# Computação Paralela e Sistemas Distribuídos - Jantar dos Filósofos

Autores: Frederico Oliveira Freitas
Luan Felipe Ribeiro Santos
Talles Souza Silva

# 1. Introdução

O objetivo deste trabalho prático consiste em implementar uma solução para o problema do Jantar dos Filósofos, utilizando-se de uma interface gráfica para demonstrar o funcionamento e execução do algoritmo.
Espera-se com isso praticar os conceitos básicos de programação paralela com o uso de *Threads* e a aplicação de semáforos.

## 2. Implementação

### Pacote Interface

### Classe: Inicial

**public class Inicial extends JFrame implements Runnable:** Definições de variáveis e componentes da tela inicial, bem como o layout que será retratado após a execução do código fonte, com o estados dos filósofos definidos em: comendo, pensando e faminto e as imagens representando tais estados.

**public Inicial(Jantar jantar):** Executa o layout inicial e a primeira tela pela qual o usuário tem acesso. Neste método estão contidas as definições de cores, fonte e dimensão da janela.

**public void paint(Graphics g):** Retrata cada estado do algoritmo através de ícones interativos que mostram a variação desses estados conforme a execução das threads.

**public void run ():** Invoca a Thread que redesenha a tela e a faz dormir durante um tempo para redesenhar novamente.
Caso não consiga, lança uma exceção.

### Classe Main

**public static void main(String[] args):** Instancia as classes Jantar e Inicial, além de invocar a tela de funcionamento que será mostrada.

**Pacote Negócio**

**Classe Filósofo**

**public Filosofo (String nome, int ID):** Método construtor para a identificação do filósofo através do nome definido e um identificador.

**public void ComFome ():** Método que define o estado do filósofo para Faminto, setando o estado dele na classe Jantar.

**public void Come ():** Método que define o estado do filósofo para Comendo, setando o estado dele na classe Jantar, com uma thread que controla o tempo em que ele permanece comendo, sendo definido o intervalo de tempo de 1000 milissegundos.

**public void Pensa ():** De modo semelhante ao método supracitado, este método define o estado do filósofo para Comendo, setando o estado dele na classe Jantar, com uma thread que controla o tempo em que ele permanece comendo, sendo definido o intervalo de tempo de 1000 milissegundos.

**public void LargarGarfo ():** Método para controle do estado de soltar o garfo do Filósofo, ou seja, o momento em que o estado dele é alterado de comendo para pensando, a partir do decremento do semáforo mutex que permite controlar o estado atual da mesa e dos filósofos.

**public void PegarGarfo ():** Método para controle do estado de soltar o garfo do Filósofo, ou seja, o momento em que o estado dele é alterado de pensando para faminto , a partir do decremento do semáforo mutex que permite controlar o estado atual da mesa e dos filósofos.

**public void TentarObterGarfos():** Método que verifica se o filósofo está com fome, e se o vizinho da esquerda e da direita não estão comendo. A partir desta verificação, o filósofo pode ou não, comer.

**public void run ():** Método que retrata o ambiente pelo qual os estados do filósofos são controlados. Inicialmente ele pensa, logo após fica com fome, e em seguida ele come. E os estados se repetem a partir do controle das threads.

**public int VizinhoDireita ():** Método para obter o filósofo vizinho da direita. Realiza uma operação de Mod, e verifica se o valor encontrado for maior do que 4, o filósofo vizinho estará na direita.

**public int VizinhoEsquerda ():** Método para obter o filósofo vizinho da esquerda. Realiza uma operação de Mod, e verifica se o valor encontrado for menor do que 0, o filósofo vizinho estará na esquerda..

**Classe Jantar**

Classe na qual tem se a criação dos semáforos da aplicação. Um semáforo mutex, sendo o principal semáforo da aplicação e um semáforo para controlar cada um dos 5 filósofos.
Além disso nela está contida o método construtor da Jantar da aplicação, a inicialização de todos filósofos e semáforos a partir de vetores para controlar as posições.

**Classe Semáforo**

**public synchronized void decrementar ():** Método que realiza a sincronização da classe onde será decrescido o contador e realiza a verificação enquanto ele permanecer em 0, espera uma nova solicitação, caso contrário, lança uma exceção, com uma mensagem de erro.

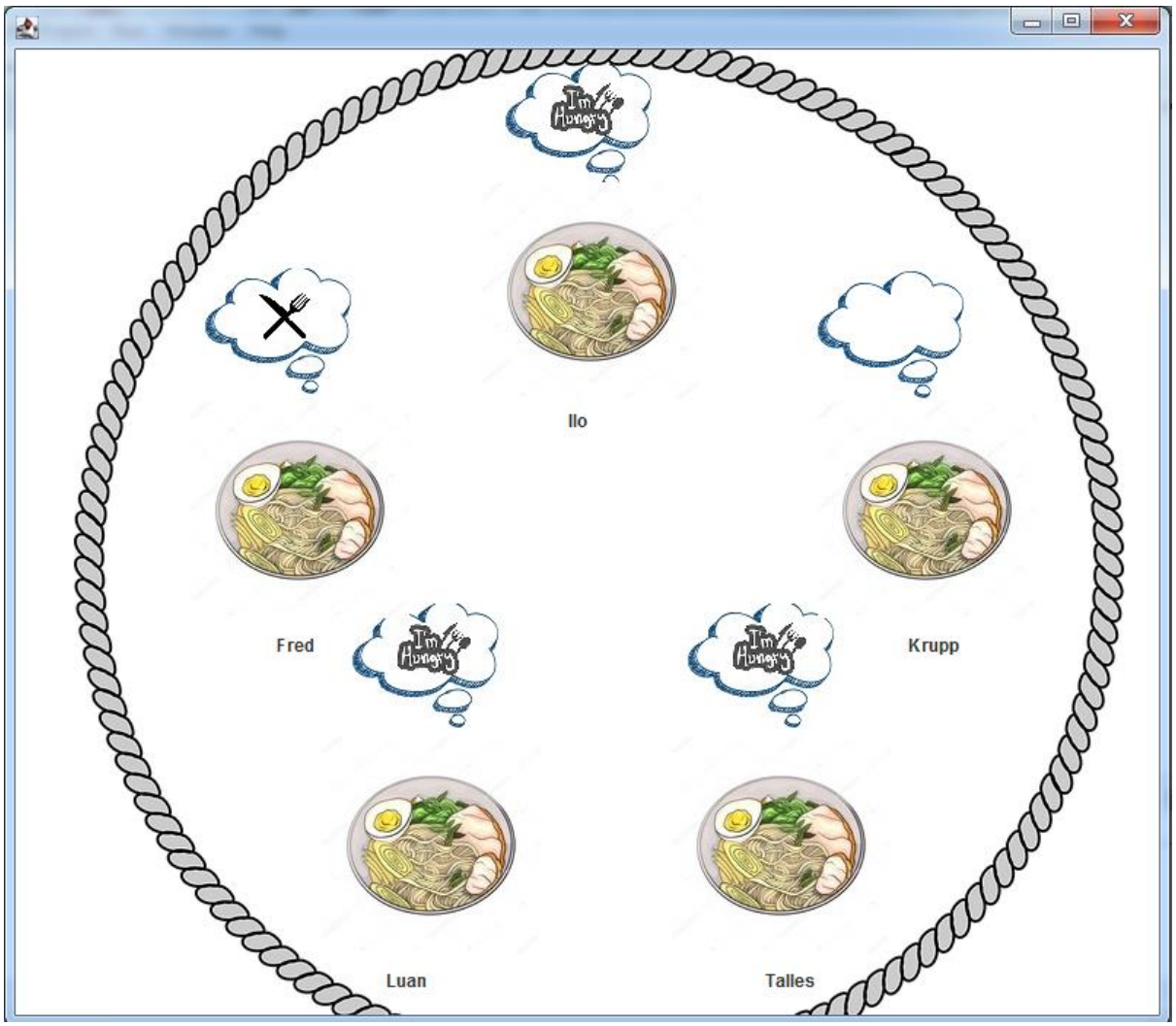
**public synchronized void incrementar ():** Método que realiza a sincronização da classe onde será incrementado o contador. Incrementa o contador da classe e notifica que a solicitação já foi executada.

## 3. Testes

Problems Javadoc Declaration Console

Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.2+9-jre\bin\javaw.exe (8 de mar de 2019 19:24:35)

```
O Filósofo Krupp está PENSANDO!
O Filósofo Luan está PENSANDO!
O Filósofo Talles está PENSANDO!
O Filósofo Fred está FAMINTO!
O Filósofo Fred está COMENDO!
```



## 4. Anexos

- Inicial.Java
- Main.Java
- Semaforo.Java
- Jantar.Java
- Filosofo.Java