# Semáforos, Condições de Corrida e Trade-offs em Sistemas Concorrentes

## O que é um Semáforo?

Um **semáforo** é uma estrutura de sincronização usada para controlar o acesso de múltiplas threads a um recurso compartilhado. Ele funciona com base em um contador interno de permissões:

- `acquire()` — tenta obter uma permissão; se não houver, a thread bloqueia.
- `release()` — devolve uma permissão, liberando outra thread.

Quando configurado com apenas 1 permissão, o semáforo se comporta como uma forma de exclusão mútua, permitindo que apenas uma thread entre na seção crítica por vez.

Além disso, semáforos em Java fornecem importantes garantias de ordem e visibilidade, conhecidas como relação **happens-before**: tudo que acontece antes do `release()` de uma thread torna-se visível para a thread que fizer `acquire()` depois.

---

## Semáforo vs Condição de Corrida

Quando várias threads acessam e modificam o mesmo recurso sem sincronização, ocorre uma condição de corrida. Isso leva a:

- resultados incorretos,
- comportamento imprevisível,
- inconsistência de memória.

O semáforo elimina a condição de corrida, pois impede que mais de uma thread execute a seção crítica ao mesmo tempo.

---

## O Trade-off: Correção vs Throughput

Embora o semáforo resolva o problema de concorrência, ele impõe um custo:

- O acesso ao recurso fica serializado.
- Menos threads conseguem trabalhar ao mesmo tempo.
- Em muitos cenários, isso reduz o throughput geral.

Em contrapartida, o semáforo garante:

- correção,
- ordem entre operações,
- visibilidade entre threads,

- comportamento determinístico.