Jacek Mosakowski – Object Oriented Programming (C#)
Lab 7 – WPF

"A picture is worth a thousand words"

To create a new project in Visual Studio, click **Create New Project** → **WPF APP .NET Core** → enter name for the program (e.g. **"Ex_01_01"**) → choose destination folder → enter whole solution name (e.g. **"Lab_01"**).

To run a program, click **F5** (debug mode) or **Ctrl+F5** (without debugging).

Exercise 1 – WPF. Desktop applications. [S]

Let's write a Windows Presentation Foundation (WPF) program. WPF is the .NET tool for Graphical User Interface (GUI). Our program will list all MP3 files in a given directory (optionally including subdirectories). We'll add to this program:

- a few **Menu** controls, e.g. to load files, to close the app, or to print author info,
- a **TextBox** to enter the target directory location, together with a **TextBlock** description of this box,
- a **CheckBox** for an option to also include subdirectories,
- a **Button** to run the application.

Links:

- https://www.wpf-tutorial.com/about-wpf/what-is-wpf/
- https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022

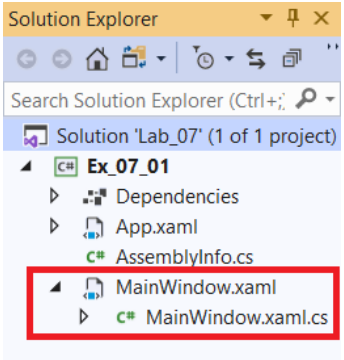
Preparations:

After creating the project, a new Visual Studio window is opened with two tabs:

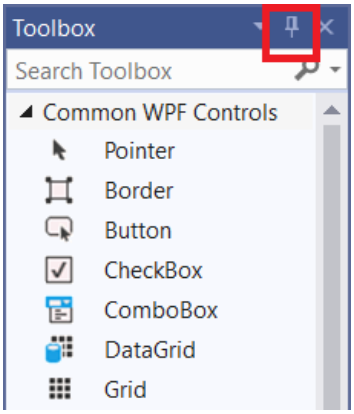
- **MainWindow.xaml** – controls the graphical part, we use XAML (Extensible Application Markup Language) here,
- **MainWindow.xaml.cs** – controls the core mechanics of the application, using C#.

Both parts are interconnected and will communicate with each other. In case any of the two tabs close, you can open them again from the Solution Explorer on the right:

- the **.xaml** file is in the main project directory,
- the **.xaml.cs** file is in the subdirectory of the .xaml file.



If the app window in the .xaml tab is too small, enlarge it with the mouse wheel. A useful tool to have is the **Toolbox**, as it contains shortcuts for buttons, controls, etc. If the Toolbox hasn't opened itself (on the left), you can do it manually by pressing Ctrl + Alt + X or by choosing View → Toolbox. It's useful to pin the Toolbox so it doesn't auto close. To do it, use the pin icon, next to the close button.



We'll start with the graphical (XAML) part, and then move to the program mechanics (C#). Below is the full listing for both parts.

WPF is the latest .NET approach to GUI. Before, Windows Forms was used instead. In general, it's currently recommended to use WPF however, as it's still under development, you may sometimes need to import the Windows Forms functionalities in a WPF project. To do so:

1. Solution Explorer, right-click the project name → Unload Project
2. Solution Explorer, right-click the project name (unloaded) → Edit [name].csproj
3. in this file, below the <UseWPF>true</UseWPF> entry, add another line:
   <UseWindowsForms>true</UseWindowsForms>
4. save and close the file
5. Solution Explorer, right-click the project name (unloaded) → Reload Project
6. add this line on the top part of the C# file:
   using System.Windows.Forms;
7. (optionally) you can modify the line above to create a "WinForms" alias:
   using WinForms = System.Windows.Forms;

Links:

- https://stackoverflow.com/questions/5129090/how-to-edit-csproj-file/5129214#5129214

Notes on xaml:

Similarly to html, if you create an **<element>**, you also need to close it using **</element>**. If it can be closed in the same line it was created, you can just add "/" at the end instead: **<element … />**.

xmlns and **xmlns:[name]** define namespaces – shortcuts to use in other code parts. Link: https://carldesouza.com/understanding-wpf-namespaces/

mc:Ignorable namespace provides definitions that are ignored by the xaml processor.

Link: https://stackoverflow.com/a/15749623

DockPanel allows us to dock elements in all directions - top, bottom, left, right. Useful to divide the window into areas. By default, the last element in the DockPanel automatically fills all the rest of the space – you can change it by adding LastChildFill="False".
Link: https://wpf-tutorial.com/panels/dockpanel/

Grid divides an area into columns and rows.
Link: https://wpf-tutorial.com/panels/grid/

Margin property takes four parameters: Left, Top, Right, Bottom to align a given element. Typically, only two values should be positive (depending on the choice of the HorizontalAlignment and VerticalAlignment), while the other two should be left zero.
Link: https://www.c-sharpcorner.com/Resources/807/set-margin-of-controls-in-wpf.aspx

---

Notes on C#:

WPF methods contain two arguments. The first, **object sender**, identifies which xaml element called the given method (e.g. a button, a menu item, etc.). The second, **RoutedEventArgs e**, contains data associated with the event. In both cases, the element which raised the event is responsible for creating and populating these arguments.
Link: https://tinyurl.com/wpfevents

InitializeComponent extracts the compiled XAML and uses it to build user interface.

“@” denotes a verbatim string, so a string which should be read exactly as it is, ignoring special characters. For example, @"C:\Windows\" is the same as "C:\\Windows\\".

Solution – file MainWindow.xaml:

```
<Window x:Class="Ex_07_01.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="MP3 Listing" Height="180" Width="400">
    <!-- This is a comment -->
    <DockPanel>
        <Menu DockPanel.Dock="Top">
            <MenuItem Header="_File">
                <MenuItem Header="_LoadPath" Click="MenuFileLoadPath_Click"/>
                <MenuItem Header="_SavePath" Click="MenuFileSavePath_Click"/>
                <Separator/>
                <MenuItem Header="_Exit" Click="MenuFileExit_Click"/>
            </MenuItem>
            <MenuItem Header="_Info">
                <MenuItem Header="_About" Click="MenuInfoAbout_Click"/>
            </MenuItem>
        </Menu>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
```

```
            <RowDefinition Height="2*"/>
        </Grid.RowDefinitions>
        <TextBlock Foreground="#0000FF" HorizontalAlignment="Left" Margin="5,10,0,0"
                   Text="Enter directory:" TextWrapping="Wrap"
                   VerticalAlignment="Top" Width="90"/>
        <TextBox Background="HotPink" HorizontalAlignment="Left" Margin="100,10,0,0"
                 Name="TextPath" Text="C:\Windows\" TextWrapping="Wrap"
                 VerticalAlignment="Top" Width="250"/>
        <CheckBox Content="Include subfolders" Grid.Row="1"
                  HorizontalAlignment="Left" IsChecked="True" Margin="100,10,0,0"
                  Name="CheckBoxSubfolders" VerticalAlignment="Top"/>
        <Button Background="LightGreen" Click="ListButton_Click"
                Content="Create my mp3 list!" Grid.Row="2"
                HorizontalAlignment="Center" VerticalAlignment="Center"/>
    </Grid>
  </DockPanel>
</Window>
```

Solution – file MainWindow.xaml.cs:

```
using System;               // needed for AppContext
using System.IO;            // needed for File
using System.Windows;       // needed for RoutedEventArgs, Window
using Microsoft.Win32;      // needed for OpenFileDialog

namespace Ex_07_01
{
    // Interaction logic for MainWindow.xaml
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

        /*************************************************************************/

        private void MenuFileLoadPath_Click(object sender, RoutedEventArgs e)
        {
            // TODO
        }

        /*************************************************************************/

        private void MenuFileSavePath_Click(object sender, RoutedEventArgs e)
        {
            // TODO
        }

        /*************************************************************************/

        private void MenuFileExit_Click(object sender, RoutedEventArgs e)
        {
            // TODO
        }

        /*************************************************************************/

        private void MenuInfoAbout_Click(object sender, RoutedEventArgs e)
        {
            // TODO
        }
```

```
        /*********************************************************************/

        private void ListButton_Click(object sender, RoutedEventArgs e)
        {
            // TODO
        }
    }
}
```

GitHub

To **add a project to GitHub** for the first time:

1. Git (from upper menu) → Create Git repository
2. if not logged in, click Log in (middle right) → GitHub → Sign in in the web browser → go back to Visual Studio
3. enter Repository Location, Name, Description (optional), Private Repository (optional) → Create and push

To **load a project from GitHub**:

1. Either:
   - **if you just opened Visual Studio:** Clone repository (upper right)
   - **if you already have the main Visual Studio editor window opened:** Git (from the upper menu) → Clone repository
2. click GitHub (bottom left)
3. if not logged in, click Log in (upper right) → GitHub → Sign in in the web browser → go back to Visual Studio
4. choose a project → double-click it or click Clone (if you can't Clone, click Open)
5. Solution Explorer → double click the .sln file [ProjectName.sln] from the list → double click Program.cs

To **send changes to GitHub**:

1. click Git Changes (on the right, next to Solution Explorer)
2. **important:** if there is a yellow box saying “Configure user name and e-mail”, click “Configure” and enter them
3. enter a commit description
4. dropdown menu (under the description) → Commit All and Sync

---