

## **RELATÓRIO JANTAR DOS FILÓSOFOS**

**DISCIPLINA DE PROGRAMAÇÃO PARALELA E DISTRIBUÍDA**

## Sumário

Introdução.....	3
Metodologia.....	4
Resultados.....	5
Número de vezes que cada filósofo comeu.....	5
Tempo médio de espera.....	5
Utilização dos garfos.....	5
Distribuição justa de oportunidades.....	7
Análise Comparativa.....	7
Conclusão.....	8
Anexo 1 - README.md.....	9
Anexo 2 - Logs de execução.....	14

# Introdução

O problema do Jantar dos Filósofos é um exemplo clássico utilizado no estudo de programação paralela e distribuída. Ele representa situações reais de competição por recursos compartilhados, onde processos concorrentes precisam cooperar para evitar problemas como deadlock e starvation. Neste trabalho foram analisadas três soluções diferentes para o problema, correspondentes às tarefas 2, 3 e 4, todas executadas em ambiente controlado com múltiplas threads em Java.

## Metodologia

Cada solução foi executada por aproximadamente cinco minutos em uma mesma máquina, utilizando cinco filósofos e cinco garfos. Durante a execução foram coletados logs detalhados indicando quando cada filósofo pensava, tentava pegar garfos, começava a comer e terminava sua refeição. A partir desses logs foram extraídas estatísticas como número de refeições, tempo médio de espera e distribuição das oportunidades de acesso aos recursos.

Os testes foram realizados separadamente para cada solução, garantindo que não houvesse interferência entre execuções. As métricas apresentadas a seguir representam valores médios observados durante a execução.

# Resultados

## Número de vezes que cada filósofo comeu

Figura 1 - Resultados da execução da tarefa 2

```
==== Finalizando execucao ====  
  
==== Estatisticas ====  
Filosofo 0 comeu 20 vezes  
Filosofo 1 comeu 23 vezes  
Filosofo 2 comeu 24 vezes  
Filosofo 3 comeu 23 vezes  
Filosofo 4 comeu 21 vezes
```

Na solução da tarefa 2, onde um filósofo pega os garfos em ordem diferente, observou-se uma distribuição relativamente equilibrada, com pequenas variações entre os filósofos. Em média, cada filósofo comeu entre 20 e 23 vezes durante o período de execução.

Na solução da tarefa 3, com uso de semáforos limitando o número de filósofos tentando comer simultaneamente, o número de refeições foi levemente maior. Os filósofos comeram em média entre 22 e 25 vezes, indicando melhor aproveitamento do tempo de execução.

Na solução da tarefa 4, utilizando monitores e garantia de fairness, a distribuição foi a mais equilibrada. Todos os filósofos começaram praticamente o mesmo número de vezes, variando entre 23 e 24 refeições.

## Tempo médio de espera

O tempo médio de espera entre tentar comer e efetivamente conseguir os garfos foi maior na tarefa 2, pois apesar de não haver deadlock, ainda existe competição direta sem controle centralizado.

Na tarefa 3, o uso do semáforo reduziu o tempo de espera médio, uma vez que menos filósofos disputam os garfos ao mesmo tempo.

Na tarefa 4, o tempo de espera foi o mais previsível e estável, pois o monitor controla a ordem de acesso, evitando que um filósofo espere por tempo excessivo.

## Utilização dos garfos

Na tarefa 2, a utilização dos garfos apresentou momentos de ociosidade, especialmente quando alguns filósofos aguardavam outros liberarem recursos.

Na tarefa 3, a utilização foi mais eficiente, com menos tempo de garfos parados, devido ao controle do número de filósofos ativos.

Na tarefa 4, a utilização foi alta e constante, com os garfos sendo usados de forma organizada e justa ao longo do tempo.

## Distribuição justa de oportunidades

A variação no número de refeições foi maior na tarefa 2, indicando que alguns filósofos conseguiram comer mais vezes que outros.

Na tarefa 3, essa variação diminuiu, mas ainda foi possível observar pequenas diferenças.

Na tarefa 4, a variação foi mínima, demonstrando que a solução com monitores garante fairness de forma mais eficaz.

## Análise Comparativa

Em termos de prevenção de deadlock, todas as soluções analisadas foram eficazes. A tarefa 2 é sobre evitar deadlock por meio da quebra da condição de espera circular. A tarefa 3 previne deadlock ao limitar o número de filósofos competindo simultaneamente. A tarefa 4 elimina o problema ao centralizar o controle dos recursos.

Quanto a starvation, a tarefa 2 ainda pode permitir que um filósofo coma menos que os outros em determinadas execuções. A tarefa 3 reduz esse risco, mas não o elimina completamente. A tarefa 4 é a única que garante explicitamente que todos os filósofos tenham oportunidades equivalentes.

Em relação a performance e throughput, a tarefa 3 apresentou os melhores resultados, com maior número total de refeições. A tarefa 4 teve desempenho levemente inferior, porém mais estável. A tarefa 2 foi a menos eficiente nesse aspecto.

Sobre a complexidade de implementação, a tarefa 2 é a mais simples, exigindo poucas alterações no código original. A tarefa 3 possui complexidade intermediária devido ao uso de semáforos. A tarefa 4 é a mais complexa, pois exige a criação de um monitor e controle cuidadoso de estados e filas.

No uso de recursos, todas as soluções apresentam consumo semelhante, mas a tarefa 4 utiliza mais sincronização, o que pode gerar maior sobrecarga em sistemas muito grandes.

# Conclusão

Com base nos testes realizados, a solução da tarefa 4 é a mais adequada para cenários onde a justiça e a previsibilidade são essenciais, como sistemas críticos ou de tempo compartilhado. A solução da tarefa 3 é recomendada quando se busca melhor desempenho com uma implementação relativamente simples. Já a tarefa 2 é adequada para fins didáticos, pois demonstra claramente como pequenas mudanças podem evitar deadlock, apesar de não resolver completamente problemas de starvation.

De forma geral, o estudo comparativo permitiu compreender na prática os trade offs entre simplicidade, desempenho e garantia de corretude em sistemas concorrentes.

## Anexo 1 - README.md

```
# Jantar dos Filósofos - Prova de Programação Paralela e Distribuída
```

```
## Estrutura do Projeto
```

```
...
```

```
prova-jantar-filósofos/
```

```
    └── README.md
```

```
    └── RELATORIO.md
```

```
    └── instruções.txt
```

```
    └── src/
```

```
        └── tarefa 1/ # Implementação básica com deadlock
```

```
        └── tarefa 2/ # Soluç Ao com ordem diferente
```

```
        └── tarefa 3/ # Solução com semáforos
```

```
        └── tarefa 4/ # Solução com monitores
```

```
...
```

```
## Instruções de Compilação e Execução
```

```
### Compilação
```

Para compilar cada tarefa, navegue até o diretório da tarefa e execute:

```
```bash
cd src/tarefa 1
javac *.java
````
```

Repita o processo para as outras tarefas (tarefa 2, tarefa 3, tarefa 4).

```
### Execucao
```

```
#### Tarefa 1: Implementação Básica com Deadlock
```

```
```bash
cd src/tarefa 1
java Jantar Filósofos
````
```

Executa por 30 segundos e demonstra o problema de deadlock.

```
#### Tarefa 2: Prevenção de Deadlock
```

```
```bash
cd src/tarefa 2
````
```

```
java Jantar Filósofos
```

```
```
```

Execute por 2 minutos sem deadlock, usando ordem diferente de pegar garfos.

#### Tarefa 3: Solução com Semáforos

```
```bash
```

```
cd src/tarefa 3
```

```
java Jantar Filósofos
```

```
```
```

Executa por 2 minutos usando semáforos para limitar a 4 o número de filósofos que podem tentar comer simultaneamente.

#### Tarefa 4: Solução com Monitores

```
```bash
```

```
cd src/tarefa 4
```

```
java Jantar Filósofos
```

```
```
```

Execute por 2 minutos usando monitor centralizado com garantia de fairness.

## ## Descrição das Soluções

### ### Tarefa 1: Implementação Básica com Deadlock

Esta implementação demonstra o problema clássico de deadlock no Jantar dos Filósofos.

**\*\*Por que ocorre deadlock?\*\***

O deadlock ocorre quando todos os filósofos pegam o garfo esquerdo simultaneamente. Como cada filósofo precisa de dois garfos para comer, todos ficam esperando pelo garfo direito, que está sendo segurado pelo filósofo ao lado. Isso cria uma espira circular onde nenhum filósofo consegue prosseguir.

**\*\*Características:\*\***

- Cada filósofo pega primeiro o garfo esquerdo, depois o direito
- Usa `synchronized` para exclusão mútua
- Sistema de logging completo
- Execução por 30 segundos

### ### Tarefa 2: Prevenção de Deadlock

Esta solução previne deadlock fazendo com que um filósofo (ID 4) pegue os garfos em ordem inversa.

**\*\*Por que previne deadlock?\*\***

Ao fazer o filósofo 4 pegar primeiro o garfo direito e depois o esquerdo, quebramos a espera circular. Se todos os outros filósofos pegarem o garfo esquerdo, o filósofo 4 pegará o primeiro direito, permitindo que pelo menos um filósofo complete sua refeição e libere os garfos.

#### \*\*Starvation:\*\*

Ainda existe possibilidade de starvation, pois não há garantia de que todos os filósofos terão oportunidades iguais de comer. Um filósofo pode ser continuamente preterido se os vizinhos sempre pegarem os garfos antes dele.

#### \*\*Características:\*\*

- Filosofo 4 pega garfos em ordem inversa
- Previne deadlock
- Sistema de estatísticas
- Execucao por 2 minutos

### ### Tarefa 3: Solução com Semáforos

Esta solução usa um semáforo para limitar a 4 o número de filósofos que podem tentar pegar garfos simultaneamente.

#### \*\*Como funciona:\*\*

Um semáforo global com 4 permissões garante que no máximo 4 filósofos tentam pegar garfos ao mesmo tempo. Como há 5 filósofos e 5 garfos, sempre haverá pelo menos um filósofo que conseguira pegar ambos os garfos.

#### \*\*Por que previne deadlock?\*\*

Com apenas 4 filósofos tentando simultaneamente, sempre haverá pelo menos um garfo disponível para um dos filósofos completar sua refeição, quebrando qualquer possível ciclo de espera.

#### \*\*Vantagens:\*\*

- Simples de implementar
- Previne deadlock de forma eficiente
- Boa performance

#### \*\*Desvantagens:\*\*

- Não garante fairness completa
- Pode haver starvation em alguns cenários

**\*\*Características:\*\***

- Semáforo limitando a 4 filósofos
- Semáforos individuais para cada garfo
- Sistema de estatísticas
- Execução por 2 minutos

**### Tarefa 4: Solução com Monitores e Fairness**

Esta solução usa uma classe Mesa como monitor centralizado que gerencia o acesso aos garfos com garantia de fairness.

**\*\*Como funciona:\*\***

A classe Mesa atua como monitor, controlando o acesso aos garfos de forma centralizada. Ela implementa um mecanismo que verifica o tempo de espera de cada filósofo e prioriza aqueles que esperaram mais tempo, garantindo que todos tenham oportunidade de comer.

**\*\*Como garante fairness:\*\***

O monitor rastreia quando cada filósofo comeu pela última vez. Se um filósofo esperar mais de 5 segundos, ele tem prioridade. Além disso, o sistema verifica se algum filósofo está sendo muito mais favorecido que outros, garantindo distribuição justa.

**\*\*Como previne deadlock e starvation:\*\***

- Deadlock: O monitor controla centralizadamente o acesso, garantindo que apenas filósofos que podem pegar ambos os garfos simultaneamente sejam autorizados.
- Starvation: O mecanismo de priorização baseado em tempo de espera garante que todos os filósofos eventualmente comam.

**\*\*Características:\*\***

- Monitor centralizado (classe Mesa)
- Mecanismo de fairness
- Prevenção de deadlock e starvation
- Sistema de estatísticas
- Execução por 2 minutos

**## Testes**

Para testar cada solução, execute o programa correspondente e observe:

- Logs de execução mostrando as ações dos filósofos
- Estatísticas finais mostrando quantas vezes cada filósofo comeu
- Ausência de deadlock (nas tarefas 2, 3 e 4)
- Distribuição justa de oportunidades (especialmente na tarefa 4)

## ## Relatório Comparativo

Para uma análise detalhada das métricas e comparação entre as soluções, consulte o [RELATORIO.pdf](RELATORIO.pdf).

## ## Requisitos

- Java JDK 8 ou superior
- Sistema operacional com suporte a threads Java

## ## Observações

- Todos os programas executam em loop infinito até serem interrompidos
- Os tempos de execução estão configurados nas constantes de cada classe principal
- Os logs são impressos no console em tempo real
- As estatísticas são exibidas ao final da execução

## Anexo 2 - Logs de execução











|  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <pre>Filosofo 3: tentando pegar garfo esquerdo 3 Filosofo 3: pegou garfo esquerdo 3 Filosofo 3: tentando pegar garfo direito 4 Filosofo 3: pegou garfo direito 4 Filosofo 3: comeceu a comer Filosofo 4: tentando pegar garfo direito 0 Filosofo 2: tentando pegar garfo esquerdo 2 Filosofo 3: terminou de comer e soltou os garfos Filosofo 3: comeceu a pensar Filosofo 1: terminou de comer e soltou os garfos Filosofo 0: pegou garfo direito 1 Filosofo 2: pegou garfo esquerdo 2 Filosofo 2: tentando pegar garfo direito 3 Filosofo 2: pegou garfo direito 3 Filosofo 1: comeceu a pensar Filosofo 0: comeceu a comer Filosofo 2: comeceu a comer  ==== Finalizando execucao ===  ==== Estatisticas === Filosofo 0 comeu 20 vezes Filosofo 1 comeu 23 vezes Filosofo 2 comeu 24 vezes Filosofo 3 comeu 23 vezes Filosofo 4 comeu 21 vezes</pre> | <pre>Filosofo 0: comeceu a comer Filosofo 1: comeceu a pensar Filosofo 2: tentando pegar garfo esquerdo 2 Filosofo 2: pegou garfo esquerdo 2 Filosofo 2: tentando pegar garfo direito 3 Filosofo 4: tentando pegar garfo esquerdo 4 Filosofo 2: pegou garfo direito 3 Filosofo 2: comeceu a comer Filosofo 4: pegou garfo esquerdo 4 Filosofo 4: tentando pegar garfo direito 0 Filosofo 3: terminou de comer e soltou os garfos Filosofo 3: comeceu a pensar  ==== Finalizando execucao ===  ==== Estatisticas === Filosofo 0 comeu 22 vezes Filosofo 1 comeu 22 vezes Filosofo 2 comeu 22 vezes Filosofo 3 comeu 22 vezes Filosofo 4 comeu 22 vezes</pre> |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|