Relatório: Executor de Tarefas com ThreadPoolExecutor

1. Objetivo

Este exemplo demonstra o uso de `ThreadPoolExecutor` em Java para gerenciar e executar múltiplas tarefas de forma concorrente. Através do uso do pool de threads, o sistema otimiza a criação, execução e término das tarefas, facilitando o gerenciamento de recursos.

2. Estrutura do Código

- **Servidor (Servidor)**:
  Gerencia a execução das tarefas utilizando um pool de threads. O método `executarTarefa` envia uma tarefa para execução e exibe estatísticas do pool, como o tamanho do pool, a contagem de threads ativas e o total de tarefas concluídas. O método `encerrarServidor` finaliza o executor.
- **Tarefa (Tarefa)**:
  Representa uma tarefa a ser executada. Cada instância registra a data de criação e, ao ser executada, simula um processamento por um período aleatório, exibindo mensagens de início e término.
- **Principal (Principal)**:
  Inicializa o sistema criando uma instância do Servidor e 100 tarefas, que são enviadas para execução. Após o envio de todas as tarefas, o servidor é encerrado.

3. Fluxo de Execução

1. A classe `Principal` instancia um objeto `Servidor`, que configura um pool de threads usando `Executors.newCachedThreadPool()`.
2. Um loop cria 100 tarefas, cada uma instanciada pela classe `Tarefa`, que registra a data de criação e simula um tempo de processamento aleatório.
3. Cada tarefa é enviada ao método `executarTarefa` do Servidor, que as executa e exibe estatísticas atualizadas do pool de threads.
4. Após o envio de todas as tarefas, o método `encerrarServidor()` é chamado para desligar o pool de threads.

4. Exemplo de Execução

Saída do Console:

```
Servidor: Uma nova tarefa chegou
Servidor: Tamanho do Pool: 5
Servidor: Contagem de Threads Ativas: 5
Servidor: Tarefas Completadas: 0
```

```
Thread-1: Tarefa Tarefa 0: Criada em: Sat Feb 10 15:30:12 BRT 2025
Thread-1: Tarefa Tarefa 0: Iniciada em: Sat Feb 10 15:30:12 BRT 2025
Thread-1: Tarefa Tarefa 0: Processando por 3 segundos
...
Thread-3: Tarefa Tarefa 0: Finalizada em: Sat Feb 10 15:30:15 BRT 2025
...
```

5. Conclusão

O uso de `ThreadPoolExecutor` permite gerenciar de forma eficiente a execução de múltiplas tarefas concorrentes, otimizando a utilização dos recursos do sistema. Ao utilizar um pool de threads, o sistema mantém estatísticas importantes, como o tamanho do pool, o número de threads ativas e o total de tarefas concluídas, o que facilita o monitoramento e a escalabilidade da aplicação. Essa abordagem é fundamental para aplicações que requerem alta performance e gerenciamento dinâmico de tarefas.