UNIVERSIDADE DO ESTADO DA BAHIA
Departamento de Ciências Exatas e da Terra, Campus I
Colegiado de Sistemas de Informação / **Semestre:** 2025.2
Disciplina: Linguagem de Programação III / **Professor:** Vagner Fonseca
Aluno: _____

Exercício 01

A Apolo Tech é responsável por um microsserviço crítico de autenticação. Para garantir a segurança e o desempenho, o microsserviço principal (AuthService) só pode se ligar à porta (SocketBind) e começar a aceitar requisições externas depois que seus quatro módulos essenciais de inicialização forem completamente carregados. Se o servidor iniciar antes da conclusão de qualquer um desses módulos (Configuração, Cache, Chaves de Criptografia, e Conexão de Log), ele pode falhar ou expor vulnerabilidades de segurança. Sabendo que o módulo de configuração leva em média 6 segundos para ser carregado, o de cache leva em média 9 segundos, o de chaves de criptografia em média 12 segundos e o da conexão de log em média 4 segundos.

A Apolo Tech precisa de uma solução robusta e concorrente para gerenciar essa dependência. Desta forma ela te contratou com o intuito de solucionar o seu problema, para isso você irá implementar um sistema de inicialização de servidor que só inicia a aceitação de conexões (simulada) depois que todos os módulos de configuração e segurança forem carregados.

Requisitos

1. Crie uma classe ServerInitializer que:
    a. Contém uma instância de CountDownLatch inicializada com o valor 4.
    b. Possui um método waitForInitialization() que chama latch.await().
    c. Possui um método startServer() que imprime a mensagem “Servidor Principal Online: Pronto para aceitar conexões (Socket.bind())” após o await() ser liberado.
2. Crie uma classe ModuleLoader que implemente Runnable. Cada instância desta classe representa o carregamento de um módulo (e.g., “Configuração”, “Segurança”, “Logs”, “Cache”).
    a. O run() deve simular um atraso no carregamento (e.g., Thread.sleep(1000) a 4000 milissegundos).
    b. Após o atraso, deve imprimir a mensagem “Módulo [Nome do Módulo] carregado.” E, em seguida, chamar latch.countDown() para sinalizar sua conclusão.
3. No método principal (main):
    a. Crie um ExecutorService (e.g., Executors.newCachedThreadPool()).
    b. Instancie o ServerInitializer com o CountDownLatch compartilhado.
    c. Submeta 4 instâncias do ModuleLoader ao Executor.
    d. Submeta a tarefa ServerInitializer.startServer() ao Executor imediatamente após a submissão dos ModuleLoaders.
    e. A mensagem do servidor só pode aparecer após as 4 mensagens de módulo carregado.

Saida esperada:
[CARREGANDO] Módulo Conexão de Log iniciando...

[CARREGANDO] Módulo Cache iniciando...
[CARREGANDO] Módulo Configuração iniciando...
[CARREGANDO] Módulo Chaves de Criptografia iniciando...
[OK] Módulo Conexão de Log carregado.
[OK] Módulo Configuração carregado.
[OK] Módulo Cache carregado.
[OK] Módulo Chaves de Criptografia carregado.

==========
Servidor Principal Online: Pronto para aceitar conexões (Socket.bind())
==========

[SISTEMA] Processo de inicialização concluído.