UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
DIM0124 - Programação Concorrente - TURMA 01

Relatório sobre implementação de banheiro unissex utilizando concorrência

João Paulo Gomes Siqueira
Rodrigo Gomes da Rocha

Natal - RN
2022

1 Metodologia

O funcionamento do banheiro depende da garantia de algumas coisas: que homens e mulheres não estejam no banheiro ao mesmo tempo, que os primeiros a chegarem sejam os primeiros a usar o banheiro (garantindo justiça) e também que nem homem nem mulheres deixem de ter oportunidade de entrar no banheiro (impedindo *starvation*).

Para garantir ordem de chegada foi usada uma fila. Assim que uma pessoa (representada por uma thread) chega para usar o banheiro, é adicionada ao fim da fila. Para garantir que homens e mulheres não estejam no banheiro ao mesmo tempo, são separadas as n primeiras pessoas do mesmo sexo na fila para serem executadas juntas.

Essa abordagem também cumpre também o último item, por evitar que, por exemplo, já tendo mulheres no banheiro, entrem mais mulheres indefinidamente sem que o banheiro esvazie. Caso isso acontecesse, aconteceria um "starvation" das threads referentes a homens.

Em toda a execução do programa existem duas threads em execução: a primeira alimentando a fila com pessoas constantemente, e a outra retirando pessoas para a execução quando elas podem ser executadas juntas. Segue-se o padrão de produtor e consumidor.

A sincronicidade entre as threads produtoras e consumidoras é garantida através do uso da fila chamada "BlockingQueue" pertencente a biblioteca de concorrência do java. A implementação dessa fila impede que as duas threads acessem a fila ao mesmo tempo.

Já a execução das threads é realizada usando o Executor FixedThreadPool. A quantidade de threads executadas é limitada pela quantidade de vagas no banheiro. Para esse programa foi pré-determinado que o banheiro tem somente quatro vagas. Nos casos em que o número de homens ou mulheres em sequência excedem o número de vagas, as threads são enviadas da mesma forma para a execução. O pool de threads tem a capacidade de lidar automaticamente com as threads em excesso, alocando as threads para executar a próxima pessoa a medida que alguma das threads, representando vaga no banheiro, for liberada.

Para adicionar as threads no executor é utilizado o método "invokeAll". A execução desse método bloqueia a thread consumidora até o fim da execução de todas as threads. Quando o consumidor é desbloqueado ele pode continuar o processo de adicionar as próximas n mulheres ou homens para a execução.

2 Corretude

A garantia de corretude é dada pela utilização da BlockingQueue, que é uma implementação de fila com “Thread safe” em Java, ou seja garante o acesso correto de somente uma Thread por vez a fila de pessoas.

Ainda podemos destacar que será respeitada as restrições de acesso ao banheiro já que o método “invokeAll” do Executor Service garante que todas as pessoas da ponta da fila, selecionadas para entrar no banheiro (variável “pessoasParaEntrar”), terão terminado sua execução, assim liberando espaço pras próximas pessoas da fila.

3 Dificuldades

Nossa primeira dificuldade foi a descoberta de uma proposta de solução, dado que se respeitasse as regras de uso do banheiro. Ao descobrirmos uma proposta de solução, utilizar uma fila e selecionar todos do mesmo sexo da ponta fila até quando possível para entrar no banheiro, nossa próxima dificuldade foi garantir que a execução esperasse todas as pessoas sairem para utilizar.

Nossa implementação usa fixedThreadPool, que, por exemplo, seja selecionados seis pessoas para entrar (seis pessoas do mesmo sexo da ponta da fila), e o banheiro tenha apenas quatro vagas, a Thread Pool se encarregará de utilizar somente quatro Threads (pessoas) por vez.

Sendo assim, nosso maior desafio era garantir que se esperasse que todas as pessoas selecionadas acabassem sua execução. Para isso estudamos a possibilidade de utilizar o método awaitTermination() do Executor Service, ou ainda o uso de CountDownLatch, mas nenhuma delas foi utilizada. Fizemos uma primeira versão em que a partir de um loop nos objetos de “pessoasParaEntrar” esperava-se o retorno de cada tarefa. Mas por fim utilizamos o método InvokeAll() do Executor Service, que receberia uma lista de Callables, submeteria sua execução e aguarda o final de todas elas, o que se encaixa perfeitamente em nossos objetivos.

2 Execução

Para executar o programa, faça o clone do projeto para sua máquina:

git clone https://github.com/jopak19/banheiro-unissex.git

Importe o projeto como um projeto Java Maven em sua IDE de escolha. A partir do arquivo BanheiroUnissex.java, onde se encontra o método main, execute/build o programa.