

Laboratório  
Concorrente

de

Programação

Lab 7 - 25.1

Strategies for  
Concurrent Access  
Control in Java

## Objetivo

O objetivo deste laboratório é exercitar a sincronização da execução de threads partindo de uma solução concorrente insegura de um problema real e desenvolvendo soluções concorrentes seguras utilizando diferentes abordagens na linguagem de programação Java, tais como: Semáforos, Monitores (synchronized) e Variáveis Atômicas (pacote `java.util.concurrent.atomic`).

Esperamos que vocês sigam os seguintes passos:

- 1) Analisem um código concorrente sem controle de concorrência que resolve um problema simples;
- 2) Desenvolvam três soluções concorrentes seguras:
  - a) Solução que usa Semáforos
  - b) Solução que usa monitores (synchronized)
  - c) Solução que usa variáveis atômicas

## Contextualização do Problema

Imagine um sistema web de e-commerce que recebe milhares de acessos concorrentes. Cada requisição pode:

- Registrar acessos (contagem de usuários que entram no site).
- Registrar compras realizadas.
- Registrar acessos finalizados sem compras ou falhas.
- Registrar falhas em requisições (por exemplo, erro 500 ou timeout).
- Para análise de desempenho, precisamos manter estatísticas seguras e consistentes:
  - Total de acessos
  - Total de compras
  - Total de falhas
  - Total de acessos sem compras ou falhas.
  - Contagem de usuários simultâneos conectados no momento (login/logout)

Tudo isso deve ser feito concorrentemente por várias threads que simulam usuários acessando o sistema.

A solução disponibilizada recebe como argumento o número de usuários simultâneos (threads) que acessam o sistema.

Lembrando que vocês devem desenvolver três versões para a solução concorrente considerando diferentes abordagens para controle de acesso concorrente. A partir do código disponibilizado, sua solução deve ser desenvolvida em etapas como descrito abaixo:

- **Etapa 1** – Desenvolver solução que faz o controle de acesso a regiões críticas usando Semáforos
- **Etapa 2** – Desenvolver solução que faz o controle de acesso a regiões críticas usando monitores (synchronized)
- **Etapa 3** – Desenvolver solução que faz o controle de acesso a regiões críticas usando variáveis atômicas

Todas as soluções devem ser desenvolvidas no diretório concurrent com os arquivos seguindo os seguintes nomes:

- 1) WebStatsSemMain.java
- 2) WebStatsSynMain.java
- 3) WebStatsAtmVarMain.java

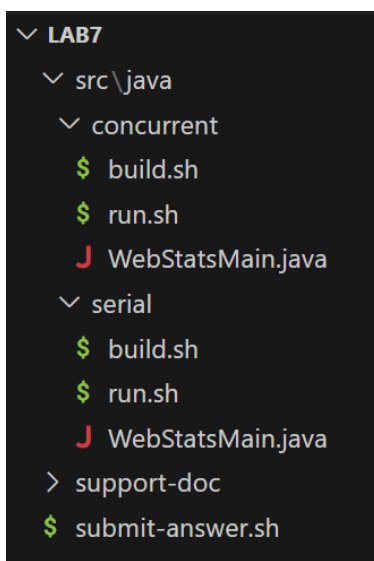
Como parte da entrega, você também deve descrever qual o resultado esperado para as estatísticas do sistema quando o código com controle de concorrência seguro é executado. Essa descrição também deve estar dentro do diretório concurrent no seguinte arquivo:

- 1) expected-result.txt

## Visão geral do código base

<https://github.com/giovannifs/fpc/tree/master/2025.1/Lab7>

O código está organizado de acordo com a hierarquia abaixo:



- **src/java/serial/**: Diretório que disponibiliza a implementação inicial do sistema sem controle de concorrência.
- **src/java/concurrent/**: Diretório onde você implementará o código solicitado ao longo deste lab. Inicialmente o conteúdo deste diretório é uma cópia do conteúdo do diretório serial.
- **scripts auxiliares nos diretórios serial e concurrent**:
  - *build.sh*: script para compilar o código do diretório
  - *run.sh*: script para executar as implementações do diretório. Você precisa alterar o [run.sh](#) do diretório concurrent para considerar as novas classes a serem desenvolvidas.
- **support-doc**: Disponibiliza documentação e tutorial sobre o uso de threads, semáforos, sincronizações e variáveis atômicas em Java.
- **submit-answer.sh**: Script que será utilizado para a submissão de sua resposta.

## Execução do código

1. Clone o repositório do código base

```
git clone [link do repositório]
```

2. Execução da versão serial do código

- a. Navegue até o diretório da implementação serial:

```
cd fpc/2025.1/Lab7/src/java/serial
```

- b. Compile o código

```
bash build.sh
```

- c. Execute a versão serial especificando como parâmetro o número de usuários simultâneos no sistema:

```
bash /run.sh 5000
```

## Prazo

19/08/2025 às 16:00h

## Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script `submit-answer.sh`, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

- 1) Crie um arquivo `lab7_matr1_matr2.tar.gz` somente com o “src” do repositório que vocês trabalharam. Para isso, supondo que o diretório raiz de seu repositório privado chama-se `lab7_pc`, você deve executar:

```
tar -cvzf lab7_matr1_matr2.tar.gz lab7_pc/src
```

- 2) Submeta o arquivo `lab7_matr1_matr2.tar.gz` usando o script `submit-answer.sh`, disponibilizado no mesmo repositório do laboratório:

```
bash submit-answer.sh lab7 path/lab7_matr1_matr2.tar.gz
```

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.