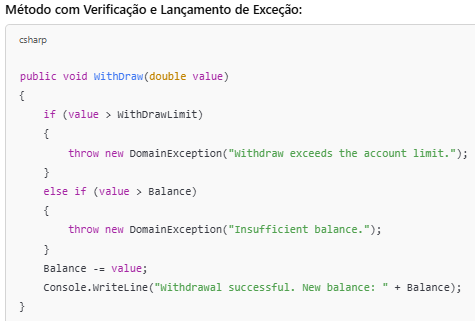
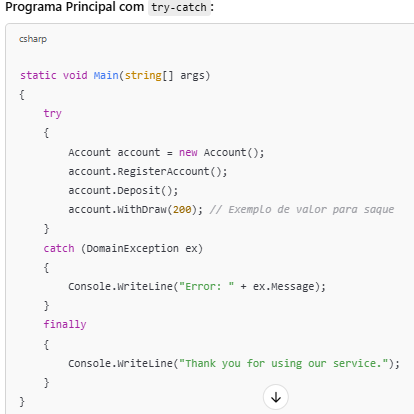
* ***Tratamento de exceções:*** 
  + **Conceito:** Tratamento de exceções é o processo de capturar e lidar com erros em tempo de execução para evitar que o programa encerre inesperadamente. Isso melhora a robustez e a confiabilidade da aplicação.
  + **O que é:** 
    - **2 forma principais de tratar exceções no C#:**
    - **Tratamento com Exceções Padrão usando try, catch, finally:** A forma mais comum de tratamento de exceções em C# é usando exceções padrão do .NET, como ArgumentNullException, etc. Elas são úteis para lidar com erros gerais no fluxo de execução. Para isso, usamos o recurso try, catch e finally para montar a verificação da exceção.
    - **Tratamento com Exceções Personalizadas usando throw e if:** As exceções personalizadas são úteis para erros específicos do domínio da aplicação. Nesses casos, criamos uma nova classe de exceção que herda de Exception ou uma de suas subclasses (ApplicationException, etc.), como sua DomainException. Com a classe criada vamos no método ou classe que queremos fazer a exceção e definimos a regra usando um if e lançando a exceção.
  + **Quando usar:** 
    - **Exceções padrões:** Use exceções padrão para erros genéricos ou erros comuns já contemplados pelo .NET, como o acesso a índices fora do array (IndexOutOfRangeException) ou leitura de arquivos inexistentes (FileNotFoundException).
    - **Exceções personalizadas:** Use exceções personalizadas para tratar erros específicos da lógica do programa, como "SaldoInsuficienteException" em um sistema bancário, ou "PedidoInvalidoException" em um sistema de pedidos.
  + **Definições:** 
    - **Sempre tentar usar o tipo específico de exceção no catch.**
    - **Try:** Delimita o bloco de código que pode causar uma exceção. Caso uma exceção ocorra, o fluxo de execução é transferido para o bloco catch correspondente
    - **Catch:** Captura e trata exceções lançadas no bloco try. Cada catch pode capturar tipos específicos de exceção, permitindo tratamento personalizado para cada tipo de erro.
    - **Finally:** Executado após o bloco try e catch, independentemente de uma exceção ter ocorrido ou não. Geralmente usado para liberar recursos, como fechar arquivos ou conexões de banco de dados.
    - **Exceções específicas no catch:** Sempre que possível, capture tipos específicos de exceções para diferenciar o tratamento de cada tipo de erro, melhorando a precisão do tratamento de exceções.
    - **Exceções Personalizadas:** Para tratar erros específicos do programa, crie uma classe de exceção personalizada herdando de ApplicationException ou outra subclasse de Exception.
  + **Vantagens:** 
    - **Melhor controle de erros:** O código pode prever, capturar e tratar erros em vez de encerrar inesperadamente.
    - **Organização:** Permite separar a lógica do programa do tratamento de erros, facilitando a manutenção e a leitura do código.
    - **Exceções personalizadas:** Ajudam a fornecer informações de erro mais específicas e detalhadas para o usuário, relacionadas diretamente ao contexto do programa.
  + **Como criar uma exceção padrão:** 
    - **Definição de Exceções Padrão:**
      * As exceções padrão são classes de erro que já vêm incluídas na biblioteca do .NET, como ArgumentNullException, InvalidOperationException, FormatException, entre outras.
      * Elas são usadas para representar erros comuns que podem ocorrer durante a execução de um programa e são úteis para fornecer informações sobre o que deu errado em um contexto específico.
    - **Tratamento da Exceção no Método:**
      * Ao escrever um método que pode falhar devido a condições esperadas, você deve estar preparado para capturar e lidar com essas exceções.
      * Utilize um bloco try para tentar executar a lógica que pode gerar uma exceção. Se uma exceção padrão ocorrer, use um bloco catch correspondente para capturar e lidar com essa exceção.
      * Por exemplo, ao tentar converter uma entrada do usuário para um número, você pode usar int.Parse() e, caso a entrada não seja um número válido, isso lançará uma FormatException.
    - **Tratamento da Exceção no Programa Principal:**
      * No ponto onde o método é chamado (geralmente no Main), utilize um bloco try-catch para tratar exceções que podem ser lançadas pelas operações dentro do método.
      * Se um erro ocorrer (como uma FormatException ao tentar converter uma string em número), o bloco catch captura a exceção e permite que você forneça uma mensagem ou tratamento adequado ao usuário.
    - **Exemplo:**



* + **Como criar uma exceção personalizada:** 
    - **Definição de Exceção Personalizada:**
      * Defina uma classe de exceção personalizada, como DomainException, que herda de ApplicationException ou diretamente de Exception.
      * Isso é útil para criar exceções específicas para o domínio da aplicação, com mensagens relevantes para o contexto do erro.
    - **Lançamento da Exceção no Método:**
      * No método em que deseja fazer a verificação, utilize uma estrutura condicional (if) para verificar cada situação específica. Se uma condição indicar um erro, use throw new DomainException("Mensagem") para lançar a exceção.
      * A mensagem passada para DomainException deve ser clara e informar qual condição não foi atendida ou qual erro ocorreu.
    - **Tratamento da Exceção no Programa Principal:**
      * No ponto onde o método é chamado (geralmente no Main ou em algum método que gerencie a execução do programa), utilize um bloco try-catch.
      * O bloco try tenta executar o código que pode lançar exceções. Se uma exceção de DomainException for lançada, o bloco catch correspondente a essa exceção a captura.
    - **Exemplo:**







* + **Diferença entre try e throw:** 
    - **Try:** O try-catch é completamente funcional por si só. Ele é usado para capturar e tratar exceções, independentemente de onde essas exceções surgem. Elas podem ocorrer naturalmente durante a execução (como uma divisão por zero ou um acesso a um índice inválido de uma lista) sem necessidade de um throw.
    - **Throw:** throw é usado para lançar ou relançar exceções, ele precisa de um try-catch em algum lugar no código para capturar e tratar a exceção que ele lança. Sem um try-catch para interceptá-la, o throw causará uma interrupção no programa, encerrando-o e exibindo o erro diretamente na saída de erro. O throw pode ser usado para alterar ou personalizar o comportamento quando uma exceção de sistema é lançada. O throw é obrigatório para exceções personalizadas e opcional para exceções de sistema.
  + **OBS/Importante:** Exceções personalizadas são um recurso importante em cenários que demandam maior clareza em erros específicos ou quando estamos lidando com regras de negócio complexas, mas é verdade que, em muitos sistemas, as exceções padrões do .NET são suficientes e podem simplificar o código.
  + **Resumo:** Tratamento de exceções em C# é o processo de capturar e gerenciar erros durante a execução, utilizando blocos try, catch e finally para exceções padrão e o comando throw para lançar exceções, seja de sistema ou personalizadas, garantindo que o programa lide adequadamente com situações de erro e mantenha sua estabilidade.