

Sistemas Distribuídos - Aulas Práticas -

Universidade do Minho
2024/2025

Material Aulas

- Guiões com exercícios
- Java (SDK 11+)
 - OpenJDK, Oracle Java
- IDE:
 - **IntelliJ IDEA**, VSCode, ...

Threads

- Fios de execução concorrentes de um programa
- Um processo tem uma ou mais threads
- Partilham recursos e comunicam entre si através de memória partilhada
- Analogia:
 - As threads de um processo são como vários cozinheiros que seguem as instruções do mesmo livro de culinária, mas não necessariamente todos na mesma página.

Threads em Java

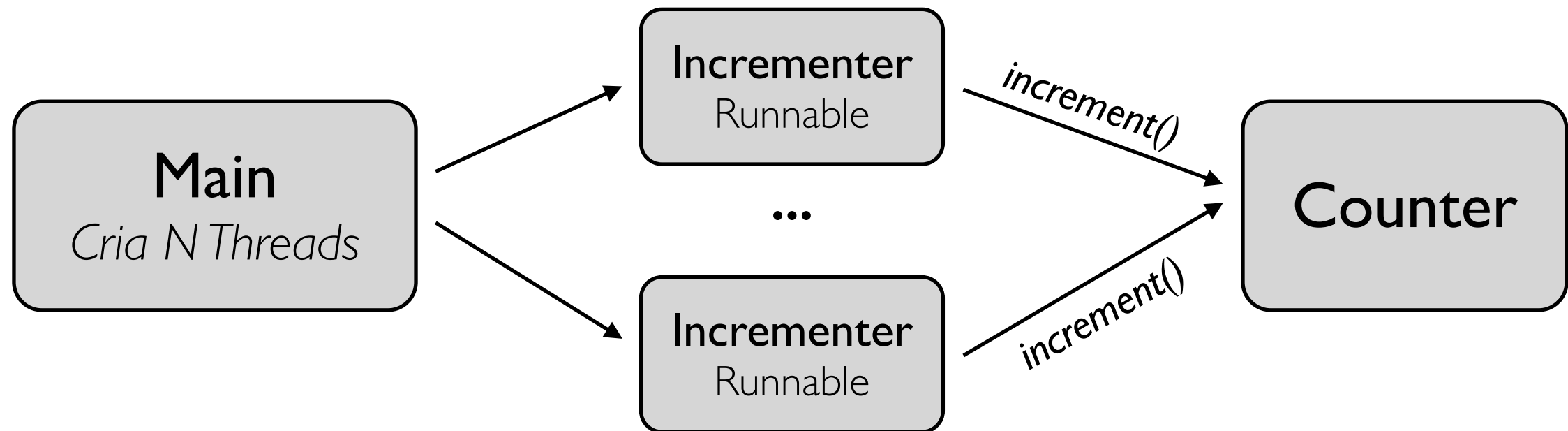
- `java.lang.Runnable`
 - interface implementada por classes cujas instâncias representam threads
 - classes que implementem esta interface têm que implementar o método `run()`
- `java.lang.Thread`
 - implementa `java.lang.Runnable`
 - classes que estendam `Thread` devem re-implementar o método `run()`
 - outros métodos relevantes: `start()`, `sleep(...)`, `join()`

Criação de uma thread:

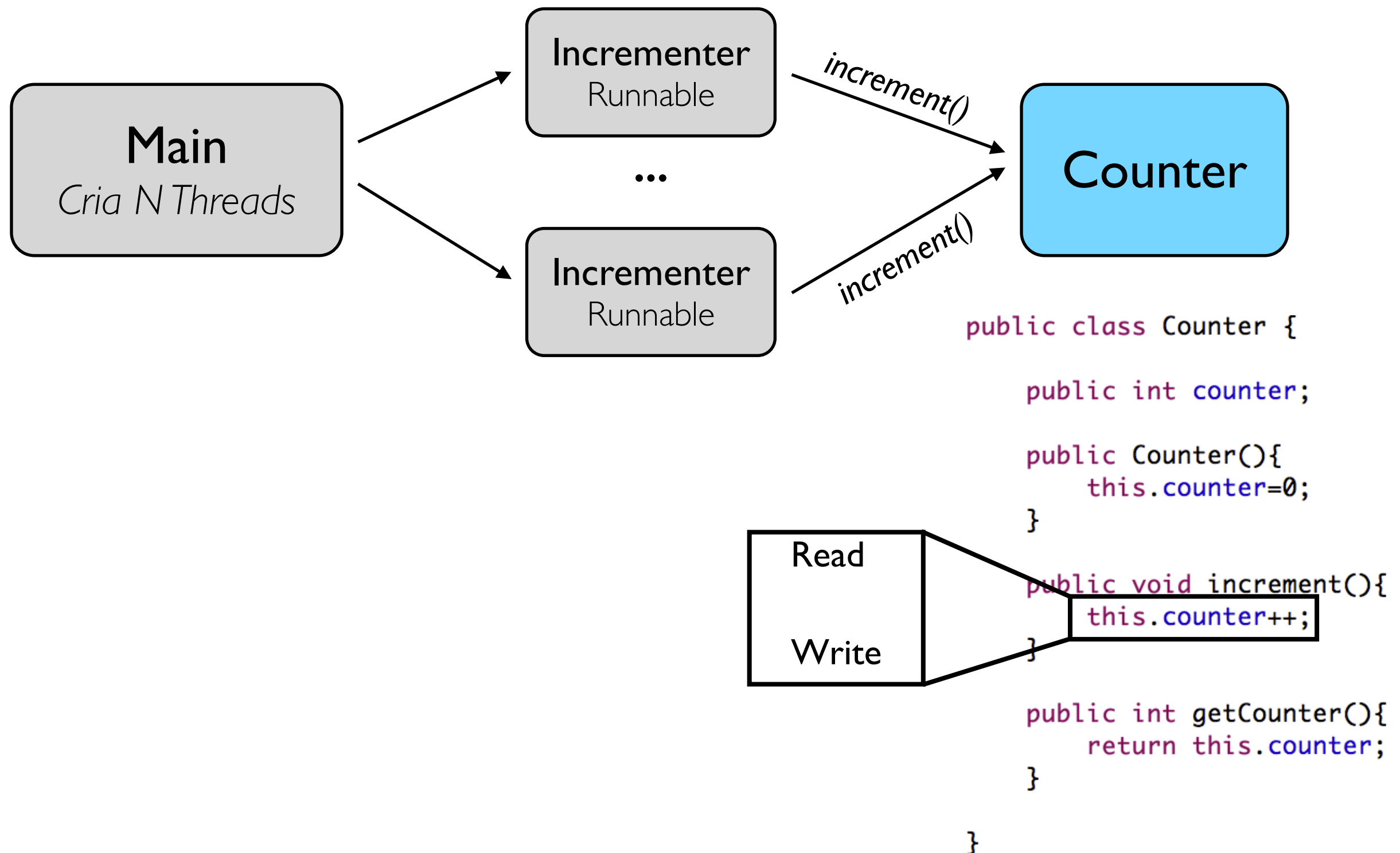
```
public class HelloRunnable implements Runnable {  
    public void run() {  
        System.out.println("Hello from a thread!");  
    }  
}  
  
public class Main {  
    public static void main(String args[]) {  
        Thread t = new Thread(new HelloRunnable());  
        t.start();  
        t.join(); // Bloqueia até a thread terminar.  
    }  
}
```

Exclusão Mútua

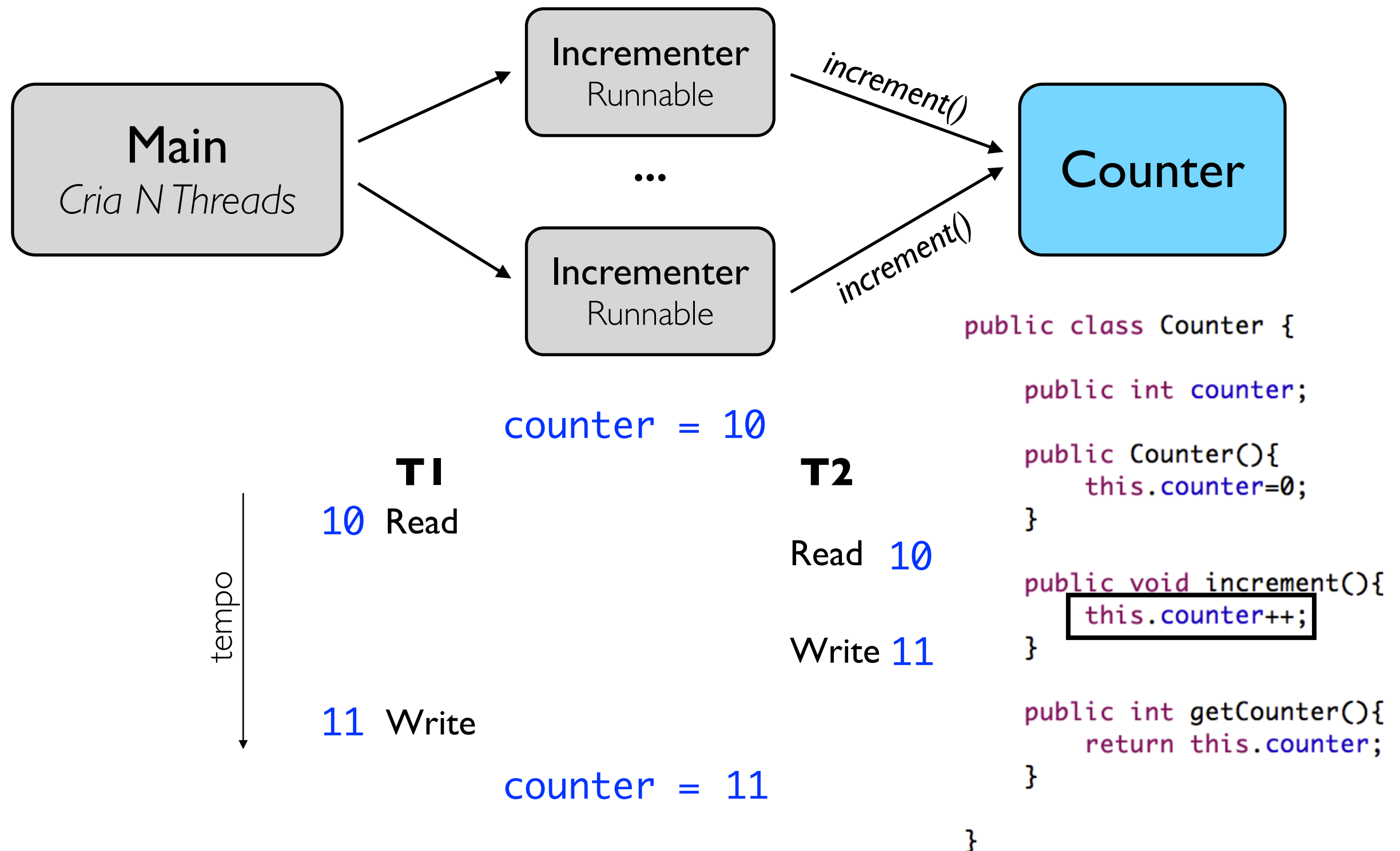
Exemplo: Programa com N threads que têm acesso a um único objecto de uma classe Counter. Cada thread deverá incrementar I vezes o contador. Thread Main imprime o valor do contador no final.



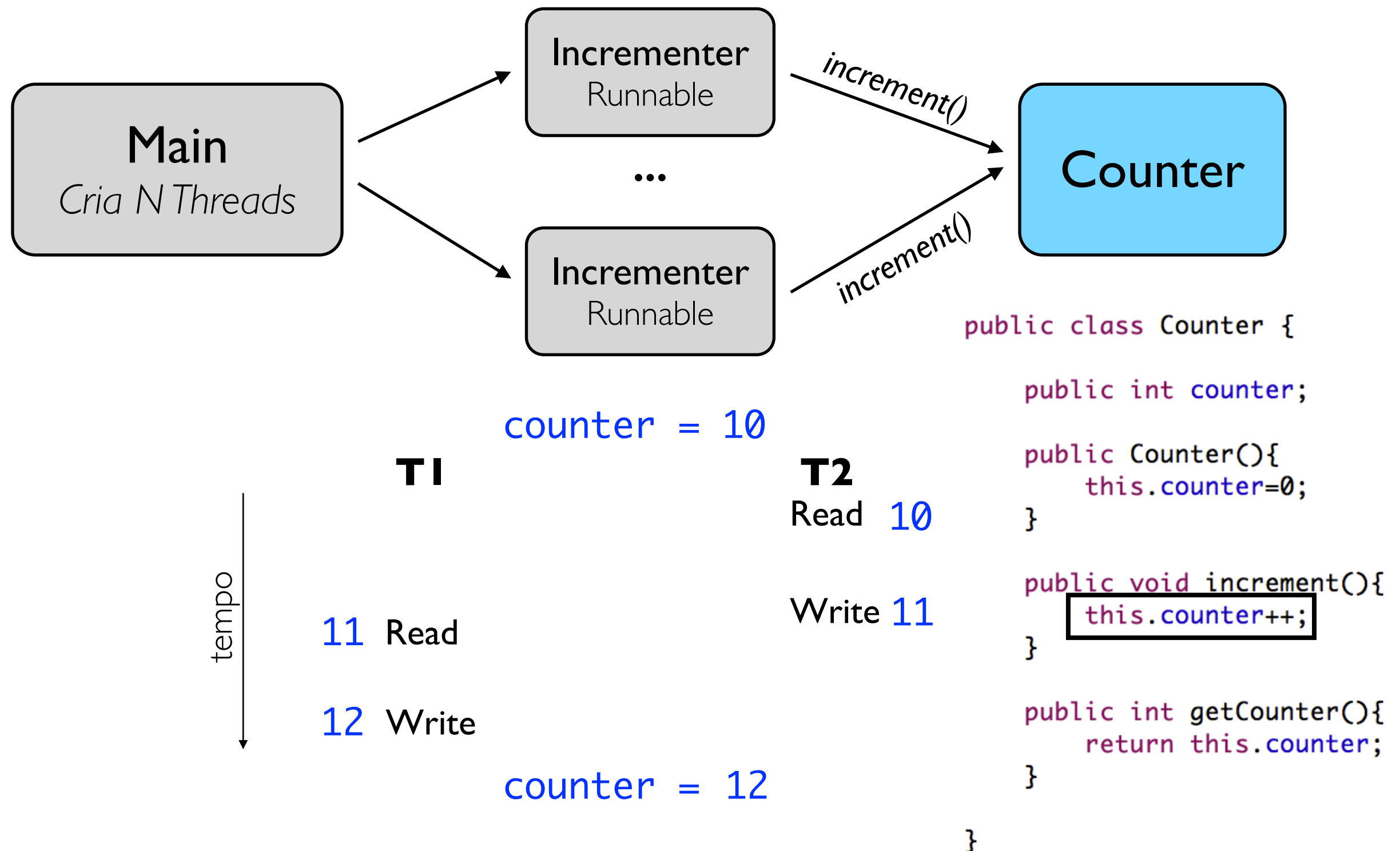
Exemplo: Programa com N threads que têm acesso a um único objecto de uma classe Counter. Cada thread deverá incrementar I vezes o contador. Thread Main imprime o valor do contador no final.



Exemplo: Programa com N threads que têm acesso a um único objecto de uma classe Counter. Cada thread deverá incrementar I vezes o contador. Thread Main imprime o valor do contador no final.



Exemplo: Programa com N threads que têm acesso a um único objecto de uma classe Counter. Cada thread deverá incrementar I vezes o contador. Thread Main imprime o valor do contador no final.



- Acessos concorrentes a recursos partilhados podem levar a resultados inesperados e a um comportamento errado do programa.
- **Exclusão mútua** é a propriedade que garante que dois processos ou threads não acedem simultaneamente a um recurso partilhado.
- Uma **secção crítica** é uma parte do programa onde os recursos partilhados são acedidos.
Proteger secções críticas do código.

```
public class Counter {  
  
    public int counter;  
  
    public Counter(){  
        this.counter=0;  
    }  
  
    public void increment(){  
        this.counter++;  
    }  
}
```

Secção crítica

Reentrant Lock

- Mecanismo utilizado para assegurar exclusão mútua
- Locks são adquiridos por threads (uma de cada vez)
- Depois de adquirido, garante acesso exclusivo aos recursos partilhados
- Métodos:
 - `ReentrantLock()` // construtor
 - `lock()` // adquirir lock e bloquear acesso
 - `unlock()` // libertar lock

<https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/ReentrantLock.html>