Curso C# Completo Programação Orientada a Objetos + Projetos

Capítulo: Tratamento de exceções

http://educandoweb.com.br

Prof. Dr. Nelio Alves

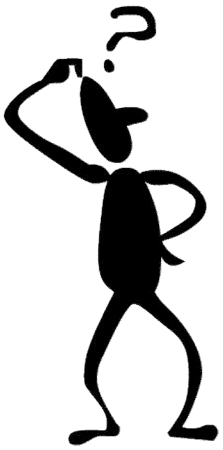
Discussão inicial sobre exceções

http://educandoweb.com.br

Prof. Dr. Nelio Alves

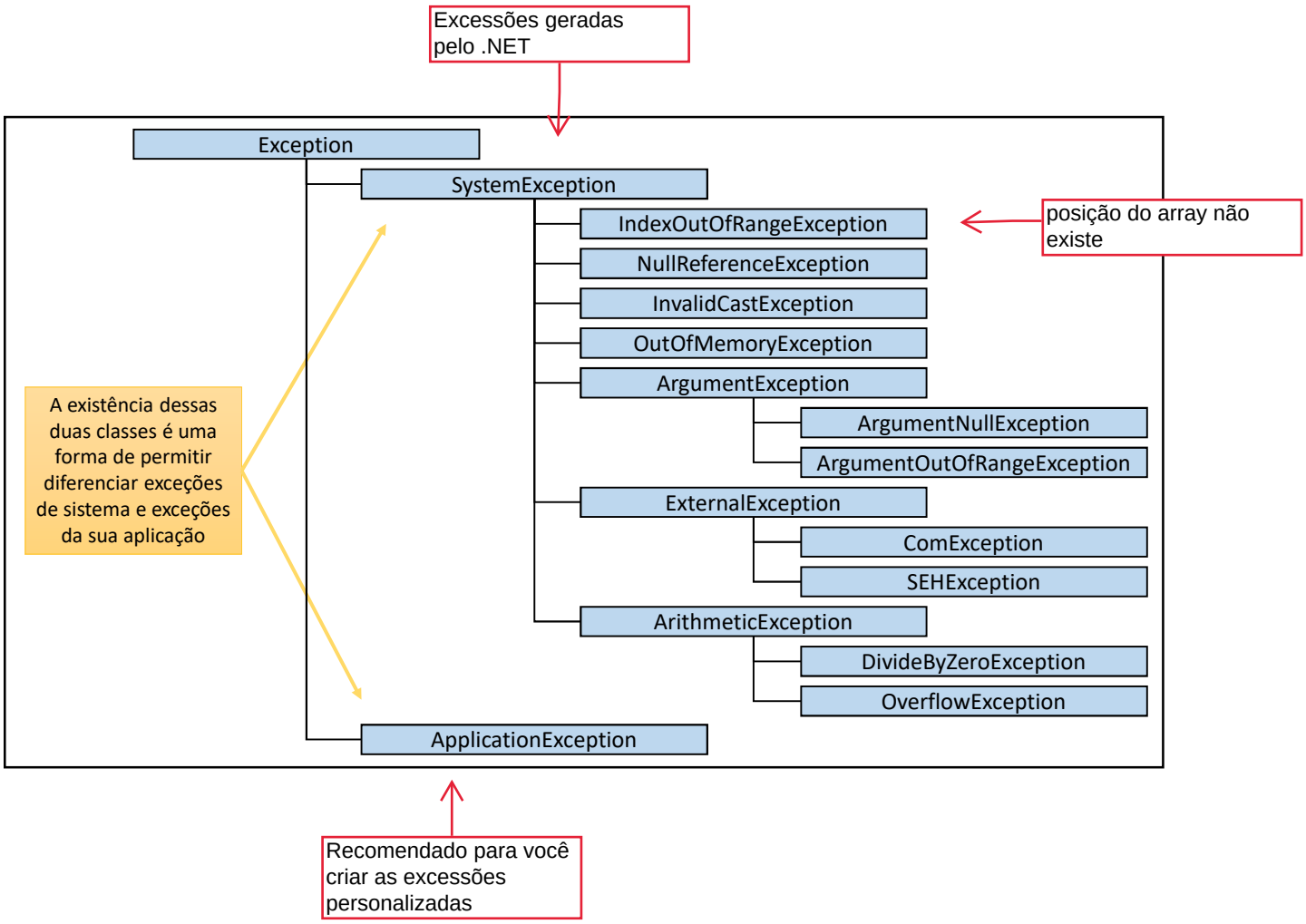
Atenção: pode ser um pouco difícil de entender vendo somente a teoria

Mas tudo ficará claro com os exemplos práticos nas próximas aulas ☺



Exceções

- Uma exceção é qualquer condição de erro ou comportamento inesperado encontrado por um programa **em execução**
- No .NET, uma exceção é um objeto herdado da classe System.Exception
- Quando lançada, uma exceção é propagada na pilha de chamadas de métodos em execução, até que seja capturada (tratada) ou o programa seja encerrado



Por que exceções?

- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas

  em vez de usar vários "if-else"

- Vantagens:
  - Delega a lógica do erro para a classe / método responsável por conhecer as regras que podem ocasionar o erro (a própria classe trata os possíveis erros dela)
  - Trata de forma organizada (inclusive hierárquica) exceções de tipos diferentes
  - A exceção pode carregar dados quaisquer

Exemplo de exceção (divisão por 0):

```
10
0

Unhandled Exception: System.DivideByZeroException: Attempted to divide by zero.
   at Course.Program.Main(String[] args) in C:\temp\projetos\Course\Course\Program.cs:line 10
Pressione qualquer tecla para continuar. . .
```

Estrutura try-catch

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Estrutura try-catch

- Bloco try
  - Contém o código que representa a execução normal do trecho de código que **pode** acarretar em uma exceção
- Bloco catch
  - Contém o código a ser executado caso uma exceção ocorra
  - Deve ser especificado o tipo da exceção a ser tratada (upcasting é permitido)
- Demo

Sintaxe

```
try {

}
catch (ExceptionType e) {

}
catch (ExceptionType e) {

}
catch (ExceptionType e) {

}
```

Código pra tratar a excessão fica dentro deos blocos "catch"

Demo

```
using System;

namespace Course {
    class Program {
        static void Main(string[] args) {

            try {
                int n1 = int.Parse(Console.ReadLine());
                int n2 = int.Parse(Console.ReadLine());

                int result = n1 / n2;
                Console.WriteLine(result);
            }
            catch (DivideByZeroException) {
                Console.WriteLine("Division by zero is not allowed");
            }
            catch (FormatException e) {
                Console.WriteLine("Format error! " + e.Message);
            }
        }
    }
}
```

Bloco finally

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Bloco finally

- É um bloco que contém código a ser executado independentemente de ter ocorrido ou não uma exceção.
- Exemplo clássico: fechar um arquivo ou conexão de banco de dados ao final do processamento.

Sintaxe:

```
try {

}
catch (ExceptionType e) {

}
finally {

}
```

```
using System;
using System.IO;

public class ProcessFile {
    public static void Main() {
        FileStream fs = null;
        try {
            fs = new FileStream(@"C:\temp\data.txt", FileMode.Open);
            StreamReader sr = new StreamReader(fs);
            string line = sr.ReadLine();
            Console.WriteLine(line);
        }
        catch (FileNotFoundException e) {
            Console.WriteLine(e.Message);
        }
        finally {
            if (fs != null) {
                fs.Close();
            }
        }
    }
}
```

Criando exceções personalizadas - PARTE 1

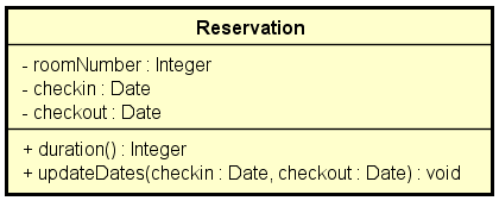
http://educandoweb.com.br

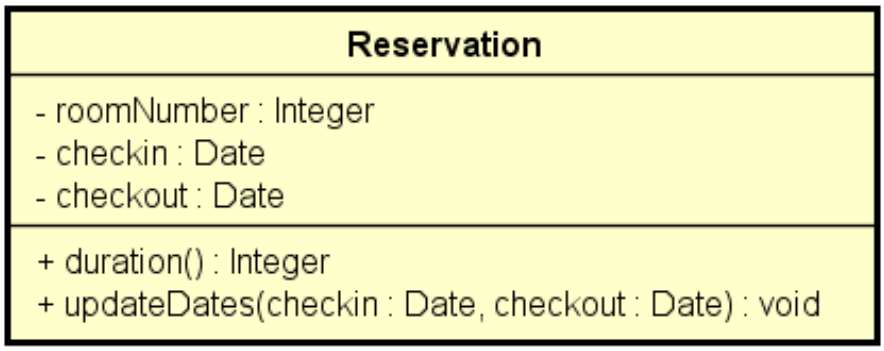
Prof. Dr. Nelio Alves

Problema exemplo

Fazer um programa para ler os dados de uma reserva de hotel (número do quarto, data de entrada e data de saída) e mostrar os dados da reserva, inclusive sua duração em dias. Em seguida, ler novas datas de entrada e saída, atualizar a reserva, e mostrar novamente a reserva com os dados atualizados. O programa não deve aceitar dados inválidos para a reserva, conforme as seguintes regras:

- Alterações de reserva só podem ocorrer para datas futuras
- A data de saída deve ser maior que a data de entrada





Examples

```
Room number: 8021
Check-in date (dd/MM/yyyy): 23/09/2019
Check-out date (dd/MM/yyyy): 26/09/2019
Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
Check-in date (dd/MM/yyyy): 24/09/2019
Check-out date (dd/MM/yyyy): 29/09/2019
Reservation: Room 8021, check-in: 24/09/2019, check-out: 29/09/2019, 5 nights
```

```
Room number: 8021
Check-in date (dd/MM/yyyy): 23/09/2019
Check-out date (dd/MM/yyyy): 21/09/2019
Error in reservation: Check-out date must be after check-in date
```

Examples

```
Room number: 8021
Check-in date (dd/MM/yyyy): 23/09/2019
Check-out date (dd/MM/yyyy): 26/09/2019
Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
Check-in date (dd/MM/yyyy): 24/09/2015
Check-out date (dd/MM/yyyy): 29/09/2015
Error in reservation: Reservation dates for update must be future dates
```

```
Room number: 8021
Check-in date (dd/MM/yyyy): 23/09/2019
Check-out date (dd/MM/yyyy): 26/09/2019
Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
Check-in date (dd/MM/yyyy): 24/09/2020
Check-out date (dd/MM/yyyy): 22/09/2020
Error in reservation: Check-out date must be after check-in date
```

Criando exceções personalizadas - PARTE 2

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Solução do problema

- Solução 1 (muito ruim): lógica de validação no programa principal
  - Lógica de validação não delegada à reserva
- Solução 2 (ruim): método retornando string
  - A semântica da operação é prejudicada
    - Retornar string não tem nada a ver com atualização de reserva
    - E se a operação tivesse que retornar um string?
  - Ainda não é possível tratar exceções em construtores
  - A lógica fica estruturada em condicionais aninhadas

Criando exceções personalizadas - PARTE 3

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Resumo da aula

- Cláusula throw: lança a exceção / "corta" o método
- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
  - Lógica delegada
  - Construtores podem ter exceções
  - Código mais simples. Não há aninhamento de condicionais: a qualquer momento que uma exceção for disparada, a execução é interrompida e cai no bloco catch correspondente.
  - É possível capturar inclusive outras exceções de sistema

https://github.com/acenelio/exceptions1-csharp

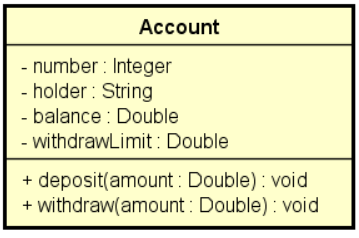
Exercício de fixação

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Exercício de fixação

Fazer um programa para ler os dados de uma conta bancária e depois realizar um saque nesta conta bancária, mostrando o novo saldo. Um saque não pode ocorrer ou se não houver saldo na conta, ou se o valor do saque for superior ao limite de saque da conta. Implemente a conta bancária conforme projeto abaixo:



Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00

Enter amount for withdraw: 100.00
New balance: 400.00
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00

Enter amount for withdraw: 400.00
Withdraw error: The amount exceeds withdraw limit
```

Examples

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 500.00
Withdraw limit: 300.00

Enter amount for withdraw: 800.00
Withdraw error: The amount exceeds withdraw limit
```

```
Enter account data
Number: 8021
Holder: Bob Brown
Initial balance: 200.00
Withdraw limit: 300.00

Enter amount for withdraw: 250.00
Withdraw error: Not enough balance
```

https://github.com/acenelio/exceptions2-csharp