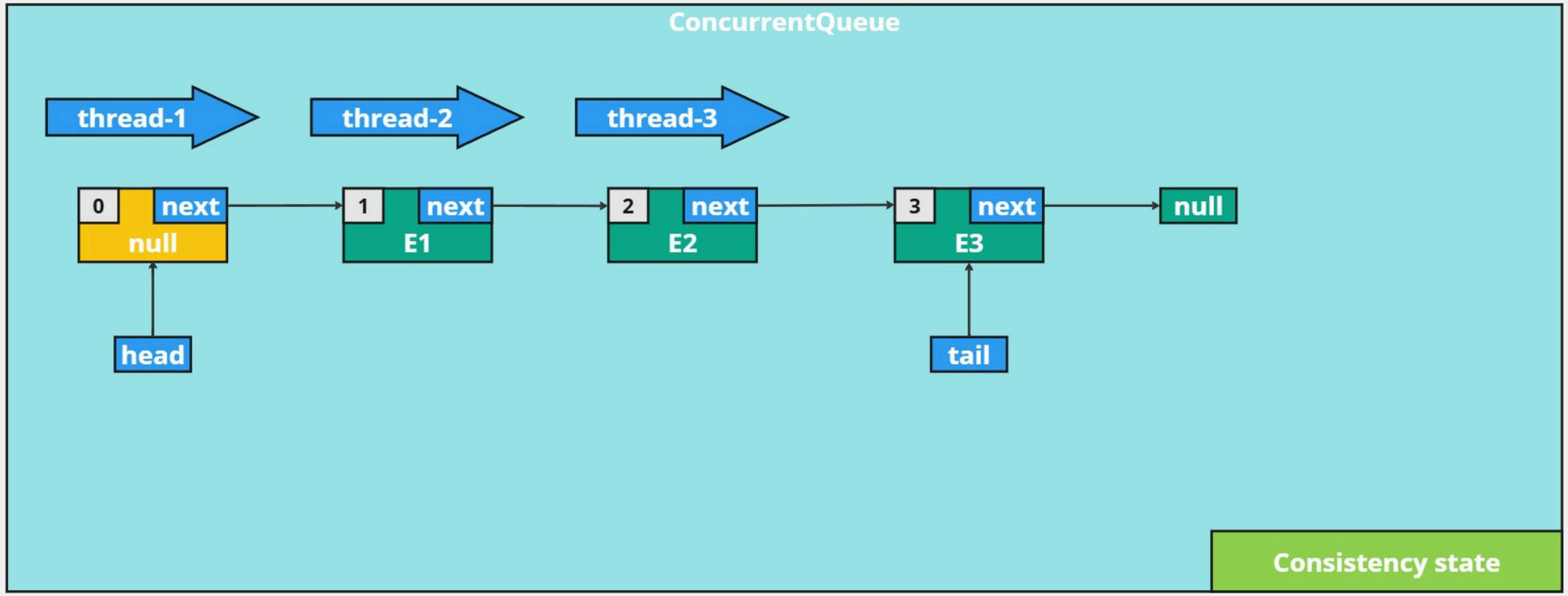
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

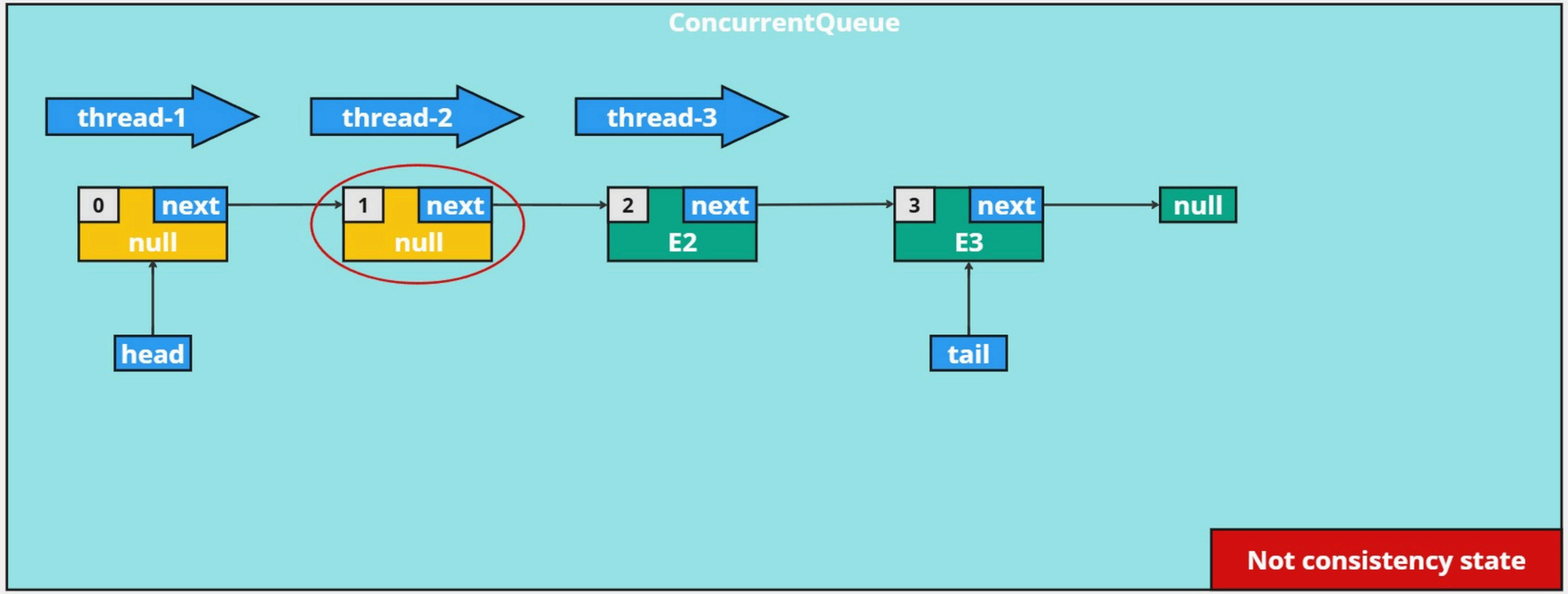
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

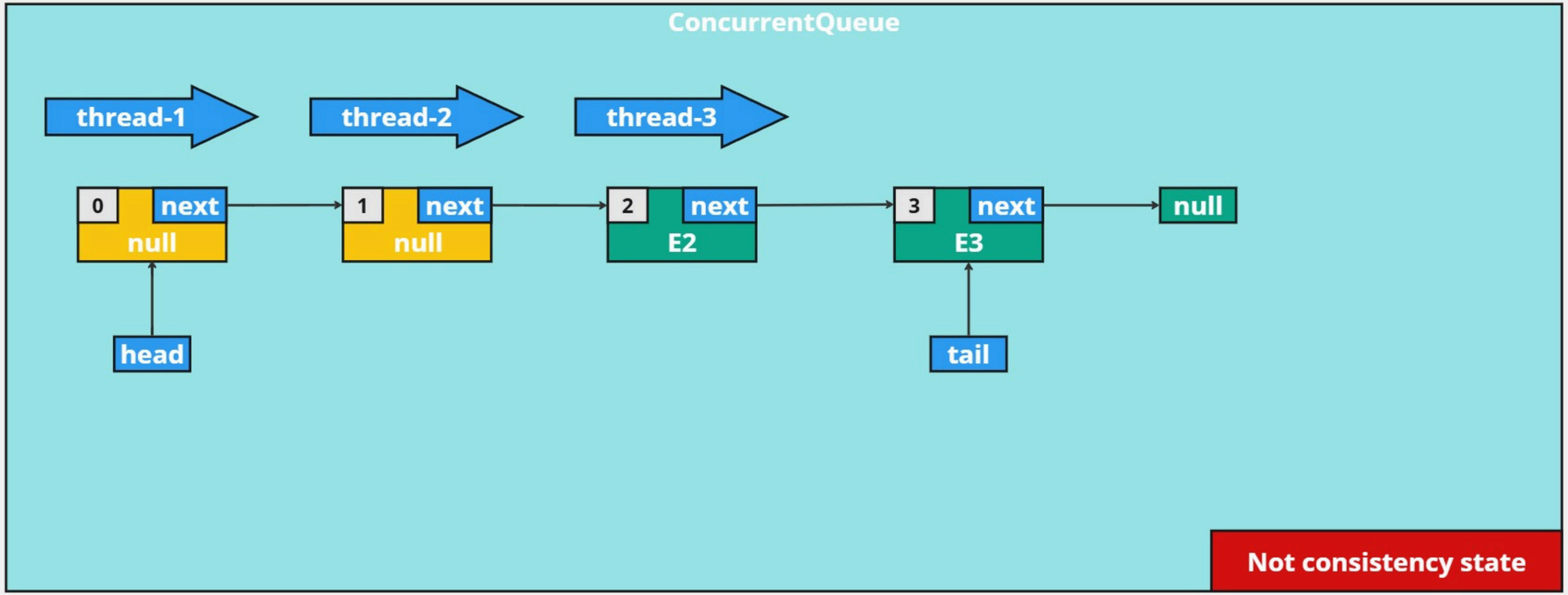
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

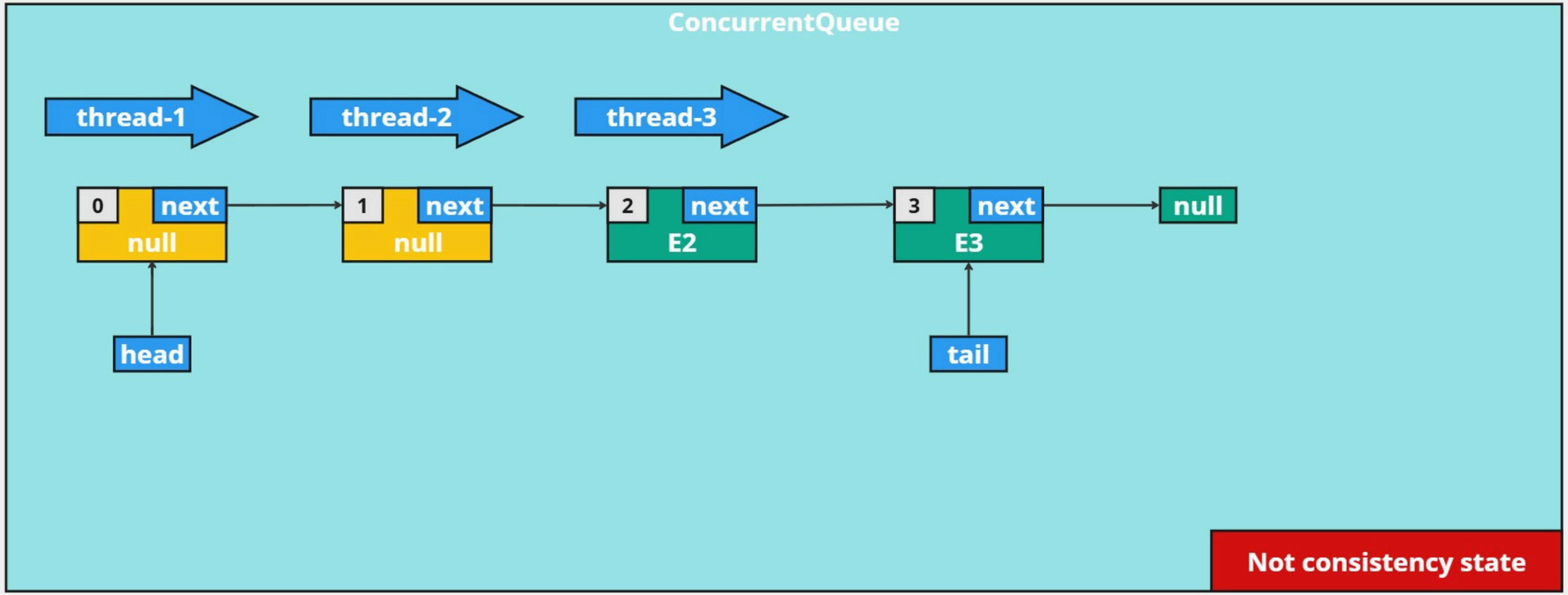
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

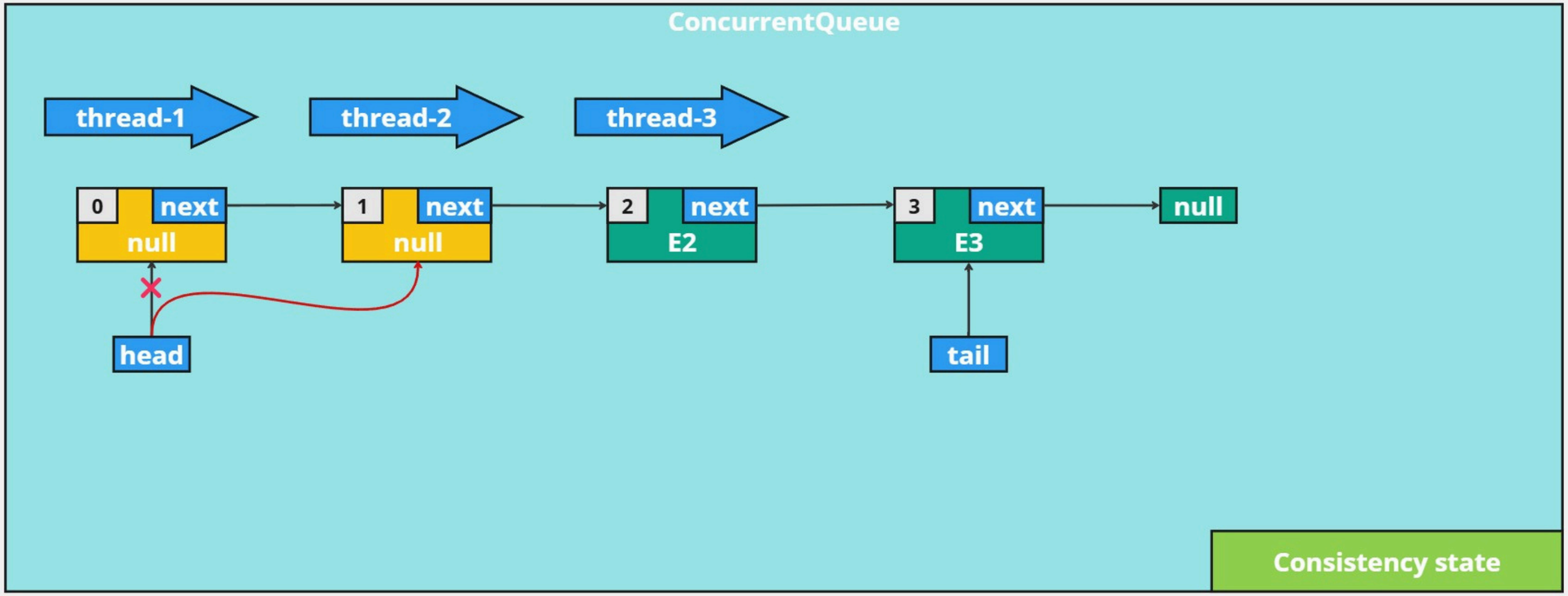
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

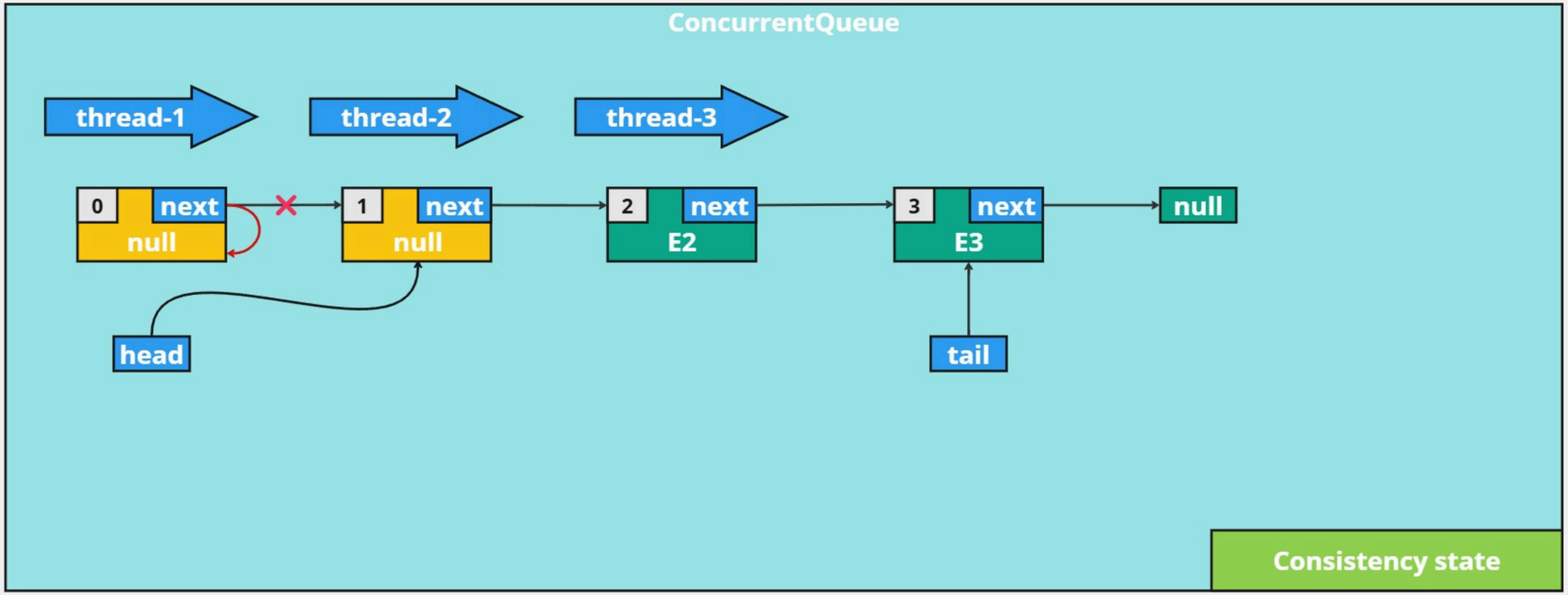
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

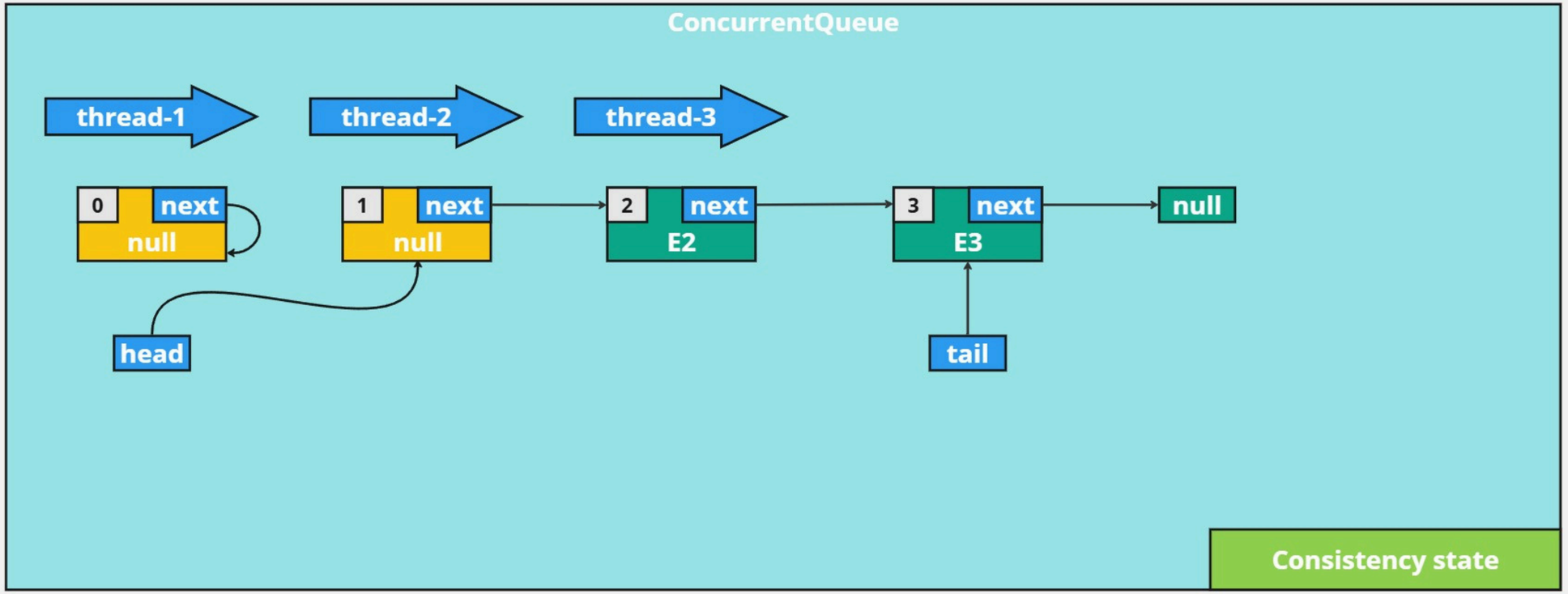
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

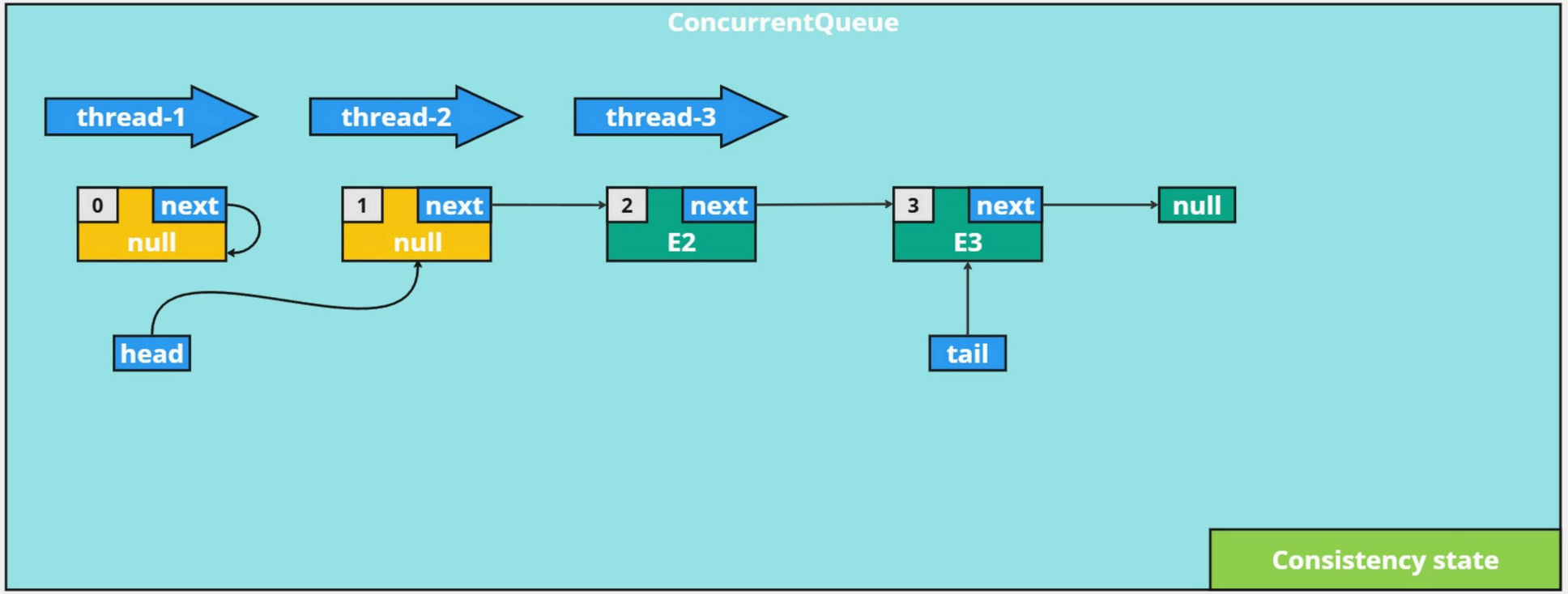
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

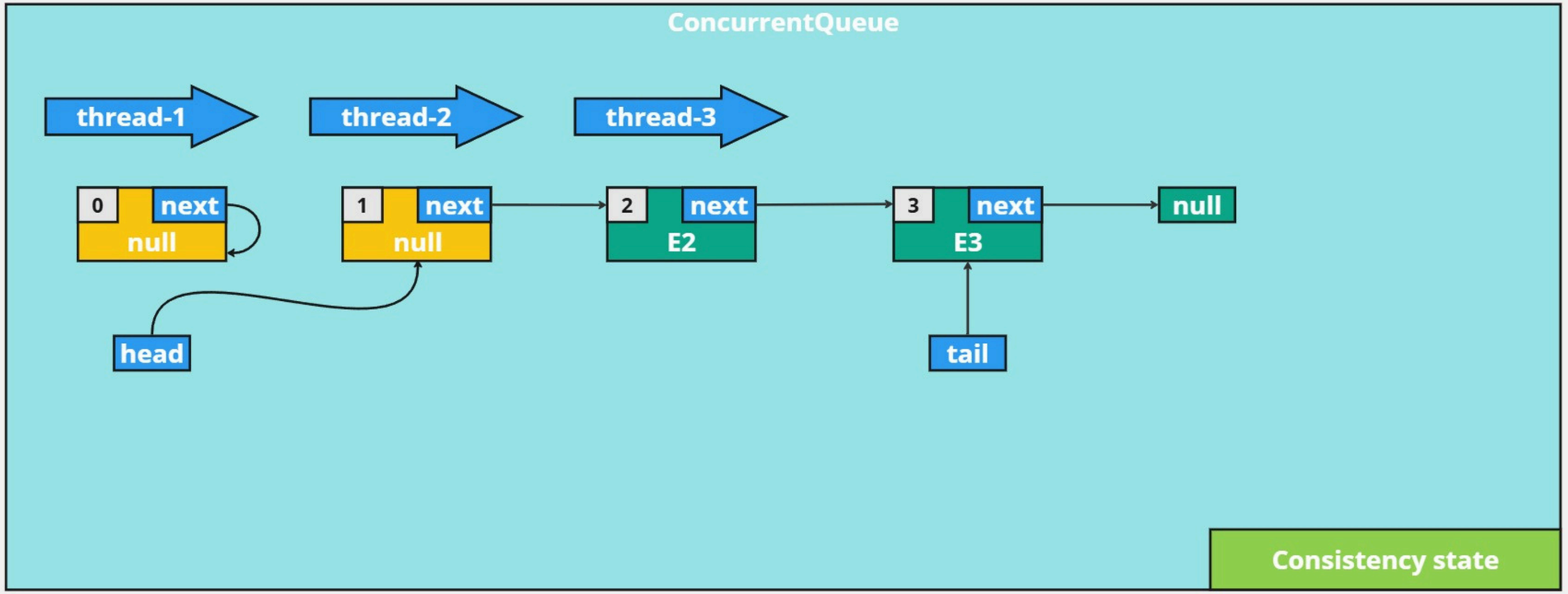
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

1

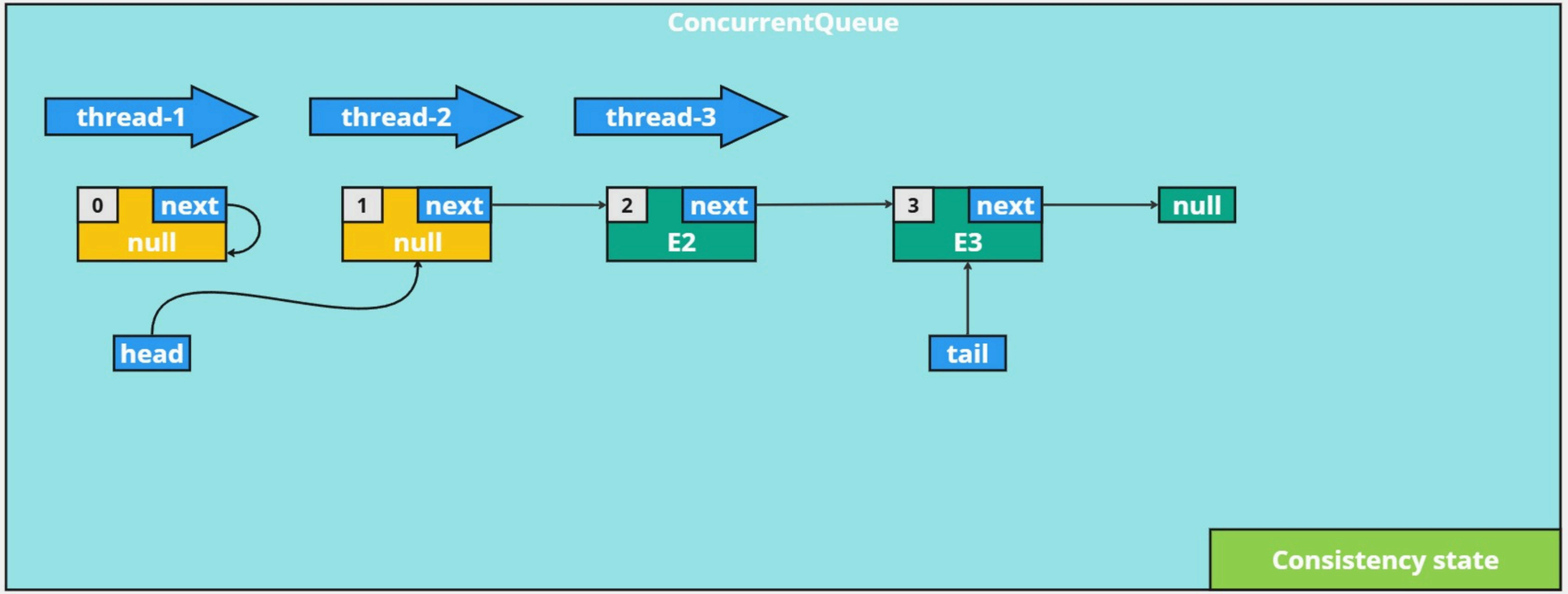
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

1

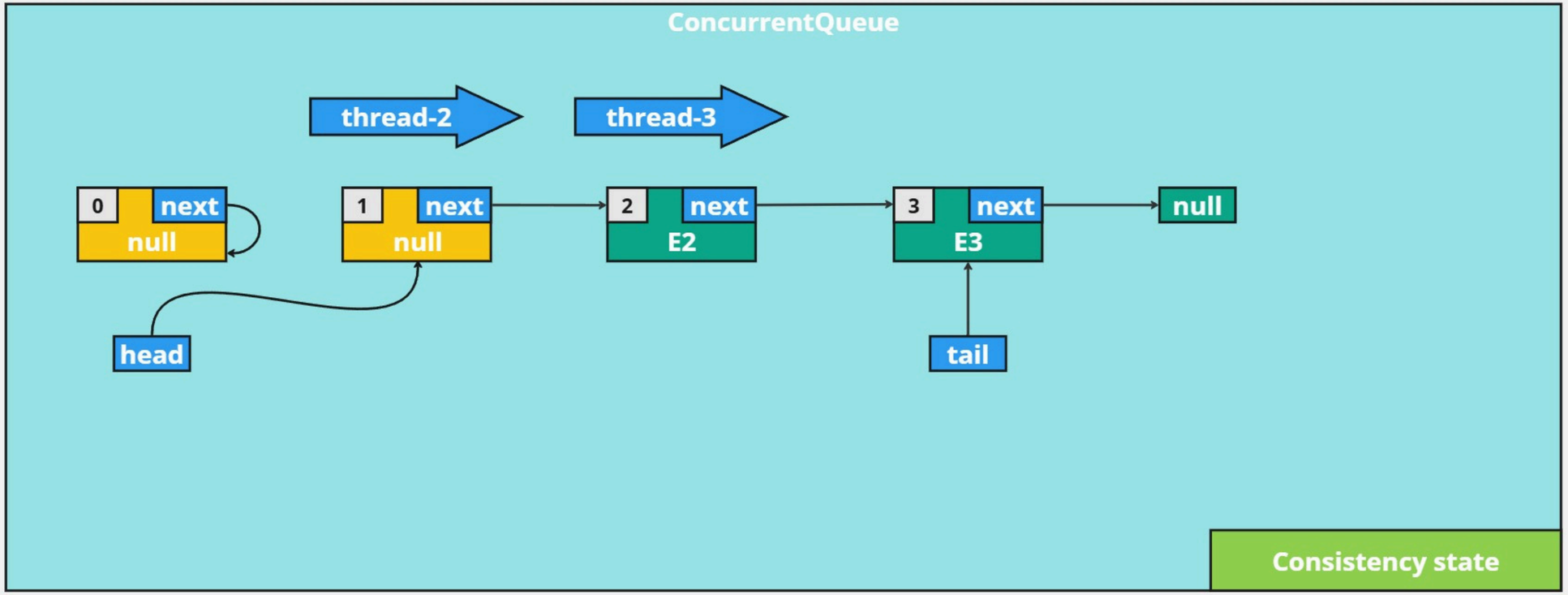
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

0

1

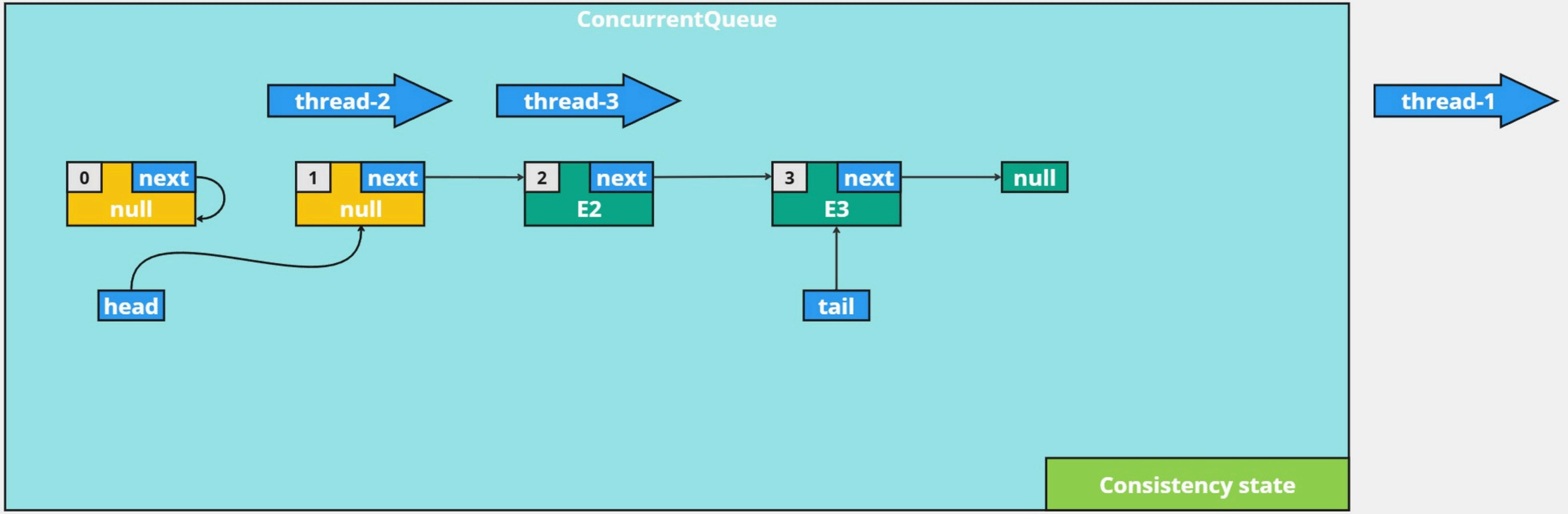
E1

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

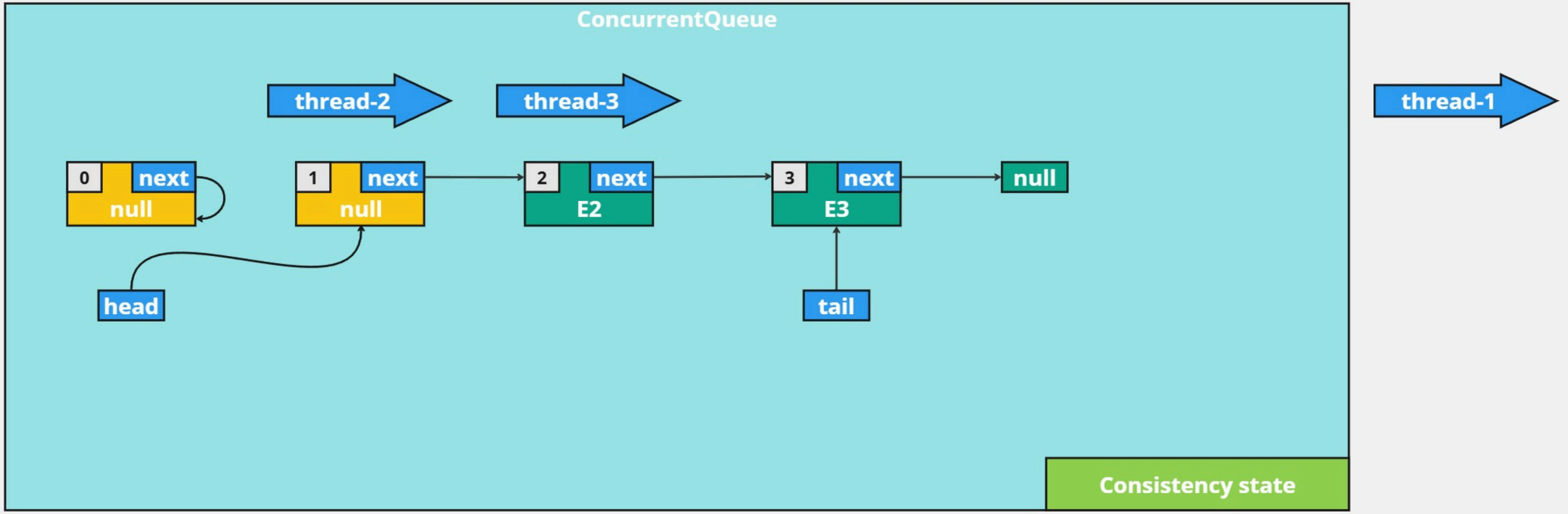
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

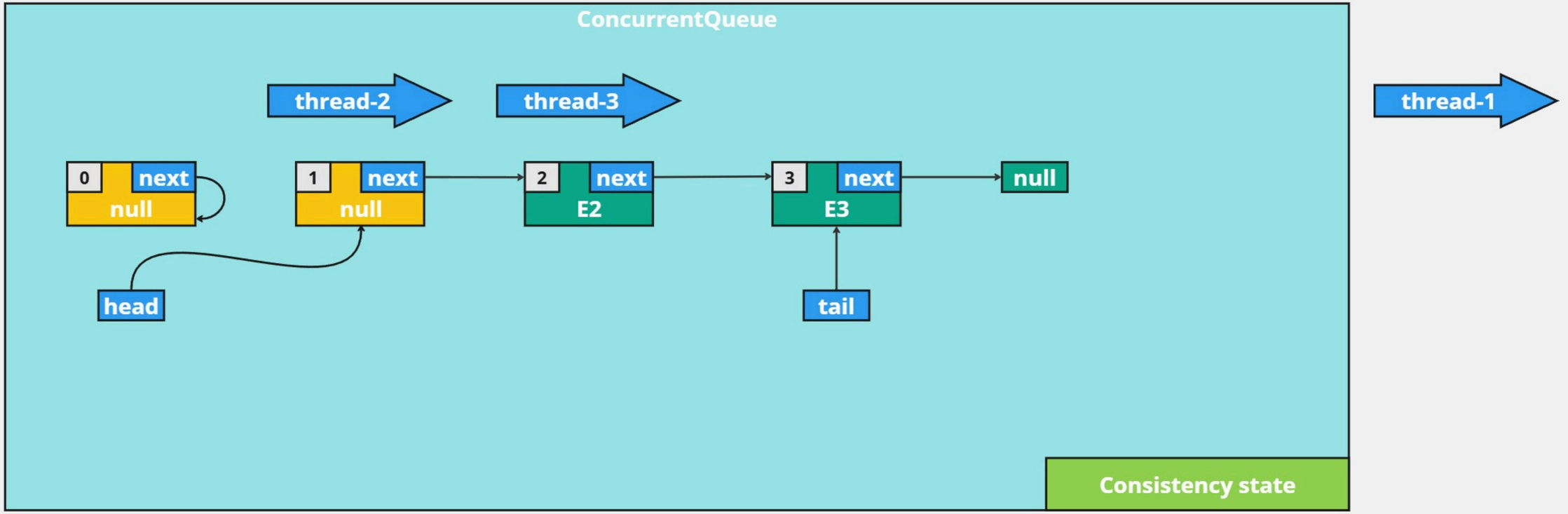
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

1

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

1



thread-1

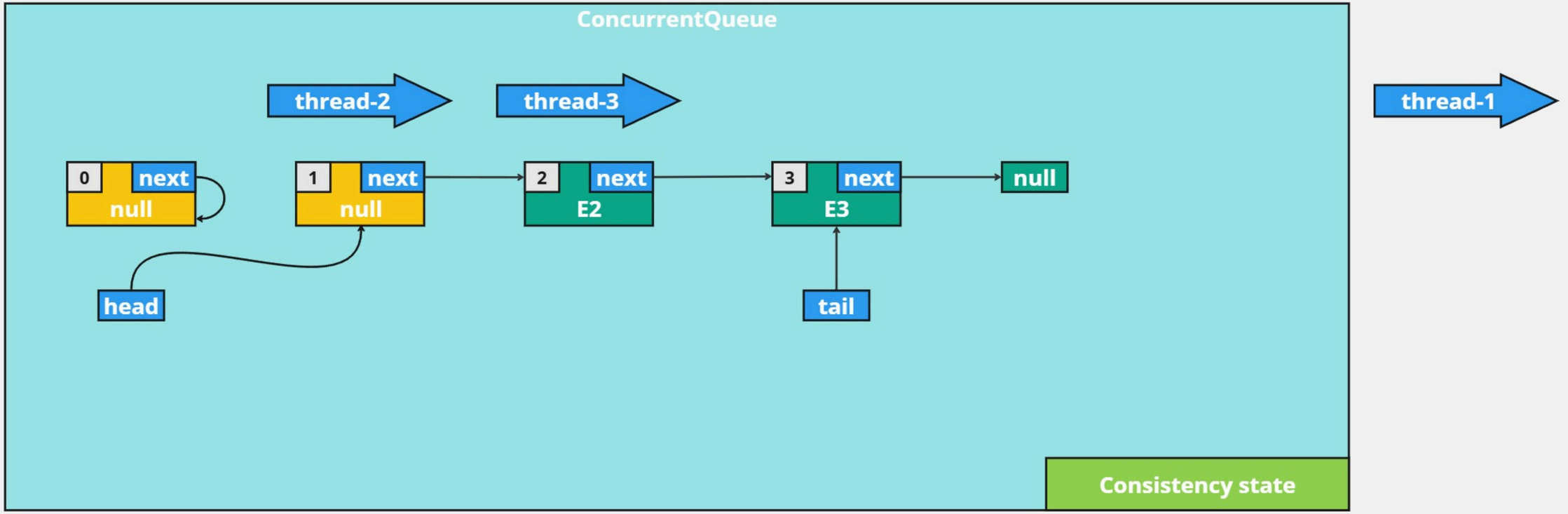
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

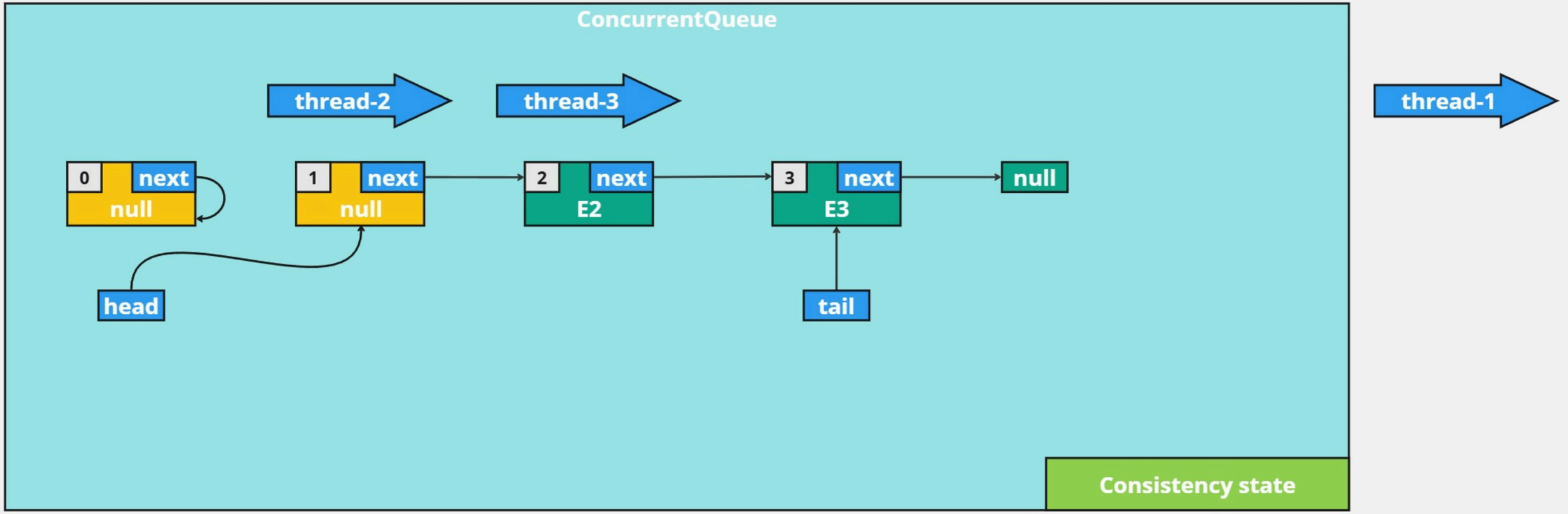
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

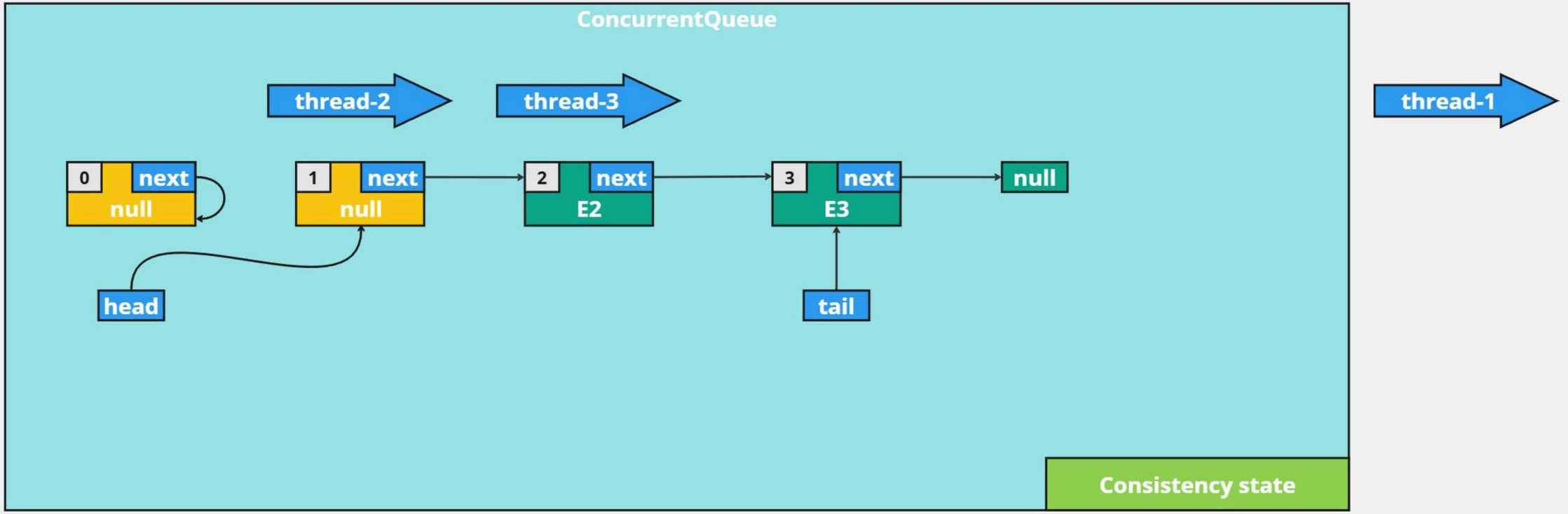
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

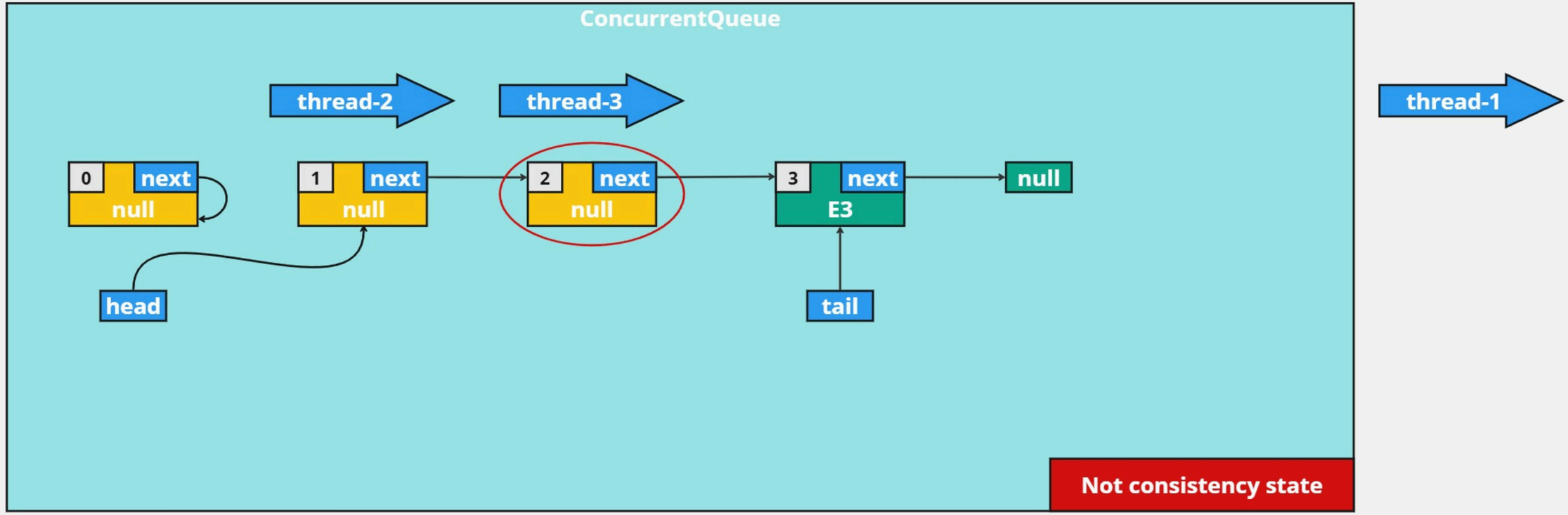
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

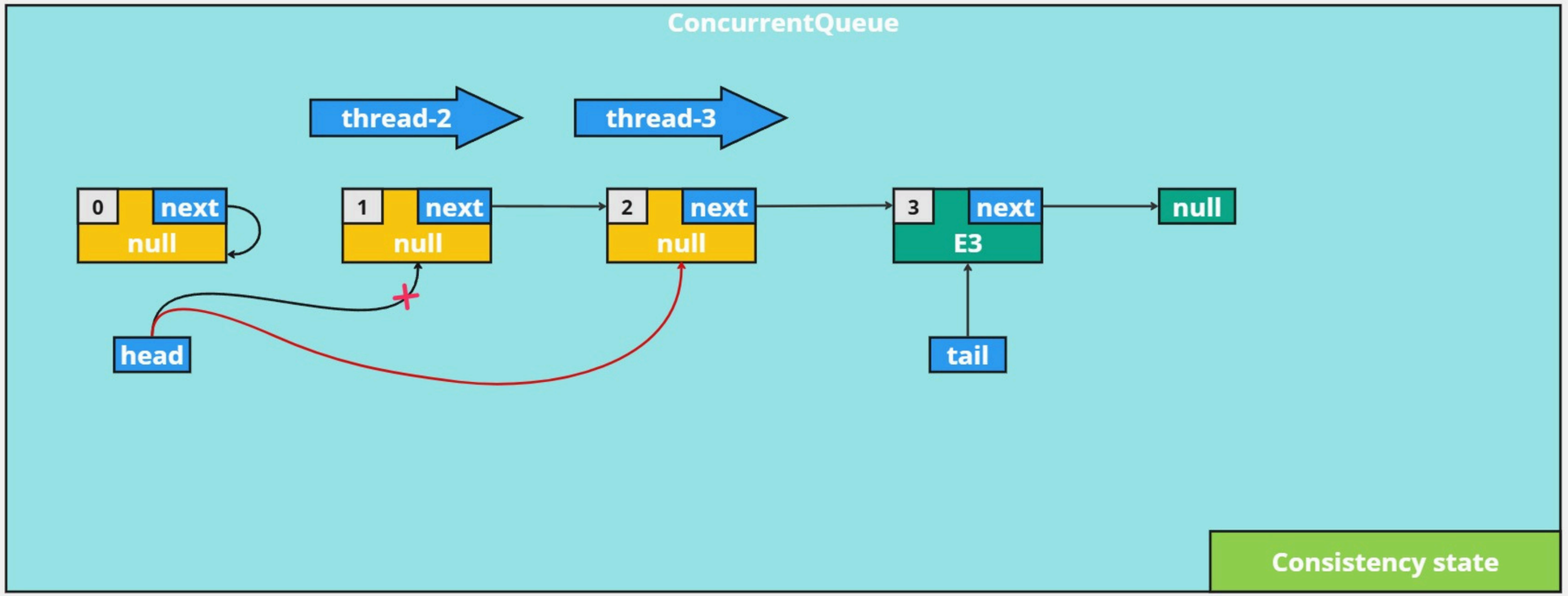
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

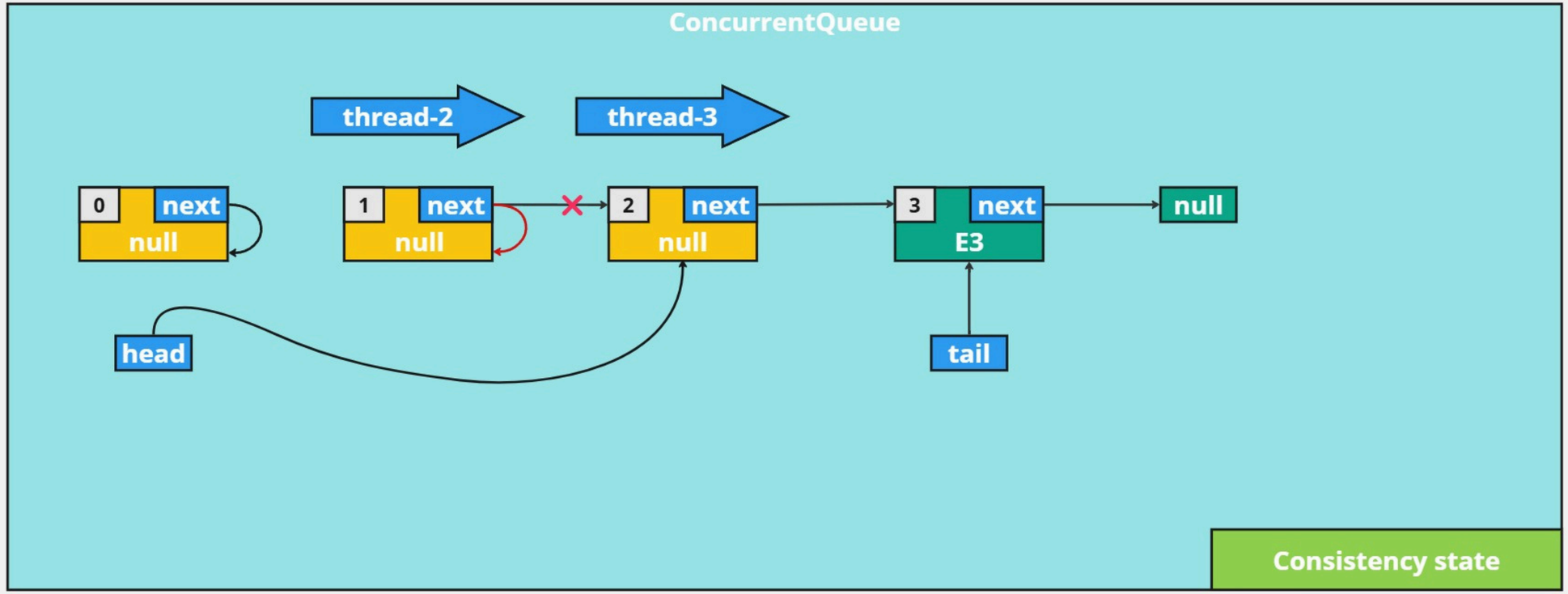
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

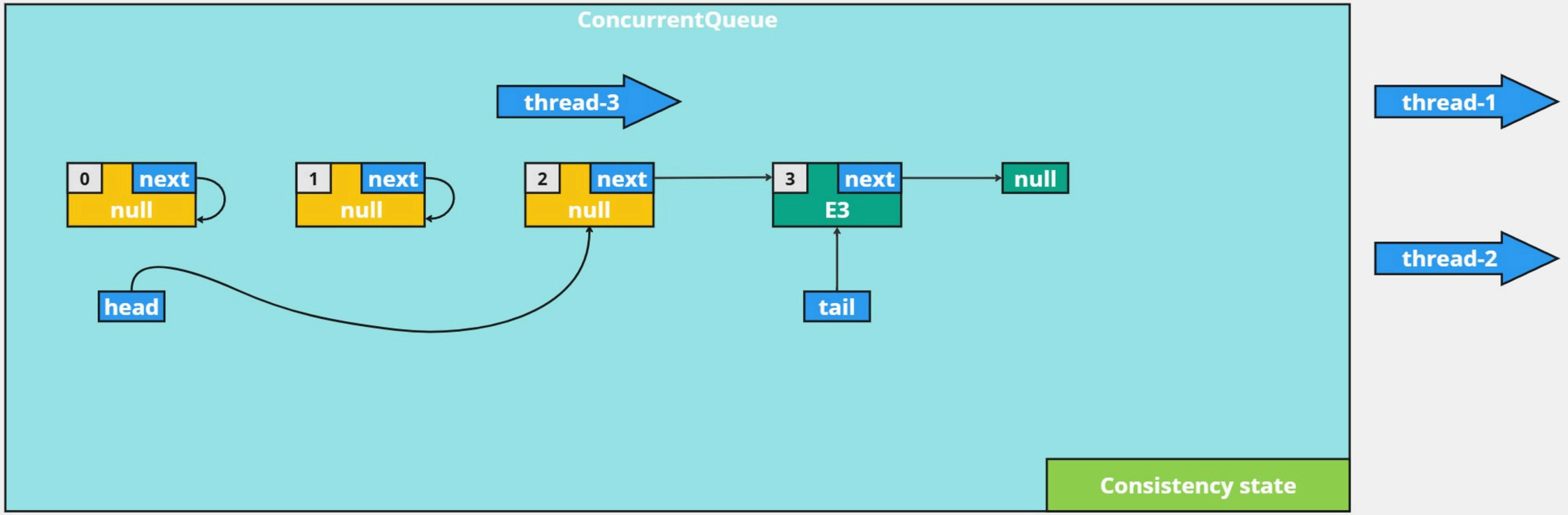
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

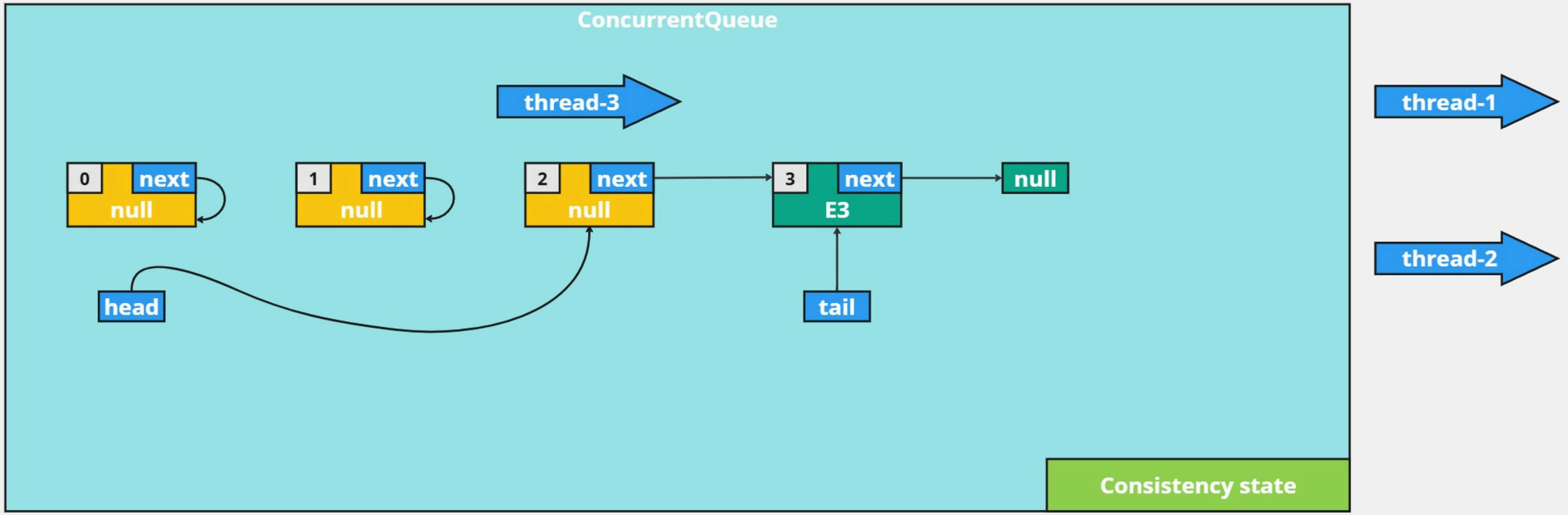
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

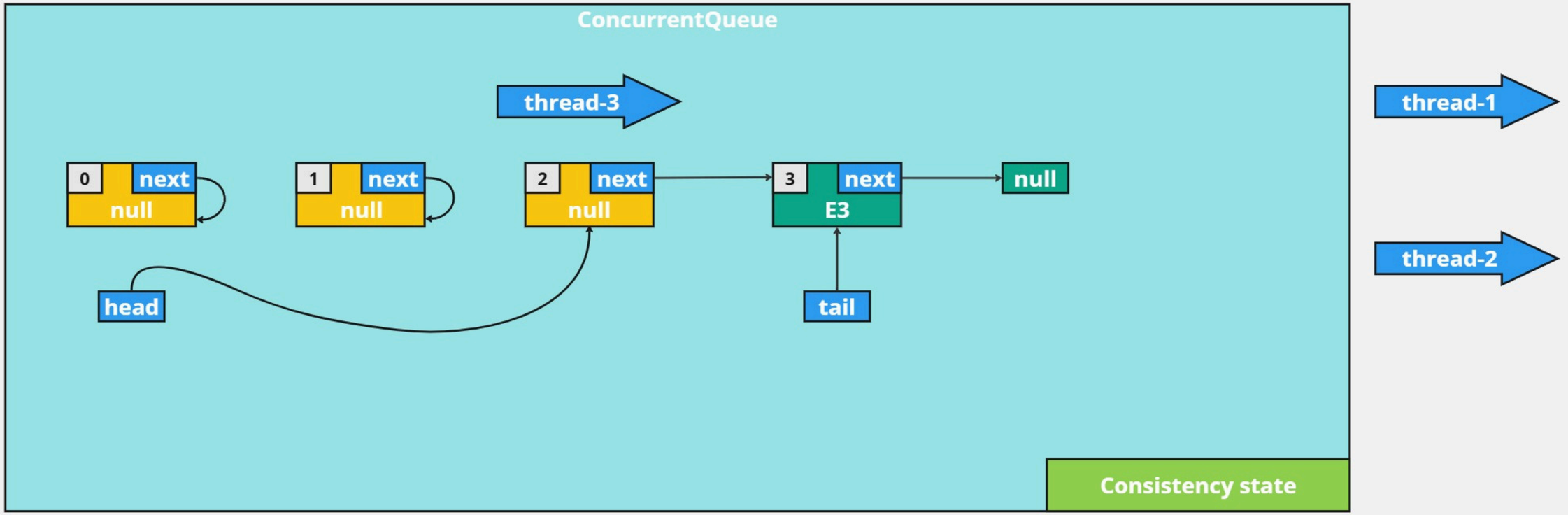
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

1

2

E2



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

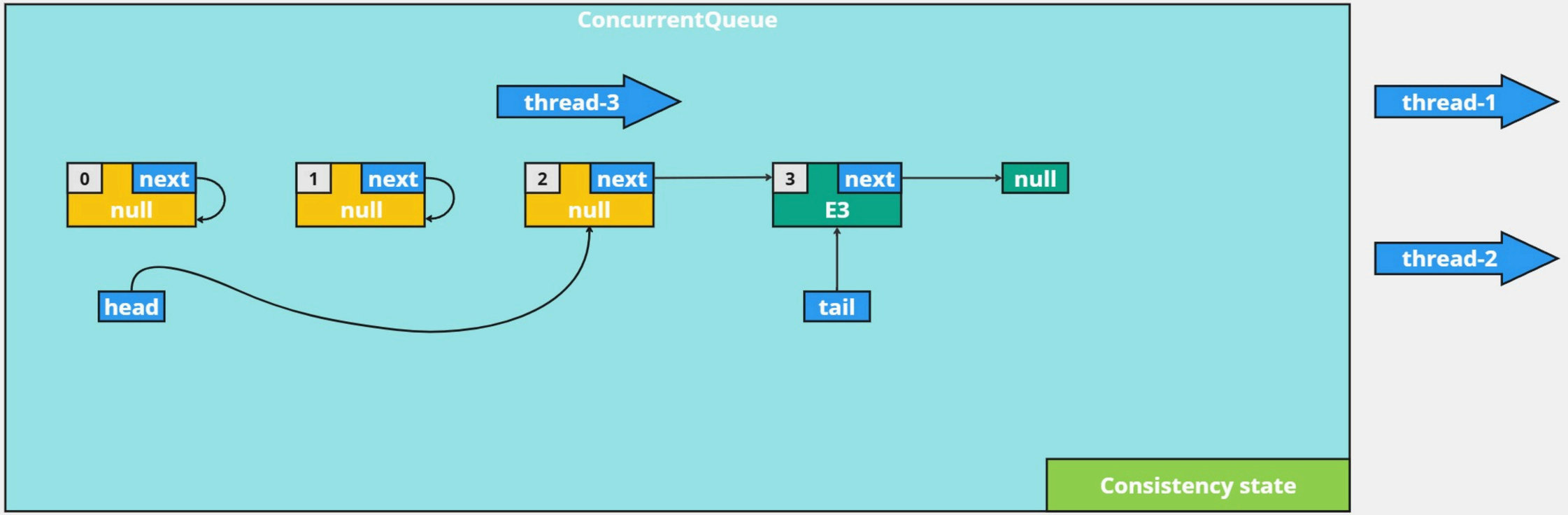
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

1

2

E2



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

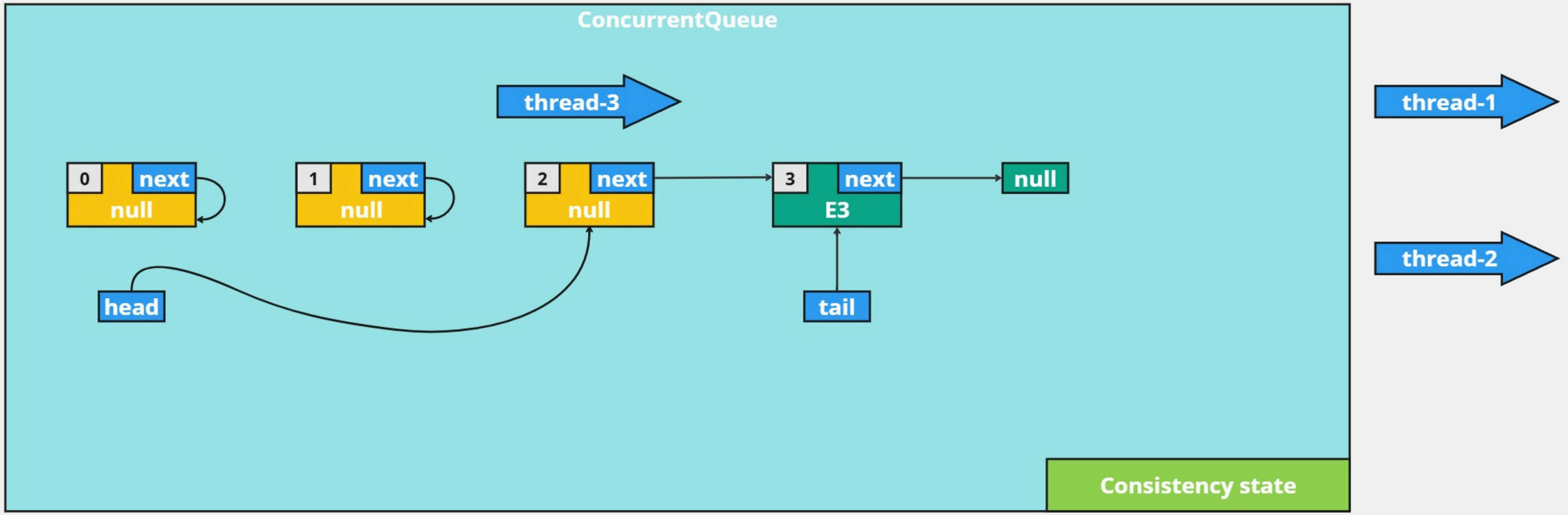
thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

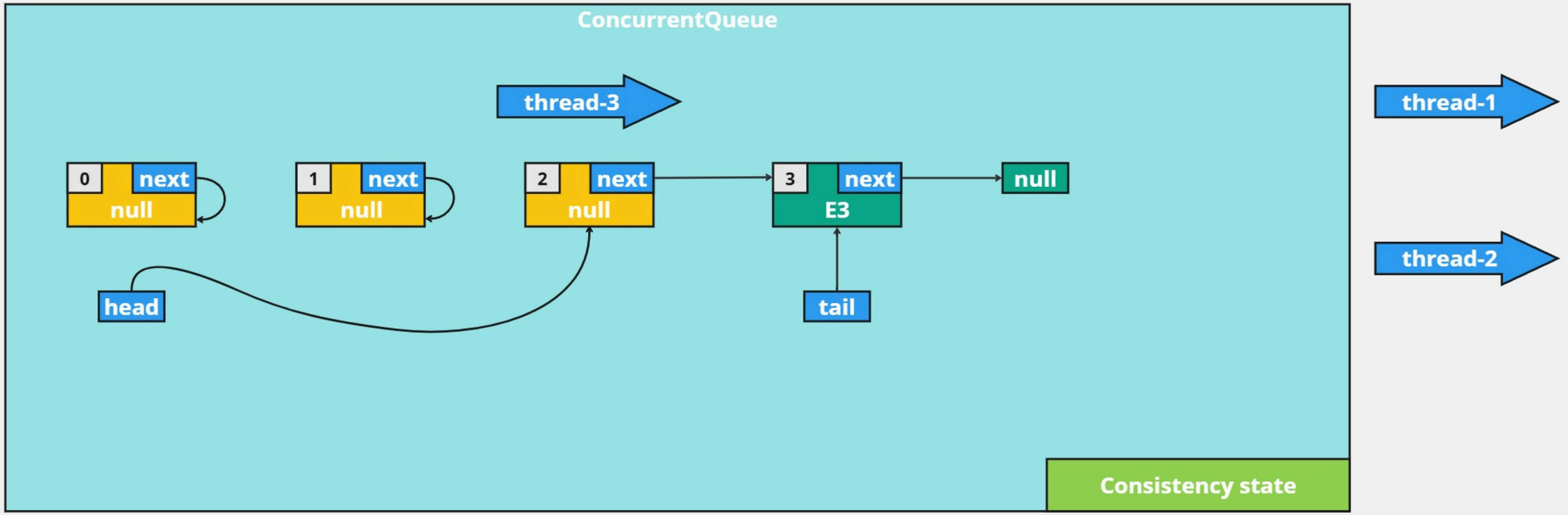
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

3



thread-1

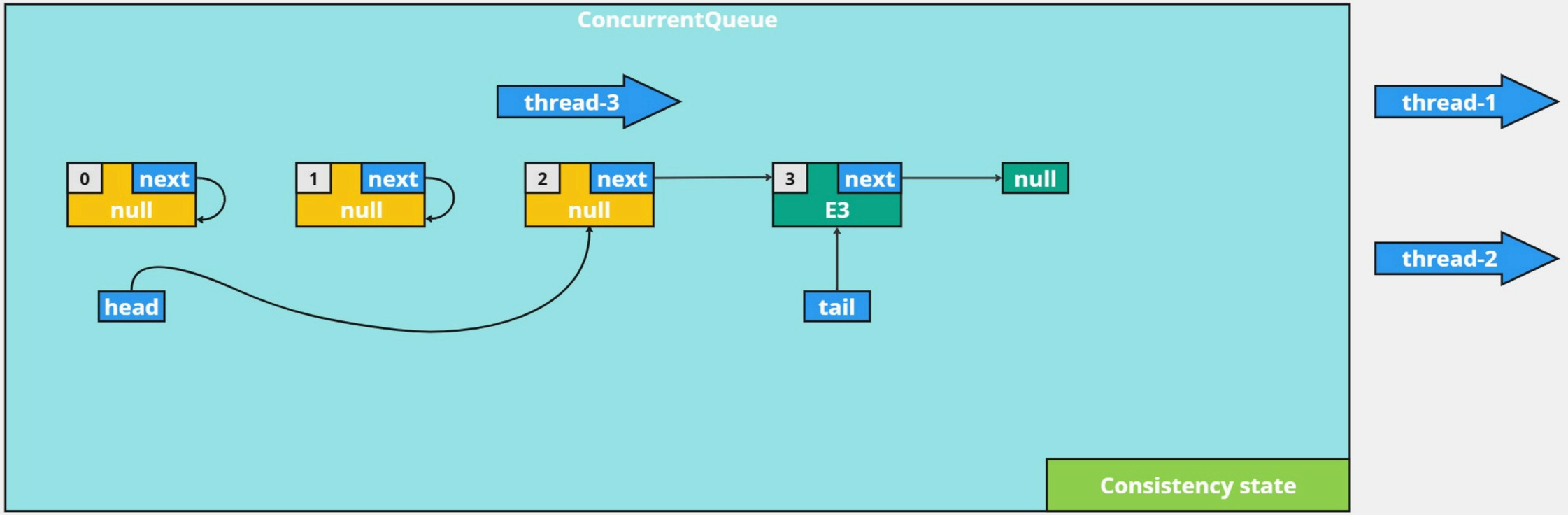
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

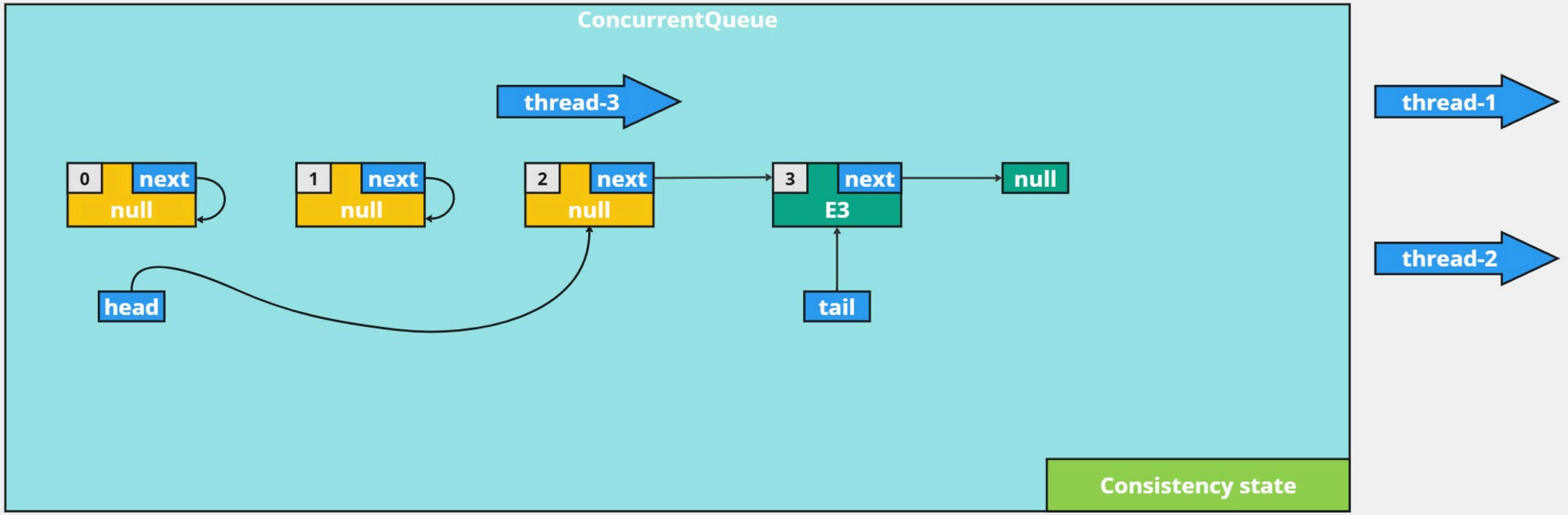
```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

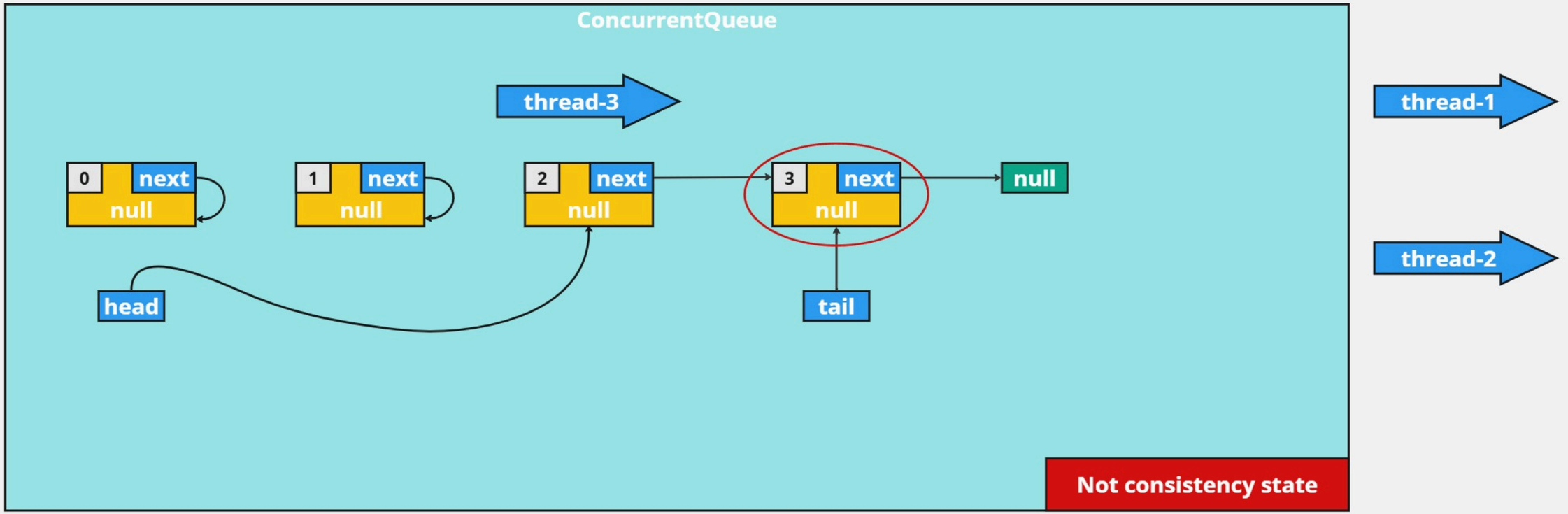
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

3

E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

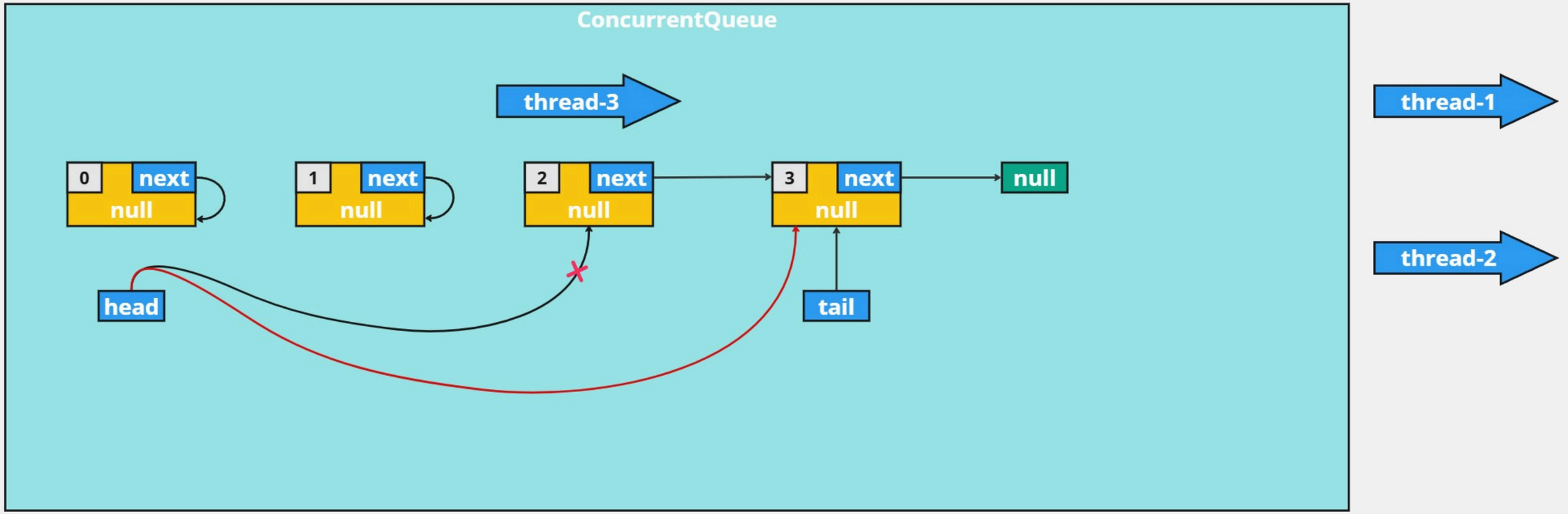
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

3

E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

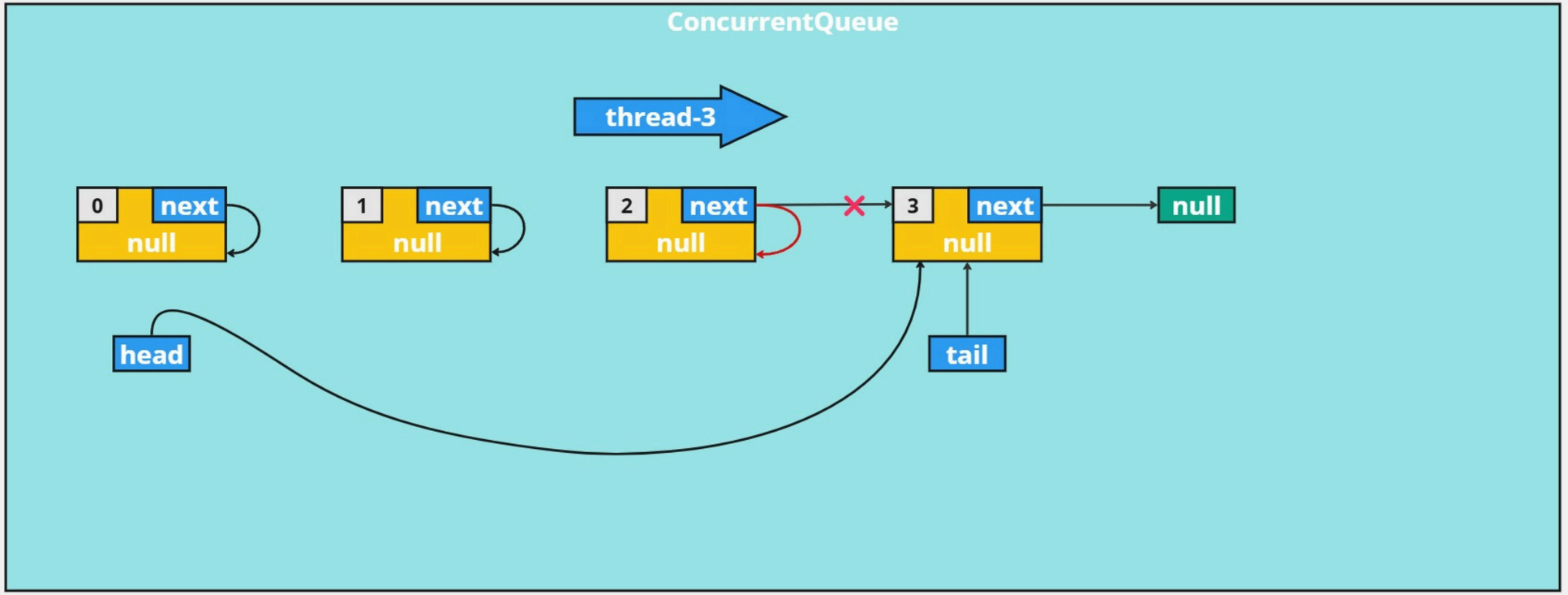
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

3

E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

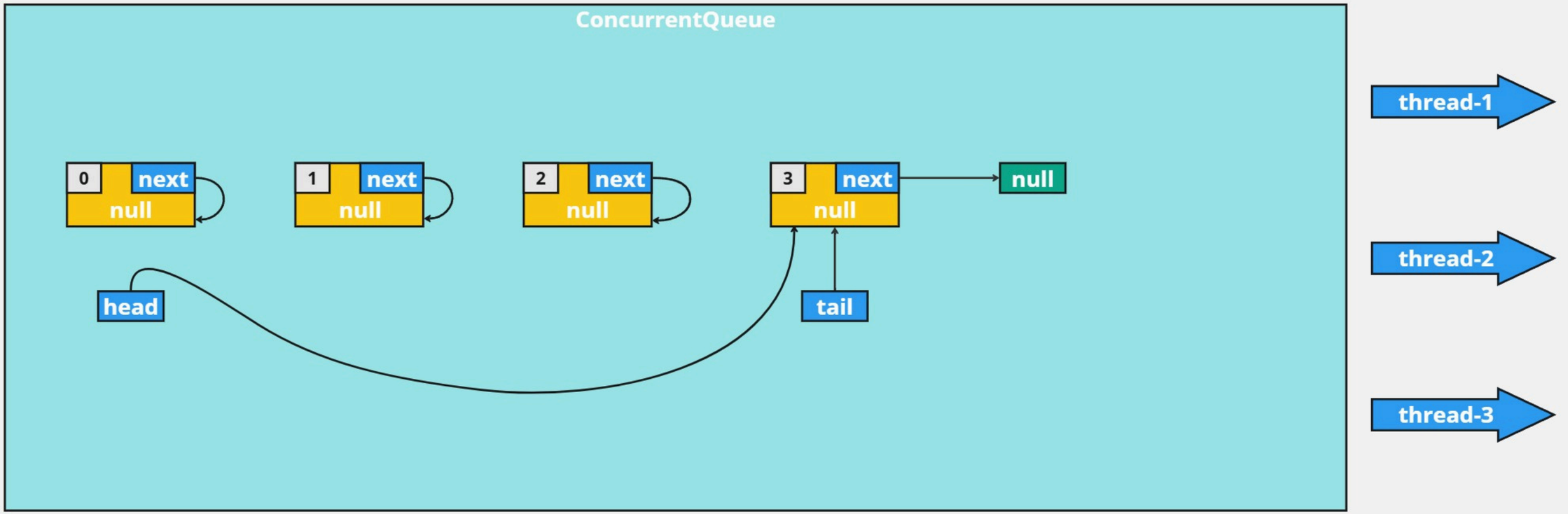
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

3

E3



thread-1

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

thread-2

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

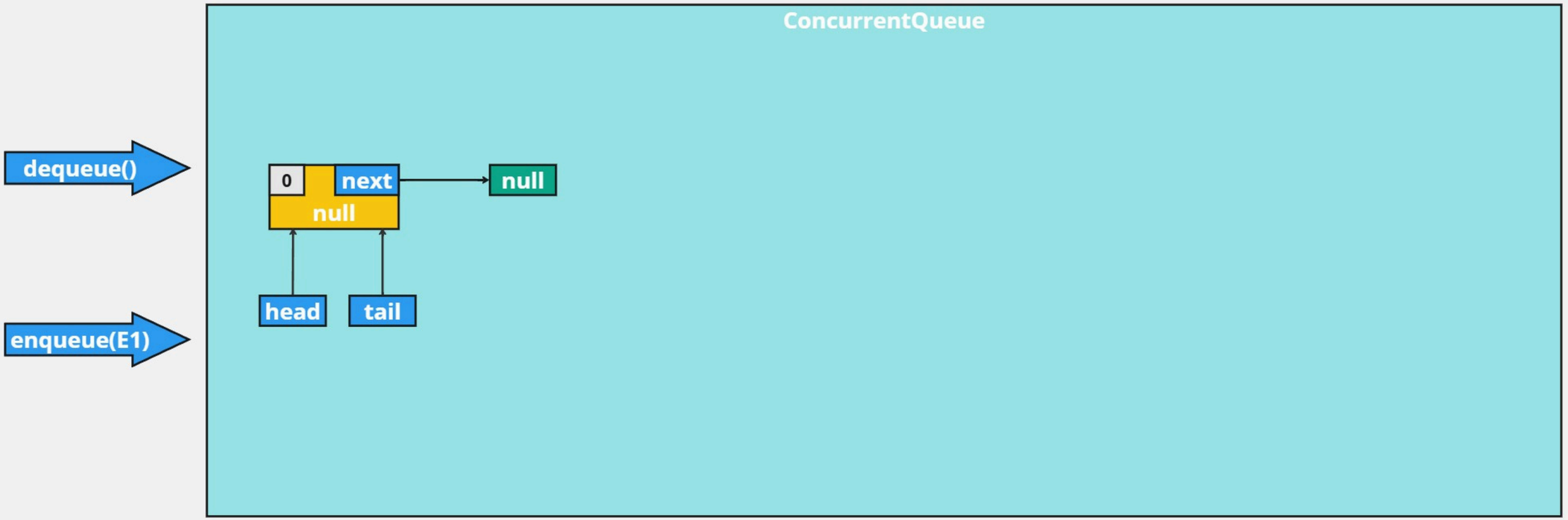
thread-3

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
private void updateHead(final Node<E> previous, final Node<E> next) {
    if (head.compareAndSet(previous, next)) {
        previous.next.set(previous);
    }
}
```

2

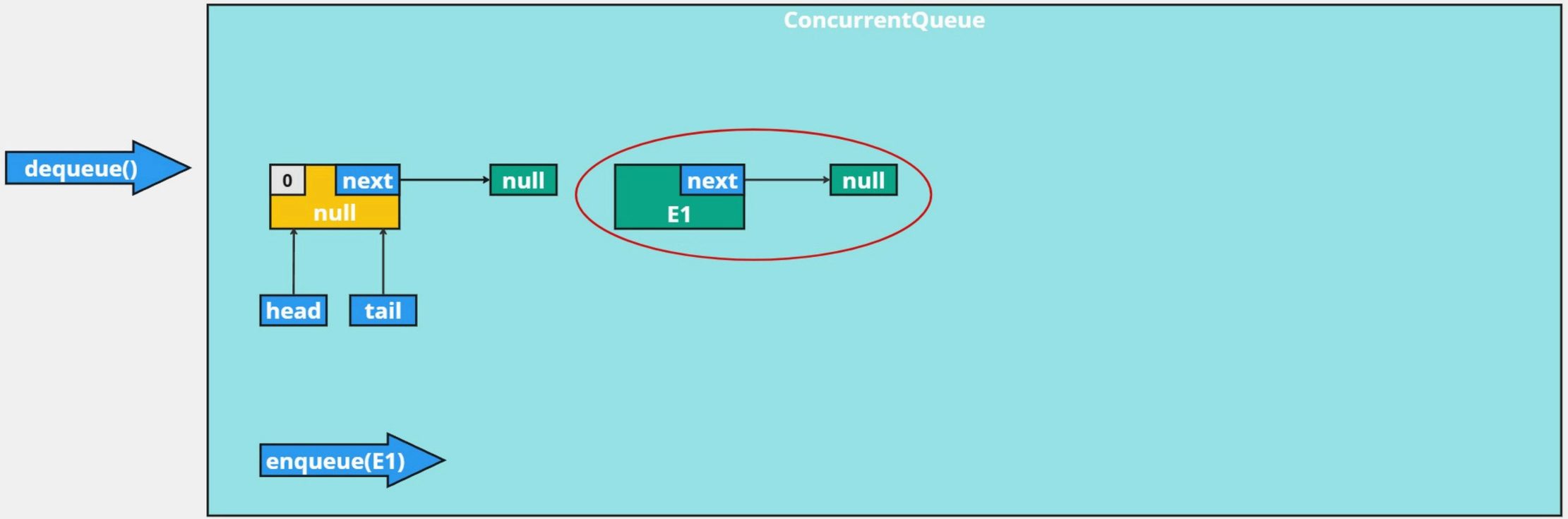
3

E3



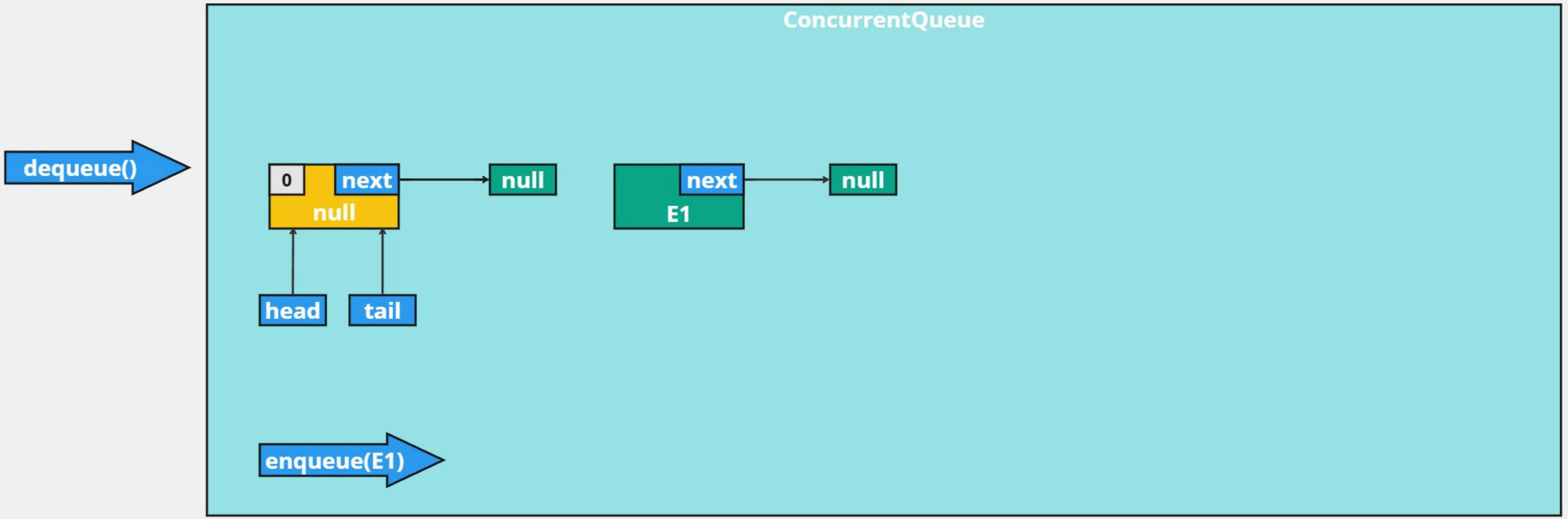

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

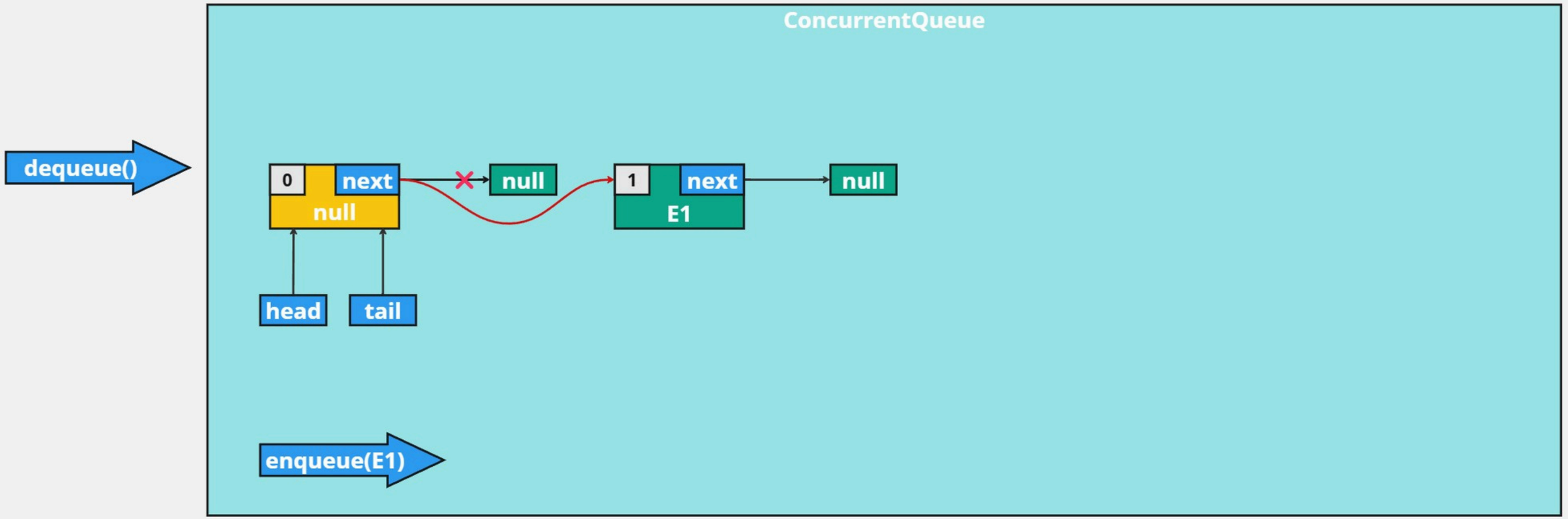
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

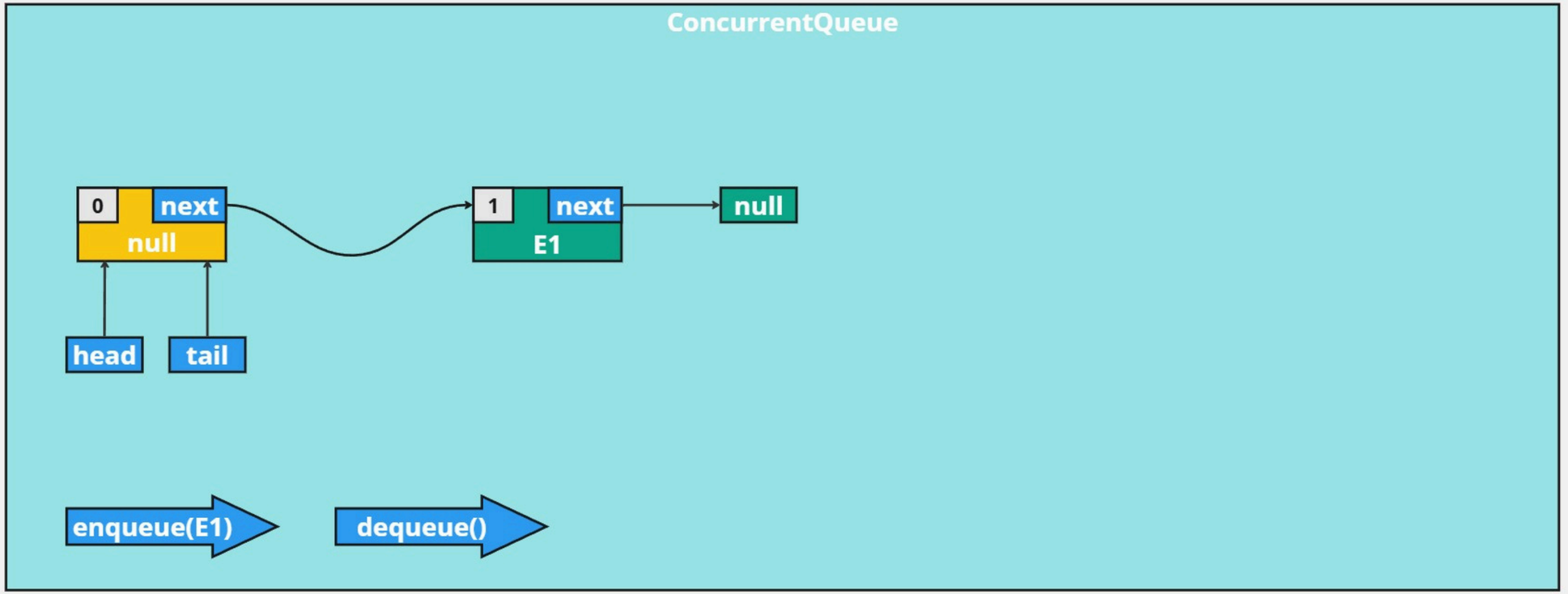
0




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0

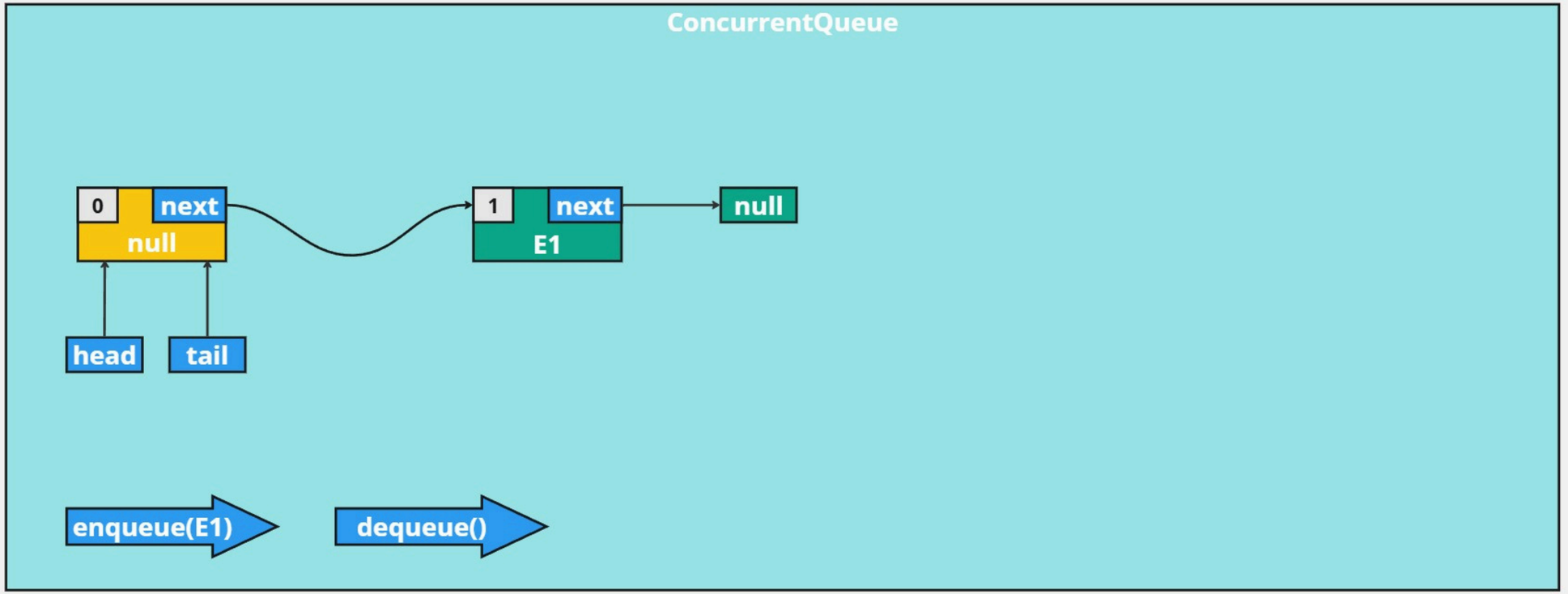



```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0



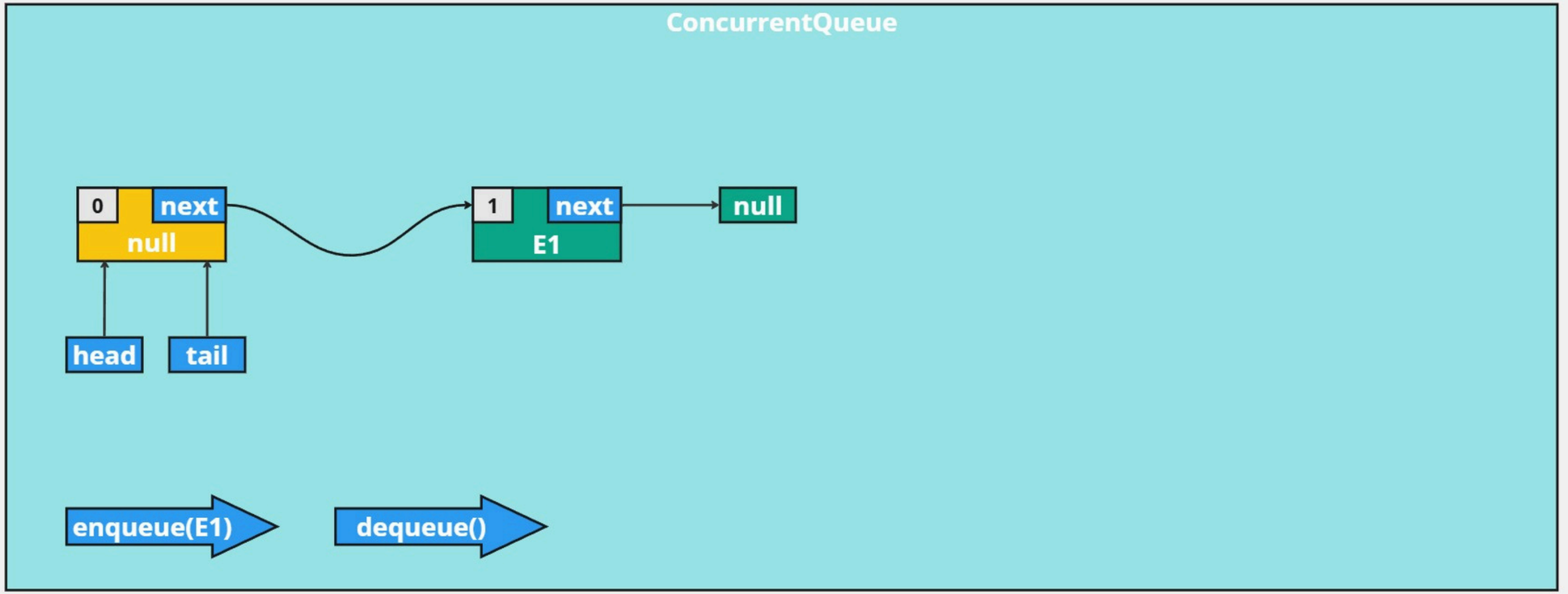

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

1

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0



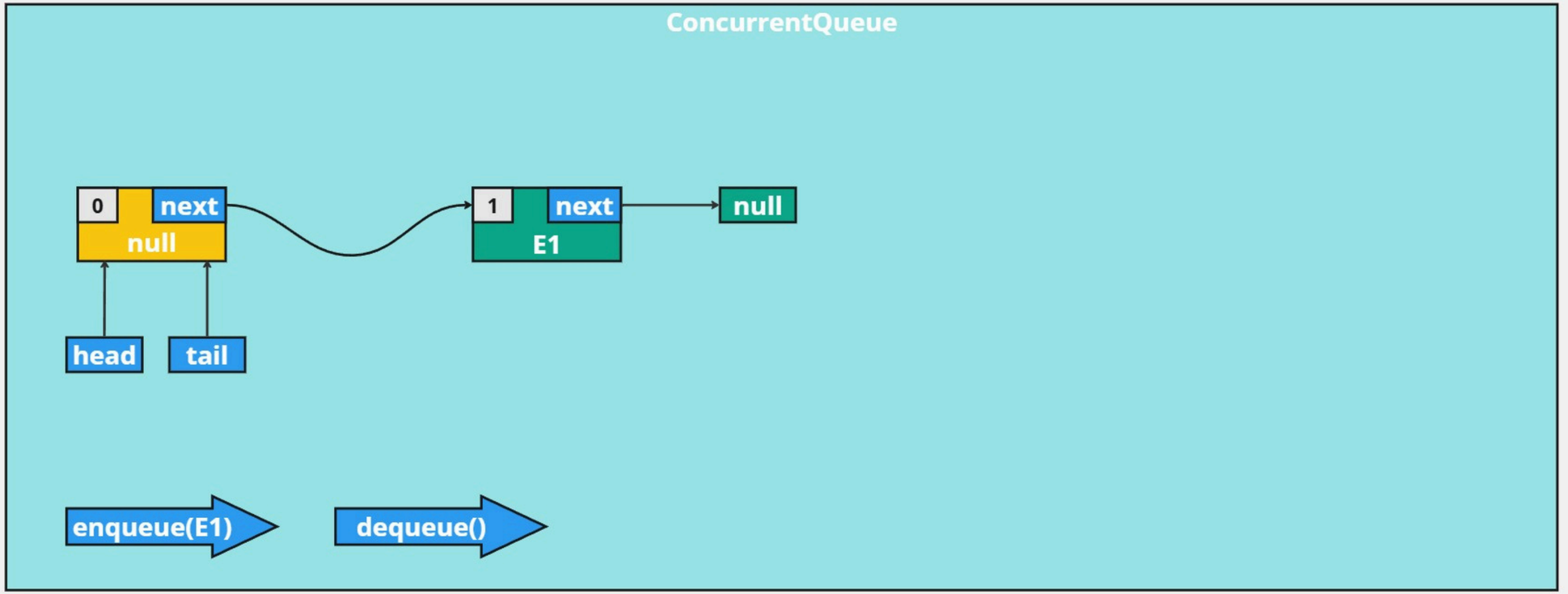

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

1

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0



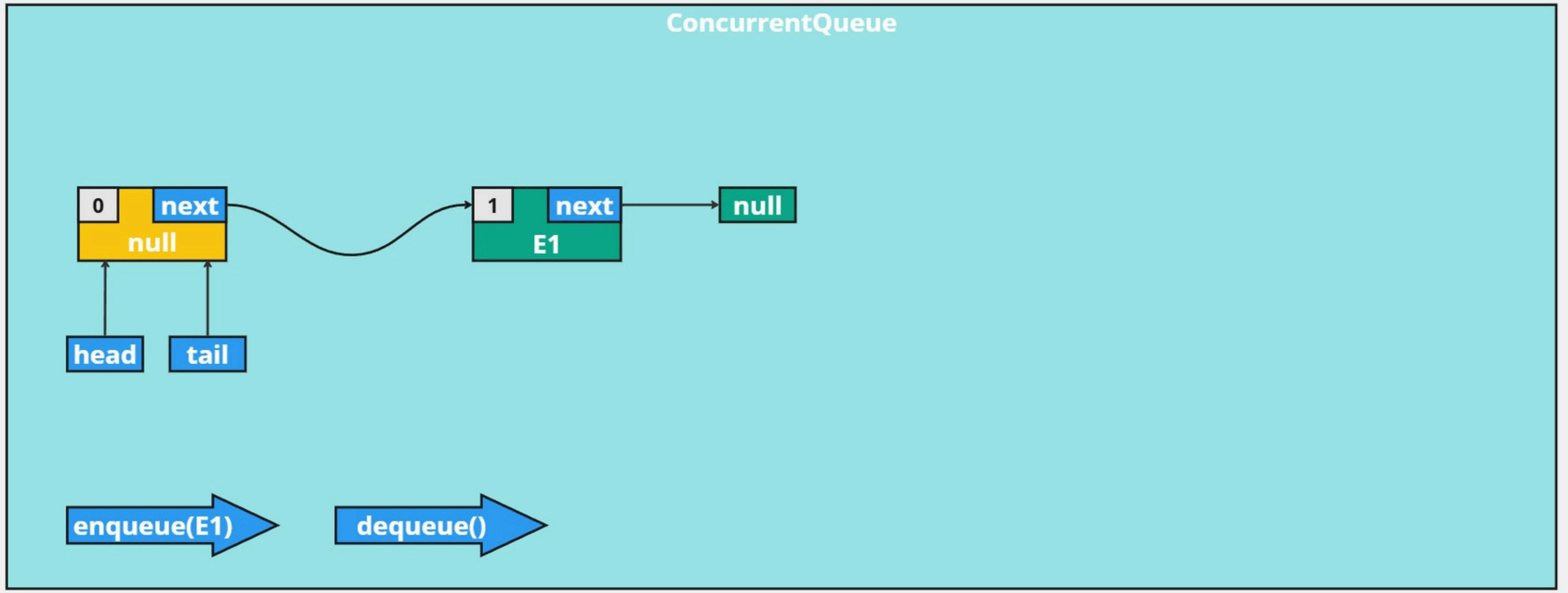

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

1

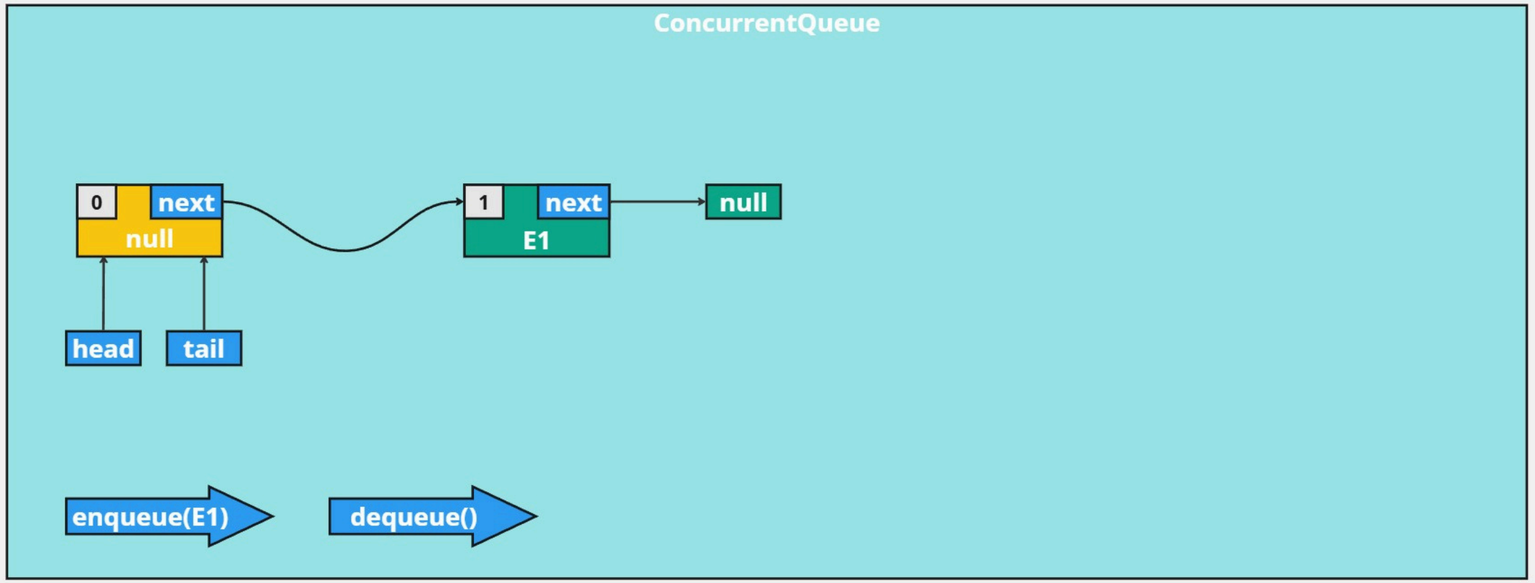
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0



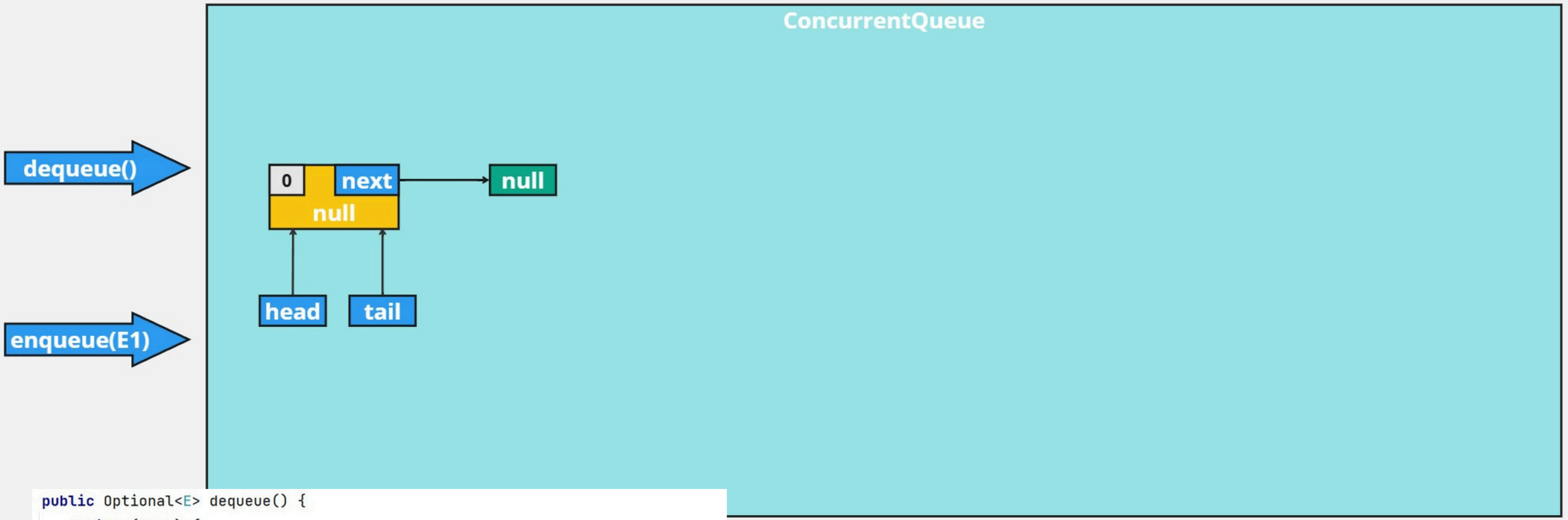

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get()) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get() && nextHead == null) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```



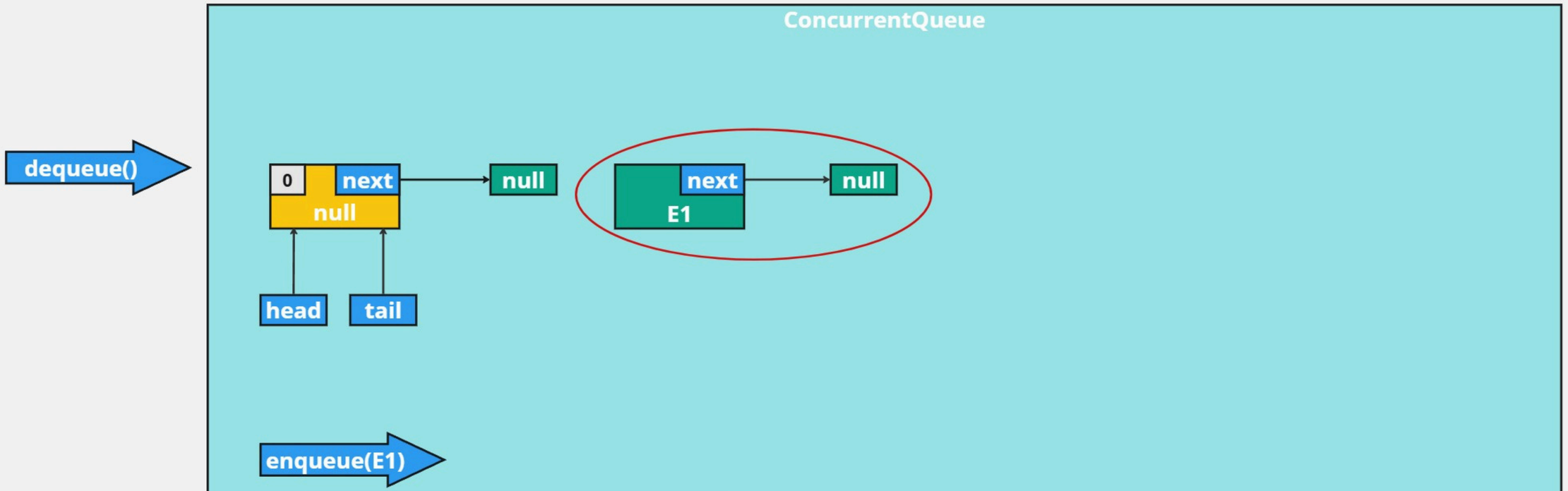

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == tail.get() && nextHead == null) {
            return empty();
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```



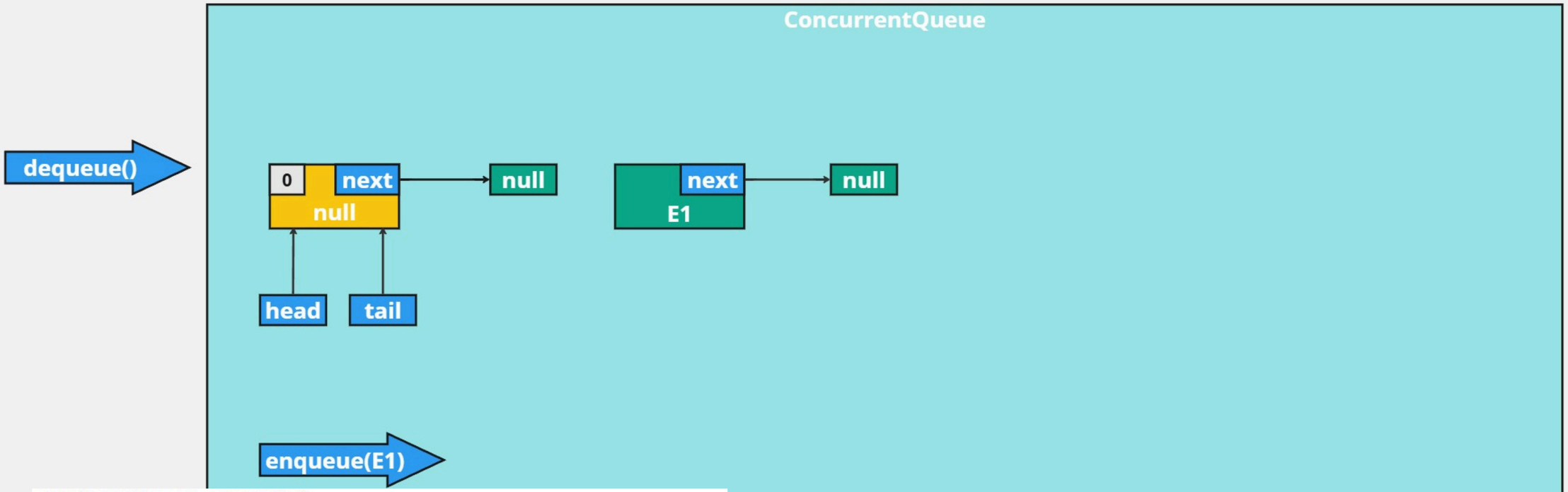

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

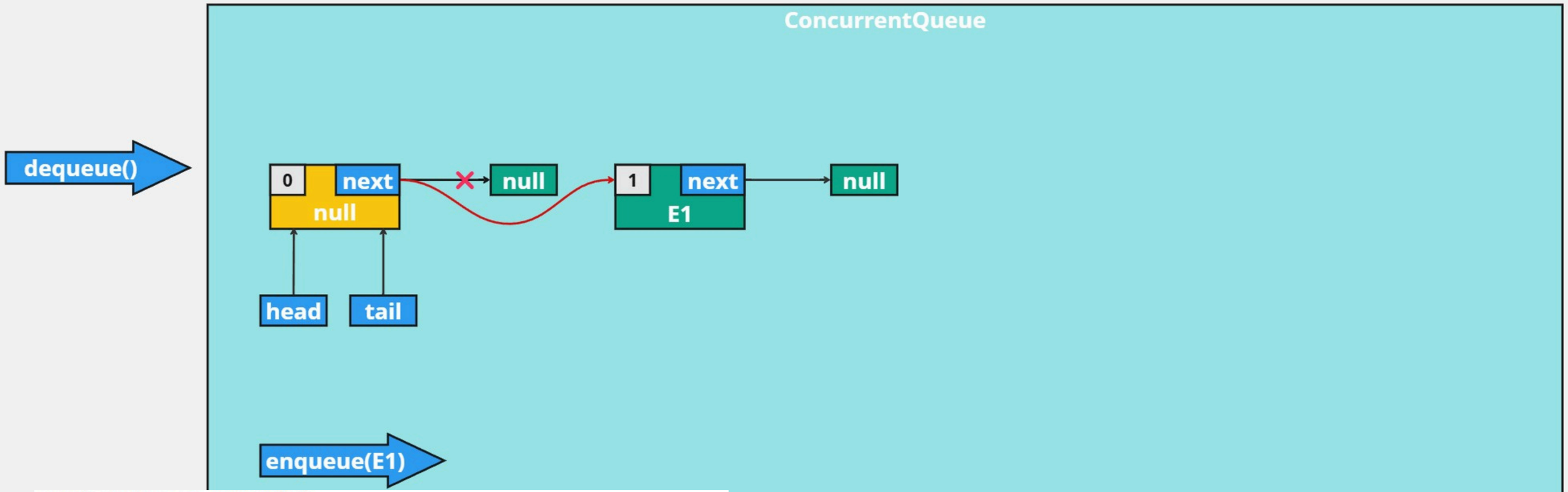
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

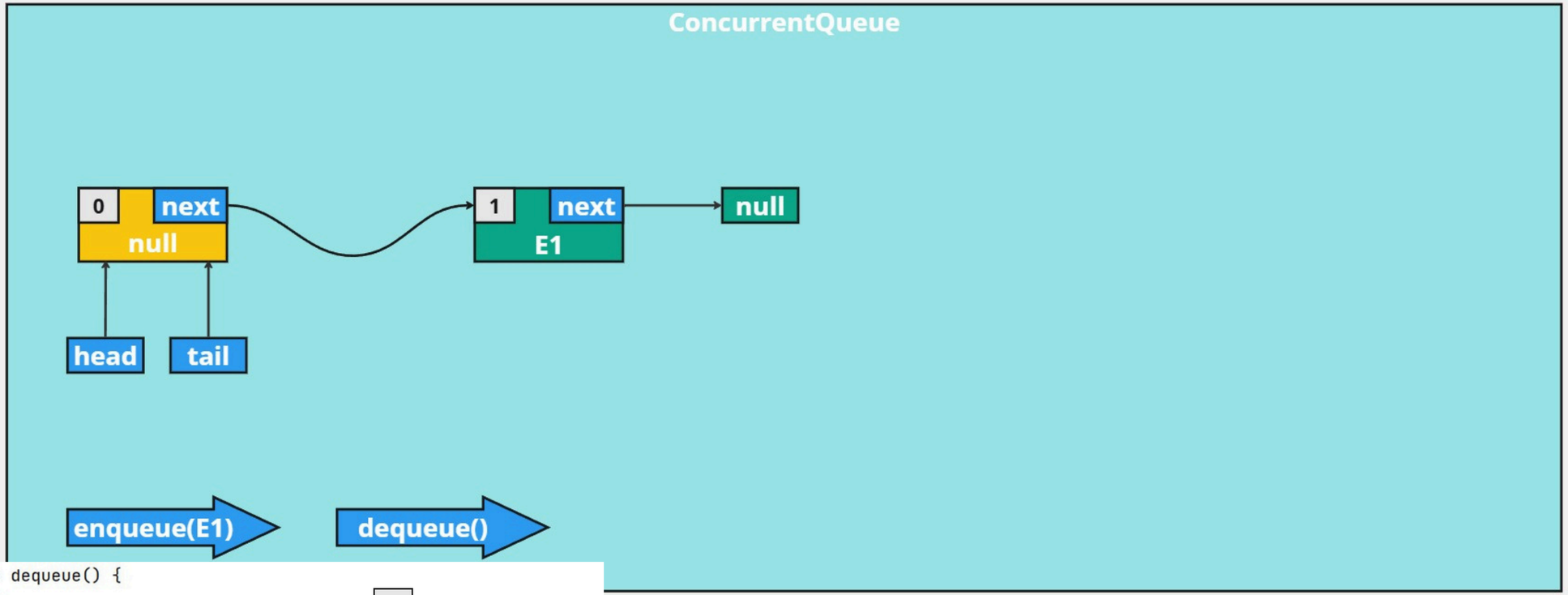
0




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0

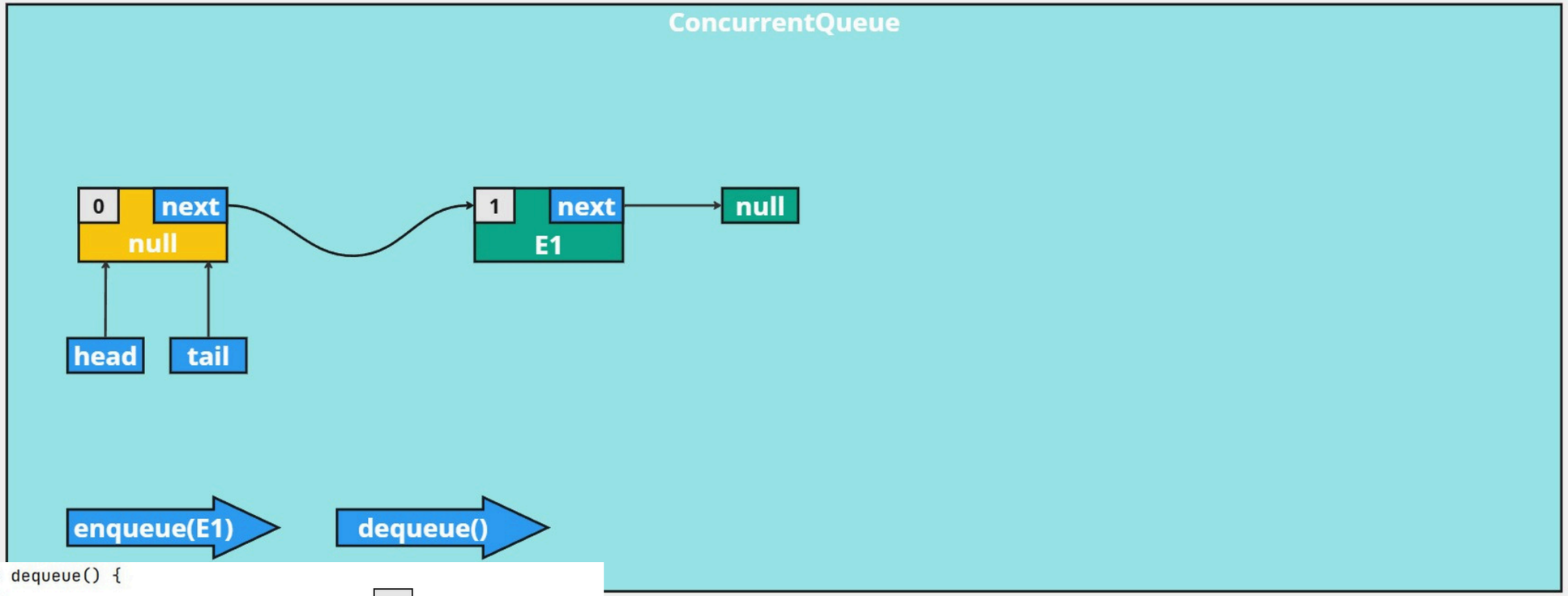



```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

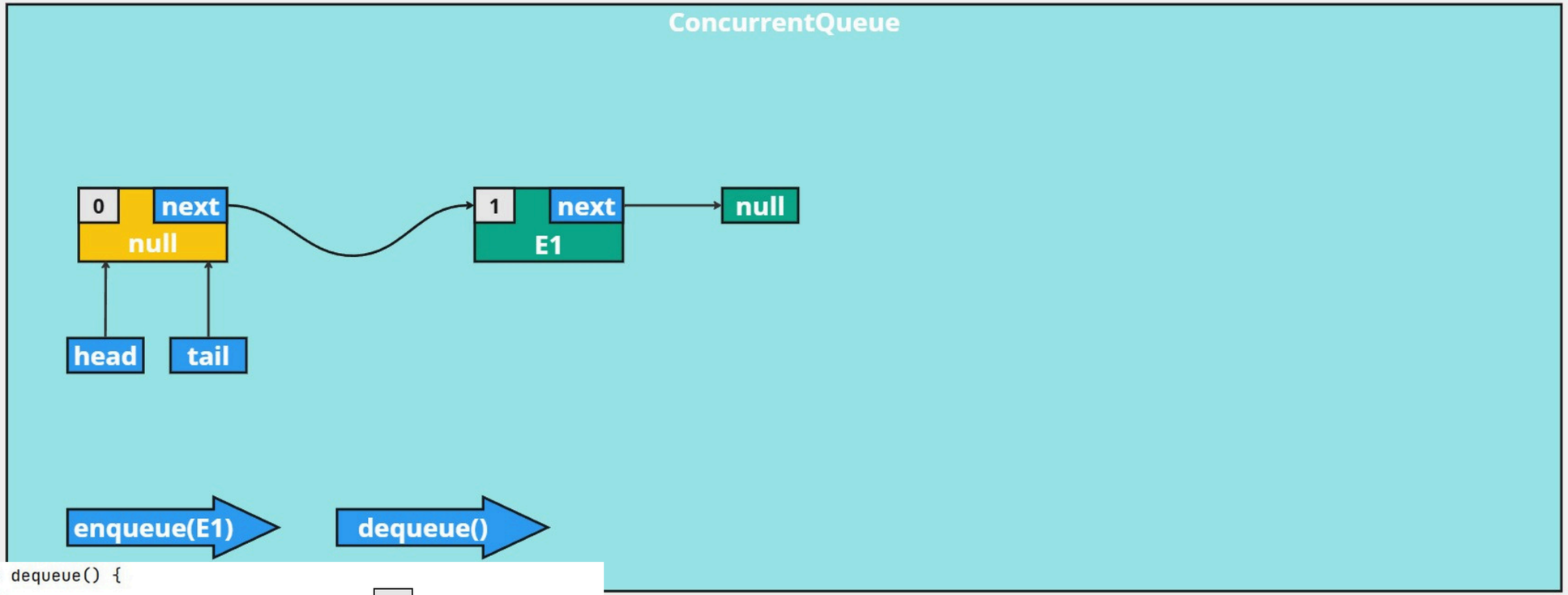
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0



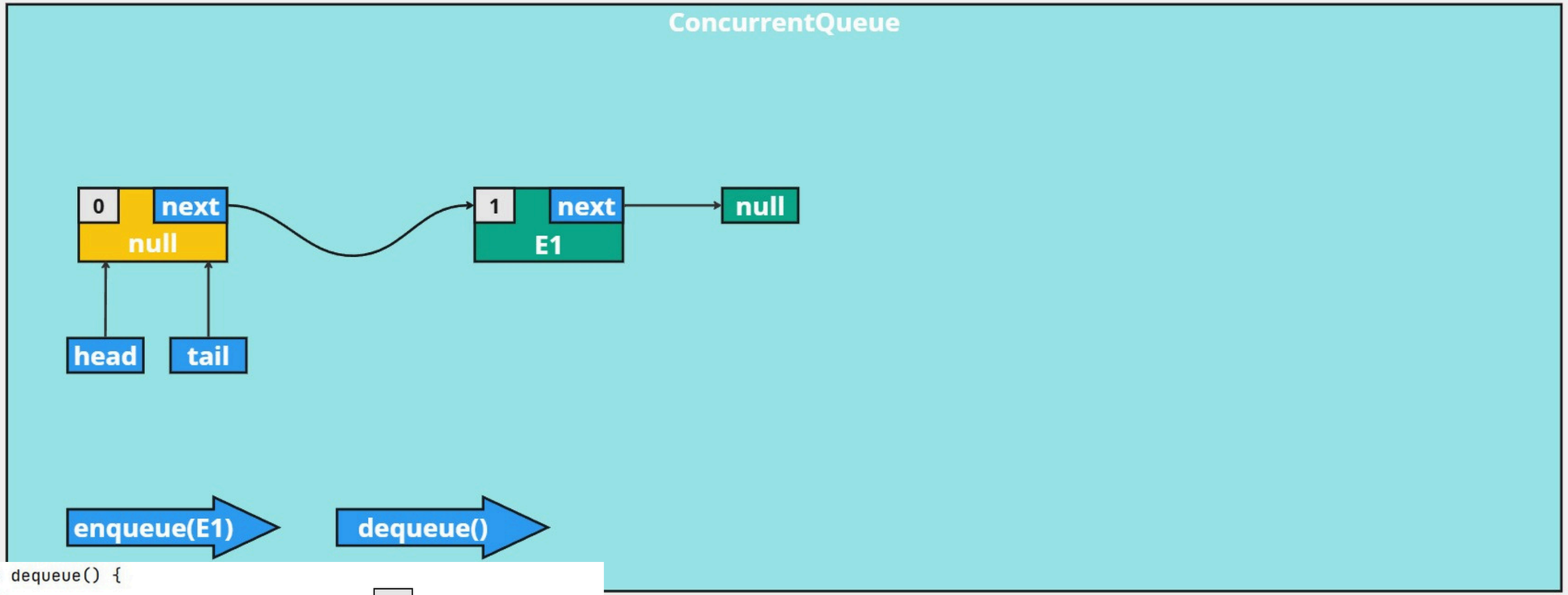

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



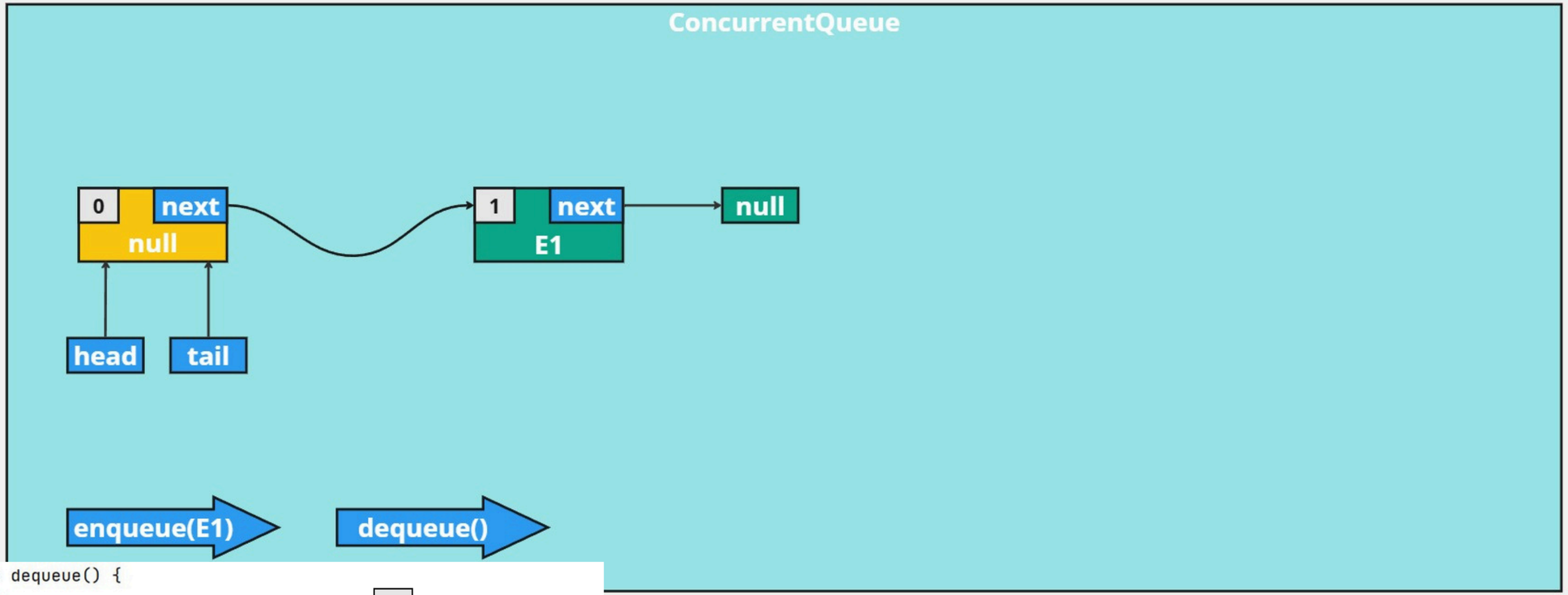

```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```



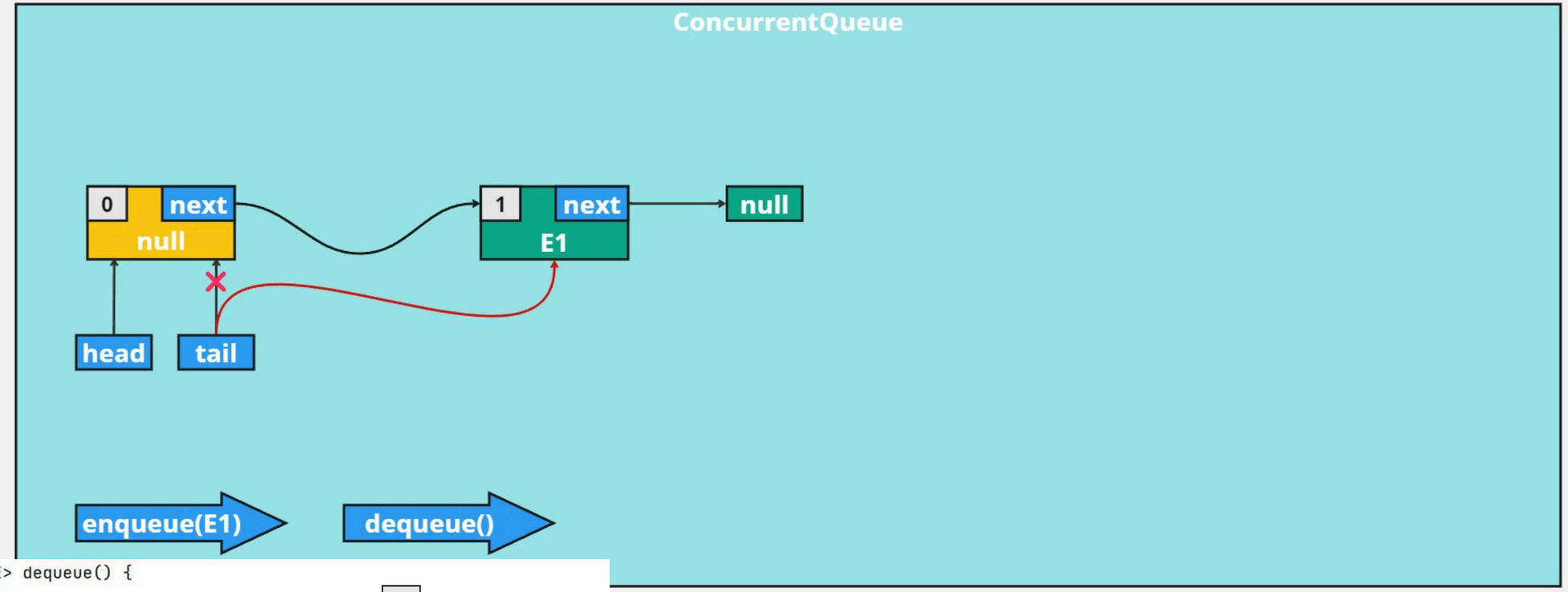

```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

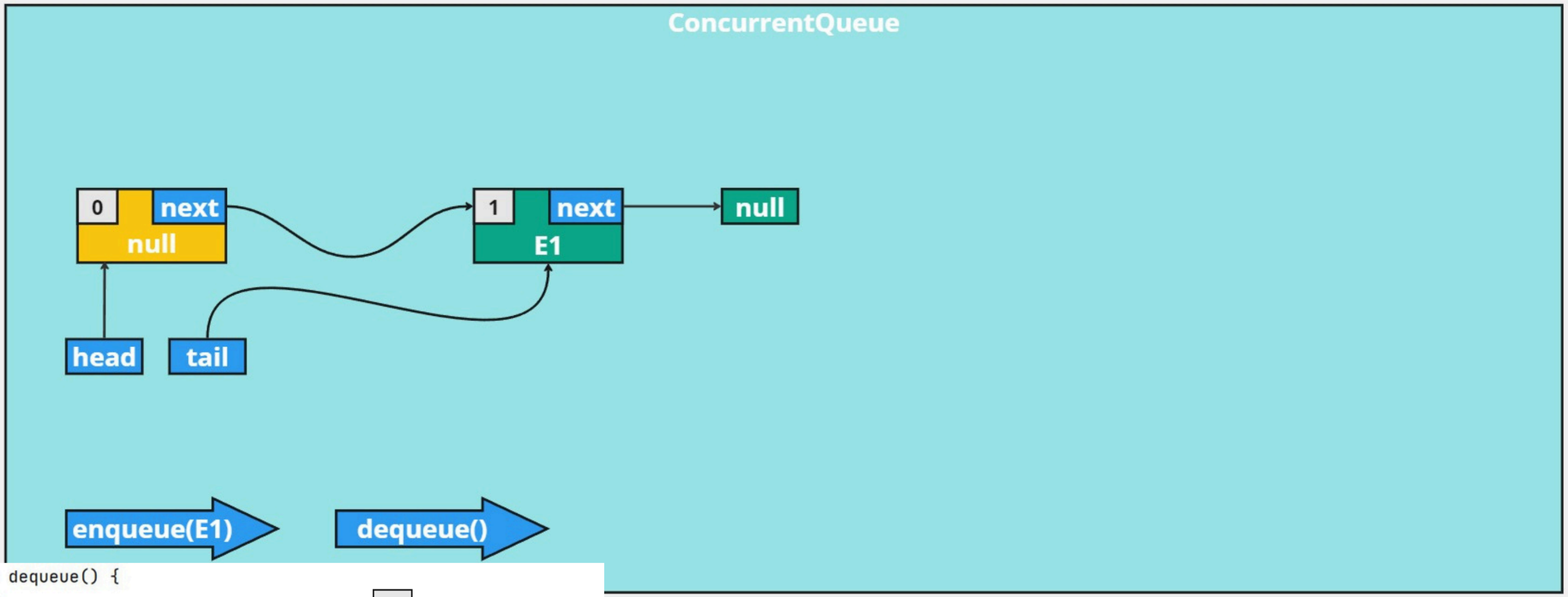
0

0

1

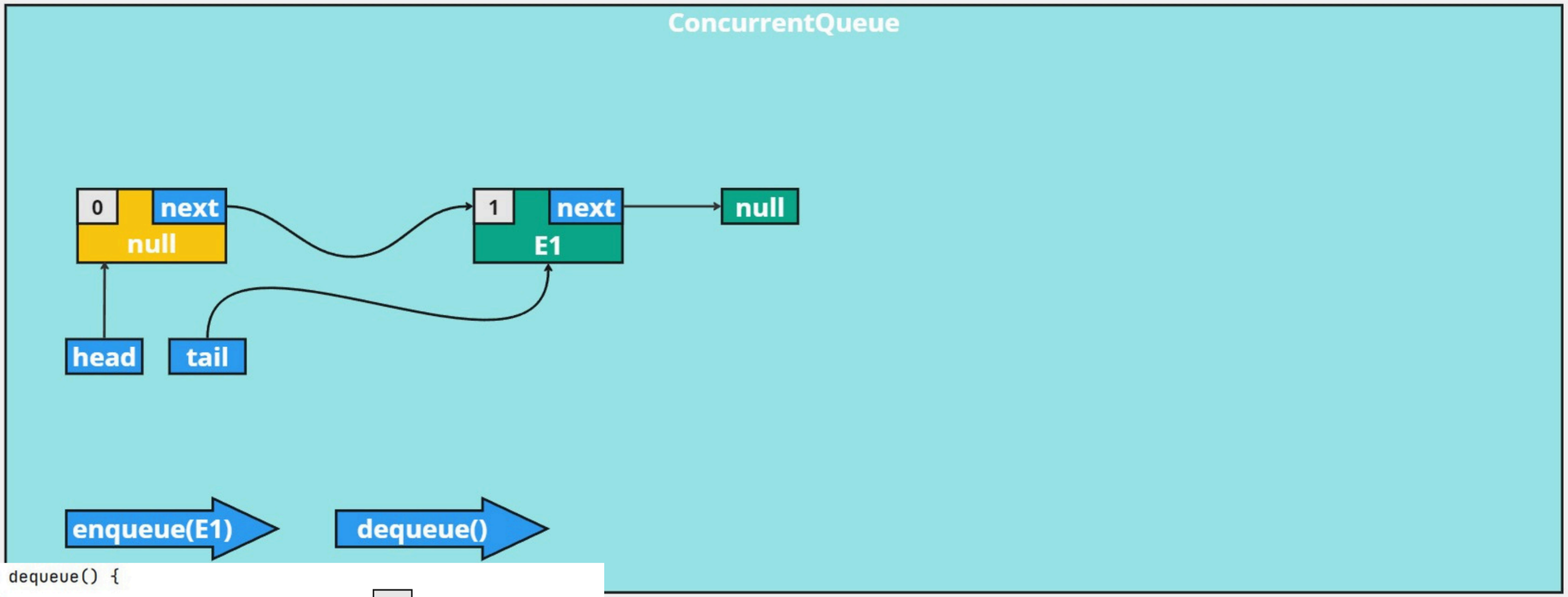
```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0




```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

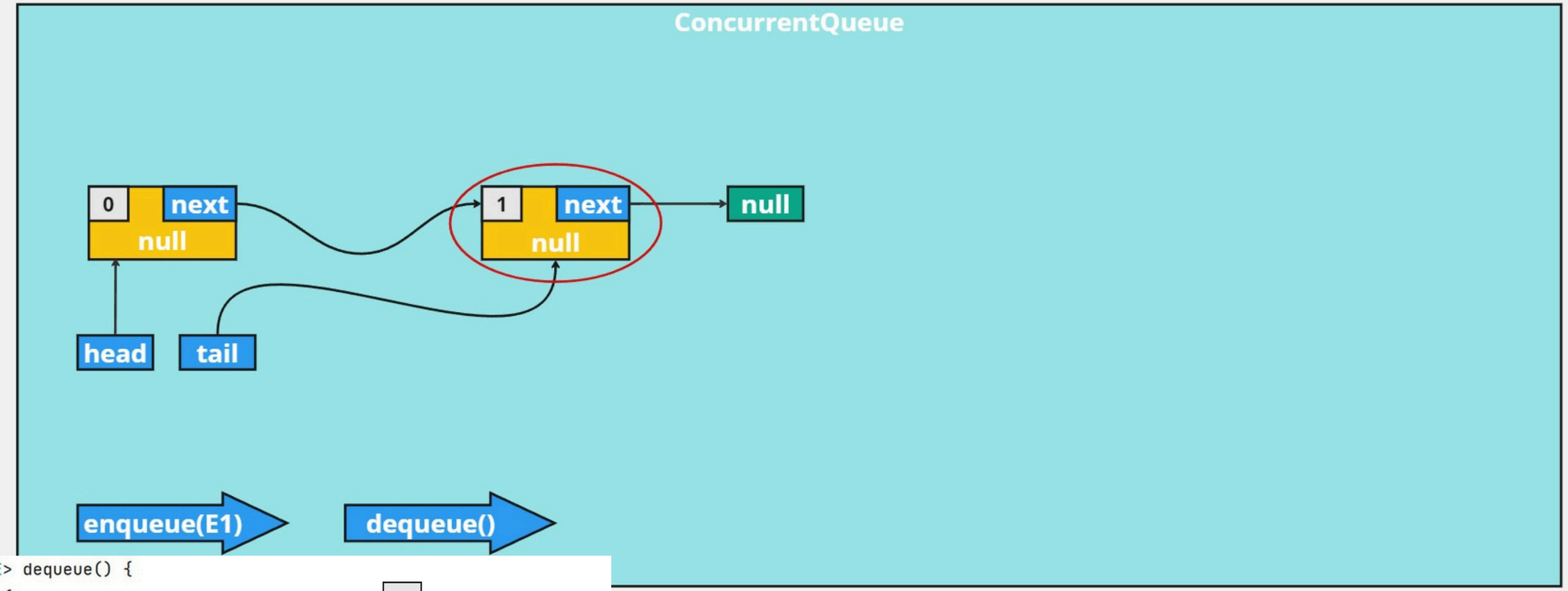
0

1

E1

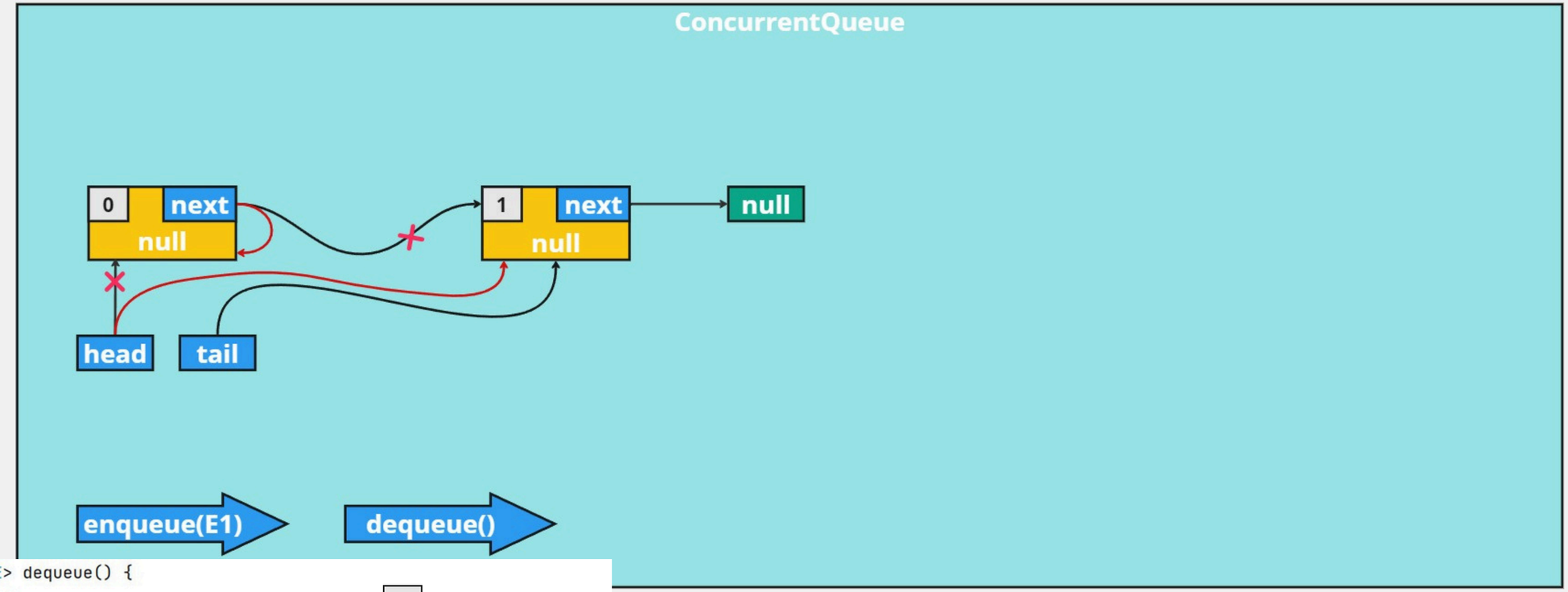
```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```




```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0

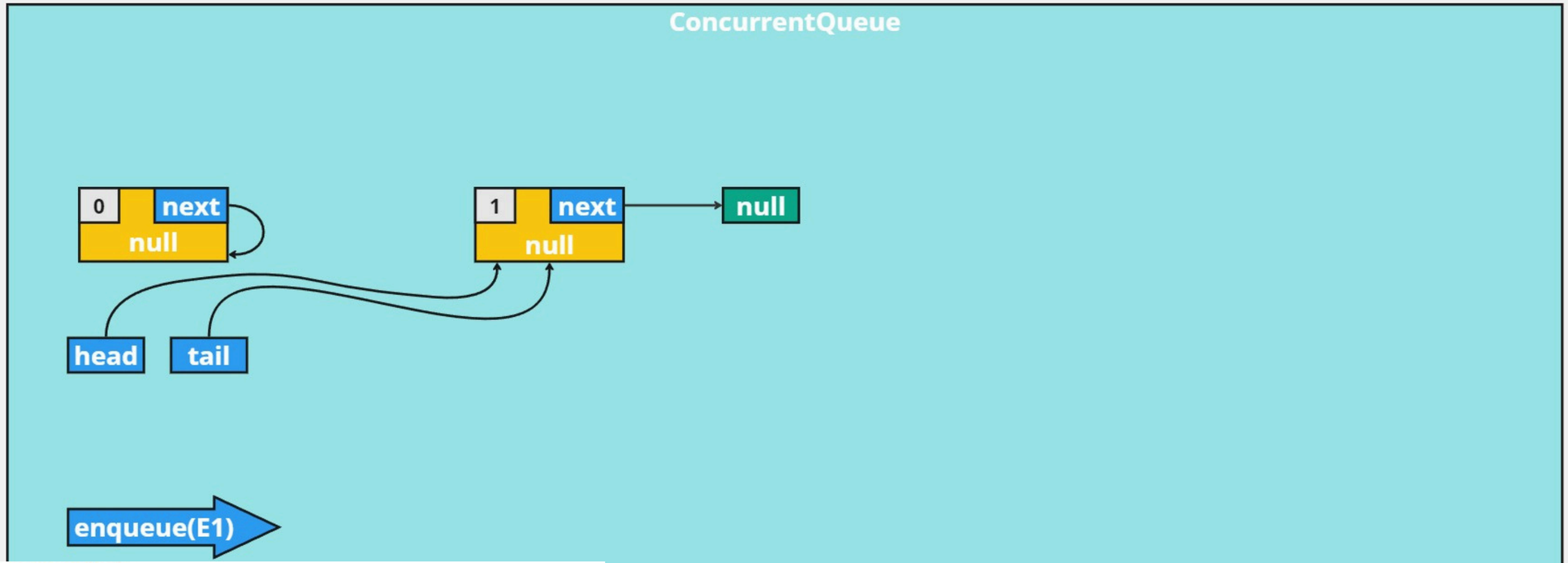
0

1

E1

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0

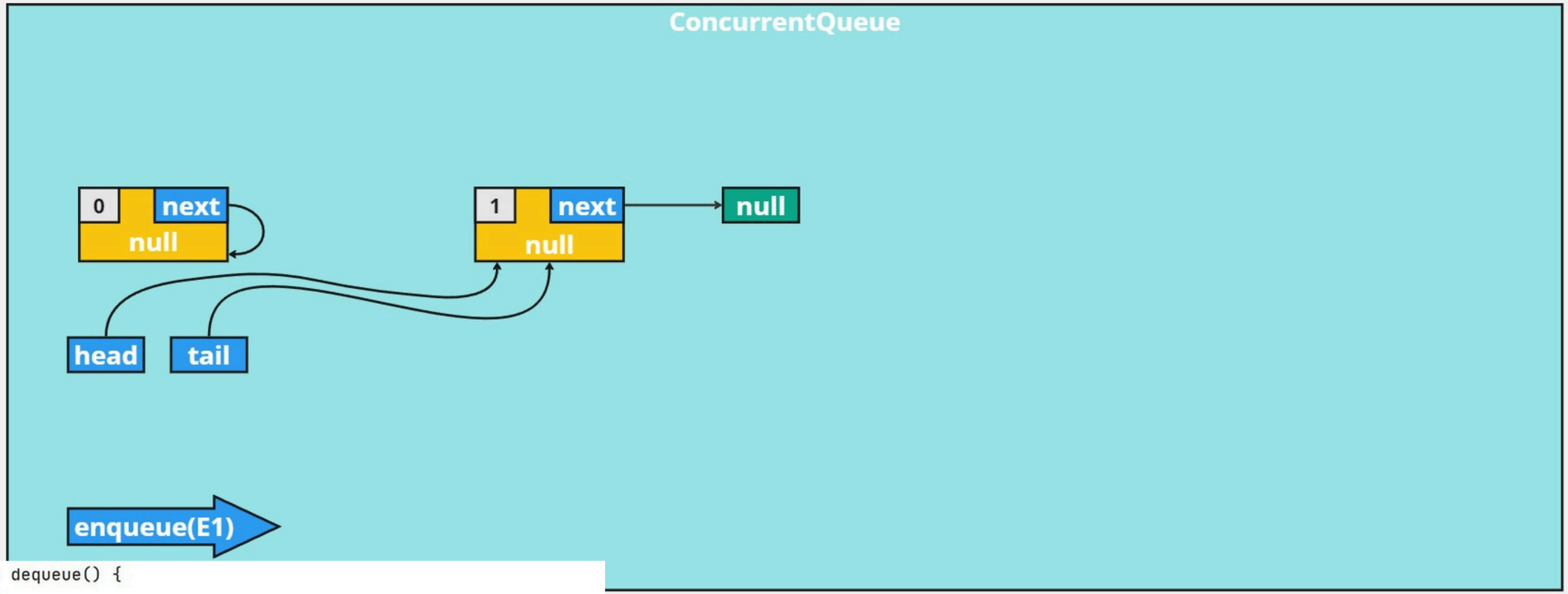



```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

0 0 1 E1

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

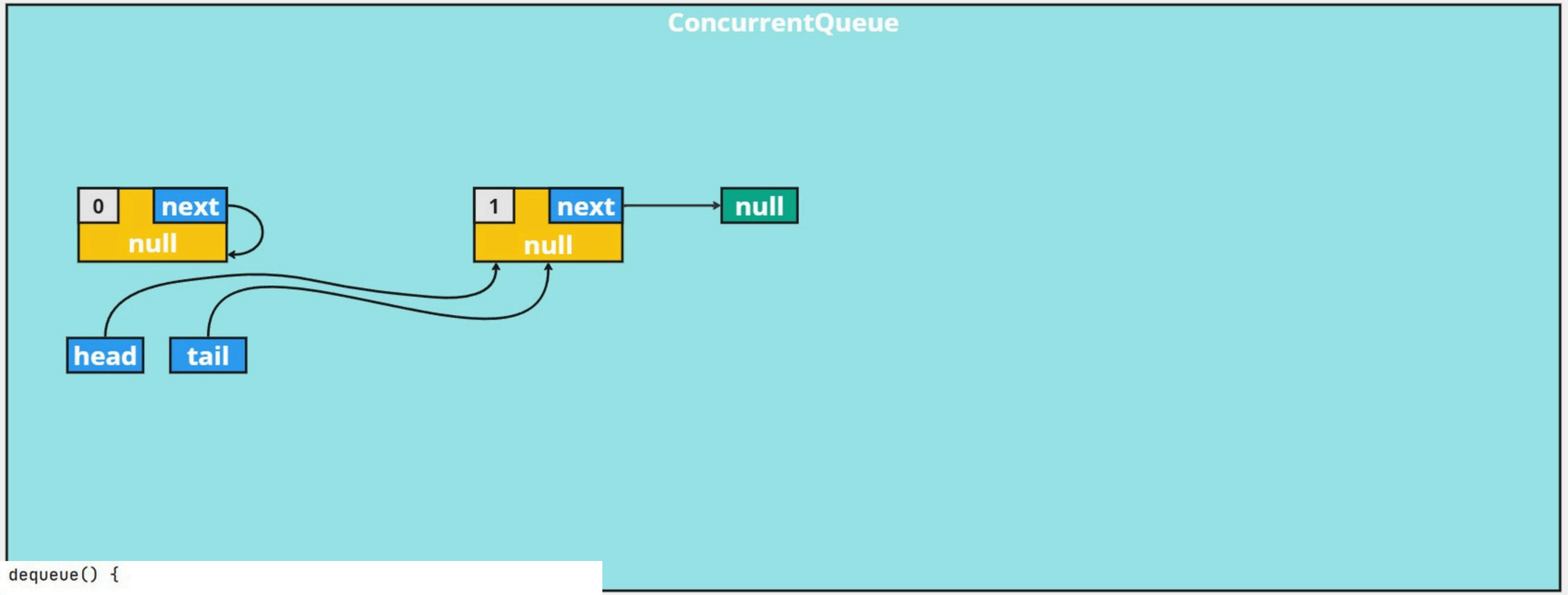
0




```
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0




```java
public Optional<E> dequeue() {
    while (true) {
        final Node<E> previousHead = head.get();
        final Node<E> previousTail = tail.get();
        final Node<E> nextHead = previousHead.next.get();
        if (previousHead == previousTail) {
            if (nextHead == null) {
                return empty();
            } else {
                tail.compareAndSet(previousTail, nextHead);
            }
        }
        final E element = nextHead.value.get();
        if (element != null && nextHead.value.compareAndSet(element, null)) {
            updateHead(previousHead, nextHead);
            return of(element);
        } else {
            updateHead(previousHead, nextHead);
        }
    }
}
```

```java
public void enqueue(final E element) {
    final Node<E> newNode = new Node<>(element);
    while (true) {
        final Node<E> previousTail = tail.get();
        if (previousTail.next.compareAndSet(null, newNode)) {
            tail.compareAndSet(previousTail, newNode);
            return;
        } else {
            tail.compareAndSet(previousTail, previousTail.next.get());
        }
    }
}
```

0

java.util.concurrent.ConcurrentLinkedQueue