Programação Orientada à Objetos

Aula 14
Tratamento de Exceções

Henrique Poyatos
henrique.poyatos@bandtec.com.br

Introdução

- Muitas vezes ao tentarmos executar um comando podemos obter um erro;
- Por exemplo, imagine que podemos tentar ler um arquivo mas ele não existe;
- Quando tentarmos executar essa operação de leitura ocorrerá um erro ou exceção;
- Seria interessante poder tratar exceções como essa, exibindo, no caso acima, uma mensagem de erro ao usuário;
- Tratar uma exceção é muito mais elegante do que simplesmente abortar o programa.

Sintaxe try...catch..finally

```
try
{
    // bloco de comandos
}
catch (<Exceção>)
{
    // bloco de comandos
}
//Daqui para baixo é opcional
catch (<Exceção>)
{
    // Podemos ter quantos blocos catch
    // for necessário
}
finally
{
    // bloco de comandos
}
```

Exemplo – Divisão por Zero

```
static void Main(string[] args)
{
    Console.WriteLine("Digite o divisor");
    int divisor = Int32.Parse(Console.ReadLine());

    //E se o usuário digitar zero?
    try
    {
        int total = 10 / divisor;
    }
    catch (DivideByZeroException ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

Explicação

- O bloco *try* contém os comandos que podem gerar uma exceção (erro);

- O bloco *catch* declara uma exceção de um determinado tipo, e dentro do seu bloco deve-se colocar o tratamento dado caso essa exceção ocorra (por ex: exibir uma mensagem de erro);

- Podem existir mais de um bloco *catch*, um para cada tipo de exceção que pode ser gerada;

- O bloco *finally* é opcional, mas se existir os comandos dentro dele serão executados independentemente de ocorrer ou não uma exceção;

Como funciona

- O programa tenta executar cada comando dentro do bloco **try**;
- Se algum erro ocorrer durante a execução de algum dos comandos, o fluxo do programa sai do bloco **try** e vai para o **catch**;
- O programa vai procurar **catch** que contém o tipo da exceção gerada;
- Se encontrar, executa o bloco de comandos dentro do catch e depois continua a execução do programa;
- Se não encontrar, o programa é **abortado**;
- Se o bloco **finally** foi declarado, ele é executado incondicionalmente após o catch.

Exemplo: Leitura de um arquivo

```
static void TestCatch2()
{
    System.IO.StreamWriter sw = null;
    try
    {
        sw = new System.IO.StreamWriter(@"C:\test\test.txt");
        sw.WriteLine("Hello");
    }
    catch (System.IO.FileNotFoundException ex)
    {
        // Put the more specific exception first.
        System.Console.WriteLine(ex.ToString());
    }
```

Exemplo: Leitura de um arquivo

```
    catch (System.IO.IOException ex)
    {
        // Put the less specific exception last.
        System.Console.WriteLine(ex.ToString());
    }
    finally
    {
        sw.Close();
    }

    System.Console.WriteLine("Done");
}
```

Forçando o erro – throw

- Você pode forçar um erro através da palavra reservada throw.

Exemplo – Throws

```
static void Main(string[] args)
{
    Console.WriteLine("Digite o divisor");
    int divisor = Int32.Parse(Console.ReadLine());
    int total = 0;
    try {
        if (divisor == 0)
        {
            throw new DivideByZeroException();
        }
        else
        {
            total = valor / divisor;
        }
    }
    catch (DivideByZeroException ex) {
        Console.WriteLine(ex.ToString());
    }
}
```

Criando Excepções Personalizadas


```
class CustomException : Exception
{
    public CustomException(string message)
    {

    }

}

private static void TestThrow()
{
    CustomException ex =
        new CustomException("Exceção Customizada em
TestThrow()");

    throw ex;
}
```