UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

Trabalho Prático – Mecanismos de Sincronização

Paulo Vitor Fernandes Andrade

Rita de Cássia Chagas da Cruz

Natal - RN

Dezembro, 2022

Sumário

1 Introdução

Este relatório tem como objetivo documentar a solução proposta ao problema apresentado. O problema em questão consiste em um banheiro unissex para o qual deve ser feito a gerência de uso de modo que existe uma capacidade máxima de lotação que deve ser respeitada e apenas pessoas de mesmo gênero podem utilizar o banheiro simultaneamente. Nesse sentido, foram aplicados elementos de programação concorrente, threads em especial, para modelar computacionalmente a situação.

2 Solução proposta

Para solucionar o problema do banheiro unissex, optamos por utilizar o método de bloqueios, com *ReentrantLock,* para fazermos a sincronização, além disso, utilizamos também variáveis para ajudar no controle da execução do programa, como as variáveis *currentGender, currentCapacity, person - que é um LinkedHashSet* - e métodos auxiliares que nos ajudaram a analisar a situação atual do banheiro. Os tópicos abaixo discorrem sobre o detalhamento de algumas perspectivas da solução proposta para o problema.

2.1 Lógica de sincronização utilizada

Para sincronização, fizemos a utilização de locks, mais especificamente o *ReentrantLock,* que funciona de maneira similar a utilizar a palavra-chave synchronized, basicamente ele possui dois métodos importantes, o lock() e unlock(). O lock() para fazer o bloqueio daquele recurso enquanto a thread está utilizando e o unlock() para liberá-lo. A ideia utilizada é que cada pessoa (thread) ao tentar utilizar o banheiro, verifique se ela é do mesmo gênero das pessoas que estão dentro do banheiro, se o banheiro não está cheio e se ela já não está lá dentro. Caso essas condições sejam verdadeiras, ai sim a thread realiza a tentativa de bloqueio do recurso com o método lock(), entra no banheiro e dá unlock(). Após realizar o seu trabalho, é dado um bloqueio novamente e, após sair, um unlock(). Além disso, para garantimos uma

condição de justiça, inicializamos o ReentrantLock com o valor true, para que a thread que será executada seja aquela que estava a mais tempo esperando, logo, evita-se o problema de starvation.

2.2 Corretude da solução

Para a garantia da corretude da solução com relação à concorrência, foi utilizada a classe ReentrantLock, que é implementação das interfaces Serializable e Lock. Pela implementação da classe nativa do Java, apenas uma thread por vez pode adquirir o lock e todo acesso ao recurso compartilhado requer que o lock seja adquirido primeiro, sendo que o ReentrantLock pertence à thread que deu lock com sucesso pela última vez.

No caso da solução implementada neste trabalho, se pessoas do gênero X estão utilizando o banheiro no momento que uma pessoa deste mesmo gênero solicita uso, considerando que o banheiro não está lotado, a última pessoa a adquirir o lock (i.e., última pessoa a entrar no banheiro) “repassa” para a pessoa solicitante.

3 Compilação e execução

Para compilar o programa, o comando é

```
$ javac Manager.java
```

e, após compilado, pode ser executado com

```
$ java Manager -maxC n
```

sendo n a capacidade máxima do banheiro. Este argumento é opcional e seu valor padrão é 5.