Created by Microsoft but extending xml

Language for creating UI, not only for WPF, but for XAMARIN, etc…

<Grid.ColumnDefinitions>

<ColumnDefinition Width="123"/> 123 pixeles de ancho

<ColumnDefinition Width="auto"/> lo que ocupen sus elementos de dentro

<ColumnDefinition Width="\*"/> The entire available space will be asigned. So if we have 2 columns, each one with have the half… If for example we have 4 columns, we ARE DIVIDING IF 4 EQUALLY COLUMNS

Filas espaciadas iguales (alto) 🡪 1/7 (Hay 6 filas, pero una ocupará el doble, por tanto, como si hubiese 7 porciones)

La primera, ocupará 2/7

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/> 🡪 Ocupará 2/7avos

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="3\*"/> 🡪 Ocupará 3/8avos

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

Partial clases:

/// <summary>

/// Lógica de interacción para MainWindow.xaml

/// Partial class, junto la autogenerada para MainWindow.xaml-->

/// La otra partial class la podríamos ver mostrando todos los archivos--> Obj-Debug-MainWindows.g.i.cs

/// Gracias a esta clase autogenerada podemos acceder a los elementos XAML por su atributo Name

/// </summary>

public partial class MainWindow : Window

Styling way 1 (See background and foreground)

<Button x:Name="pointButton"

Click="pointButton\_Click"

Background="#444444"

Foreground="White"

Content="."

Margin="5"

Grid.Row="5"

Grid.Column="2"/>

Styling way 2---App.xaml File para que los mismos static resources estén disponibles para toda la aplicación:

**App.xaml**

<Application.Resources>

<SolidColorBrush x:Key="numbersColor" Color="#666666"/>

<SolidColorBrush x:Key="operatorsColor" Color="Green"/>

<SolidColorBrush x:Key="foregroundColor" Color="White"/>

</Application.Resources>

<Button x:Name="divisionButton"

Click="OperationButton\_Click"

Background="{StaticResource operatorsColor}"

Foreground="White"

Content="/"

Margin="5"

Grid.Row="1"

Grid.Column="3"/>

Styling way 3 🡪 Implicit style 🡪 No quiere decir que en algún botón en particular tu no puedas sobreescribir la property y ponerle un foreground de otro color…

<Application.Resources>

<Style TargetType="Button">

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="25"/>

</Style>

**Stylin way 4 🡪 Explicit styles ---**

<Application.Resources>

<SolidColorBrush x:Key="numbersColor" Color="#444444"/>

<SolidColorBrush x:Key="operatorsColor" Color="Orange"/>

<SolidColorBrush x:Key="foregroundColor" Color="White"/>

<Style TargetType="Button">

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="25"/>

<Setter Property="Margin" Value="5"/>

</Style>

<Style TargetType="Button" x:Key="numberButtonsStyle">

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="Margin" Value="5"/>

<Setter Property="Background" Value="{StaticResource numbersColor}"/>

</Style>

<Style TargetType="Button" x:Key="operatorButtonsStyle" BasedOn="{StaticResource numberButtonsStyle}">

<Setter Property="Background" Value="{StaticResource operatorsColor}"/>

</Style>

<Style TargetType="Button" x:Key="additionalButtonsStyle" BasedOn="{StaticResource numberButtonsStyle}">

<Setter Property="Background" Value="LightGray"/>

<Setter Property="Foreground" Value="Black"/>

</Style>

<Style TargetType="Label">

<Setter Property="FontSize" Value="70"/>

</Style>

</Application.Resources>