Manipulação de Exceções em Threads - Tratamento de Exceções Não Controladas em um Grupo de Threads

Objetivo do Projeto

Este documento apresenta os testes e resultados obtidos ao explorar dois cenários distintos relacionados à manipulação de exceções em threads no Java. Os cenários incluem a execução do programa sem manipulação global de exceções e com manipulação global de exceções.

🗒️ Cenários teste

Cenário 1: Sem Manipulação Global de Exceções

Neste cenário, as exceções lançadas pelas threads são tratadas individualmente. Caso uma exceção não seja tratada, a thread será encerrada.

Modificação no Código

No arquivo Task.java, foi implementada uma lógica para lançar uma exceção ao tentar dividir um número por zero.

Testes e Resultados

Teste: Executar o programa e observar o comportamento quando uma exceção é lançada.

Resultado: Quando uma exceção é lançada, o método uncaughtException do grupo de threads é chamado, imprimindo o rastreamento da pilha e interrompendo todas as outras threads do grupo. O programa é encerrado após a interrupção das threads.

Cenário 2: Com Manipulação Global de Exceções

Neste cenário, a manipulação de exceções é feita de forma global, capturando exceções não tratadas de qualquer thread dentro do grupo.

Modificação no Código

Nenhuma alteração adicional foi necessária, pois o grupo de threads já estava configurado para manipular exceções.

Testes e Resultados

Teste: Executar o programa e observar o comportamento quando uma exceção é lançada.

Resultado: O comportamento é idêntico ao do Cenário 1. O método uncaughtException é chamado, imprimindo o rastreamento da pilha e interrompendo todas as outras threads do grupo. O programa é encerrado após a interrupção das threads.

🔧 Conclusão:

🔗

Os testes demonstraram que, independentemente da abordagem utilizada (local ou global), o comportamento final do programa é o mesmo. O grupo de threads manipula as exceções lançadas, garantindo que todas as threads sejam interrompidas e que o programa finalize corretamente ao detectar um erro crítico. Essa abordagem é útil para garantir que falhas inesperadas sejam tratadas de forma segura e previsível dentro de um sistema multithreading.

💻 Desenvolvedor

🔗

Marcio Fonseca
👨‍💻

=======