**Universidade Federal do Pampa – UNIPAMPA**

**Engenharia de Software**

**Sistemas Operacionais**

Trabalho Prático 1

**Problemas Clássicos**

Alunos:

Ícaro Machado Crespo – 1801560688

Ketrin Diovana Alves Rodrigues Vargas – 1801570702

Alegrete, 13 de outubro de 2019.

# Introdução

Este trabalho visa a implementação de problemas clássicos da computação no que diz respeito à *Threads*. Estas serão desenvolvidas na linguagem java e seu código se encontra no link abaixo para acesso. O objetivo é implementar os problemas: (i) produtor-consumidor com buffer limitado; (ii) jantar dos filósofos; (iii) problema da montanha russa, de maneira que seja possível solucionar o problema eficaz e com cobertura de testes para assegurar a integridade do sistema. O desenvolvimento fora realizado por Ícaro Crespo e Ketrin Vargas e vem como parte da aquisição de nota da disciplina de Sistemas Operacionais, ministrada pela Profª. Dra. Aline Vieira de Mello.

O problema do produtor-consumidor tem como prerrogativa a questão de abastecimento de produtos. A “vida” do produtor é baseada em manter populado os produtos para que o consumidor possa os utilizar. Já o problema do jantar dos filósofos consiste em todos os filósofos quererem jantar, porém necessitam de dois garfos para comer a massa. Os garfos sãos dispostos entre todos os filósofos de forma que recebam apenas um e tenham que aguardar que o filósofo ao lado largue o seu. Por fim, o problema da montanha russa é baseado em um carrinho deste brinquedo que carrega passageiros, com uma capacidade máxima, onde eles podem entrar através das diversas portas do carrinho. O carrinho inicia o trajeto assim que atingir a sua capacidade máxima.

# Desenvolvimento

O trabalho fora desenvolvido em JAVA, onde ambos os membros desenvolveram em conjunto os três problemas impostos. Fora utilizado como ferramenta de sincronização e controle de versionamento um repositório, denominado “Problemas\_Classicos”, na plataforma Github.

Para o desenvolvimento do problema clássico do produtor-consumidor, foram feitas classes para o consumidor, produtor, o buffer de dados e uma main para rodar aquela parte do sistema.

Em relação ao problema do produtor-consumidor, foram utilizadas as classes Buffer, Consumidor e Produtor, além de uma classe Main para iniciar aquele escopo do sistema. Já o problema do jantar dos filósofos, foram utilizadas as classes Filósofo, Garfo e Escalonador, e da mesma maneira, uma classe Main para executar este trecho. No que diz respeito ao problema da montanha russa, foram elicitadas e montadas as classes Carrinho, Maquinista, MontanhaRussa, Fila e AlimentaFila que possui a função de gerar passageiros a entrarem no brinquedo. Além disso, houve a criação de uma classe Main.

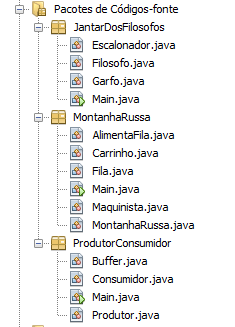


Figura 1: Distribuição das classes aos problemas do trabalho.

A estrutura do projeto consiste na divisão de testes e código fonte da solução. Os códigos estão alocados em */src*, separados por pacotes de acordo com o problema clássico. O mesmo é válido aos testes, no que diz respeito a sua separação. Sua localização é */test*. Abaixo são apresentadas figuras de 2 a 8 que demonstram os métodos e trechos específicos relevantes para desempenhar a função básica de cada problema.

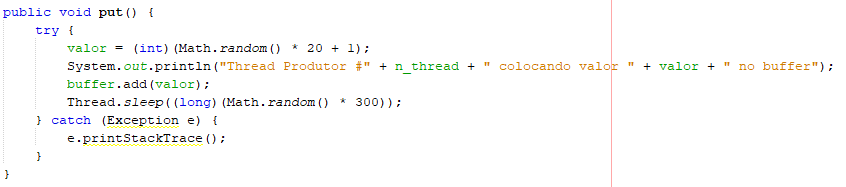


Figura 2: Método de adição à Thread, produtor.

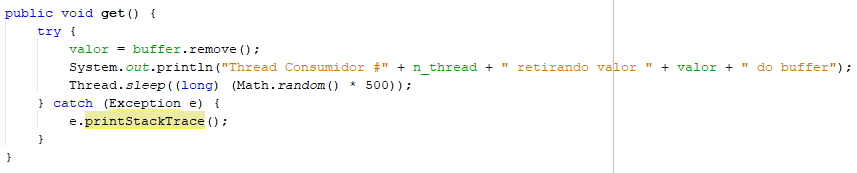


Figura 3: Método de remoção à Thread, consumidor.

Vale ressaltar que ambas as classes, Produtor e Consumidor, estendem de Thread, para poderem trabalhar sobre o mesmo conjunto de dados e em threads,

# Resultados Obtidos

O sistema teve com princípio de testes a inserção de grandes valores para checar a amplitude dos dados e se gerariam problemas ao sistema.

Apresentar **COMO FOI TESTADO O SISTEMA**

Explicar **COMO O TESTE REALIZADO GARANTE QUE O SISTEMA FUNCIONA**

# Conclusão

Pudemos notar no desenvolvimento dos algoritmos aspectos que complementaram as aulas ministradas. É percebido a importância da programação concorrente, principalmente ao se trabalhar com Threads, já que o sistema se torna mais dinâmico e rápido, podendo alocar recursos de maneira distinta às suas semelhantes. É notório destacar que a programação concorrente necessita de um cuidado maior dos programadores, ´pois podem gerar erros de dados inconsistentes com valores sendo manipulados e alterados por mais de uma Thread.

# Anexos

Esta seção mostra os anexos utilizados na elaboração deste documento. Serão apresentados abaixo as referências bibliográficas, lista de figuras e de links que auxiliaram o desenvolvimento do trabalho e deste relatório.

Figura 1: Distribuição das classes aos problemas do trabalho.

Figura 2: Método de adição à Thread, produtor.

Figura 3: Método de remoção à Thread, consumidor.

Figura 4:

Link 1: Utilizando Threads parte 1 – Devmedia. <https://www.devmedia.com.br/utilizando-threads-parte-1/4459> Acessado em 5 de outubro de 2019.