Especificação do Exercício de Concorrência

Infraestrutura de Software - 2024.1

Objetivo

O exercício de concorrência consiste num conjunto de exercícios, sem necessidade de interface gráfica, onde os alunos deverão desenvolver utilizando concorrência na linguagem Java. Que deverá ser feito individual ou em dupla seguindo as especificações descritas aqui.

Especificação

O desenvolvimento da atividade de cada dupla deve ser feito **APENAS** por seus integrantes, podendo os monitores da disciplina auxiliar com conceitos, explicações e conhecimentos gerais relevantes para e sobre a atividade. As consequências da detecção de plágio ficam a critério do professor.

O exercício deve ser desenvolvido no GitHub. A única linguagem de programação permitida para uso é Java. É recomendado que use a última versão disponível da linguagem.

Cada dupla deve apresentar suas soluções para o seu respectivo monitor que foi atribuido na planilha de temas.

O exercício será dividido em 3 entregas, cada entrega terá 2 problemas que deverão ser implementados, que ao final da 3 terceira entrega, terá 6 problemas implementados. Todos os problemas devem ser resolvidos para que funcionem de forma concorrente e adequada, aplicando os conhecimentos adquiridos na sala de aula e nas monitorias.

Contexto

Um cientista muito renomado decidiu que estava na hora de migrar 1% da população do planeta Terra para outro planeta. Para isso, ele criou, nesse novo planeta, uma nova cidade chamada New City. E para que as atividades comuns de uma cidade funcionem rapidamente, vários profissionais estão trabalhando em conjunto. Arquitetos e Engenheiros de Software também estão trabalhando na implementação de alguns sistemas dessa cidade. Logo, você deve estudar, discutir e apresentar as soluções para os problemas listados abaixo.

Entregas

Entrega 1 (21/06/24)

Problema 1: Sistema Bancário

O banco criou um sistema para compartilhar uma mesma conta entre os membros de uma família. Onde é possível fazer depósitos e saques, que podem acontecer simultaneamente. Deseja-se manter a uniformidade do saldo da conta, que por sua vez é utilizada por mais de um membro da família. Sua tarefa é criar duas operações (saque e depósito) que alteram uma variável chamada saldo e garantir que um cliente não consiga sacar mais do que a conta tem de saldo.

Observações:

- Lembre-se de simular várias pessoas (threads) sacando e depositando simultaneamente na mesma conta.
- Crie uma thread para cada pessoa que deseja fazer uma operação.

Problema 2: Construção de uma Ponte

Uma ponte está sendo construída, mas o espaço da ponte será bem estreita, com apenas uma única faixa. Mas, os carros (threads) trafegam nas duas direções. Portanto, é necessário alguma forma de sincronizar para que os carros não colidam. Na prática, quando a ponte está vazia, um carro (da esquerda ou da direita) pode entrar nela. Uma vez que o carro entra na ponte, ele precisa atravessá-la e depois sair. Só então um próximo carro (da esquerda ou da direita) pode usar a ponte. Sua tarefa é implementar um programa que simula essa situação. Considere ainda duas versões desse mesmo problema, uma onde a ponte tem o controle de fluxo (com sincronização) e outra onde a ponte não tem controle de fluxo (sem sincronização).

Observações:

- Crie uma thread para cada carro.

Entrega 2 (05/07/24)

Problema 3: Barbeiro da Cidade

Uma barbearia consiste em uma sala de espera com n cadeiras e a sala do barbeiro contendo a cadeira do barbeiro. Se não houver clientes para atender, o barbeiro vai dormir. Se um cliente entra na barbearia e todas as cadeiras estão ocupadas, o cliente sai da loja. Se o barbeiro estiver ocupado, mas houver cadeiras disponíveis, o cliente senta-se em uma das cadeiras livres. Se o barbeiro estiver dormindo, o cliente acorda o barbeiro. Escreva um programa para coordenar o barbeiro e os clientes.

Observações:

- Dica: Dá pra simular o tempo de descanso do barbeiro utilizando o sleep.
- Crie uma thread para cada cliente.

Problema 4: Restaurante

Um restaurante foi criado com apenas 5 lugares. Se um cliente chegar quando um dos lugares estiver vazio, ele/ela poderá sentar imediatamente. Mas, se o cliente chegar quando os cinco lugares estiverem ocupados, isto significa que os 5 clientes sentados estão jantando juntos e será preciso esperar todos os cinco clientes saírem antes que o próximo cliente possa sentar em um dos lugares.

Observações:

- Lembre-se que os clientes que não foram atendidos devem esperar numa fila.
- Crie uma thread para cada cliente.

Entrega 3 (19/07/24)

Problema 5: Transporte

New City está evoluindo e deseja criar um novo sistema para os ônibus da cidade. Onde a filosofia é que ninguém deve ir em pé no ônibus, ou seja, quando o ônibus antige sua capacidade máxima de assentos, ninguém mais entra. Logo, você deve modelar um sistema para resolver o seguinte problema: Os passageiros chegam a um ponto de ônibus e esperam por um ônibus. Quando o ônibus chega, todos os passageiros tentam entrar, mas quem chega enquanto o ônibus está embarcando tem que esperar pelo próximo ônibus. A capacidade do ônibus é de 50 pessoas. Se houver mais de 50 pessoas esperando, algumas terão que esperar pelo próximo ônibus. Quando todos os passageiros em espera tiverem embarcado, o ônibus poderá partir. Se o ônibus chegar e estiver sem passageiros na parada, ele deverá partir imediatamente.

Observações:

- Repare que não é formada uma fila para entrar no ônibus, quando o ônibus chega, todos tentam entrar ao mesmo tempo até que as cadeiras esgotem.
- Crie threads para cada passageiro.
- Simule que o tempo do ônibus passar na parada varia entre 1 e 3 segundos.

Problema 6: Banheiro Unisex

Está sendo implantado um novo projeto de banheiros unisex espalhados pela cidade. Mas há algumas regras nesses banheiros e você deve garantir que essas regras sejam seguidas.

1. Não pode haver mulheres e homens no banheiro ao mesmo tempo.
2. A capacidade máxima do banheiro é de 3 pessoas.

Observações:

- Perceba que as pessoas podem demorar mais ou menos tempo dentro do banheiro.
- Perceba também que isso pode gerar um deadlock, então você deve garantir que isso não aconteça.
- Crie threads para cada pessoa.

Restrições

As estruturas de dados permitidas são Array e ArrayList.

Classes, métodos e modificadores que abstraem o gerenciamento de threads e o controle de concorrência são proibidos, em especial os seguintes:

Permitido	Não permitido
• `ReentrantLock` e `Condition` • `CountDownLatch`, `CyclicBarrier`, `Phaser` e `Semaphore` • `SingleThreadedExecutor` e `SingleThreadedScheduledExecutor`	• Métodos e expressões síncronos (`synchronized`, `wait()` e `notify()`) • `ReadWriteLock` e `StampedLock` • Variáveis atômicas (ex. `AtomicBoolean`) • `volatile` • `BlockingQueue`, `ConcurrentMap` e `ConcurrentNavigableMap` • Queues da biblioteca `java.util.concurrent` • Threads leves • Vector • SwingWorker

Critérios gerais de avaliação

- **20% - Código:** boas práticas: código limpo, bem estruturado e indentado corretamente; nomes claros para objetos e métodos; comentários relevantes sobre o funcionamento e implementação.
- **40% - Concorrência:** uso adequado de concorrência, observando as restrições impostas, sem erros ou *bugs*.
- **40% - Explicação da Solução:** a solução deve ser explicada em detalhes na apresentação;
    - Como a dupla chegou na solução;
    - Porque escolheram essa solução;
    - Como a solução resolve o problema;
- Pontos poderão ser deduzidos da nota geral da entrega nas seguintes situações:
    - Erros de construção (*build*) **(-1)**;
    - Todos os problemas devem ser resolvidos utilizando concorrência. (**Não será considerado soluções sem a utilização de concorrência**);

Como e até quando entregar

As soluções deve ser entregue no classroom da disciplina na atividade correspondentes por apenas um membro da equipe. A entrega deve conter o link do GitHub com o commit que deverá ser corrigido pelos monitores. **Lembrando que apenas serão considerados os *commits* até a data de entrega**. Atrasos podem não receber a pontuação completa, à critério do professor. As datas limites de entrega para cada atividade estão abaixo. Além disso, os alunos devem marcar com o seu respectivo monitor a apresentação, a comunicação será feita através do discord da disciplina.

- Entrega 1: 21/06/24
- Entrega 2: 05/07/24
- Entrega 3: 19/07/24