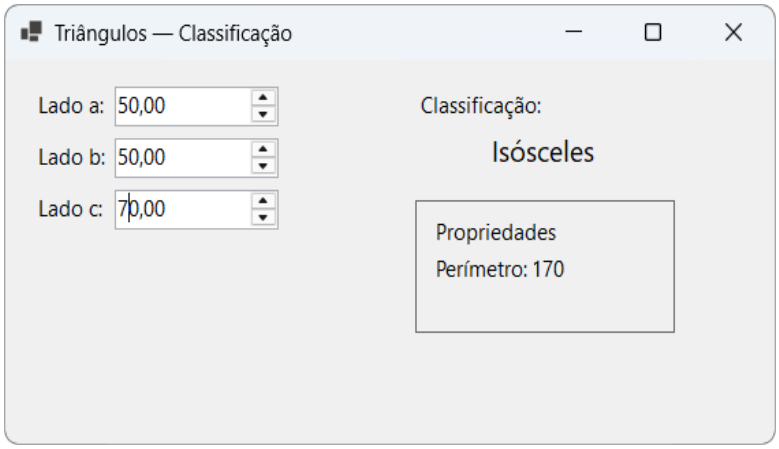
EXERCÍCIO 01 – CLASSIFICAÇÃO DE TRIÂNGULOS

Parte 1 — Construção do Formulário

Crie um projeto **Windows Forms** chamado TriangulosSimples. Renomeie a classe Form1 para FormTriangulo, insira manualmente os controles (conforme imagem abaixo) e configure-os:



1. **Entradas dos lados do triângulo**
   - numLadoA, numLadoB, numLadoC — três controles NumericUpDown, cada um representando um lado do triângulo.
   - Configure: DecimalPlaces = 2, Increment = 0.10, Minimum = 0, Maximum = 1000.
2. **Classificação**
   - lblClassificacao — uma Label para exibir o resultado da classificação (Equilátero, Isósceles, Escaleno ou Inválido).
   - Configure: AutoSize = false, TextAlign = MiddleCenter.
   - Por padrão, exibir apenas "—".
3. **Propriedades**
   - pnlPropriedades — um Panel para exibir informações adicionais quando o triângulo for válido.
   - Dentro do painel, coloque:
     - lblPerimetroTitulo (Label com texto “Perímetro:”)
     - lblPerimetro (Label que exibirá o valor do perímetro).
4. **Dica**
   - lblDica — uma Label com o texto: "Dica: a + b > c, a + c > b, b + c > a".
   - Este Label só será exibido quando o triângulo for inválido.

Parte 2 — Implementação do método AtualizarTriangulo

Implemente um método chamado AtualizarTriangulo no FormTriangulo. Ele deverá:

1. **Ler os valores** de numLadoA, numLadoB, numLadoC.
2. **Verificar validade** do triângulo usando as desigualdades:
    - a + b > c && a + c > b && b + c > a.
3. **Se inválido**:
    - lblClassificacao.Text = "Não forma triângulo"
    - lblClassificacao.BorderStyle = None
    - pnlPropriedades.Visible = false
    - lblDica.Visible = true
4. **Se válido**:
    - Classificar como **Equilátero**, **Isósceles** ou **Escaleno**.
        1. **Equilátero**: todos os lados iguais.

            a == b && b == c
        2. **Isósceles**: exatamente dois lados iguais.

            (a == b || a == c || b == c) **e** não equilátero.
        3. **Escaleno**: todos os lados diferentes.

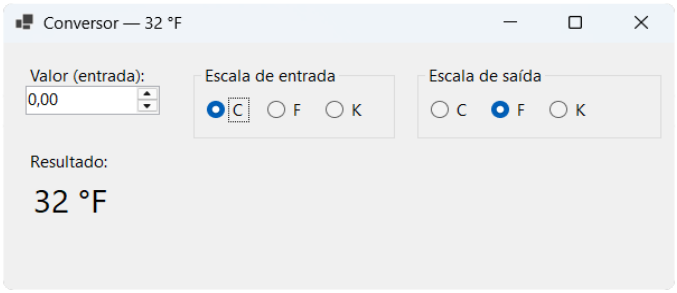
            a != b && a != c && b != c
    - Exibir a classificação em lblClassificacao.
    - Se for **Equilátero**, mudar o BorderStyle de lblClassificacao para FixedSingle; caso contrário, None.
    - Calcular o perímetro e exibir em lblPerimetro.
    - Tornar pnlPropriedades.Visible = true e lblDica.Visible = false.

Parte 3 — Ligação dos eventos

Configure os eventos ValueChanged dos três NumericUpDown para chamar o método AtualizarTriangulo.

EXERCÍCIO 02 – CONVERSOR DE TEMPERATURA

Parte 1 — Construção do Formulário
Crie um projeto **Windows Forms** chamado ConversorTemperatura. Renomeie a classe Form1 para FormConversor, insira manualmente os controles (conforme imagem abaixo) e configure-os:



1. **Entrada de valor**
   - numValor — um controle NumericUpDown para receber a temperatura de entrada.
   - Configure: DecimalPlaces = 2, Increment = 0.10.
   - Defina Minimum e Maximum de acordo com a escala escolhida (será ajustado dinamicamente no código).
2. **Escala de entrada**
   - grpEntrada — um GroupBox para agrupar os RadioButton de entrada.
   - Dentro dele, coloque três RadioButton:
     - rdbEntradaC (Celsius)
     - rdbEntradaF (Fahrenheit)
     - rdbEntradaK (Kelvin)
   - Garanta que um deles esteja sempre marcado (por padrão, Celsius).
3. **Escala de saída**
   - grpSaida — um GroupBox para agrupar os RadioButton de saída.
   - Dentro dele, coloque três RadioButton:
     - rdbSaidaC (Celsius)
     - rdbSaidaF (Fahrenheit)
     - rdbSaidaK (Kelvin)
   - Garanta que um deles esteja sempre marcado (por padrão, Fahrenheit).
4. **Resultado**
   - lblResultadoTitulo — Label com o texto **"Resultado:"**.
   - lblResultado — Label para exibir o valor convertido.
     - Configure a fonte maior (ex.: 18pt), AutoSize = true.
     - Valor inicial: "—".

Parte 2 — Implementação do método AtualizarConversao
Implemente um método chamado AtualizarConversao no FormConversor. Ele deverá:
1. **Ler o valor de entrada** do numValor.
2. **Converter a temperatura** para Kelvin (como unidade intermediária).
   - Se rdbEntradaC estiver marcado: K = valor + 273.15
   - Se rdbEntradaF estiver marcado: K = (valor + 459.67) * 5 / 9
   - Se rdbEntradaK estiver marcado: K = valor
3. **Converter de Kelvin para a escala de saída**.
   - Se rdbSaidaC: resultado = K - 273.15

- Se rdbSaidaF: resultado = K * 9 / 5 - 459.67
- Se rdbSaidaK: resultado = K

4. **Atualizar a interface**:
   - Exibir o valor convertido em lblResultado, com duas casas decimais.
   - Exibir a unidade ao lado (°C, °F ou K).
   - Atualizar também o título do formulário (this.Text) para mostrar o valor atual, no formato:
     - "Conversor — 25,00 °C"

Parte 3 — Ligação dos eventos

Configure os seguintes eventos para chamar o método AtualizarConversao:

1. ValueChanged do numValor.
2. CheckedChanged dos três RadioButton de entrada.
3. CheckedChanged dos três RadioButton de saída.

Extras (opcionais)

Ajuste dinâmico de limites do NumericUpDown

- Se a escala de entrada for Celsius: Minimum = -273,15, Maximum = 1000.
- Se for Fahrenheit: Minimum = -459,67, Maximum = 1832.
- Se for Kelvin: Minimum = 0, Maximum = 1273,15.

EXERCÍCIO 03 – MINI TABELA PERIÓDICA

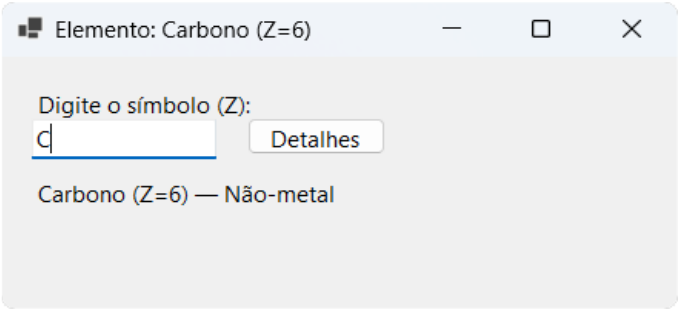
Parte 0 – Criação de Classe

Neste exercício iremos criar uma classe chamada Elemento. Para isso, clique em Adicionar Classe no projeto, a definindo da seguinte forma:

```
internal class Elemento
{
    public string Simbolo { get; set; }
    public string Nome { get; set; }
    public int NumeroAtomico { get; set; }
    public string Classe { get; set; }
}
```

Parte 1 — Construção do Formulário

Crie um projeto **Windows Forms** chamado MiniTabelaPeriodica. Renomeie a classe Form1 para FormElementos, insira manualmente os controles (conforme imagem abaixo) e configure-os:



1. **Entrada do símbolo**
    - txtSimbolo — TextBox para digitar o símbolo químico (ex.: H, O, Na, Fe).
    - Configure: MaxLength = 2.
2. **Exibição**
    - lblInfo — Label para mostrar **Nome (Z) — Classe** do elemento quando encontrado.
    - Valor inicial: "—".
3. **Ação**
    - btnDetalhes — Button que exibe uma MessageBox com os dados do elemento.
    - Deve ficar **habilitado apenas quando o símbolo for válido**.

Parte 2 — Implementação do método AtualizarUI

Implemente no FormElementos um método chamado AtualizarUI que:

1. Lê o texto de txtSimbolo.
2. Procura o elemento em uma List<Elemento> (comparação por símbolo).
3. Se **encontrar**:

   lblInfo.Text = "{Nome} (Z={n}) — {Classe}".

   btnDetalhes.Enabled = true.
4. Se **não encontrar**:

   lblInfo.Text = "—".

   btnDetalhes.Enabled = false.

Parte 3 — Ligação dos eventos

Conecte os seguintes eventos e **apenas chame** AtualizarUI neles (a lógica principal fica isolada no método):

1. txtSimbolo.TextChanged → atualiza em tempo real.
2. btnDetalhes.Click: se houver elemento válido, mostrar uma MessageBox com **Símbolo, Nome, Z, Classe**.

   Para exibir uma MessageBox, basta executar MessageBox.Show("Mensagem a ser exibida aqui");

Parte 4 — Dados (List)

No FormElementos.Load, ou seja, assim que o seu formulário for carregado, crie a lista:

```
elementos = new List<Elemento>
{
   new Elemento { Simbolo="H",  Nome="Hidrogênio", NumeroAtomico=1,  Classe="Não-metal" },
   new Elemento { Simbolo="He", Nome="Hélio",      NumeroAtomico=2,  Classe="Gás nobre" },
   new Elemento { Simbolo="C",  Nome="Carbono",    NumeroAtomico=6,  Classe="Não-metal" },
   new Elemento { Simbolo="O",  Nome="Oxigênio",   NumeroAtomico=8,  Classe="Não-metal" },
   new Elemento { Simbolo="Na", Nome="Sódio",      NumeroAtomico=11, Classe="Metal alcalino" },
   new Elemento { Simbolo="Fe", Nome="Ferro",      NumeroAtomico=26, Classe="Metal de transição" }
};
```