




Unidade 2

Seção 3

Programação Orientada a Objetos II

Webaula 3

Programação em Java utilizando elementos para
sincronização em Java



O objetivo desta webaula é apresentar os elementos de sincronismo de threads e classes que encapsulam a complexidade das threads, além de introduzir elementos mais específicos e complexos da programação concorrente.

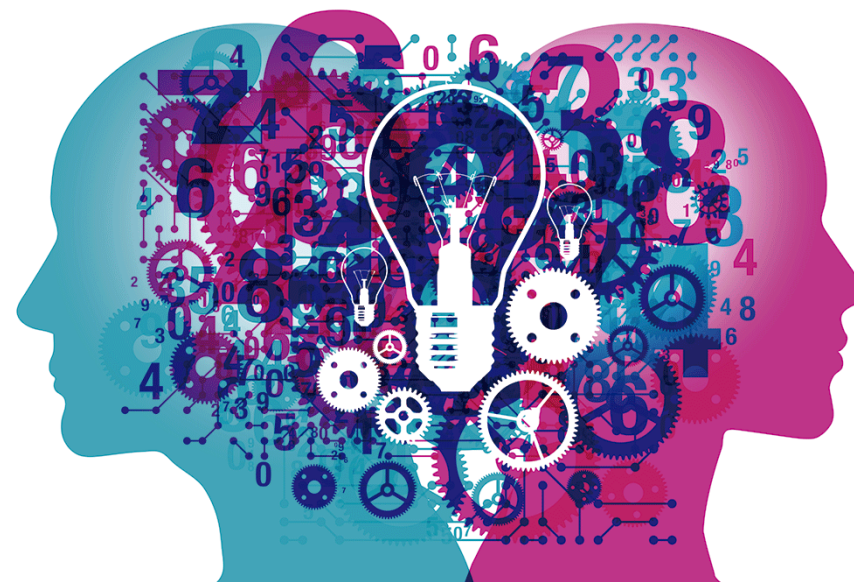
Introdução

Os recursos computacionais podem ser comparados a produtos de supermercados: existem diversos itens a serem comprados. Todavia, se dois clientes diferentes forem comprar o mesmo item, ao mesmo tempo, será necessário algum tipo de controle para evitar conflitos.

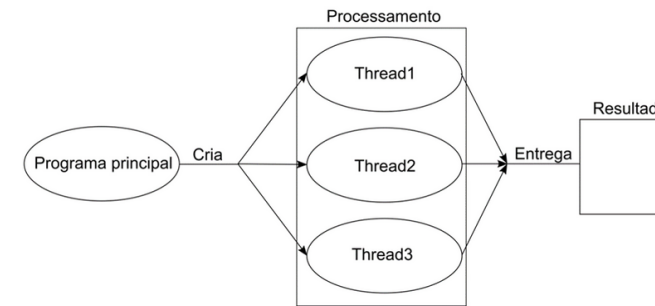
As threads em um sistema computacional podem ser comparadas aos clientes, e os itens do mercado são relacionados à memória, aos dispositivos de entrada e de saída ou a outros recursos do computador. Como forma de utilizar o hardware de maneira mais eficiente ou fazer com que uma aplicação seja capaz de executar diversas tarefas ao mesmo tempo, utilizamos as threads.

Exceções em um código orientado a objetos e paralelo

Existem diversas formas de tratar as exceções em um código orientado a objetos e paralelo, todavia existem problemas que podem ocorrer em sistemas concorrentes que não geram erros que são tratados pelos mecanismos do próprio Java. [Clique na imagem para saber mais.](#)



Pense em uma classe básica em que quantidades de elementos são somadas ou subtraídas. Nessa situação, imagine que diversas threads utilizam a mesma instância dessa classe para centralizar essas estatísticas. Esse tipo de cenário em programação é bem comum, e a imagem apresenta um caso em que um programa principal cria três threads para fazer um processamento e, ao final, todas elas escrevem seus dados em apenas um elemento central. **Clique na imagem para ampliá-la.**



Fonte: elaborada pelo autor.

O método sincronizado

É importante pontuar que quando duas threads fazem o acesso ao mesmo tempo, são necessárias estratégias da própria linguagem de programação, juntamente ao sistema operacional, para garantir que certos métodos sejam acessados por apenas uma thread por vez. Para isso existe a palavra reservada **synchronized**.

Essa palavra é utilizada no método que se deseja garantir que cada thread fará acesso de forma única para evitar inconsistência dos dados (HORSTMANN, 2016). Para isso, a *Java Virtual Machine* (JVM), que faz a execução dos códigos junto ao sistema operacional, consegue garantir que cada thread acesse os métodos de forma individual (FURGERI, 2015).

```
package U2S3;

1. public class ContadorSync {
    private int quantidadeAlunosCurso;

2.     public synchronized void incrementa()
        {
            quantidadeAlunosCurso++;
        }

3.     public synchronized void decrementa()
        {
            quantidadeAlunosCurso--;
        }
}
```

Fonte: elaborada pelo autor.

Contagem com métodos sincronizados

No quadro, temos um exemplo do uso do mecanismo de sincronização. Nas linhas 2 e 3 foram utilizados **métodos `synchronized`**, com isso, quando diversas threads acessarem esses métodos, a própria JVM, junto ao sistema operacional, consegue garantir que o acesso seja individual. Com os métodos **`synchronized`**, é possível produzir códigos paralelos confiáveis.

As classes *Timer* e *TimerTask*

Para evitar os problemas das questões de modelagem e ainda aumentar o acoplamento das classes que utilizam threads é threads, é possível usar as classes *Timer* e *TimerTask* (HORSTMANN, 2016). Para que o sistema funcione, é preciso implementar as duas classes, pois na classe *Timer* o método que inicia o processamento espera uma instância da classe *TimerTask*. Com isso, a parte que cuida do controle da thread (*Timer*) fica separada da que faz o processamento (*TimerTask*). Dessa maneira, a coesão das classes aumenta, criando formas de manutenção mais amplas.

Vantagens do uso do *Timer* e *TimerTask*

É importante mencionar que os sistemas *Customer Relationship Management* (CRM) são responsáveis pelo controle da relação para envio de propagandas, de brindes e de outros elementos entre uma empresa e seus clientes. Esses sistemas possuem milhares de linhas de código. Nesse sentido, o uso do *Timer* e *TimerTask* representa um aumento da coesão das classes e encapsulamento, assim, as manutenções ficam facilitadas.



Tome nota!

Estudamos, nesta webaula, parte de programação paralela. Você pode conhecer diversas formas de implementar esse poderoso recurso, porém, a escolha correta da maneira de fazer o processamento garante que um sistema seja mais robusto, com manutenção mais fácil e ainda que seus componentes possam ser reaproveitados. Não esqueça de recorrer ao livro didático para aprofundar os seus estudos!

Vídeo de encerramento







Bons estudos!