UFRN - UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
IMD - INSTITUTO METRÓPOLE DIGITAL
CURSO DE GRADUAÇÃO EM TECNOLOGIA DA INFORMAÇÃO

TRABALHO PRÁTICO - MECANISMOS DE SINCRONIZAÇÃO

Docente: Prof. Everton Ranielly de Sousa Cavalcante

Discentes: 20200050108 Lucas Henrique de Alencar Rodrigues
20200043883 Humberto Vitalino da Silva Neto

Natal
Dezembro de 2022

INTRODUÇÃO

Para a implementação do problema do “banheiro unissex”, desenvolvemos um software feito em Java, utilizando-se da IDE Eclipse, versão 2022-09.

O projeto, hospedado neste link, conta com 3 classes principais que serão exploradas com mais detalhe posteriormente. Para o versionamento, utilizamos das tecnologias disponibilizadas pelo Git e Github.

CLASSE “PRINCIPAL”

A classe principal, que conta com o método Main, de modo geral, é responsável por inicializar todas as threads que serão utilizadas durante a execução. Para isso, inicialmente o programa requisita os dados específicos da execução (capacidade de pessoas e quantidades de pessoas na fila).

Após inserir os dados, o banheiro é inicializado com a capacidade de pessoas adicionadas pelo usuário e, de forma aleatória, define a quantidade de homens e mulheres até chegar ao número máximo de pessoas definido também pelo usuário no passo anterior. As pessoas são adicionadas à ArrayList<> de pessoa.

Por fim, o método Main inicia todas as threads e aguarda o fim da execução do programa.

Além do método Main, a classe Principal conta com outros dois métodos privados, sendo eles o método inserirValores(), e o método proximaPessoaSeraHomem().

O método inserirValores() é responsável por requisitar do usuário os dados necessários para a execução, e o método proximaPessoaSeraHomem() é responsável por, de forma aleatória, definir se a pessoa será homem ou mulher.

```
private static boolean proximaPessoaSeraHomem() {
  if (random.nextBoolean()) {
    return true;
  } else {
    return false;
  }
}
```

Imagem 01 - Método proximaPessoaSeraHomem()

CLASSE “PESSOA”

A classe “Pessoa” possui 4 atributos privados: Nome, sexo, tempo de uso do banheiro e o banheiro que será utilizado. Conta também com todos os getters, setters e construtores.

```
public class Pessoa extends Thread{

    private String nome;
    private Sexo sexo;
    private int tempoUsoBanheiro;
    private Banheiro banheiro;
```

Imagem 02 - Classe Pessoa e seus atributos privados

Como a classe extende a classe Thread, também é necessário sobrescrever o método “Run”.

```
@Override
public void run() {
    banheiro.tentarUsarBanheiro(this);

}
```

Imagem 03 - Sobrescrita do método Run

No seu construtor, a classe define de forma randômica o tempo de utilização do banheiro e seu sexo é definido na classe Principal, como foi dito anteriormente.

```
public Pessoa() {
    Random rng = new Random();
    this.tempoUsoBanheiro = (rng.nextInt() % 6 + 10)*1000;
}

public Pessoa(String nome, Sexo sexo, Banheiro banheiro) {
    super();
    this.nome = nome;
    this.sexo = sexo;
    this.banheiro = banheiro;
    Random rng = new Random();
    this.tempoUsoBanheiro = (rng.nextInt() % 6 + 10)*1000;
}
```

Imagem 04 - Construtores da classe Pessoa

Quando essa classe é executada na classe principal, o método “Run” faz uma requisição para a classe Banheiro, por meio do método tentarUsarBanheiro(), passando como parâmetro o próprio this.

CLASSE “BANHEIRO”

A classe “Banheiro” possui 4 atributos privados: quantidade máxima de pessoas, sexo da vez um atributo para definir quantas pessoas estão utilizando o banheiro nesse

momento. Também possui dois semáforos: um semáforo para definir se o banheiro está vazio e outro semáforo para manter a ordem da fila. A quantidade máxima de pessoas utilizando o banheiro é definido pelo usuário na classe Principal.

```
public class Banheiro extends Thread{

    private int qtdMaxima;
    private Sexo sexoVez;
    private int qtdPessoasUsandoBanheiro;

    private Semaphore semUsandoBanheiro;
    private Semaphore semOrdemChegada;
```

Imagem 05 - Classe Banheiro e seus atributos

O método mais importante da classe é tentarUsarBanheiro(). Que serve para que o banheiro possa ser utilizado pelas pessoas e, com o auxílio dos semáforos, manter a ordem de sua utilização.

Além do método tentarUsarBanheiro(), a classe conta com outros três métodos para auxiliar no uso do banheiro. São estes: usandoBanheiro(), sairBanheiro() e pedirPraUsarBanheir().

O método usandoBanheiro() serve para dar um Sleep na utilização da thread Banheiro pelo tempo definido na classe Pessoa. O método sairBanheiro() atualiza o valor da quantidade de pessoas que estão utilizando o banheiro no momento atual (logo após uma das pessoas sair do banheiro) e também serve para resetar o sexo para NENHUM, caso o banheiro esteja vazio. Por fim, o método pedirPraUsarBanheiro() atualiza o semáforo que verifica se tem alguma pessoa utilizando o banheiro, atualiza o parâmetro que define a quantidade de pessoas que estão utilizando o banheiro e define o sexo das pessoas que estão utilizando o banheiro no momento.

```
private synchronized void pedirPraUsarBanheiro(Pessoa pessoa) throws InterruptedException {
    while(!this.sexoVez.equals(pessoa.getSexo()) && !this.sexoVez.equals(Sexo.NENHUM)) {
        this.wait();
    }

    this.semUsandoBanheiro.acquire();
    this.sexoVez = pessoa.getSexo();
    this.qtdPessoasUsandoBanheiro++;
}
```

Imagem 06 - Método Sincronizado pedirPraUsarBanheiro

```
private synchronized void sairBanheiro(Pessoa pessoa) {
    this.qtdPessoasUsandoBanheiro--;

    if(this.qtdPessoasUsandoBanheiro == 0) {
        this.sexoVez = Sexo.NENHUM;
        this.notifyAll();
    }
}
```

Imagem 07 - Método sincronizado sairBanheiro

TIPO ENUM “SEXO”

Para tornar o código mais legível, também foi definida a Enumeração chamada Sexo, que possui 3 valores possíveis: NENHUM, HOMEM e MULHER. Esse tipo é utilizado tanto para definir o sexo das pessoas que estão na fila, quanto para definir qual o sexo das pessos que estão utilizando o banheiro no momento atual.

```
public enum Sexo {
    HOMEM("Homem"),
    MULHER("Mulher"),
    NENHUM("Nenhum");


    private String label;

    private Sexo(String label) {
        this.label = label;
    }

    public String getLabel() {
        return label;
    }
}
```

Imagem 08 - Enum Sexo

SINCRONIZAÇÃO DO SISTEMA E CORRETUDE DA SOLUÇÃO

Para realizar a sincronização do sistema, e por consequência, garantir a corretude da solução, utilizamos dois semáforos. Um para definir se o banheiro se encontra vazio (*semUsandoBanheiro*) e outro para garantir que a fila seja obedecida (*semOrdemChegada*). Esses semáforos são utilizados nos métodos da classe Banheiro. Para garantir a sincronização, também se usa a palavra-chave *syncronized* nos métodos “pedirPraUsarBanherio” e “sairBanheiro”. A escolha de mantê-los sincronizados é essencial para que a fila siga de maneira correta, para que não haja problemas no compartilhamento do parâmetro “qtdPessoasUsandoBanheiro”.

EXECUÇÃO DO SISTEMA

Para executar o programa que está hospedado neste link, deve-se usar o terminal do sistema operacional ou executar a aplicação do eclipse.

Utilizando o terminal do sistema operacional: Abrir o terminal, entrar na pasta Executável .jar, executar o comando 'java .jar BanheiroUnissex.jar', Informar a capacidade de pessoas no banheiro, se informar um valor invalido será utilizado o valor padrão, Informar a quantidade de pessoas que vão usar o banheiro, se informar um valor invalido será utilizado o valor padrão (100).

Utilizando o eclipse para executar o programa: Executar a classe Principal.java, Informar a capacidade de pessoas no banheiro, se informar um valor invalido será utilizado o valor padrão, informar a quantidade de pessoas que vão usar o banheiro, se informar um valor invalido será utilizado o valor padrão (100).