Jantar dos Filósofos

Trabalho Prático 2

Disciplina: Sistemas Operacionais II – 2023/02

Professor: Douglas Donizeti de Castilho Braz

Aluno: Tárcio Henrique Rodrigues Santos

Venho neste trabalho apresentar os dados encontrados durante os testes caseiros.

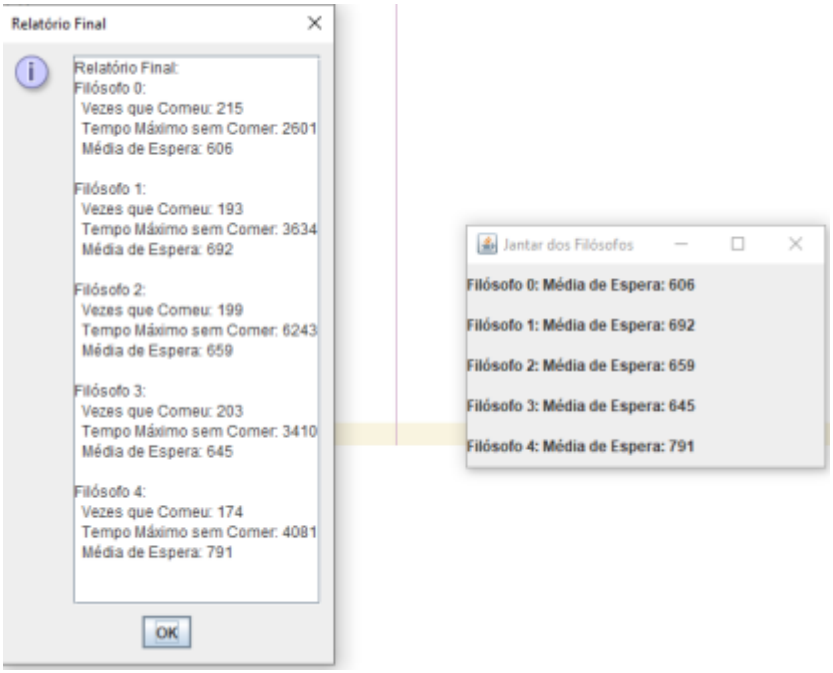
Implementação propria:

Implementação do livro:

Lembrei de respeitar cada nuance dos pedidos do projeto como:
interface gráfica na implementação própria e ainda adicionei a interface a do livro e relatório de dados depois de 180 segundos.

O código implementa essa lógica utilizando threads em Java, com cada filósofo sendo uma instância de thread. Semáforos são empregados para controlar o acesso aos garfos, e cada filósofo passa por um ciclo contínuo de pensar, tentar pegar os garfos e comer. Além disso, são registradas métricas, como o número de vezes que cada filósofo comeu, o tempo máximo que ficou sem comer e a média de espera para cada filósofo. Essas métricas são posteriormente utilizadas para gerar um relatório após um período específico de execução. O código também inclui uma interface gráfica (UI) .

Seguem os dados coletados via interface do programa (o mesmo pode ser interrompido a qualquer momento):



Agora a minha implementação conta com classes diferentes e visual mais intuitivo.

Existem três classes principais: Filosofo, JantarUI, e Main. A classe Filosofo define o comportamento individual de cada filósofo, utilizando semáforos para controlar o acesso aos garfos e garantir que apenas filósofos adjacentes possam comer simultaneamente.

A classe JantarUI é uma interface gráfica Swing que exibe o estado atual de cada filósofo (pensando, com fome, comendo) e encerra a simulação após um tempo predefinido, exibindo um relatório final.

A classe Main inicia a simulação, criando instâncias dos filósofos e iniciando suas threads. O código utiliza semáforos e estratégias para evitar condições de deadlock, garantindo que os filósofos possam alternar entre os estados sem ficarem bloqueados.

O código inclui também a classe Semaforo, que implementa semáforos clássicos para controle de concorrência. Os semáforos são usados para evitar condições de corrida e coordenar o acesso aos recursos compartilhados, como os garfos no problema do jantar dos filósofos.

Ao final da simulação, um relatório é exibido, apresentando estatísticas como o número de vezes que cada filósofo comeu, o tempo máximo de espera, e a média de espera. O código é estruturado para evitar deadlock e garantir uma execução concorrente segura do problema do jantar dos filósofos.

```
// Método de sincronização da classe onde será decrescido o contador
   public synchronized void decrementar ()
   {

      // Enquanto o contador for igual a 0, ele aguarda e trata a exceção
      while (this.contador == 0)
      {
         try
         {
            // Espera uma nova solicitação
            wait();
         }
         catch (InterruptedException ex)
         {
            // Exibe uma mensagem de controle de erro
            System.out.println("ERROR>" + ex.getMessage());
         }
      }

      // Caso tenha saído do while acima, então decrementa o
      // contador da classe
      this.contador--;

   }
```

essa classe que remete ao decrescimento do status dos filósofos.