Tratamento de exceções

Por que tratar exceções?

Os principais motivos são:

Segurança:

No momento em que você captura os erros, e os trata para que eles não sejam expostos para o usuário você evita que informações sensíveis possam ser utilizadas por usuários mal-intencionados.

Melhoria da robustez do programa:

O tratamento de exceções ajuda a tornar o programa mais robusto. Se ocorrer um erro durante a execução do programa e não houver tratamento adequado, o programa pode travar ou fornecer resultados incorretos. O tratamento de exceções permite que o programa lide com essas situações de maneira controlada e continue a execução, se possível.

Manutenção mais fácil:

Ao lidar com exceções, é possível separar o código normal do código de tratamento de erros. Isso facilita a manutenção do código, pois os blocos de tratamento de exceções são isolados e podem ser modificados ou estendidos independentemente do código principal.

Compreensão e diagnóstico de erros:

O tratamento de exceções permite que os desenvolvedores capturem informações detalhadas sobre os erros que ocorrem durante a execução do programa. Isso é crucial para diagnosticar problemas e corrigi-los de maneira eficaz.

Melhoria da experiência do usuário:

Ao tratar corretamente os erros, o usuário final não terá contato com um programa que pode ficar quebrando. E caso quebre, ele receberá apenas informações definidas pelo programador, tornando sua experiência mais agradável.

Testabilidade:

O tratamento de exceções facilita a criação de casos de teste específicos para situações de erro, ajudando os desenvolvedores a garantir que o software se comporte corretamente em condições adversas.

Tratando exceções

Estrutura Try-Catch

O primeiro tratamento de exceção que vamos abordar é o bloco try catch.

O bloco Try Catch é dividido em dois:

O bloco Try: responsável por conter o código que que representa a execução padrão do trecho de código que **PODE** vir a acarretar um erro (exceção).

O bloco Catch:

Responsável por conter o código que virá a ser executado **CASO** ocorra um erro (exceção).

Já o bloco finally é aquele que é executado **INDEPENDENTEMENTE** de ocorrer um erro ou não.

Um **exemplo** bom para demonstrar como funciona o bloco try-catch é o seguinte:

Imagina que você tem um programa que pede pra você digitar um número de 1 a 10. E sem querer ou por falta de atenção, você digita um número maior que 10 ou até mesmo uma letra. O programa vai quebrar e apresentar um erro, porém, se utilizarmos a estrutura try para colocar a parte do código que possibilita o usuário a errar e logo em seguida colocar no catch a parte de código que você quer que aconteça caso o erro aconteça, o programa não quebrará!

Existem 3 maneiras de criar exceções no seu projeto:

Uma muito ruim: A lógica de validação é feita dentro do programa principal

Uma ruim: Nesse caso, a solução é criada através de métodos que retornam Strings com uma mensagem de erro

Uma boa: A solução mais adequada é a com o tratamento de exceções, utilizando o Try-Catch e o finally.