Relatório: Tratamento de Exceções e Interrupção de Threads

1. Objetivo

O objetivo deste exercício é demonstrar o uso de threads, grupos de threads e o tratamento de exceções em Java. O código implementa a execução de múltiplas threads que realizam cálculos aleatórios, com manejo de exceções (divisão por zero) e controle de interrupção das threads.

2. Estrutura do Código

2.1. Classe `grupoThreads`

- **Descrição:**
  - Extende a classe `ThreadGroup` para criar um grupo de threads personalizado que trata exceções não capturadas.
- **Método `uncaughtException`:**
  - Este método sobrecarregado captura exceções não tratadas nas threads do grupo, exibe a exceção e interrompe as threads restantes do grupo.

2.2. Classe `Tarefa`

- **Descrição:**
  - Implementa a interface `Runnable` para ser executada em várias threads.
- **Funcionamento:**
  - Cada thread executa uma tarefa de cálculo com um valor aleatório, podendo gerar uma exceção de divisão por zero.
  - Se uma exceção for gerada, a thread a captura e exibe uma mensagem.
  - A thread também verifica se foi interrompida e, caso tenha sido, exibe uma mensagem e encerra a execução.

2.3. Classe `Principal`

- **Descrição:**
  - Orquestra a execução das threads utilizando o grupo de threads personalizado.
- **Funcionamento:**
  - Cria o grupo de threads `grupoThreads` e a tarefa `Tarefa`.
  - Inicializa cinco threads que executam a tarefa em paralelo.

3. Exemplo de Execução

Saída do Console:

1 : 1000 1 : 1000 1 : 1000 2 : 1000 2 : 1000 2 : 1000 3 : 1000 3 : 1000 3 : 1000 4 : Exceção de divisão por zero 5 : Interrompida A thread 4 lançou uma exceção

java.lang.ArithmeticException: / by zero at Tarefa.run(Principal.java:11) ...
Terminando o restante das threads

4. Modificações Realizadas

- **Tratamento de Exceções:**
  - Foi adicionada uma captura para exceções de divisão por zero no método `run` da classe `Tarefa`, para impedir a falha inesperada da thread devido a esse erro.
- **Controle de Interrupção:**
  - A verificação e a resposta à interrupção de threads foi implementada, permitindo que as threads se autointerrompam caso necessário.

5. Testes Realizados

- **Cenário 1:** Execução normal das threads, sem exceções.
  - Resultado esperado: Todas as threads executam o cálculo sem problemas e exibem os resultados.
- **Cenário 2:** Geração de exceção de divisão por zero.
  - Resultado esperado: A thread que tenta dividir por zero é interrompida, e as exceções são tratadas corretamente.
- **Cenário 3:** Interrupção manual de uma thread.
  - Resultado esperado: A thread interrompida exibe a mensagem "Interrompida" e encerra a execução.

6. Conclusão

Através deste exercício, foi possível compreender a importância do tratamento de exceções e do controle de interrupção de threads no contexto de programação concorrente. A personalização do grupo de threads foi útil para centralizar o manejo de exceções e garantir que, em caso de erro em uma thread, o restante do grupo fosse adequadamente interrompido. As modificações feitas no código permitiram testar diferentes cenários e validar o comportamento das threads em condições de exceção e interrupção.