Mecanismos de Sincronização

O Banheiro Unissex

Programação Concorrente - Bacharelado em Tecnologia da Informação
Instituto Metrópole Digital (IMD) - Universidade Federal do Rio Grande do Norte (UFRN)
Aluno: Gabriel Estácio de Souza Passos

Introdução

O objetivo deste estudo é aplicar conceitos de programação concorrente utilizando mecanismos de sincronização. Para tanto, foi utilizada a linguagem de programação Java, na sua versão 17, através da IDE Eclipse. A máquina que executou o programa possui como especificações: sistema operacional Windows, com 4GB de memória RAM e um processador Intel Core I3 de 10º geração, com capacidade de processamento de 2.1 GHz.

Metodologia

1. **Estrutura da Solução**

   O projeto possui uma interface Person, que é implementada por duas classes: Man e Woman, que fazem uma implementação básica dos gêneros envolvidos no problema, possuindo um atributo "gender", que será responsável por controlar quem poderá entrar na seção crítica. Essas classes, porém, não possuem a implementação de threads. Entrando efetivamente na parte da concorrência, o projeto possui uma classe chamada Bathroom, que implementa um banheiro, onde no nosso problema, é a implementação da seção crítica. Bathroom um atributo "capacity", que armazena a capacidade máxima de threads permitidas ali dentro simultaneamente, um atributo chamado "current_gender", que é responsável pelo controle do tipo de thread que pode acessar a seção crítica em um dado momento, e um terceiro atributo chamado "occupants", que armazena numa lista as threads atualmente dentro da seção crítica. Além das classes citadas e da classe Main, há duas classes chamadas ManThread e WomanThread, que implementam as threads, simulando o "comportamento" de homens e mulheres no nosso problema.

   A ideia geral é que, quando as threads são iniciadas, elas tentem entrar no banheiro (seção crítica). Ao tentar entrar, há três possibilidades: 1) o banheiro está lotado. Neste caso, o sistema exibe um comunicado que o banheiro está em sua capacidade máxima e suspende a thread até que haja vagas no banheiro; 2) o banheiro está sendo usado por pessoas do gênero oposto e isso não pode ser trocado no momento, pois o banheiro não está vazio. Assim, o sistema exibe um comunicado que o banheiro está sendo usado pelo gênero oposto e a thread é suspensa até que o banheiro esvazie; 3) o banheiro está vazio. Neste caso, o atributo que armazena o gênero que estava usando o banheiro por último se mantém igual, caso seja o mesmo gênero da próxima thread na fila, ou muda para o gênero oposto, caso o gênero seja diferente. Após isso, o sistema adiciona a thread na lista de ocupantes do banheiro (simbolizando que aquela pessoa entrou no banheiro) e imprime um comunicado que a thread está liberada para entrar na seção crítica Após isso, a próxima thread da fila é reativada.

Após terem entrado no banheiro (seção crítica), as threads precisam sair e liberar espaço para as próximas threads. O método que controla essa saída é executado após o método de entrada, e sua lógica consiste em retirar a thread da lista de ocupantes do banheiro e reativar a próxima thread da fila.

2. **Lógica de sincronização**
   Para que houvesse sincronização, foi utilizada uma abordagem com monitor e variável de condição. Os métodos do monitor são declarados como métodos *synchronized*, fazendo com que o banheiro só seja acessado quando o lock for liberado. A suspensão e reativação das threads são realizadas pelos métodos wait() e notify(), respectivamente, baseadas nas lógicas das variáveis de condição, que no nosso caso são *quantity* e *current_gender* da classe Bathroom.

   A partir das três possibilidades de quando uma thread tenta entrar na seção crítica, descritas no tópico anterior, a aplicação da suspensão e reativação das threads funciona através de laços condicionais: no 1º caso, a thread fica bloqueada enquanto o banheiro estiver lotado; já no 2º caso, a thread fica bloqueada enquanto o gênero que está utilizando o banheiro é diferente do seu e o banheiro não está vazio (se estivesse, seria possível fazer a troca do gênero do banheiro); e no 3º caso, não há laço, apenas alteração do gênero, caso necessário, e adição da thread na lista de "ocupantes do banheiro".

   Desta forma, não é necessário que uma thread de gênero diferente da primeira thread que acessou a seção crítica aguarde que todas as threads do gênero oposto ao seu sejam executadas para, então, ela ser executada. Threads de gêneros diferentes podem alternar seu uso da seção crítica.

Instruções de Compilação e Execução

Para compilar o projeto, basta executar o seguinte comando em um terminal dentro da pasta *src* do projeto:

javac Main.java concurrency/ManThread.java concurrency/WomanThread.java support/Bathroom.java support/Man.java support/Person.java support/Woman.java

Para executar o projeto, basta executar o seguinte comando em um terminal dentro da pasta *src* do projeto:

java Main <x> <y>

onde <x> e <y> são argumentos que serão passados para o programa, sendo <x> o valor da capacidade máxima do banheiro, e <y> o número de threads "Man" e "Woman" a serem criadas e executadas pelo programa.

Para compilar e executar o programa pelo Eclipse, é necessário apenas criar uma nova Run Configuration como Java Application e adicionar os dois argumentos descritos acima.