05 – Como utilizar POO no ambiente gráfico do C#?

Abra um novo projeto, mas desta vez, não escolha uma aplicação Console. Escolha Windows Form Application.

A diferença para a aplicação de Console é a presença de um objeto Form (Formulário), que é uma janela padrão Windows . Neste formulário, colocaremos os objetos visuais para compor nossa aplicação.

🖉 poovisual1 - Microsoft Visual C# 2010 Express						– 0 ×
<u>File Edit View Project Debug Data Format Tools Window H</u> elp						
i 🗊 🖼 - 🔯 🛃 🖇 🗠 🖄 - 🔍 - 🕨 🙋 - 👘 - 🔛 - 🔛 - 🔛 - 🔛 - 🔛 - 🔛 - 🔛						
[尊] 臣 수 비 교 사 표 (금 전 路 徹) 바 많 많 않 응 한 않 않 [11 년] '김 명, 말,						
/j Form1.cs [Design]* ×					Solution Explorer	~ ₽ ×
Torona de Objetos no ambiente gráfici				,	Solution 'poorie Solution 'poorie Solution 'poorie Propertie Solution 'poorie Formlos Formlos Formlos Program.	ual1' (1 project) s es 1.Designer.cs 1.resx .cs
					Properties Form1 System.Windo	
Error List				- ₽ >	ShowInTaskbar	True
O Errors O Warnings O Messages	File	Line	Column	Project	Size SizeGripStyle StartPosition Tag Text The text associated w	614; 356 Auto WindowsDefaultLo Utilização de Obje v
🚜 Error List 🔳 Output				15, 15	<u>∓</u> ⊟614>	x 356

Para inserir objetos visuais, utilize a barra de ferramentas (toolbox) que deve estar à esquerda, ocuta, mas que ao aproximar o mouse ela deve se abrir (instalação padrão do VS. Estas configurações podem ser alteradas).

🛤 poovis	ual1 - Microsoft Visual C# 2010 Express		– O ×
<u>File</u> Edit	<u>View Project Debug Data Format Too</u>	s <u>Wi</u> ndow <u>H</u> elp	
: 🛅 🔛	• 😂 • • • • 🖌 🔊 • • • • •	🧃 🚽 🚽 🖓 🔁 환 🚆 -	
:住口曰	る 訓 市 み 山 民 新 廃 道 (の) 翌	맛 및 수 학 화 하! [FT] 14 및 팩 _	
Tlb-			Calculate Contants
- Incolade	TextBox		
	ToolTin		
× -	TreeView		Solution poovisual1 (1 project)
Pi 📻	WebBrowser		> Properties
a V Cor	tainers		> 🔄 References
S 🕨	Pointer		✓ III Form1.cs
765 **	FlowLayoutPanel		Form1.Designer.cs
×**	GroupBox		Program cc
	Panel		Fighanics
	SplitContainer	b	
	TabControl		
	TableLayoutPanel		
✓ Mer	nus & Toolbars		Properties 🝷 🕂 🗙
k	Pointer		Form1 System.Windows.Forms.Form +
-10	ContextMenuStrip		1 4 m 4 m
8	MenuStrip		> Padding 0.0.0.0
	StatusStrip		RightToLeft No
0.	ToolStrip		RightToLeftLayo False
	ToolStripContainer		Showlcon True
✓ Dat	1	+ ₽×	ShowInTaskbar True
k	Pointer		> Size 614; 356
	Chart PindingNavigator	File Line Column Project	SizeGripStyle Auto
000	Bindingivavigator		StartPosition WindowsDefaultLo
	DataGridView		Text Utilização de Obie
	DataSht View		
× Cor	nnonents		The text accoriated with the control
k	Pointer		The text associated with the control.
Ready			

Em seguida, escolha o objeto desejado, clique e arraste até o formulário. Pode-se também dar um duplo clique do mouse sobre o objeto.

Ao selecionar qualquer objeto (desde o formulário até os objetos inseridos), podemos alterar seus atributos na janela de Propriedades (geralmente na direita, embaixo, ou pressionando a tecla F4 do teclado, lembrando que também podemos ocultar ou mover de posição esta janela).

Existem alguns atributos padrões a todos os objetos e existem atributos particulares para cada objeto. Por exemplo, a maioria dos objetos possuem a propriedade Text. Ela controla o texto que é exibido no objeto. A propriedade (Name), que aparece entre parênteses mesmo, define o nome do objeto, como se fosse um nome de variável.

Vamos inserir então 9 objetos, sendo 3 label, 3 textbox e 3 button, distribuindo-os como no exemplo:

🔛 Form1		—	
label1			
label2			
label3			
button1	button2	button3	

Em seguida, clicando nos objetos e alterando as propriedades Text, coloque os textos como na figura em seguida:

🔜 Form1	—	×
ld:		
Nome:		
F-mail:		
Gravar no objeto Limpar dados R	lecuperar do objeto	

Será preciso adequar o tamanho de alguns objetos. Podemos utilizar a propriedade Size de cada um deles, ou clicando com o mouse e redimensionando no próprio formulário.

Vamos alterar as propriedades (Name) dos objetos TextBox e Button. Na sequência, coloque txtld, txtNome e txtEmail nas textboxs e btnGravar, btnLimpar e btnRecuperar nos button.

O título do formulário também pode ser alterado na propriedade Text do formulário.

Em seguida, vamos começar a programação. Para criar uma nova classe (como criamos nas aplicações Console), não poderemos utilizar o arquivo que define o código gráfico. Iremos criar um novo arquivo no projeto específico para isso. Para isso utilize uma das opções:

- 1) Menu Project, ADD Class
- 2) Atalho do teclado Shift + Alt + C
- 3) Em Solution Explorer (sobre a janela de propriedades na direita), botão direito do mouse, add Class exemplo na figura a seguir.

File Edit View Project Debug Data Iools Window He うちょうない A A A A A A A A A A A A A A A A A A A	() · · · · · · · · · · · · · · · · · · ·	Ŧ								
Form Lcs (Design) × Integração com objetos Integração com objetos Id:			New Item Existing Item New Folger Windows Eorm User Control	Ctrl+Shift+A Shift+Alt+A		B <u>u</u> ild Rebuild Pu <u>b</u> lish Add <u>Refer</u> Add <u>Servi</u> Set as St <u>a</u> Debug	rence ce Reference rtUp Project	Solution	Explorer et lution 'poo poovisual Proper Refere Form1 Form1 Progra	visual1' (1 project) 11 11 11 11 11 11 11 12 12 12 12 12 12
Gravar no objeto Limpar dado	s Recuperar do objeto		<u>C</u> lass	Shift+Alt+C	×	Cu <u>t</u> Paste Remo <u>v</u> e Rena <u>m</u> e P <u>r</u> operties	5	Ctrl+X Ctrl+V Del Alt+Enter	File Folder	+ 🕂 × t Properties + poovisual1.csproj C:\Users\Sergio\docu
Error List O Errors ① 0 Messages Description			File	Line	(Column	Project	Project	File	

Dê o nome para sua classe (neste caso, chamei de Agenda)

Add New Item - poovisual1						×
Installed Templates	Sort by:	Default IIII		Search Installed Templates		٩
Visual C# Items	(H)	Class	Visual C# Items	Type: Visual C# Items		
Online Templates		Class	visual es iterns	An empty class definition		
	° c ≢	Interface	Visual C# Items			
		Windows Form	Visual C# Items			
		User Control	Visual C# Items			
		User Control (WPF)	Visual C# Items			
		About Box	Visual C# Items			
		ADO.NET Entity Data Model	Visual C# Items			
		ADO.NET EntityObject Generator	Visual C# Items			
		ADO.NET Self-Tracking Entity Generator	Visual C# Items			
		Application Configuration File	Visual C# Items			
	-	Application Manifest File	Visual C# Items	~		_
<u>N</u> ame: agenda.cs						
				Add	Cance	1

E sua classe estará criada e pronto para ser implementada:



Repare que foi criada uma aba na janela principal, e para voltar ao formulário basta selecionar a aba Form1.CS. Na Solution Explorer também é possível dar dois cliques no objeto que deseja visualizar na janela principal (seja na classe ou no formulário).

Vamos criar a classe agenda, criando os atributos e os métodos a seguir:

```
class agenda
{
    public int Id { get; private set; }
    public string Nome { get; private set; }
    public string Email { get; private set; }

    public void GravaPessoa(int NovoId, string NovoNome, string NovoEmail)
    {
        this.Id = NovoId;
        this.Nome = NovoNome;
        this.Email = NovoEmail;
    }
}
```

Agora, vamos voltar ao form1 (formulário gráfico). Vamos fazer os objetos acionarem métodos de acordo com algumas ações, que neste caso, serão cliques nos botões.

Como queremos que um objeto seja preenchido para depois ser recuperado durante a execução do programa, precisamos criar uma instância de um objeto do tipo Agenda. Para criarmos a instância que possa ser utilizada em todos os objetos deste formulário, iremos instanciar o objeto de maneira global.

Se clicarmos duas vezes em qualquer lugar do formulário, que não tenha nenhum objeto, apenas o formulário, iremos abrir o método Form_Load, como na imagem:

🕰 poovisual1 - Microsoft Visual C# 2010 Express		– 0 ×
File Edit View Project Debug Data Tools Window Help		
: 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		
10.222.4倍至2。		
<mark>∕∕a Form1.cs* ×</mark> agenda.cs Form1.cs [Design]*	-	Solution Explorer 🛛 🔻 🕂 🗙
🗧 🎨 poorisual1.Form1 🔹 🖓 Form1_Load(object sender, EventArgs e)		🔚 🗿 🛃 🔳
<pre> { public partial class Form1 : Form { public Form1() { InitializeComponent(); } } </pre>	*	Solution 'pooriual1' (1 project) CP pooriual1 En Properties En References En Berdia.cc En Form1.cs En Form1.ress Form1.ress Form1.ress Program.cs
<pre>private void Form1_Load(object sender, EventArgs e) {</pre>		Properties • 7 ×
}		
100 % • <	>	
Error List	+ # ×	
O DErrors 🛕 O Warnings 🕕 O Messages		
Description File Column Project	^ 	
📸 Error List 📕 Output		
Ready Ln 19	Col	I 2 Ch 2 INS

Para instanciarmos um objeto que possa ser utilizado em todo o formulário, iremos fazer isso logo depois da instanciação do Form, como na imagem:



agenda Registro = new agenda();

Neste caso, criamos um objeto chamado Registro, que é instanciada da classe Agenda (nome da classe que criamos no início).

Agora vamos aos Clicks dos botões. Para isso, para criarmos um método Click de um botão de forma fácil, basta dar dois cliques sobre o botão que deseja programar o evento Click. Vamos fazer primeiramente o Gravar Objetos. Iremos inserir o código:

poovisual - Microsoft Visual C# 2010 Express	– a ×
jile Edit <u>V</u> iew Project Debug Data Iools <u>W</u> indow <u>H</u> elp	
13111-121111111111111111111111111111111	
다 방법사 個 물 일부	
Form1.cs* × agenda.cs Form1.cs [Design]*	✓ Solution Explorer ✓ ₽ ×
* poovisual1.Form1 v 😵 btnGravar_Click(object sender, EventArgs e)	- 🕒 😰 🖅
<pre>private void Form1_Load(object sender, EventArgs e) { } private void btnGravar_Click(object sender, EventArgs e) { Registro.GravaPessoa(Convert.ToInt32(txtId.Text),txtNome.Text,txtEmail.Text); } </pre>	Image: Solution 'poovisual' (1 project) Image: Solution 'poovisual' I
Free List	
Concess	
Description File Line Column Project	
🖧 Error List 🔳 Output	
esdy Ln 29	Col 90 Ch 90 INS

private void btnGravar_Click(object sender, EventArgs e)
{
Registro.GravaPessoa(Convert.ToInt32(txtId.Text),txtNome.Text,txtEmail.Text);
}

Agora vamos fazer o botão que irá limpar os dados dos objetos Texts, como na imagem:

g poovisual - microsoft visual C# 2010 Express		
<u>File</u> <u>Edit</u> <u>View</u> <u>Project</u> <u>Debug</u> D <u>a</u> ta <u>T</u> ools <u>Window</u> <u>H</u> elp		
· · · · · · · · · · · · · · · · · · ·		
Form1.cs* × agenda.cs Form1.cs [Design]*	-	Solution Explorer 🛛 🔻 🕂 🗙
🛜 🛠 poovisual1.Form1 🗸 😵 btnLimpar_Click(object sender, EventArgs e)	-	🗟 🚱 🖬 🗵
<pre>private void btnGravar_Click(object sender, EventArgs e)</pre>		🤜 Solution 'poovisual1' (1 project)
	<u>^</u>	✓
Pagistra GravaPassaa(Convert ToInt32(tytId Tayt) tytNome Tayt	+ + + + Empil Toy+).	> 📴 Properties
Registro. Gravaressoa (Convert. Tornes2(Exclusive), Excholle. Text	., CALLMAII. TEAC/,	> References
		V Eprm1.cs
6		Form1.Designer.cs
private void btnLimpar_Click(object sender, EventArgs e)		Form1.resx
		Program.cs
tyttd Toyt - ""		
txtNome.Text = "";		
txtEmail.Text = "";		Properties - A ×
}		-
		21 41 🖾
100.9%	~	
Error List	- 4 ×	
🔽 🗘 0 Errors 🛕 0 Warnings 🚯 0 Messages		
Description File	Line Column Project	
Ready	Ln 36 Co	132 Ch 32 INS

```
private void btnLimpar_Click(object sender, EventArgs e)
{
    txtId.Text = "";
    txtNome.Text = "";
    txtEmail.Text = "";
}
```

Por fim, vamos criar o botão que pega os dados de um objeto e coloca nos objetos Texts.

poovis	sual i - Microsoft Visual C# 2010 Express								· · ·
<u>File</u> Edit	t <u>V</u> iew <u>P</u> roject <u>D</u> ebug D <u>a</u> ta <u>T</u> ools <u>W</u> indow <u>H</u> elp								
🛅 🛅	• 😂 🛃 🐉 🕹 🖄 🔊 • (* •) 🕨 🚺 🔸 🖓	र्य 🕾 🔊 🛃 🖕							
1 🗔 🐁	<u>_</u> ↔ @								
V. Form	agenda cs Form1 cs [Design]*						-	Solution Explorer	≁ û ×
J 👷	enviruali Ferma	- Resumerar Click(abject	conder EventAres e)				_		
olbo	txtId Text = "".	• @• burkecuperar_crick(object	sender, EventArgs e/				+	Solution 'poovisua	11' (1 project)
×	tytNomo Toyt = ""						^	 	
, D	tytEmeil Text = ""							> 📴 Properties	
ata S	txtEmail.Text = ";							> References	
purc	3							 agenua.cs Form1.cs 	
CR								🔛 Form1.	Designer.cs
E	private void btnRecuperar_Click(object	sender, EventAr	gs e)					Sorm1.	esx
	{							Program.cs	
	<pre>txtId.Text = Registro.Id.ToString(</pre>);							
	<pre>txtNome.Text = Registro.Nome;</pre>								
	<pre>txtEmail.Text = Registro.Email;</pre>							Properties	- ₽ ×
	}								-
								8 21 1	
	-								
100 %	6 • <						>		
Error	List						₹ ₽ ×		
0	0 Errors 🔥 0 Warnings 🚺 0 Messages								
	Description		File	Line	Column	Project			
						-			
📸 E	rror List 🧮 Output								
Ready						Lp.42	60	ии сь <i>и</i> и	INS
riceduy						611-10	- 00	Cline	1145

```
private void btnRecuperar_Click(object sender, EventArgs e)
{
    txtId.Text = Registro.Id.ToString();
    txtNome.Text = Registro.Nome;
    txtEmail.Text = Registro.Email;
}
```

Pronto. Basta rodar.

Não podemos gravar sem preencher dados. Para isso faça o preenchimento com os dados corretos, pois não utilizamos nenhum tipo de validação.

Exercício:

Crie uma interface chamada Conta que implementa um método Resultado, que recebe como parâmetro dois números do tipo Double, e retorna o resultado de uma operação matemática double da seguinte forma:

double resultado(double num1, double num2);

Implemente agora quatro classes, que herdam esta interface Conta, uma para cada uma das operações básicas (+, -, / e *).

Crie um formulário que possui o seguinte formato:

🔡 Calculadora	-	×
Num 1		
Num2		
+ -	*	/
Resulta	do	
	0	

São dois textBox para os números 1 e 2 da operação, 4 botões para as operações e 1 label para o resultado, além das label para rotular cada campo.

Ao clicar um dos botões de operação, pegar números digitados nos textBoxes, instanciar o objeto correto para a operação que o botão representa e enviar os parâmetros, recebendo o resultado em IblResultado para exibílo.