Relatório: Sincronização de Threads com CyclicBarrier

1. Objetivo

Este programa utiliza `CyclicBarrier` para sincronizar múltiplas threads que buscam um número específico dentro de uma matriz grande. O objetivo é garantir que todas as buscas terminem antes de processar os resultados finais.

2. Estrutura do Código

- **MatrizSimulada (`MatrixMock`)**:
  Gera uma matriz de números aleatórios e conta quantas vezes um número alvo aparece.
- **Resultados (`Results`)**:
  Armazena o número de vezes que o número alvo foi encontrado em cada linha da matriz.
- **Pesquisador (`Searcher`)**:
  Cada thread percorre uma parte da matriz e conta quantas vezes o número alvo aparece.
- **Agrupador (`Grouper`)**:
  Executa após todas as threads terminarem e soma os resultados parciais.
- **Principal (`Main`)**:
  Configura o ambiente e cria as threads de pesquisa, utilizando um `CyclicBarrier` para sincronizar todas.

3. Fluxo de Execução

1. A classe `Principal` cria a matriz e define os parâmetros da busca.
2. São iniciadas cinco threads `Pesquisador`, cada uma processando um bloco de linhas.
3. Cada `Pesquisador` conta as ocorrências do número e armazena no objeto `Resultados`.
4. Todas as threads chamam `barreira.await()`, bloqueando até que todas terminem.
5. O `Agrupador` soma os resultados e exibe o total.

4. Exemplo de Saída

```
MatrizSimulada: Existem 50010 ocorrências do número 5 nos dados gerados.
Thread-0: Processando linhas de 0 a 2000.
Thread-1: Processando linhas de 2000 a 4000.
...
Thread-4: Processando linhas de 8000 a 10000.
```

```
Thread-0: Linhas processadas.
...
Agrupador: Processando os resultados...
Agrupador: Resultado total: 50010.
```

5. Conclusão

CyclicBarrier permite sincronizar múltiplas threads, garantindo que todas completem seu trabalho antes de continuar a execução. Essa abordagem é útil para tarefas paralelas que precisam de um ponto de sincronização.