

# Windows Presentation Foundation (WPF)

Лекція 06

# План

---

1. Поняття системи WPF
2. Архітектура WPF
3. Створення проекту з графічним інтерфейсом
4. Елементи керування WPF
5. Режим дизайну
6. Властивості
7. Приклад

# Поняття системи WPF

---

**WPF (Windows Presentation Foundation)** - це потужний інструментарій для розробки візуально привабливих та функціональних застосунків для Windows. Він надає можливість створювати візуально привабливі та функціональні інтерфейси користувача з 3D-анімацією, різноманітними кольорами та ефектами, при цьому зменшуючи складність коду.

## Переваги:

- WPF використовує векторну графіку, що дозволяє створювати інтерфейси, які масштабуються без втрати якості на різних екранах.
- WPF використовує апаратне прискорення сучасних графічних карт.
- WPF використовує мову розмітки XAML для опису інтерфейсу користувача.
- WPF надає широкі можливості для налаштування та розширення функціональності

# Поняття системи WPF

---

WPF є невід'ємною частиною .NET Framework та служить для створення сучасних графічних інтерфейсів. На відміну від традиційних WinForms-додатків, де за відображення елементів інтерфейсу та графіки відповідали компоненти операційної системи (User32, GDI), WPF використовує DirectX для рендерингу.

Ключова відмінність WPF полягає в тому, що значна частина роботи з візуалізації, від простих кнопок до складних 3D-моделей, перекладається на графічний процесор, що забезпечує апаратне прискорення графіки та підвищує продуктивність.

# Поняття системи WPF

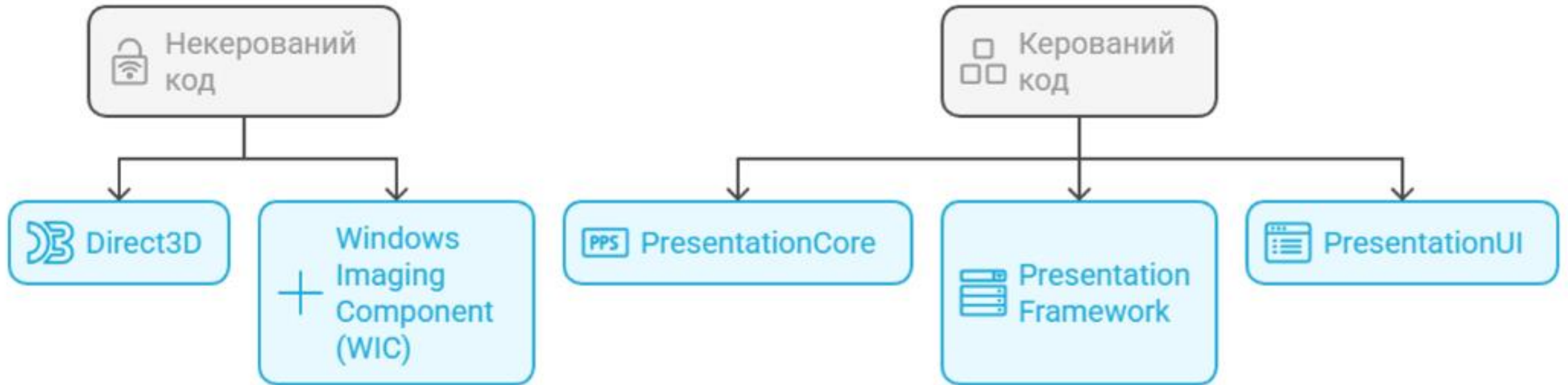
---

Розробка інтерфейсу в WPF базується на декларативній мові розмітки **XAML (Extensible Application Markup Language)**, використовується для опису елементів інтерфейсу. XAML базується на **XML(Extensible Markup Language)** та дозволяє створювати складні UI-структури декларативно.

XML використовується для зберігання та передачі даних. XAML є одним із варіантів використання XML, адаптованим спеціально для опису інтерфейсу WPF.

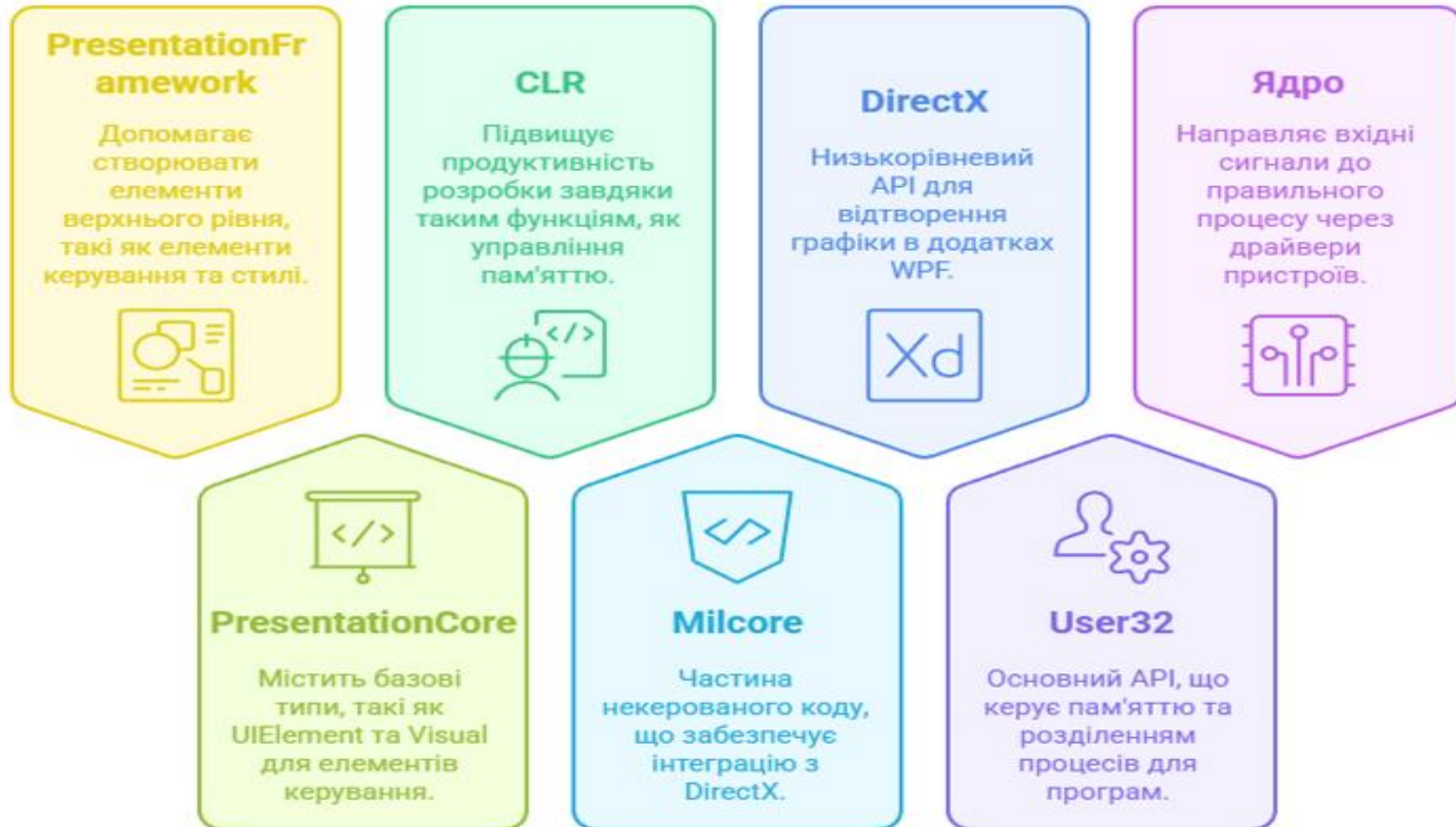
# Архітектура WPF

---

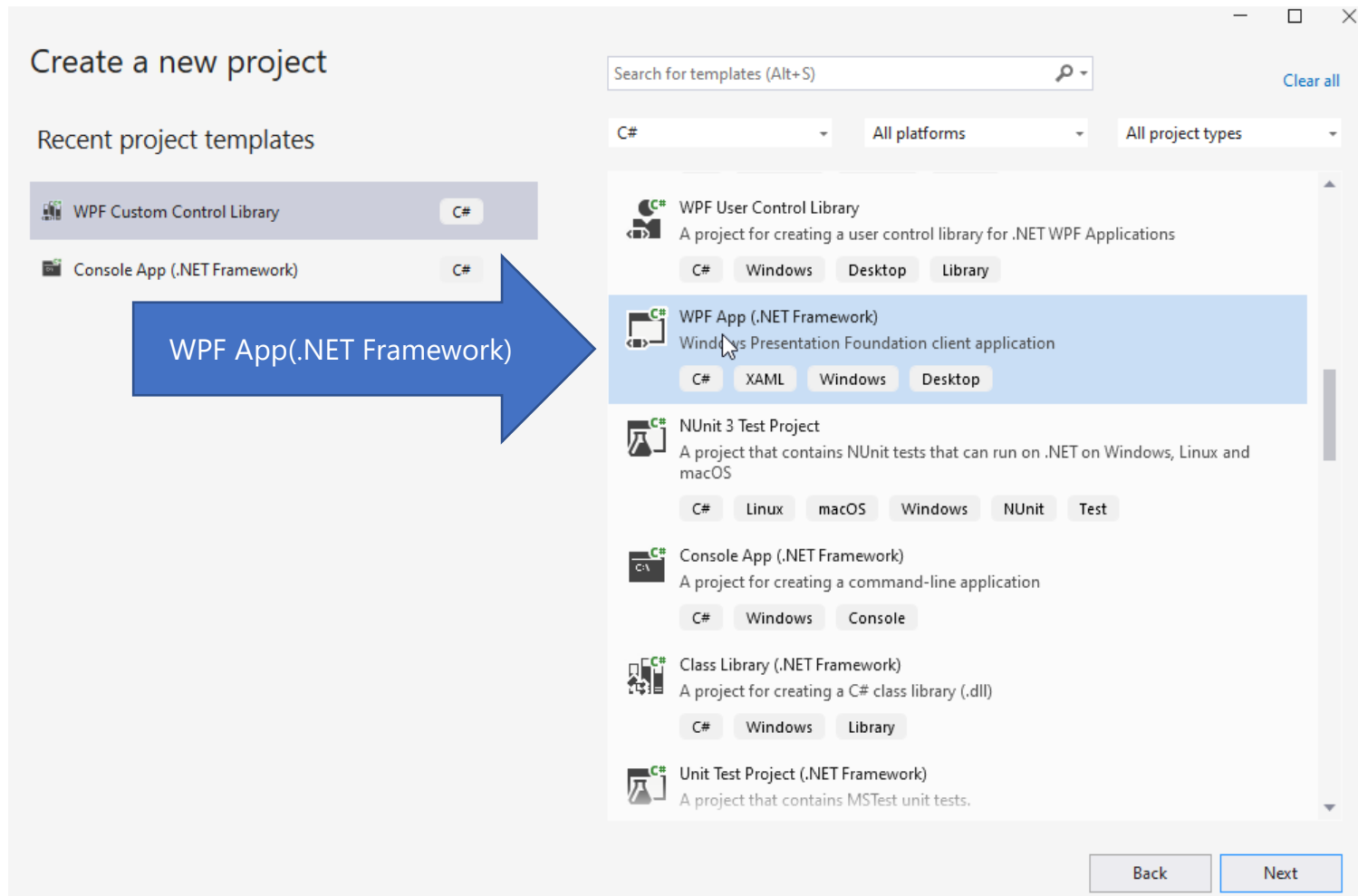


# Компоненти архітектури WPF

---



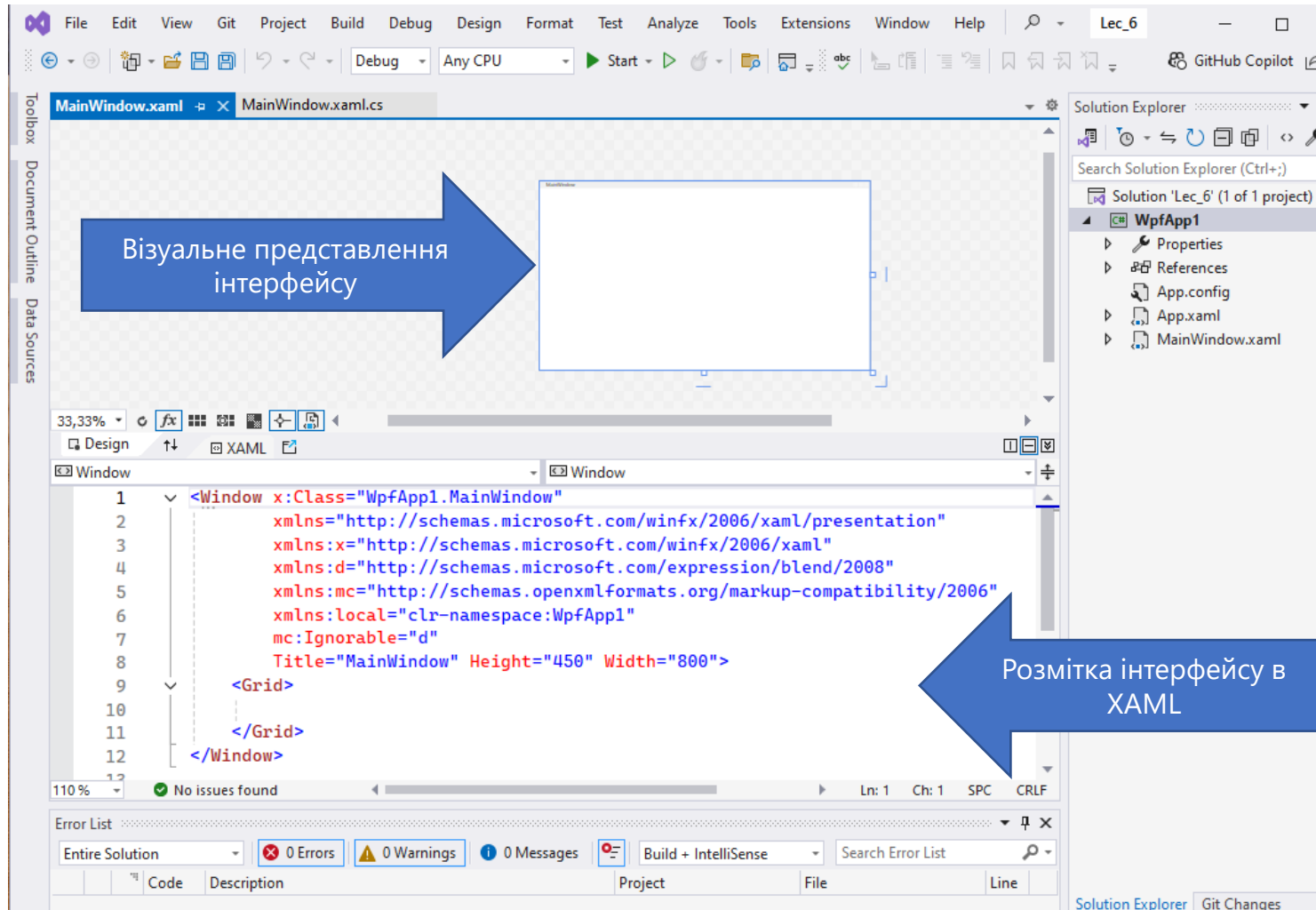
# Створення проекту з графічним інтерфейсом



Першим кроком є створення нового проекту. Для цього потрібно обрати **WPFApp(.NET Framework)**



# Створення проекту з графічним інтерфейсом



Visual Studio за замовчуванням створює два файли

Файл XAML  
(MainWindow.xaml)  
Файл CS (MainWindow.xaml.cs)

У вікнах XAML такі теги записуються за замовчуванням.

Сітка є першим елементом за замовчуванням.

# Файл MainWindow.xaml. Простір імен XAML

---

XAML використовує простий синтаксис для визначення елементів інтерфейсу користувача та їхніх властивостей.

Кожен елемент, схожий на елементи XML, укладений у початковий і кінцевий теги (наприклад, `<Window>...</Window>`) або може бути представлений у самозакриваючій формі (наприклад, `<Window />`).

Елемент XAML відповідає певному класу C#. Наприклад, елемент XAML `<Button>` асоціюється з `System.Windows.Controls.Button` класом у C#.

Атрибути елемента XAML відображають властивості відповідного класу C#.

# Файл MainWindow.xaml. Простір імен XAML

---

```
<Window x:Class="WpfApp1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

Базовий простір імен, який охоплює всі класи WPF, включаючи елементи управління, які застосовуються при побудові інтерфейсу користувача. Так як цей простір імен оголошено без префікса, то він поширюється на весь XAML-документ.

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

- це простір імен XAML. Він включає різні властивості утиліт XAML, які дозволяють впливати на те, як XAML-документ слід інтерпретувати.

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
xmlns:local="clr-namespace:WpfApp1"
```

```
mc:Ignorable="d"
```

```
Title="MainWindow" Height="450" Width="800">
```

```
<Grid>
```

```
...
```

```
</Grid>
```

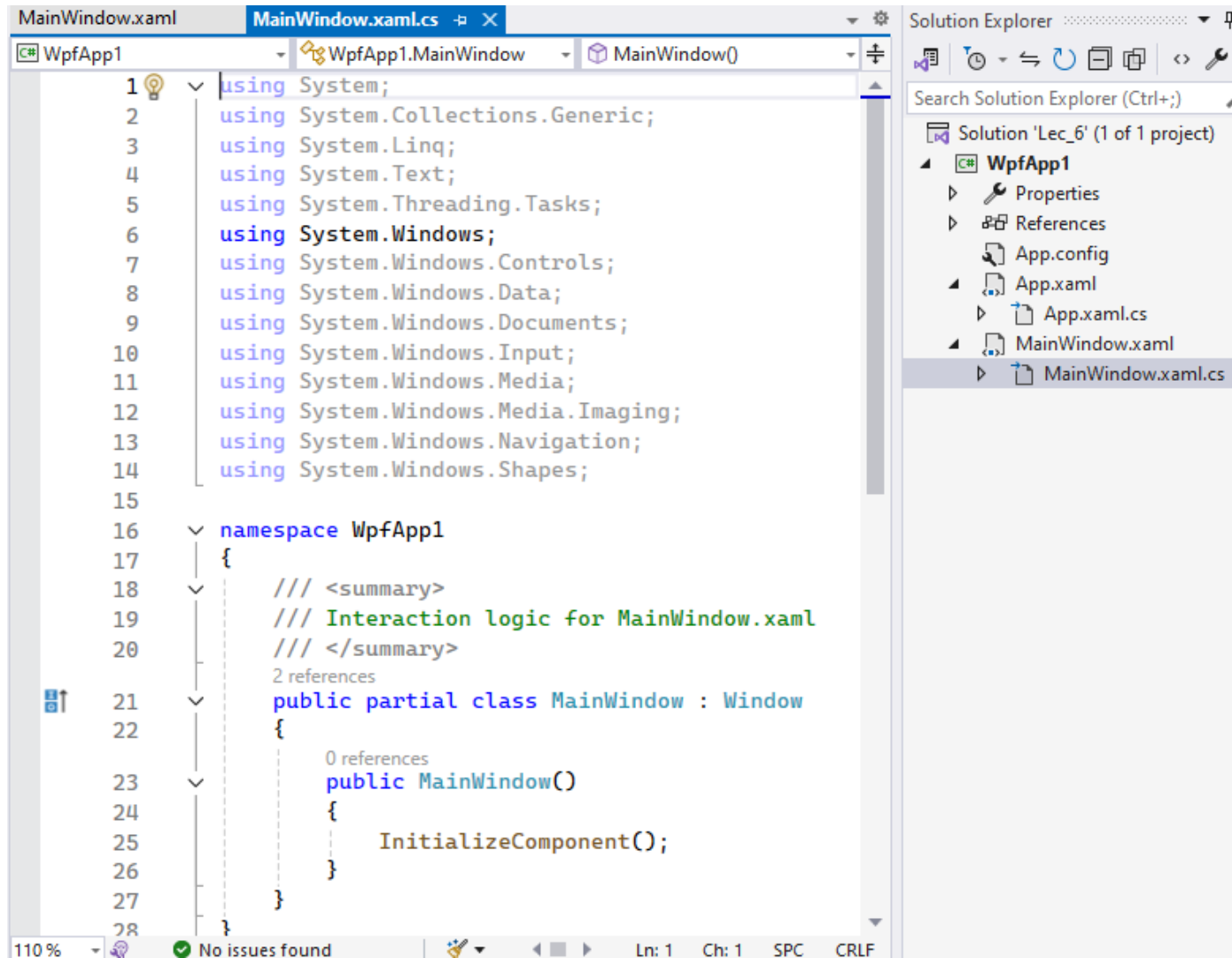
```
</Window>
```

Префікс `x`, що використовується, у визначенні `xmlns:x` означає, що ті властивості елементів, які укладені в цьому просторі імен, будуть використовуватися з префіксом `x` - `x.Name` або `x.Key`.

`x:Class="XamlApp.MainWindow"` - тут створюється новий клас `MainWindow` і відповідний файл коду, куди буде прописуватися логіка для даного вікна програми.

!!! Ці простори імен не еквівалентні тим просторам імен, які підключаються за допомогою директиви `using c#`.

# Створення проекту з графічним інтерфейсом

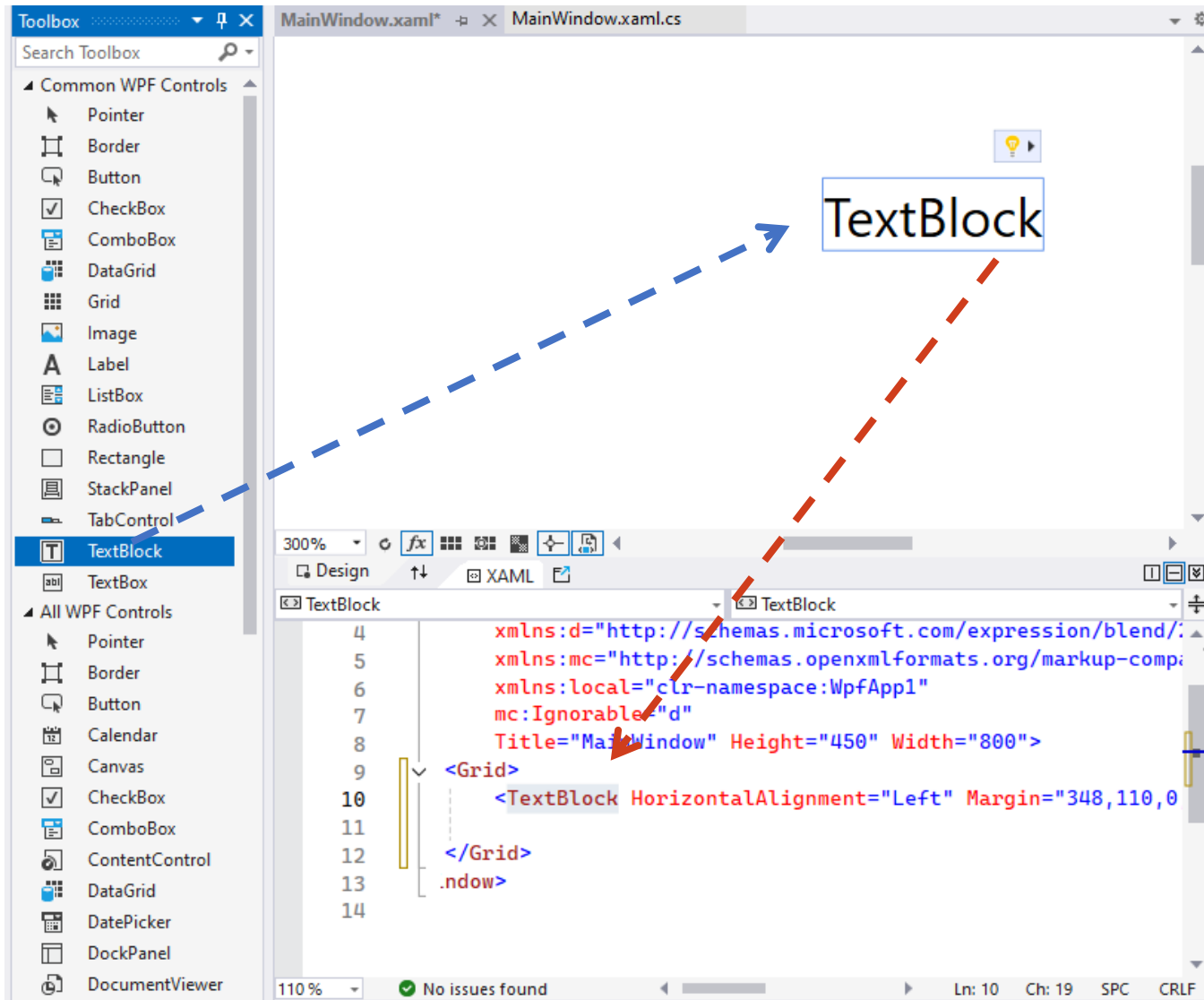


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Windows;
7 using System.Windows.Controls;
8 using System.Windows.Data;
9 using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15
16 namespace WpfApp1
17 {
18     /// <summary>
19     /// Interaction logic for MainWindow.xaml
20     /// </summary>
21     2 references
22     public partial class MainWindow : Window
23     {
24         0 references
25         public MainWindow()
26         {
27             InitializeComponent();
28         }
29     }
30 }
```

**MainWindow.xaml** основне вікно дизайну . Для опису дизайну використовуємо xaml

**MainWindow.xaml.cs** містить відповідний код, який описує функціонал.

# Створення проекту з графічним інтерфейсом



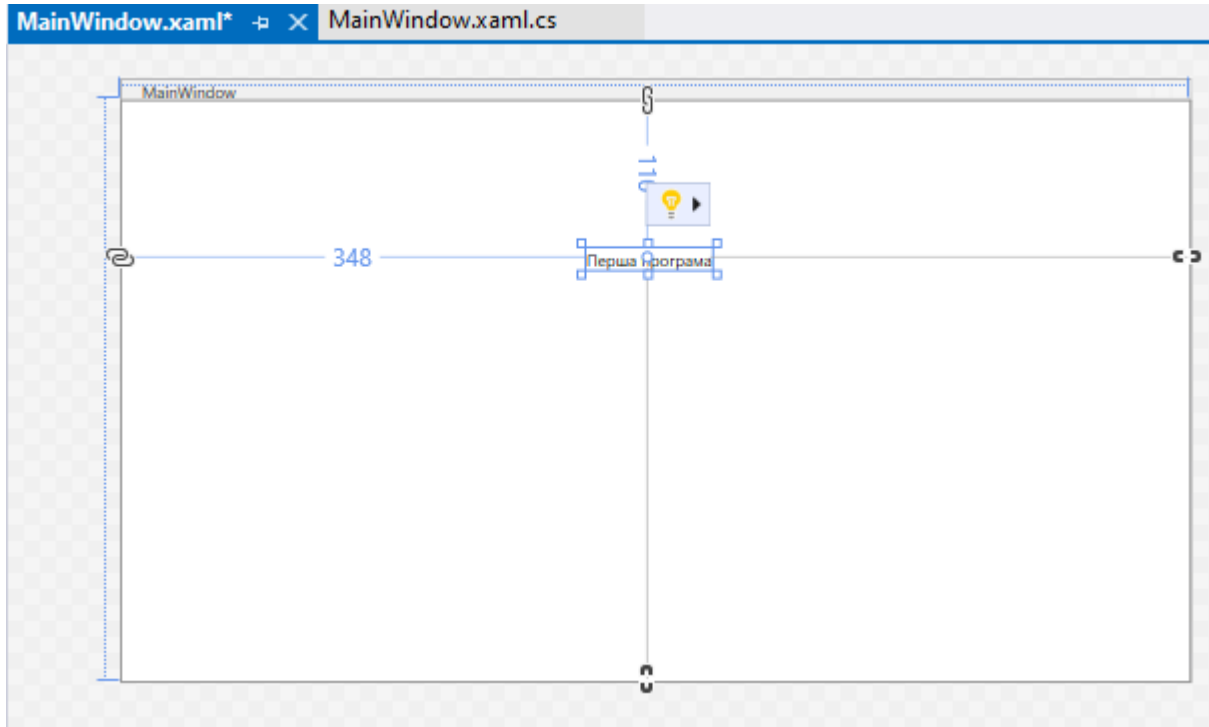
У панелі інструментів обрати **TextBlock**.

Перетягнути обраний елемент до вікна дизайну.

**TextBlock** з'явиться у вікні дизайну.

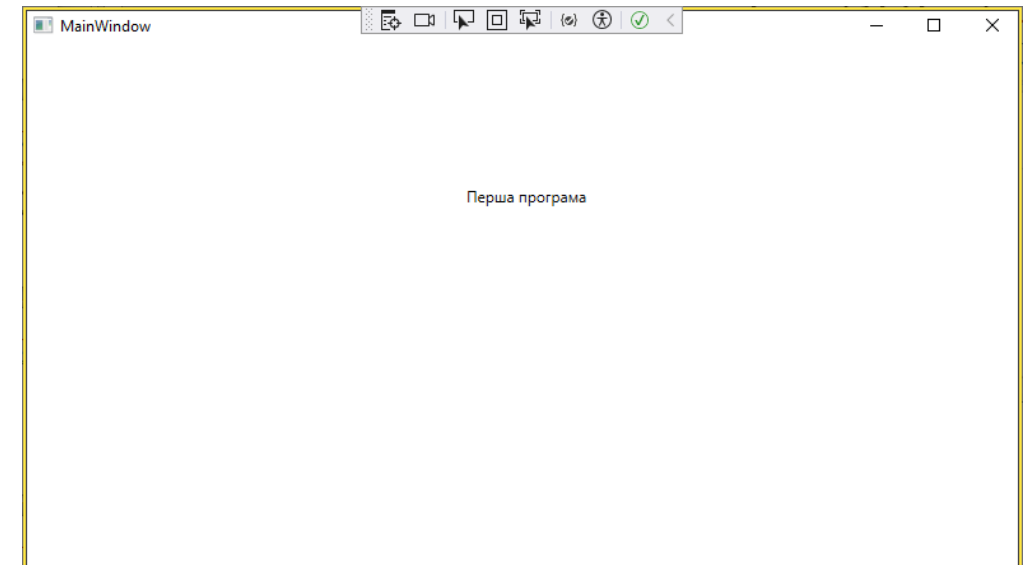
В кодї XAML можна побачити доданий **TextBlock**.

# Створення проекту з графічним інтерфейсом



Змініть текст на «*Перша програма*».

Натисніть кнопку Пуск. З'явиться вікно, де знаходиться надпис: «*Перша програма*»



# Створення кнопки

---

Створення кнопки в XAML:

- Додайте елемент `<Button>` в потрібне місце.
- Задайте атрибути кнопки:
  - ✓ `Content` - текст, який відображається на кнопці.
  - ✓ Розміри кнопки.
  - ✓ `Click` - обробник події натискання на кнопку.

```
<Button Content="Клацніть по мені" Margin="208,165,208,217" Click="Button_Click"/>
```

Створення обробника події `Click` в C#:

- У файлі C# створіть метод-обробник події `Click`. Цей метод буде викликатися при натисканні на кнопку.
- У середині методу напишіть код, який буде виконуватися при натисканні, наприклад, виведення повідомлення.

```
MessageBox.Show("Привіт, мій перший Windows Presentation Foundation!");
```

# Властивості

Режим дизайну в WPF дозволяє візуально створювати та редагувати інтерфейс користувача за допомогою WYSIWYG-редактора (What You See Is What You Get).

Кнопка має багато властивостей, які дозволяють налаштувати її зовнішній вигляд та поведінку:

**Content**- текст або інший вміст кнопки.

**Background**- колір фону кнопки.

**Foreground**- колір тексту кнопки.

**FontFamily**- шрифт тексту кнопки.

**FontSize**- розмір тексту кнопки.

**Width**- Ширина кнопки.

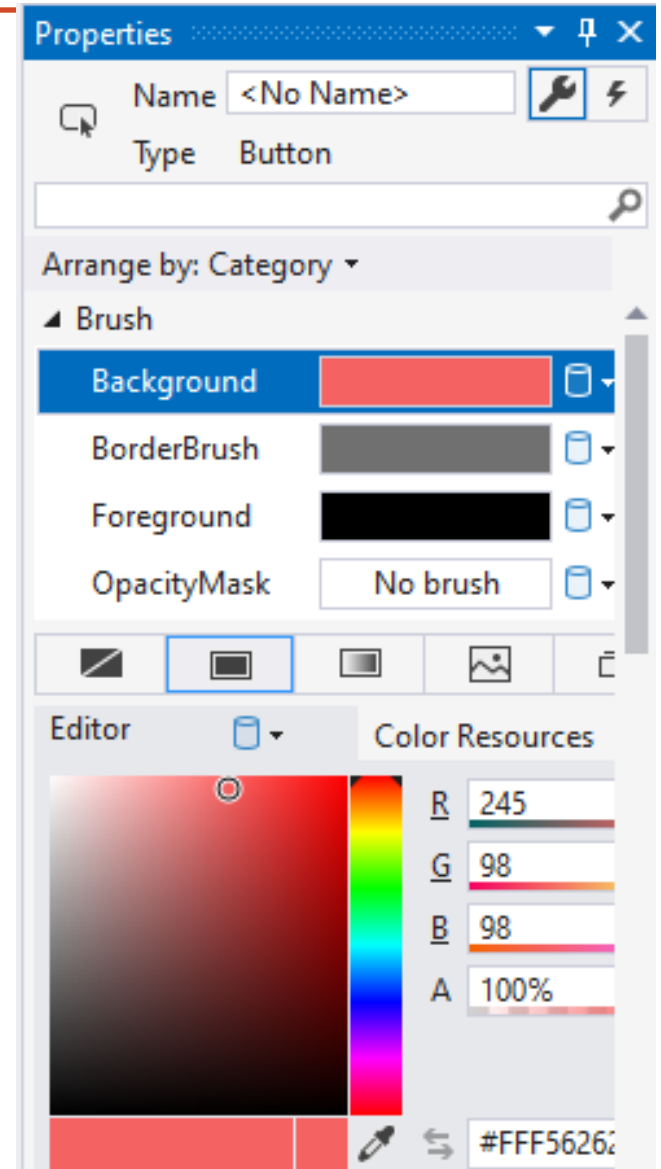
**Height**- висота кнопки.

**Margin**- відстань від кнопки до інших елементів.

**Padding**- внутрішній відступ кнопки.

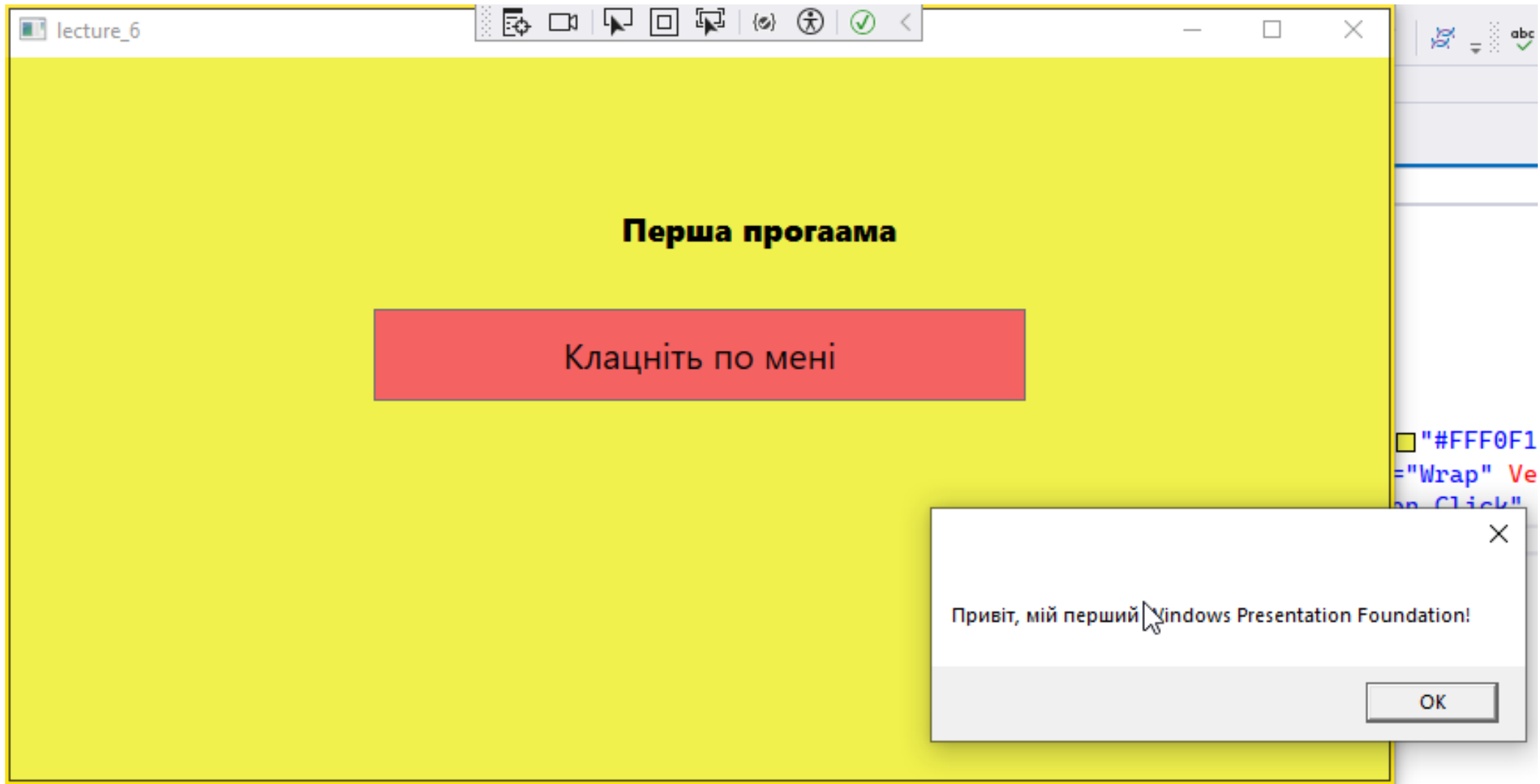
**IsEnabled** - визначає, чи доступна кнопка для натискання.

**Visibility** - визначає, чи відображається кнопка.





# Створення кнопки



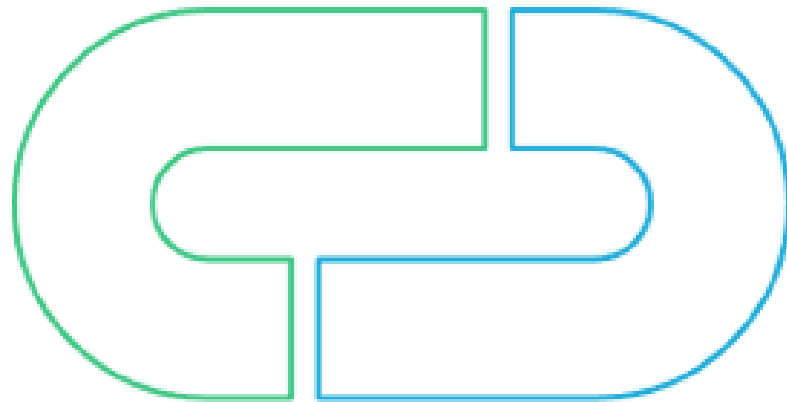
# Режим дизайну

---

Режим дизайну в WPF надає візуальний інтерфейс для створення інтерфейсів користувача. Ключові аспекти:

## Visual Studio Designer

Інтегрований інструмент для редагування WPF



## XAML

Використовується для опису інтерфейсу користувача у WPF

# Дизайн користувальницького інтерфейсу

---



## Панелі макетів

Організація елементів керування на екрані.  
Grid, StackPanel, Canvas

Створення стилів для уніфікованого вигляду.

## Стилі та шаблони



Прив'язка властивостей елементів керування до ViewModel для оновлень, для динамічного оновлення інтерфейсу



## Прив'язка даних

# Режим дизайну

---

Основні теги XAML:

<Window> - кореневий елемент, що представляє вікно програми;

<Grid> - контейнер для розміщення елементів у вигляді сітки;

<StackPanel> - контейнер для розміщення елементів один за одним (вертикально або горизонтально);

<DockPanel> - контейнер для розміщення елементів з прив'язкою до країв контейнера;

<Canvas> - контейнер для розміщення елементів з абсолютним позиціонуванням;

<Button> - кнопка;

<Label> - текстовий напис;

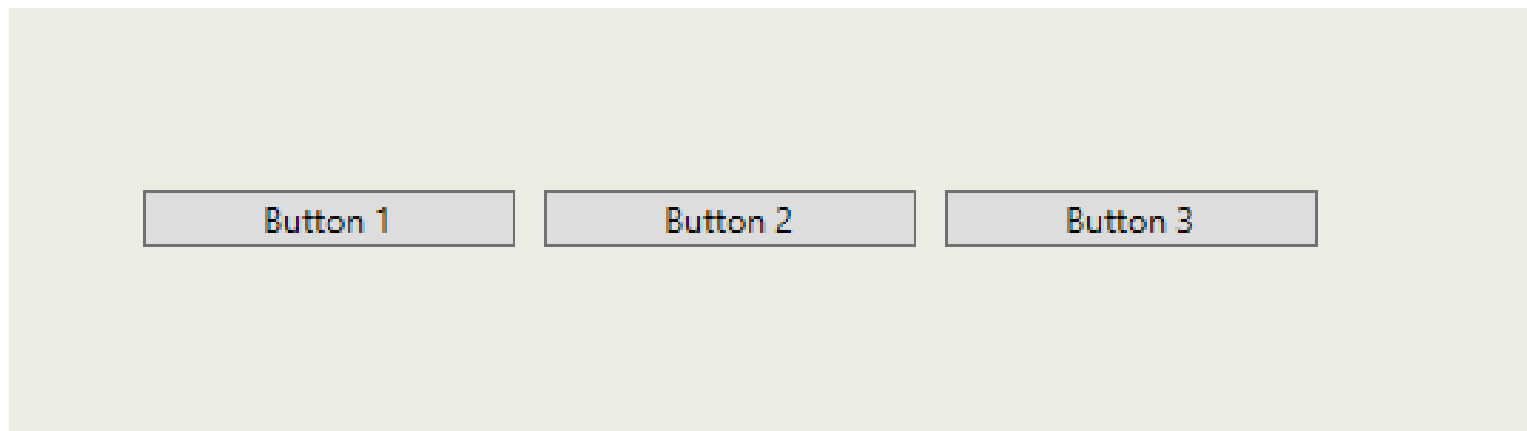
<TextBox> - поле для введення тексту;

<Image> - зображення;

<ListBox> - список елементів;

...

```
<Grid x:Name="lecture_6" Height="434" VerticalAlignment="Center"
Background="#FFFEDEE3">
  <StackPanel Orientation="Horizontal" HorizontalAlignment="Center"
VerticalAlignment="Center">
    <Button Content="Button 1" Margin="5" Width="131"/>
    <Button Content="Button 2" Margin="5" Width="131"/>
    <Button Content="Button 3" Margin="5" Width="131"/>
  </StackPanel>
</Grid>
```



# Режим дизайну у С#

---

```
public MainWindow(){
    InitializeComponent(); CreateButtons();
}
private void CreateButtons(){
    StackPanel stackPanel = new StackPanel();
    stackPanel.Orientation = Orientation.Vertical;
    stackPanel.HorizontalAlignment = HorizontalAlignment.Center;
    stackPanel.VerticalAlignment = VerticalAlignment.Center;

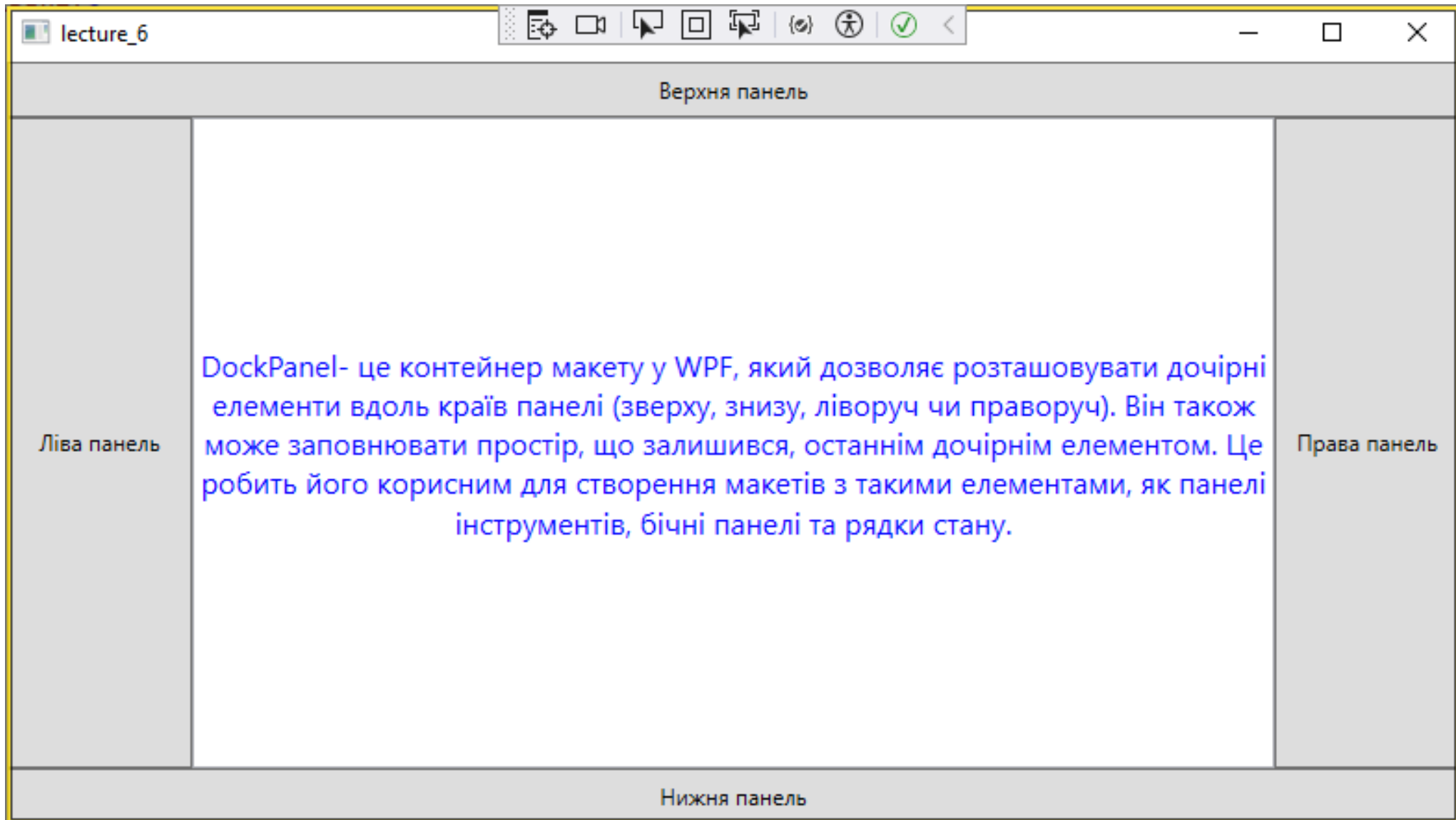
    Button button1 = new Button();
    button1.Content = "Button 1";
    button1.Margin = new Thickness(5);
    Button button2 = new Button();
    button2.Content = "Button 2";
    button2.Margin = new Thickness(5);

    stackPanel.Children.Add(button1);
    stackPanel.Children.Add(button2);
    lecture_6.Children.Add(stackPanel);
}
```



```
<DockPanel>
    <Button DockPanel.Dock="Top" Height="30" Click="Button_Click">Верхня
панель</Button>
    <Button DockPanel.Dock="Bottom" Height="30" Click="Button_Click_3">Нижня
панель</Button>
    <Button DockPanel.Dock="Left" Width="100" Click="Button_Click_1">Ліва
панель</Button>
    <Button DockPanel.Dock="Right" Width="100" Click="Button_Click_2">Права
панель</Button>
    <TextBox Text="DockPanel– це контейнер макету у WPF, який дозволяє
розташовувати дочірні елементи вдовж країв панелі. Він також може заповнювати
простір, що залишився, останнім дочірнім елементом. Це робить його корисним для
створення макетів з такими елементами, як панелі інструментів, бічні панелі та рядки
стану."
        TextAlignment="Center"
        VerticalContentAlignment="Center"
        FontSize="16"
        Foreground="Blue"
        TextWrapping="Wrap"/>
</DockPanel>
```

# Результат використання DockPanel





**WrapPanel**— це елемент керування макету у WPF, який має дочірні елементи і розташовує їх у послідовному порядку, зліва направо, з перенесенням на новий рядок, коли простір у поточному рядку закінчується.

Розмір дочірніх елементів **WrapPanel** не змінює. Він просто розташовує їх у доступному просторі.

**WrapPanel** не додає додаткових пробілів між дочірніми елементами. Якщо ви хочете додати пробіли, потрібно встановити поля або відступи для дочірніх елементів.

Це корисно для створення гнучких макетів, які адаптуються до різних розмірів вікна.

# Режим дизайну

---

```
<WrapPanel>
```

```
    <TextBlock Text="..." Margin="5" Padding="5" Background="LightBlue" FontSize="16"  
TextWrapping="Wrap"/>
```

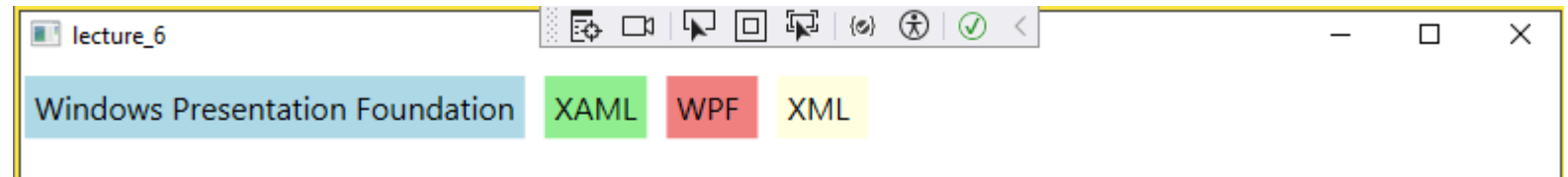
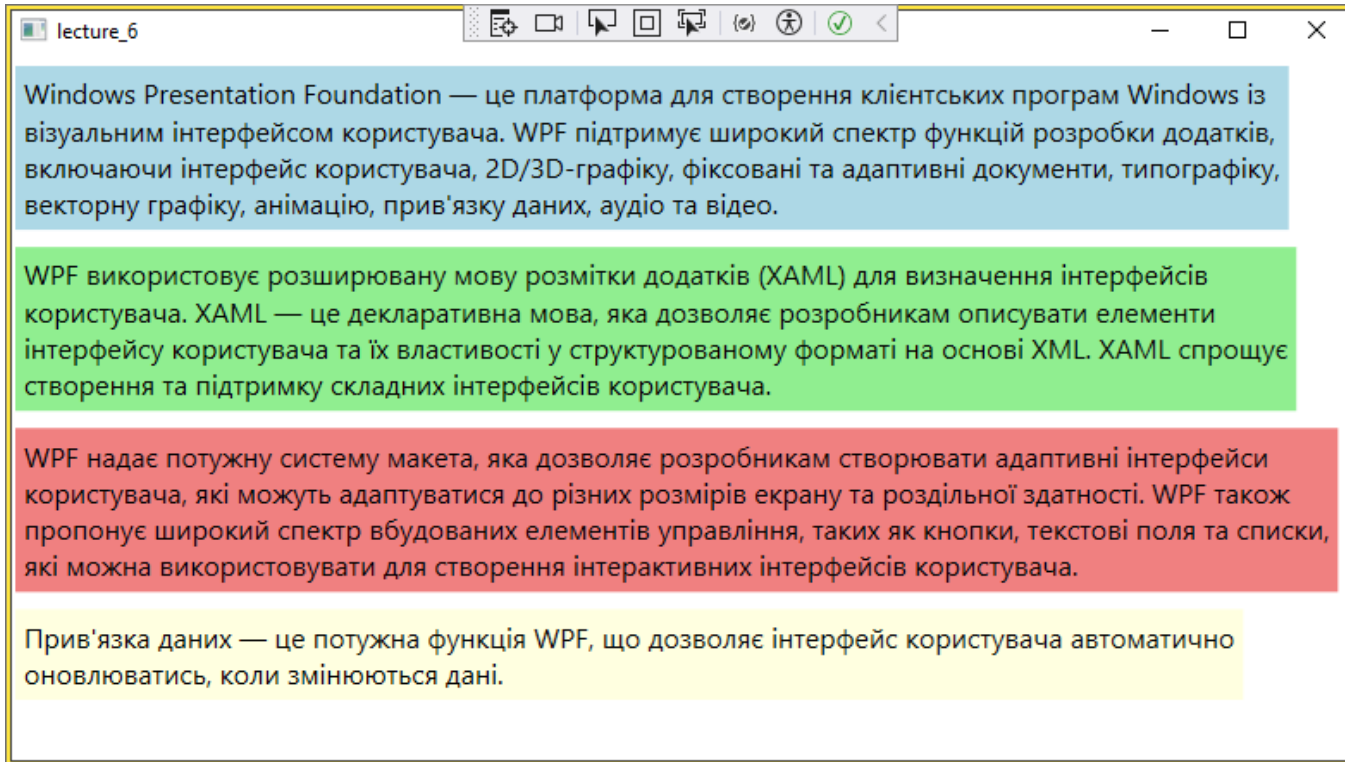
```
    <TextBlock Text="..." Margin="5" Padding="5" Background="LightGreen" FontSize="16"  
TextWrapping="Wrap" />
```

```
    <TextBlock Text="..." Margin="5" Padding="5" Background="LightCoral" FontSize="16"  
TextWrapping="Wrap"/>
```

```
    <TextBlock Text="..." Margin="5" Padding="5" Background="LightYellow"  
FontSize="16" TextWrapping="Wrap"/>
```

```
</WrapPanel>
```

# Результат використання WrapPanel

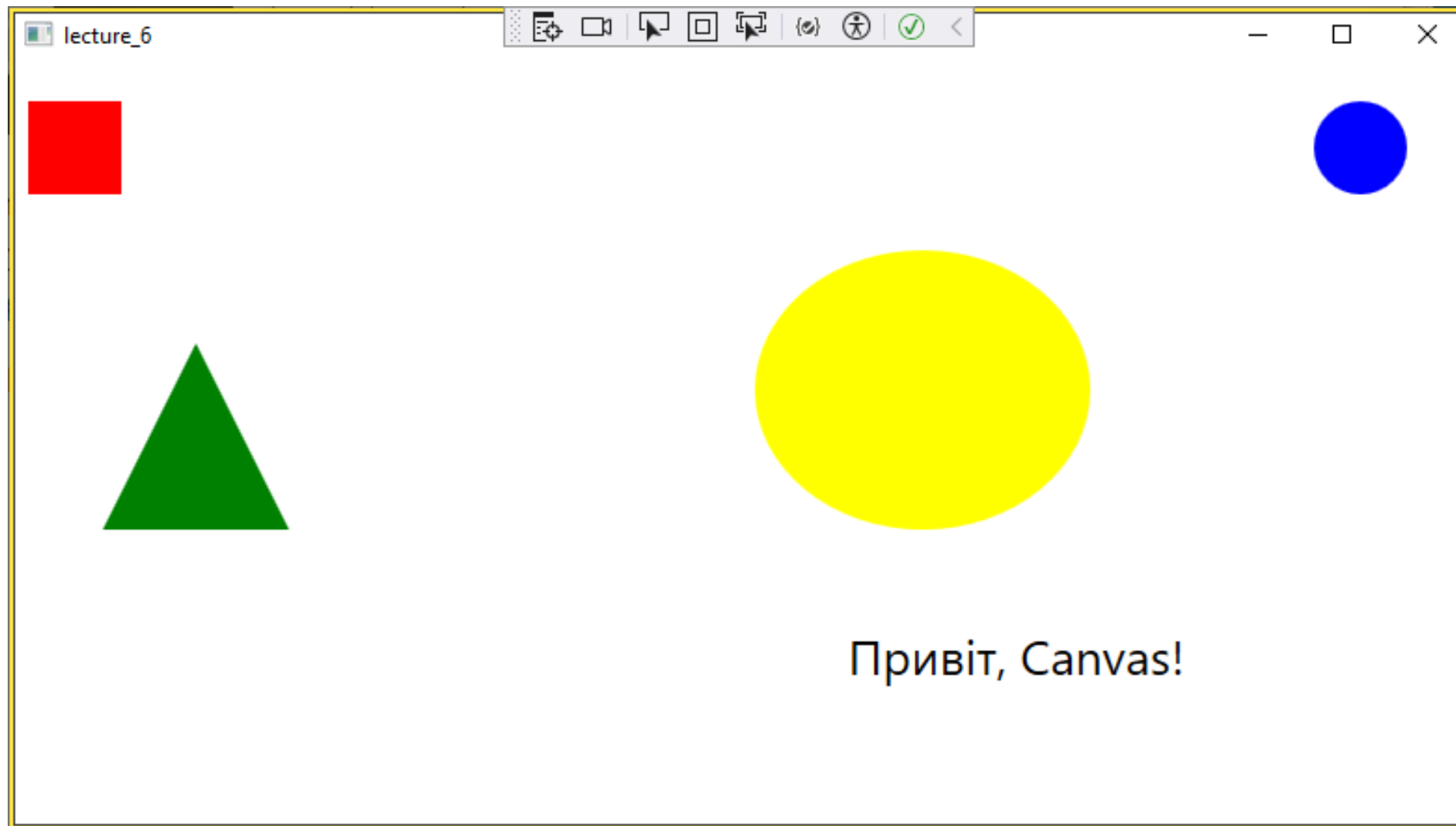


**Canvas**- це панель компоунвання WPF, яка дозволяє розміщувати дочірні елементи з використанням абсолютних координат. Це означає, що ви вказуєте точне положення кожного елемента всередині Canvas від його верхнього лівого кута.

```
<Canvas>
    <Rectangle Fill="Red" Width="50" Height="50" Canvas.Left="10"
Canvas.Top="20" />
    <Ellipse Fill="Blue" Width="50" Height="50" Canvas.Left="700"
Canvas.Top="20" />
    <TextBlock Text="Привіт, Canvas!" Canvas.Left="450" Canvas.Top="300"
FontSize="26" />
    <Polygon Points="100,150 150,250 50,250" Fill="Green" />
    <Ellipse Fill="Yellow" Width="180" Height="150" Canvas.Left="400"
Canvas.Top="100" />
</Canvas>
```

# Результат використання Canvas

---

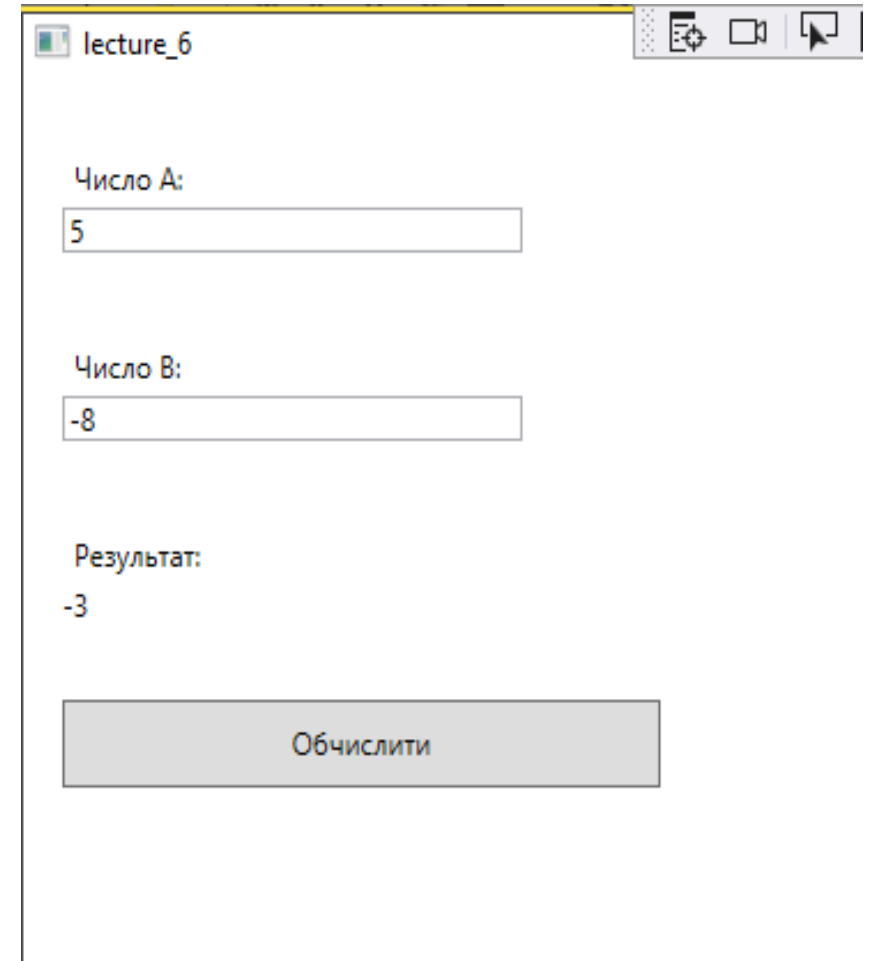


# Приклад

Створити віконний додаток для підрахунку суми двох значень, наприклад  $a + b = ?$

Для цього потрібно:

- Створити вікно з текстом і текстовими полями для введення чисел A та B, кнопку "Обчислити" і поле для відображення результату.
- Використовуємо **Grid** для організації елементів інтерфейсу.
- Призначимо обробник події **Button\_Click** для кнопки.



# Приклад

---

```
<Grid HorizontalAlignment="Center" Width="800">
    <Label Content="Число A:" HorizontalAlignment="Left" Margin="25,25,0,0"
VerticalAlignment="Top"/>
    <TextBox x:Name="A" HorizontalAlignment="Left" Margin="25,50,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>

    <Label Content="Число B:" HorizontalAlignment="Left" Margin="25,100,0,0"
VerticalAlignment="Top"/>
    <TextBox x:Name="B" HorizontalAlignment="Left" Margin="25,125,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>

    <Label Content="Результат:" HorizontalAlignment="Left" Margin="25,175,0,0"
VerticalAlignment="Top"/>
    <TextBlock x:Name="txtResult" HorizontalAlignment="Left" Margin="25,200,0,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="200"/>

    <Button Content="Обчислити" Click="Button_Click" Margin="25,246,0,130"
HorizontalAlignment="Left" Width="260"/>
</Grid>
```

# Windows Presentation Foundation

---

[Easy WPF in C# Windows Presentation Foundation for Beginners](https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-for-beginners/?utm_source=adwords&utm_medium=udemyads&utm_campaign=Search_DSA_Beta_Prof_Ia.EN_cc.ROW-English&campaigntype=Search&portfolio=ROW-English&language=EN&product=Course&test=&audience=DSA&topic=&priority=Beta&utm_content=deal4584&utm_term=._ag_162511579404._ad_696197165418._kw_.de_c._dm_.pl._ti_dsa-1677053911088._li_9194581._pd_.&matchtype=&gad_source=1&gclid=CjwKCAiAqrG9BhAVEiwAaPu5zvdJnWD-tnCGKeN3I-xRgIP4OPVnCC4r3pR8z8DDyCtcypZGzmK-6BoCDX8QAvD_BwE&couponCode=2021PM25) -

[https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-for-beginners/?utm\\_source=adwords&utm\\_medium=udemyads&utm\\_campaign=Search\\_DSA\\_Beta\\_Prof\\_Ia.EN\\_cc.ROW-English&campaigntype=Search&portfolio=ROW-English&language=EN&product=Course&test=&audience=DSA&topic=&priority=Beta&utm\\_content=deal4584&utm\\_term=.\\_ag\\_162511579404.\\_ad\\_696197165418.\\_kw\\_.de\\_c.\\_dm\\_.pl.\\_ti\\_dsa-1677053911088.\\_li\\_9194581.\\_pd\\_.&matchtype=&gad\\_source=1&gclid=CjwKCAiAqrG9BhAVEiwAaPu5zvdJnWD-tnCGKeN3I-xRgIP4OPVnCC4r3pR8z8DDyCtcypZGzmK-6BoCDX8QAvD\\_BwE&couponCode=2021PM25](https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-for-beginners/?utm_source=adwords&utm_medium=udemyads&utm_campaign=Search_DSA_Beta_Prof_Ia.EN_cc.ROW-English&campaigntype=Search&portfolio=ROW-English&language=EN&product=Course&test=&audience=DSA&topic=&priority=Beta&utm_content=deal4584&utm_term=._ag_162511579404._ad_696197165418._kw_.de_c._dm_.pl._ti_dsa-1677053911088._li_9194581._pd_.&matchtype=&gad_source=1&gclid=CjwKCAiAqrG9BhAVEiwAaPu5zvdJnWD-tnCGKeN3I-xRgIP4OPVnCC4r3pR8z8DDyCtcypZGzmK-6BoCDX8QAvD_BwE&couponCode=2021PM25)

[Easy WPF in C# Windows Presentation Foundation for Beginners](https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-for-beginners/?couponCode=2021PM25) -

<https://www.udemy.com/course/easy-wpf-in-c-windows-presentation-foundation-for-beginners/?couponCode=2021PM25>