1. Tratamento de Exceções Não Controladas em Threads

- **Objetivo:** Demonstrar como capturar exceções não tratadas dentro de uma *thread* usando `UncaughtExceptionHandler`.

2. Introdução

Em Java, quando uma *thread* lança uma exceção não capturada, o programa pode falhar sem um aviso adequado. Para evitar isso, podemos usar `UncaughtExceptionHandler`, que permite capturar e tratar exceções inesperadas em *threads*.

Este exemplo simula um erro ao tentar converter uma string inválida para um número e usa um manipulador de exceções para registrar informações detalhadas sobre o erro.

3. Descrição do Código Original

O código contém três classes principais:

Excecoes.java

- Implementa `UncaughtExceptionHandler`, capturando exceções não tratadas em *threads*.
- Exibe informações detalhadas sobre a exceção:
  - Nome da *thread* onde ocorreu o erro.
  - Tipo e mensagem da exceção.
  - Rastreamento da pilha (`Stack Trace`).
  - Estado da *thread* no momento da exceção.

Tarefas.java

- Implementa `Runnable`, simulando uma operação que gera uma exceção (`Integer.parseInt("TTT")`).
- Como a string `"TTT"` não pode ser convertida para número, uma exceção `NumberFormatException` é lançada.

Principal.java

- Cria e inicia uma *thread* com `Tarefas`.
- Define `Excecoes` como o manipulador de exceções da *thread* (`setUncaughtExceptionHandler(new Excecoes())`).
- Aguarda a conclusão da *thread* com `join()` e exibe a mensagem `"Thread has finished"`.

4. Modificações Realizadas

- **Tradução:** Todos os nomes de classes, métodos e variáveis foram traduzidos para o português.
- **Adição de Mensagens de Depuração:** Melhoramos a saída para indicar quando a exceção foi capturada.
- **Teste com Diferentes Exceções:** Alteramos `Integer.parseInt("TTT")` para `int x = 1 / 0;` para simular uma `ArithmeticException` e observar o comportamento.

5. Testes Realizados e Resultados Obtidos

Cenário 1: Código Original

Saída esperada:
1. A *thread* inicia e tenta converter `"TTT"` para um número.
2. Uma exceção `NumberFormatException` ocorre.
3. `Excecoes` captura e exibe detalhes do erro.
4. O programa continua normalmente.

Saída real (exemplo):

```
Uma execao foi capturada
Thread: 14
Excecao: java.lang.NumberFormatException: For input string: "TTT"
Stack Trace:
java.lang.NumberFormatException: For input string: "TTT"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:6
    ...
Estado da Thread: TERMINATED
Thread has finished
```

O manipulador de exceções funcionou corretamente, evitando a falha inesperada do programa.

Cenário 2: Teste com `int x = 1 / 0;`

- A exceção lançada foi `ArithmeticException: / by zero`.
- O `Excecoes` capturou e registrou o erro normalmente.

Saída real (exemplo):

```
Uma execao foi capturada
Thread: 14
Excecao: java.lang.ArithmeticException: / by zero
Stack Trace:
java.lang.ArithmeticException: / by zero
    at Tarefas.run(Tarefas.java:6)
```

```
    ...
Estado da Thread: TERMINATED
Thread has finished
```

Confirmamos que o `UncaughtExceptionHandler` *captura qualquer exceção não tratada dentro da* thread**.

6. Conclusão

Este exemplo demonstrou: - Como capturar exceções não controladas em *threads* usando `UncaughtExceptionHandler`. - A importância de registrar erros detalhadamente para facilitar depuração. - Que o programa continua executando normalmente, mesmo após uma falha na *thread*.

Esse conceito é útil para aplicações que precisam lidar com erros sem interromper completamente sua execução.