04. Entendendo Volatile

Date	@30/08/2022
Categoria	Java
Curso	Threads em Java 2: programação concorrente avançada

Tópicos

1. Resolvendo o desgin
2. Várias threads no valor
3. Entendendo Volatile
4. Variáveis, threads e memória!
5. Acesso direto à memória
6. Opcional: Simulando o problema
7. Mãos à obra: Refatorando o servidor
8. Para saber mais: java.util.concurrent.atomic
9. Download do código fonte

Entendendo Volatile

O que aprendemos?

- Threads possuem um cache.
    - Esse cache faz com que nem sempre todas as variáveis serão vistas e atualizadas de maneira atômica.

- A palavra chave `volatile` evita o uso desse cache e faz que as threads sempre acessem a memória principal.
- Como alternativa, podemos utilizar as classes do pacote `java.util.concurrent.atomic`
  - Vimos a classe `AtomicBoolean` como alternativa ao uso do `volatile`

Para saber mais: java.util.concurrent.atomic

Vimos na aula a classe `AtomicBoolean` para não precisar usar `volatile` ou `syncronized` ao acessar a uma variável. A classe faz parte do pacote `java.util.concurrent.atomic` onde podemos encontrar outras classes, como por exemplo `AtomicInteger` e `AtomicLong`.

https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/atomic/package-summary.html

No tutorial da Oracle é apresentado um pequeno exemplo como seria uma classe usando `syncronized` comparado com `AtomicInteger`. Vale à pena conferir:

https://docs.oracle.com/javase/tutorial/essential/concurrency/atomicvars.html