DESENVOLVIMENTO DE PROJETOS COM C#

GILBERTO S. T. JUNIOR
GILBERTOTAKEYA@YAHOO.COM.BR

Aula 01

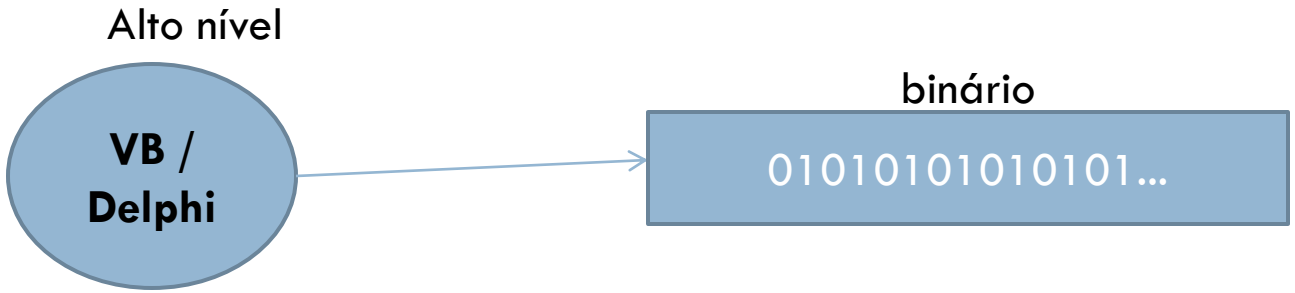
Introdução ao C# e a padrões de projeto

Sobre o C#

- Case sensitive
- Orientado a objetos
- Linguagem interpretada e não compilada
- Quebra de paradigma, estamos deixando o modelo Monolítico(tudo dentro de um exe) e partimos para um modelo SOA(Service Oriented Architeture)

Ambiente gerenciado (CLR)

- Linguagem compilada



- Linguagem interpretada

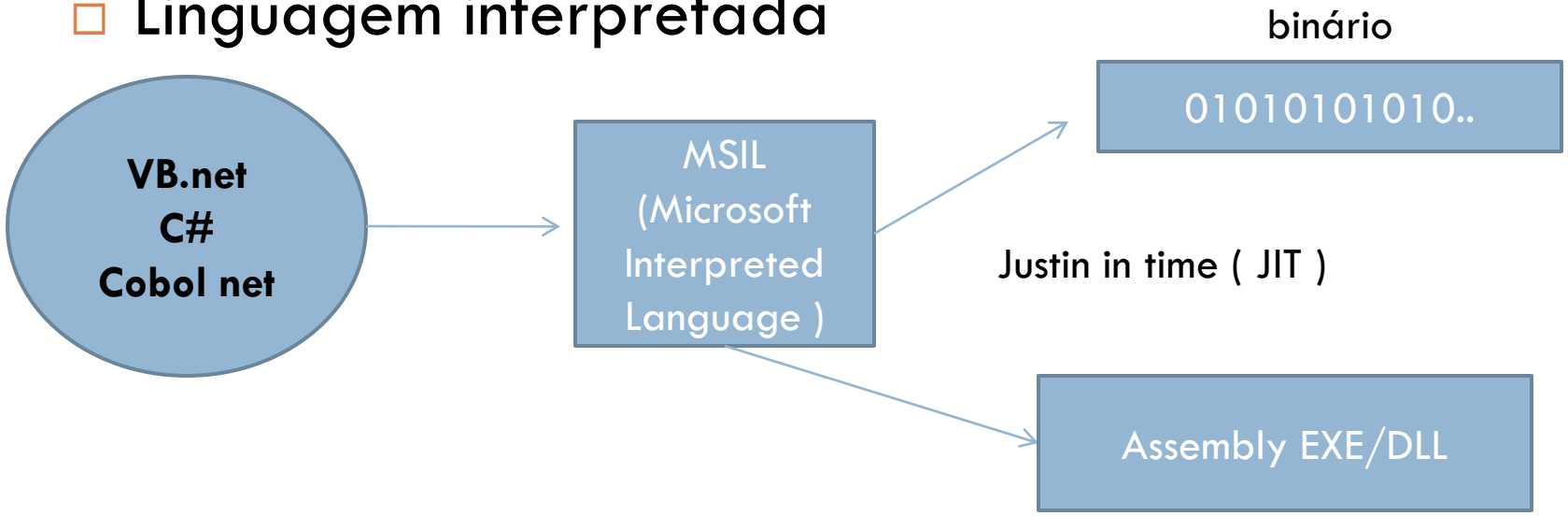
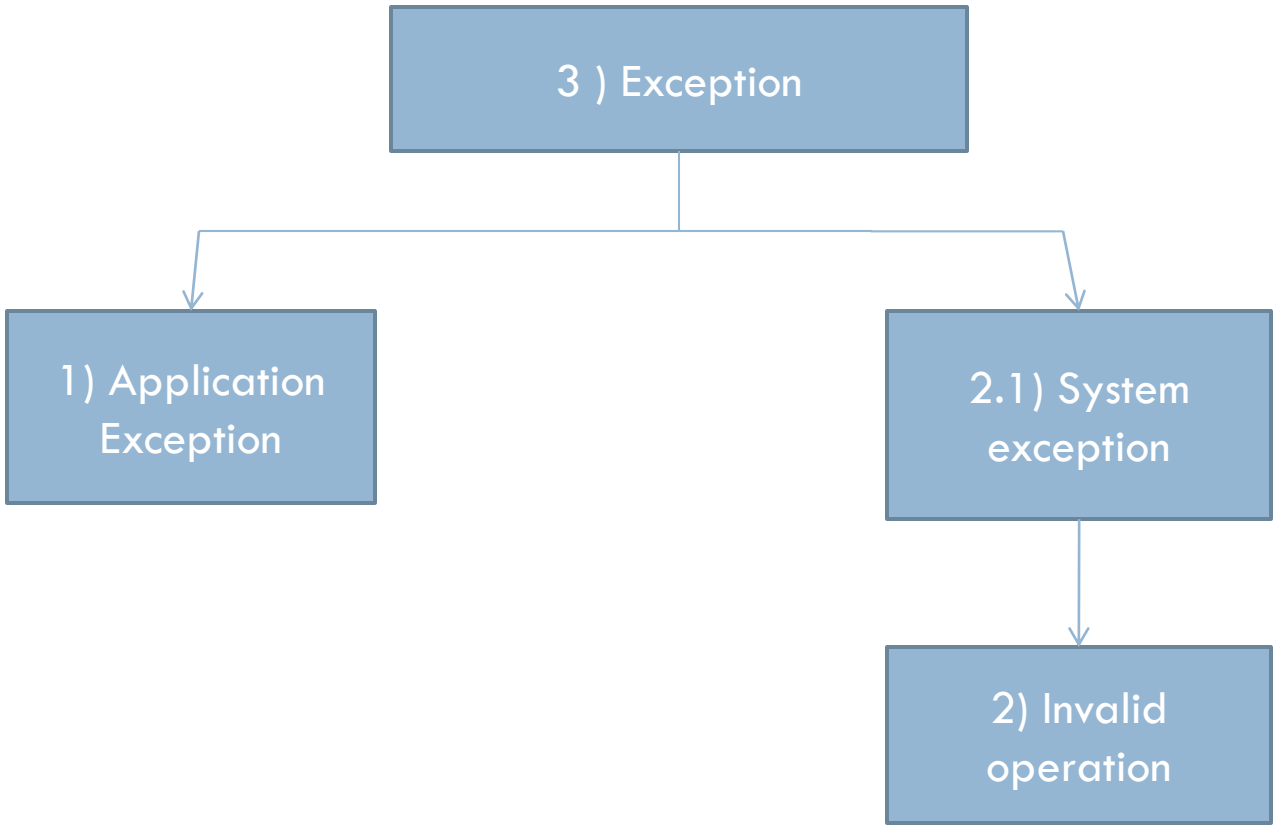


Diagrama de exceção



Exceções

```
//Verifica se tem exceção
try // Todo objeto é um texto, a exceção ocorrerá no momento
    da conversão...
 {    // Se o usuário informar letra ele vai gerar uma exceção e
    dará erro de
     // Exception
     UInt32 codigo = Convert.ToUInt32( txtCodigo.Text );
}
catch (Exception excecao) // Exceção geral
{
    MessageBox.Show(Excecao.Message);
}
```

Nomenclatura

- Button – btnOqueFazaBagaça
- ComboBox - cmbOqueFazaBagaça
- Listbox – lstOqueFazaBagaça
- TextBox- txtOqueFazaBagaça
- Label – lblOqueFazaBagaça
- Linklabel – linklblOqueFazaBagaça
- MaskedTextBox – msktxtOqueFazaBagaça
- PictureBox – pictOqueFazaBagaça
- RadioButton - rdbOqueFazaBagaça

Definições

- Namespace – Espaço com nome definido
- Class – classe que contém metodos, atributos pertencentes a classe.
- Public – público para todos no namespace
- Protected- Pode utilizar na classe e na herança
- Private – Pode utilizar somente na classe

Tipos de variavéis

- string – conjunto de caracteres
- Int16 – equivalente ao short
- Int32 – equivalente ao int
- Int64 – equivalente ao long

Padrões de projeto

- Nome do projeto será da seguinte forma:

NomeEmpresa.Departamento.Produto.Camada

Ex: Empresa: Blink

Departamento: Programação

Produto: Mobile

Camada: Model

Granularidade

- Quando trabalhamos com orientação a objetos, temos que saber utilizar de forma correta a granularidade.

- O que é?

Dividir blocos corretamente.

- O que ganhamos?

Baixo acoplamento(divisão) e alta coesão(Evitar manutenções em vários lugares (efeito cascata))

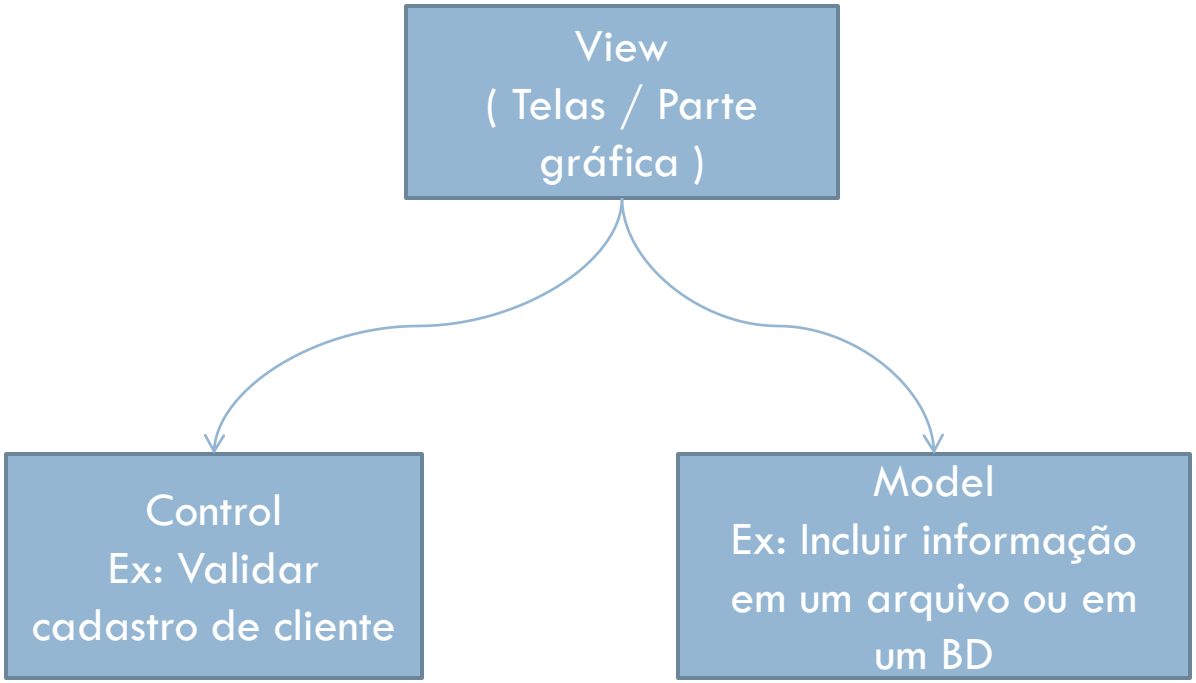
Padrões de projeto

- Iremos utilizar o MVC (Model / View / Controller) para desenvolvimento, com isso iremos dividir o nosso projeto em várias soluções.O que ganhamos?

1. Maior facilidade de manutenção
2. Mais pessoas trabalhando em vários projetos ao mesmo tempo

- Model – Modelo
- View – Visão, interface gráfica
- Controller – Validações no projeto

Exemplo de MVC



OBS: Em um projeto poderá ter mais de uma Control e Model.

Comandos básicos

- Pegar texto do objeto TextBox

string j = Objeto.text;

- Setar texto no objeto

Objeto.text = “Texto”;

Comandos básicos

- Condicional

```
if( variavel1 comparador variavel2)
{
    bloco se a condição acima for verdadeira
}
else
{
    bloco se a condição acima for falsa
}
```

Comparador(==, !=, >=, <=)

Comandos básicos

- Verificar se o texto digitado esta vazio

```
//Verifica se o campo esta vazio
if( txtNome.Text == string.empty )
{
    MessageBox.Show("Favor preencher o nome!");
    return; //Não permite a execução da linha abaixo
}
```

Propriedades de objeto

- TabIndex

>> Propriedade que permite com que o usuário dê o tab e que vá para o campo da frente. Verificar o TabIndex

- Verificar as demais propriedades ou solicitar que elas sejam explicadas em sala de aula

Exercício

- Criar um projeto para o departamento financeiro da empresa Zé Roela Corporation. O financeiro pediu uma software que faça o cadastro de clientes e que grave em arquivo texto todos os clientes novos. O nome do software deverá ser RoelaSoft.

- O cadastro de cliente, deverá conter as seguintes validações:

- CPF tem que ser 11 e CNPJ tem que ser 14.
- O campo de CPF e CNPJ não pode contar caracteres.
- O usuário deverá informar as seguintes informações do cliente: Código, nome, RG, CPF, CEP, endereço.
- Salvar as informações do cliente em um arquivo texto.
- Utilizar MVC para o desenvolvimento do projeto

Arquivos - Escrita

1) Declarar a classe System IO

```
using System.IO;
```

2) Gravando no arquivo...

```
//Instanciar a classe de escrita para escrever o arquivo
StreamWriter escreverArquivo = new
StreamWriter(@"C:\Cliente.txt", true);

//Escrever no arquivo
escreverArquivo.WriteLine(codigo);
escreverArquivo.WriteLine(nome);

//Fechar o arquivo
escreverArquivo.Close();
```

Arquivos - Leitura

1) Declarar a classe System IO

```
using System.IO;
```

2) Lendo o arquivo...

```
//Instanciar a classe de leitura para escrever o arquivo
StreamReader rd = new StreamReader(@"c:\Cliente.txt");
//Enquanto não for final de arquivo...
while(!rd.EndOfStream)
{
    string linha = rd.ReadLine();
    Console.WriteLine(linha);
}
rd.Close();
```