CENTRO UNIVERSITARIO UNIEURO

Sistemas de Informação - SI

Programação Concorrente e Distribuída

Relatório referente ao Exercício Prático 02 - EP02

Problemática da Montanha Russa

Gustavo Lopes dos Santos

CPD: 53158

Lucas Vinicius Lima Braga de Amorim

CPD: 54771

Brasília/DF

Maio 2022

RESUMO

Para exemplificar a visualização do problema, vamos usar o exemplo de uma Montanha Russa. A montanha Russa possuí três processos que devem ser seguidos em ordem para seu funcionamento, deve existir um trilho, um carrinho e pessoas para aproveitarem o passeio. Na problemática explorada, usamos conceitos aprendidos na matéria de Programação Concorrente e Distribuída em um programa escrito em Java.

SÚMARIO

1. **Formulação do Problema**
2. **Implementação**
3. **Manual de utilização do programa**
4. **Resultados**

1. Formulação do Problema

Os três processos citados são executados de forma paralela, assim, um algoritmo sequencial não teria a possibilidade de realizar a simulação de uma montanha russa. Neste paralelismo, enquanto os passageiros estão entrando na fila, os carros estão sendo ocupados e quando chegar a sua lotação máxima de 4 passageiros eles são liberados para acessar o trilho. Somente um carro pode estar no trilho por vez.

Apesar do uso de threads para realizar o paralelismo, é necessária uma sincronização para o acesso de uma área crítica, visto que as threads trabalham de forma simultânea, logo a leituras das variáveis e a execução das funções ocorrem todas de uma só vez.

Neste sistema, além do enfileiramento dos passageiros, a ocupação do(s) carro(s) e a volta no trilho, também foi implementando funções para calcular o tempo de chegada dos passageiros na fila, desembarque do carro, e cálculos dos tempos mínimo, máximo e máximo do passageiro na fila e o tempo total e individual dos carros no trilho.

2. Implementação

É utilizado quatro classes (MontanhaRussa, Carro, Passageiro e a Main), sendo que as classes Carro e Passageiro utilizam a interface Runnable.

Descrição das Classes:

- MontanhaRussa: É guardada as variáveis iniciais do sistema (N: quantidade de passageiros, M: quantidade de carros, C: capacidade do carro, TE: tempo de embarque e desembarque passageiros, TM: tempo da volta no trilho, TP_min e TP_max tempo da chegada dos passageiros), é criada também as variáveis do semáforo para controlar a quantidade de carro no trilho e da ocupação do carro, a variável para controlar a quantidade de passageiros que rodou no trilho e também a fila dos passageiros utilizando a biblioteca Queue (LinkedList).
- Passageiro: Esta classe é usada, para inserir os passageiros na fila. Cada passageiro é um thread diferente, com isso é utilizando o método synchronized, para permitir que um passageiro entre na fila por vez. A Fila é criada a partir da biblioteca LinkedList citado na classe MontanhaRussa. Quando um passageiro entra na fila começamos a pegar seu tempo inicial.
- Carro: Nesta classe, é realizado o embarque e desembarque dos passageiros e a volta no trilho. Cada instancia da classe Carro, é criado um thread e logo é iniciada, que valida se a quantidade de passageiro na fila é igual a capacidade do carro ou se está nos últimos passageiros para realizar a ocupação do carro. A cada passageiro que entra no carro aguardamos o TE. Este passageiro é retirado da fila da MotanhaRussa, e inserido em uma fila criada na própria classe. Está fila criada na classe Carros verifica quantos passageiros estão no carro, sendo que a quantidade desses passageiros tem

que ser menor ou igual a C. Após a ocupação do carro a thread executa a função que simula o trilho, e, começa a contar o tempo que a thread está no trilho. Depois do termino do TM chama a funcionalidade de desembarque de funcionário. No desembarque, é guardado o valor da saída dos passageiros e quantos passageiros saíram. A quantidade de passageiros que realizaram o desembarque é necessária, para fechar a simulação do brinquedo e calcular a média do tempo de espera dos funcionários na fila. Na função de embarcar funcionário, e no trilho utilizamos os semáforos criados na classe MotanhaRussa.

- Main: Nesta classe que está o método de inicialização do programa. Além de possui a criação dos objetos das classes MontanhaRussa, Passageiros e Carros, e ainda, demostrar o relatório dos tempos gastos durante a execução do brinquedo. Para inicializar, é exibido um menu para o usuário selecionar uma opção, nesta opção serão preenchidas as variáveis de inicialização da classe MontanhaRussa. Para a manipulação dos threads, é criado duas filas com o LinkedList, uma de passageiros e Carros. Os objetos Passageiros e Carros, dentro das filas é possível controlar e manipular a execução dos threads. E, buscar os tempos dos passageiros na fila e dos carros no trilho. Estes tempos, são demostrando como um relatório no fim da execução do programa. Neste relatório são exibidos o mínimo, máximo e a média do tempo que os passageiros estiveram na fila e o tempo no trilho de cada carro ou tempo de todos os carros no trilho.

3. Manual de utilização do programa

3.1. Rodar o programa o programa em um executável Java.

3.1.1. Escolher uma das 4 opções.

```
Escolha:[1] para 1 carro e 52 passageiros
        [2] para 2 carros e 52 passageiros
        [3] para 3 carros e 100 passageiros
        [4] carros e passageiros personalizados
Escolha: 4
```

3.1.2. Caso escolheu personalizada, digitar os valores nos campos corretos.

```
Entre com o número de pessoas no parque:8
Entre com a quantidade de carros: 2
```

3.1.3. As Threads Passageiros e Carros serão criadas.

```
Passageiro criado Id: 0
Passageiro criado Id: 1
Passageiro criado Id: 2
Passageiro criado Id: 3
Passageiro criado Id: 4
Passageiro criado Id: 5
Passageiro criado Id: 6
Passageiro criado Id: 7
Carro 0 Criado
Carro 1 Criado
```

3.1.4. As Threads Passageiros entram na fila, quando 4 passageiros estão na fila elas embarcam em um carro, se não houver nenhum outro carro na montanha russa, esse carrinho vai começar a rodar.

```
Entrou na fila: Passageiro 0
Entrou na fila: Passageiro 1
Entrou na fila: Passageiro 2
Entrou na fila: Passageiro 3
O passageiro 0 embarcou no carro 0
O passageiro 1 embarcou no carro 0
O passageiro 2 embarcou no carro 0
O passageiro 3 embarcou no carro 0
Carinho 0 entrou no trilho.
```

3.1.5. Após o tempo de passeio, o carro volta e os passageiros desembarcam do carro, o carro fica livre para um novo passeio, e assim até finalizar a fila.

```
Carinho 0 voltou de viagem.
```

3.1.6. Resultado

```
Tempo minímo da fila: 20s
Tempo máximo da fila: 89s
Tempo Média da fila: 49s
Carro 0 tempo movimentado: 120s
Carro 1 tempo movimentado: 120s
Tempo movimentado: 240
```

4. Resultados

Resultado dos Valores pré-estabelecidos.

Caso 1:

```
Tempo minímo da fila: 20s
Tempo máximo da fila: 997s
Tempo Média da fila: 505s
Carro 0 tempo movimentado: 1560s
Tempo movimentado: 1560
```

Caso 2:

```
Tempo minímo da fila: 20s
Tempo máximo da fila: 716s
Tempo Média da fila: 319s
Carro 0 tempo movimentado: 840s
Carro 1 tempo movimentado: 720s
Tempo movimentado: 1560
```

Caso 3:

```
Tempo minímo da fila: 20s
Tempo máximo da fila: 1282s
Tempo Média da fila: 544s
Carro 0 tempo movimentado: 960s
Carro 1 tempo movimentado: 960s
Carro 2 tempo movimentado: 1080s
Tempo movimentado: 3000
```