**Segundo trabalho de Computação Concorrente**

**Alunos: Carlos Cozzolino e Thiago de Oliveira**

**DREs: 115086800 e 111466197**

1. **Implementação:**

O problema proposto começa pedindo a implementação de uma função que segue a ideia de Produtores/Consumidores e, seguindo a ideia dos monitores de Leitores/Escritores, uma classe chamada “Basic” foi criada. Essa classe terá algumas varáveis globais como: Uma fila de prioridade ‘queue’ para adicionarmos e removermos o conteúdo do arquivo de entrada, 2 booleans para dizermos quando a fila está vazia ou cheia, o tamanho N dos blocos e a quantidade de números no arquivo. Primeiramente começamos com a função ‘plus’ que recebe como parâmetro uma String com o conteúdo a ser adicionado, utilizamos o synchronized para fazermos a exclusão mútua, avisamos que a fila não está mais vazia trocando o valor de ‘isEmpty’ (que se inicia como verdadeiro) para ‘false’ e notificando a próxima thread.

Com o método de adicionar a fila criado, tínhamos que fazer um para remover o conteúdo e para isto criamos o método ‘remove’ que também será synchronized pelo mesmo motivo do anterior. Aqui precisamos tomar um pouco mais de cuidado já que as múltiplas threads não podem consumir antes de ter algo produzido e nem consumir algo que não existe, então, aqui elas ficarão em espera enquanto o valor de “isEmpty” for verdadeiro **e**  o valor de “isDone” não for falso.

Depois disso, checamos se a fila está vazia e se “isDone” é verdadeiro, caso essas condições sejam atendidas a função terminou sua execução, se não forem atendidas duas coisas podem acontecer: a fila tem algo para ser removido e iremos adicionar à uma String, posteriormente retornando-a, ou, a fila está vazia mas ainda temos dados a serem produzidos pelo Produtor e neste caso iremos marcar “isEmpty” como true novamente e a função remove volta para o seu estado de espera.

Além destes métodos, a classe “Basic” ainda tem o “setIsDone” que é utilizado para mudar o valor de “isDone” pela classe “Producer”.

Com isto terminamos a primeira ideia de implementação de um monitor para Produtor e Consumidor, agora podemos falar destas classes em si começando pela “Producer”. Na classe produtora temos 2 variáveis globais, uma delas é o monitor “Basic” e outra é o arquivo para ser lido. Essa classe novamente é a mais simples já que só precisamos nos preocupar em ler o arquivo e adicionar a fila de prioridade, para isto fazemos o scan da primeira linha e dispensamos, já que ele só tem a quantidade de elementos total do arquivo e não precisamos dele para ordenar mais tarde, e criamos um loop onde enquanto tiver algo para ler no arquivo chamamos a função “plus” para adicionar o que foi lido na fila de prioridade, quando o laço termina, colocamos o valor de “isDone” como verdadeiro e terminamos.

Assim terminamos a classe “Producer” e podemos descrever a “Consumer”, na primeira implementação tínhamos uma outra classe “Writer” que era usada para escrever o novo arquivo de texto com os números ordenados. A classe foi removida e englobada na “Consumer” que tem como variáveis globais o monitor “Basic”, o monitor “monitor”, uma lista “vector” que seria nosso array global, a String “path” com o diretório onde o arquivo será salvo, o tamanho “N” do bloco e a quantidade de elementos totais do arquivo inicial.

Começando explicando a ideia implementada para o consumidor, dentro de um loop infinito começamos removendo um elemento da nossa fila com a função “remove” do monitor “Basic”, caso nossa String tenha um valor nulo, significa que toda a fila foi tratada e não temos mais blocos para ordenar, quebrando o laço infinito e terminando esta parte da execução. Caso o valor não seja nulo, então iremos remover os espaços em brancos da String com a função “join” e adicionar ao String array “splite”, que será convertido em um array de inteiros e então ordenado, depois disso iremos adicionar cada elemento a lista “vector” e na nossa primeira implementação essa lista era enviada para a classe “Writer” escrever.

Problemas iniciais:

Além de termos implementado a classe “Writer” erroneamente, tínhamos um possível problema no monitor “Basic” onde ele poderia produzir de forma rápida demais, sobrecarregando a nossa fila. Como o produtor tem uma lógica de execução, acaba levando um tempo maior na sua execução e para evitar esse possível problema criamos um “wait()” para quando a fila atingisse uma determinada capacidade, ela iria esperar alguém ser removido para continuar sua execução. Com isso, para evitar uma possível inanição das threads, adicionamos um “notifyAll()” quando o tamanho da fila atingisse a capacidade – 1 valores. Apesar de na prática isso nunca ter acontecido(provavelmente porque mesmo tendo uma lógica por trás do construtor ele continua sendo executado de maneira muito rápida), decidimos manter a ideia.