Relatório: Uso de Condições (wait/notify) para Armazenamento de Eventos

1. Objetivo

Demonstrar o uso dos métodos `wait()` e `notify()` em conjunto com a sincronização (`synchronized`) para coordenar a comunicação entre threads. O exemplo simula um sistema de armazenamento de eventos (datas), onde uma thread produtora adiciona eventos e uma thread consumidora remove esses eventos.

2. Estrutura do Código

- **ArmazenamentoEventos:**
  Gerencia uma lista de eventos com tamanho máximo.
  - `adicionar()`: Método sincronizado que adiciona um evento (data atual) à lista. Caso o armazenamento esteja cheio, a thread espera.
  - `remover()`: Método sincronizado que remove um evento da lista. Caso o armazenamento esteja vazio, a thread espera.
- **Produtor:**
  Implementa `Runnable` e adiciona 100 eventos ao armazenamento utilizando o método `adicionar()`.
- **Consumidor:**
  Implementa `Runnable` e remove 100 eventos do armazenamento utilizando o método `remover()`.
- **Principal:**
  Inicializa o armazenamento de eventos, cria e inicia as threads Produtor e Consumidor.

3. Fluxo de Execução

1. Um objeto `ArmazenamentoEventos` é criado com um tamanho máximo de 10.
2. A thread Produtora tenta adicionar 100 eventos, mas aguarda quando o armazenamento atinge o tamanho máximo.
3. Simultaneamente, a thread Consumidora tenta remover 100 eventos, aguardando se o armazenamento estiver vazio.
4. As operações de `wait()` e `notify()` permitem que as threads alternem a execução, mantendo o armazenamento dentro do limite definido.

4. Exemplo de Execução

Saída do Console:

Adicionado: 1 Adicionado: 2 Adicionado: 3 ... Removido: 10: Sun Feb 09 12:34:56 GMT 2025 Adicionado: 10 Removido: 9: Sun Feb 09 12:34:57 GMT 2025 ...

5. Conclusão

A utilização dos métodos wait() e notify() permite uma coordenação eficiente entre threads produtoras e consumidoras, evitando condições de corrida e garantindo que a lista de eventos seja manipulada de forma segura. Este mecanismo assegura que o produtor espere quando o armazenamento está cheio e que o consumidor espere quando está vazio, mantendo a integridade dos dados.