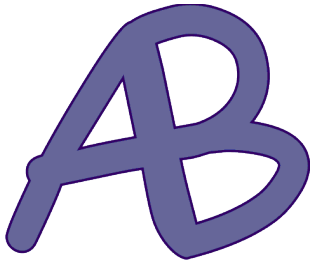
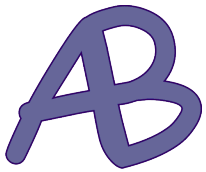
Wizualne systemy programowania

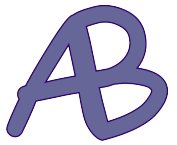


Wykład
WPF



XAML

Struktura dokumentu XAML



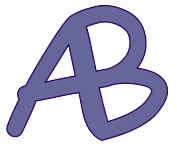
```
1   <Window x:Class="Wykład___wpf_01.MainWindow"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5           xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6           xmlns:local="clr-namespace:Wykład___wpf_01"
7           mc:Ignorable="d"
8           Title="MainWindow" Height="450" Width="800">
9       <Grid>
10
11      </Grid>
12  </Window>
```

Wygląd aplikacji zdefiniowany jest w dokumencie XAML (eXtensible Application Markup Language)

Jest to stworzony przez Microsoft, na podstawie języka XML, język opisu interfejsu aplikacji WPF.

Podobnie jak HTML jest językiem opisu wyglądu strony internetowej, także bazującym na XML.

Struktura dokumentu XAML



```
1   <Window x:Class="Wykład___wpf_01.MainWindow"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5           xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6           xmlns:local="clr-namespace:Wykład___wpf_01"
7           mc:Ignorable="d"
8           Title="MainWindow" Height="450" Width="800">
9       <Grid>
10
11      </Grid>
12  </Window>
```

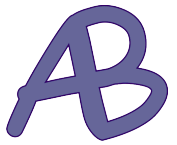
Tytuł i rozmiar okna

Zawartość okna

Klasa Window - okno programu - zapewnia obramowanie, panel tytułowy oraz przyciski maksymalizuj, minimalizuj i zamknij.

Okno WPF jest zdefiniowane w pliku XAML (.xaml), gdzie element <Window> jest rdzeniem (root), oraz z pliku z kodem źródłowym (.cs) (CodeBehind).

Struktura dokumentu XAML



```
1   <Window x:Class="Wykład___wpf_01.MainWindow"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5           xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6           xmlns:local="clr-namespace:Wykład___wpf_01"
7           mc:Ignorable="d"
8           Title="MainWindow" Height="450" Width="800">
9       <Grid>
10
11      </Grid>
12  </Window>
```

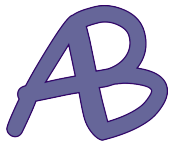
Tytuł i rozmiar okna

Zawartość okna

Klasa Window - okno programu - zapewnia obramowanie, panel tytułowy oraz przyciski maksymalizuj, minimalizuj i zamknij.

Okno WPF jest zdefiniowane w pliku XAML (.xaml), gdzie element <Window> jest rdzeniem (root), oraz z pliku z kodem źródłowym (.cs) (CodeBehind).

Struktura dokumentu XAML



Znacznik XAML musi zostać zamknięty, albo poprzez odrębny, zamykający znacznik albo dodanie ukośnika na jego końcu, tuż przed nawiasem zamykającym:

<Button></Button> Lub <Button />

Wiele kontrolek zezwala na umieszczanie treści pomiędzy znacznikiem otwierającym a zamykającym - zawartością znacznika.

Np. np. w kontrolce przycisku można ustawić napis jaki ma się na nim wyświetlać:

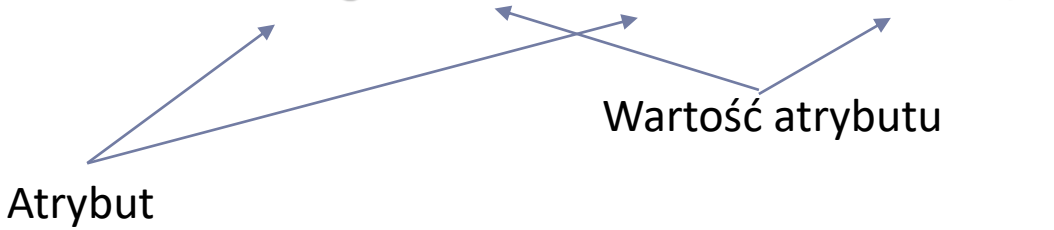
<Button>Napis na przycisku</Button>

Język XAML jest "case-sensitive", czyli odróżnia duże i małe liter w nazwach elementów. Nazwa kontrolki musi odpowiadać typowi należącemu do .NET. To samo dotyczy nazw atrybutów, które odpowiadają właściwościom kontrolki.

Struktura dokumentu XAML

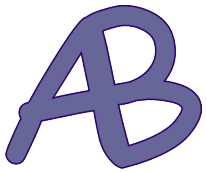
Atrybuty (opisujące właściwości kontrolki) Możemy umieszczać wewnątrz znacznika otwierającego:

```
<Button FontWeight = "Bold" Content = "A button" />
```



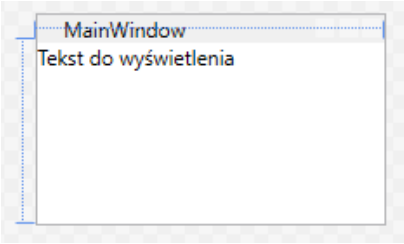
Możemy też wszystkie atrybuty kontrolki zdefiniować w postaci znaczników podrzędnych - notacja *nazwaKontrolki.właściwość*.

```
<Button>
        <Button.FontWeight>Bold</Button.FontWeight>
        <Button.Content>A button</Button.Content>
</Button>
```



Podstawowe kontrolki

Podstawowe kontrolki: TextBlock



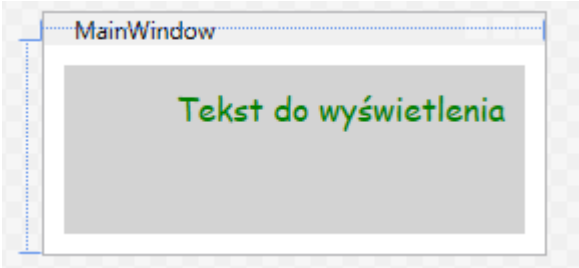
TextBlock pozwala wyświetlać na ekranie tekst.

- Może zawierać jedynie tekst (stringi).
- Stosuje się najczęściej dla wielo-liniowych stringów.

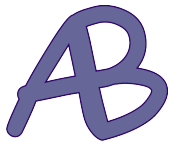
```
<TextBlock>Tekst do wyświetlenia</TextBlock>
```

- Kontrolka posiada atrybuty formatujące tekst:

```
<TextBlock
    FontSize="12pt"
    FontFamily="Comic Sans MS"
    Foreground="Green"
    Background="LightGray"
    Margin="10px"
    Padding="10px"
    TextAlignment="Right">
    Tekst do wyświetlenia
</TextBlock>
```



Podstawowe kontrolki: TextBlock

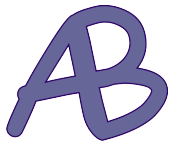


Zawijanie tekstu:

1. **Ręczne** - za pomocą znacznika <LineBreak />
2. **Automatyczne** – za pomocą własności TextWrapping="Wrap" - pozwala na zawijanie wierszy w momencie gdy tekst nie może pomieścić się w danej linii.
3. **Obcinanie tekstu** – za pomocą własności TextTrimming="CharacterEllipsis" W momencie gdy kontrolka nie może pomieścić w sobie więcej tekstu wyświetlany jest wielokropek (...).

   Jest to powszechna metoda zasygnalizowania, że tekstu jest więcej, jednak niewystarczająca ilość miejsca uniemożliwia jego pokazanie.

Podstawowe kontrolki: Label



Kontrolka Label wyświetla tekst (ale w prostszy, wymagający mniej zasobów sposób).

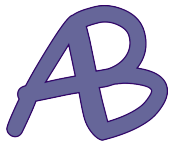
```
<Label Content="Tekst do wyświetlenia." />
```

Jest stosowana najczęściej dla krótkich, jedno-linijkowych tekstów

Inne możliwości kontrolki Label:

- Dodawanie ramki (obwódki)
- Zagnieżdżanie innych kontrolek, np. obrazków
- Wykorzystanie klawiszy dostępu, aby do aktywowania powiązanych kontrolek

Podstawowe kontrolki: TextBox



Kontrolka TextBox jest podstawową kontrolką pozwalającą na wprowadzanie tekstu.

```
<TextBox />
```

```
<TextBox Text=„Witaj Świecie!/>
```

TextBox pozwala na:

- pisanie tekstu, zarówno w jednej linii jak i w wielu liniach,
- edycję tekstu (zaznaczanie, kopiowanie wklejanie),
- sprawdzanie pisowni

```
<TextBox
     AcceptsReturn="True"
     TextWrapping="Wrap"
     SpellCheck.IsEnabled="True"
     Language="pl-PL"
 />
```

Akceptuje „Enter”

Zawijanie tekstu

Sprawdzanie pisowni

Podstawowe kontrolki: Button

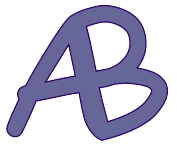
W odróżnieniu od poznanego wcześniej Windows Forms, w WPF przycisk (Button) jest konstrukcją bardzo elastyczną.

W najprostszej wersji: `<Button>Wykonaj</Button>`

Wewnątrz przycisku możemy umieścić także inną kontrolkę, a nawet cały układ kilku kontrolek:

```
<Button Padding="5"
        Height="40px">
    <StackPanel Orientation="Horizontal">
        <Image Source="Pictures/ab.png" />
        <TextBlock Margin="5,0" FontSize="15pt">Tekst przycisku</TextBlock>
    </StackPanel>
</Button>
```

Podstawowe kontrolki: Button



Obsługa przycisku - zdarzenie kliknięcia

Aby oprogramować reakcję przycisku możemy subskrybować do niego zdarzenia Click

```
<Button Click="Button_Wykonaj_Click">Wykonaj</Button>
```

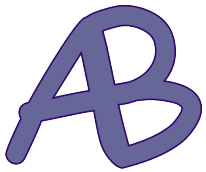
W celu obsłużenia kliknięcia musimy w kodzie c# stworzyć metodę o wybranej nazwie.

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private void Button_Wykonaj_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Akcja wykonana");
    }
}
```

Parametrami metody są:

- „`object sender`” - referencja do obiektu, który wysłał zdarzenie
- „`RoutedEventArgs e`” - delegat zdarzenia zawierający informacje o jego parametrach.



Panele

Panele to kontenery w których umieszczamy inne kontrolki.

Panele zarządzają układem okna. Panel można wykorzystać do podzielenia dostępnej przestrzeni na mniejsze kawałki. W każdym z nich możemy umieścić kontrolkę bądź kolejny panel.

Panele

WrapPanel (panel zwijający) - ustawia elementy podrzędne jedno obok drugiego w poziomie (domyślnie) bądź w pionie. Gdy zabraknie miejsca przechodzi do nowej linii.

StackPanel (stos) - zamiast przenosić nowy element do kolejnej linii, sam się rozszerza tworząc miejsce na nowy element. Każdy z elementów jest rozciągany tak, aby stos zajął całą dostępną przestrzeń w pionie lub poziomie.

DockPanel (panel dokujący) - pozwala dokować element podrzędne do krawędzi panelu. Ostatni element wypełni cała pozostałą przestrzeń (jeśli nie określimy jego pozycji).

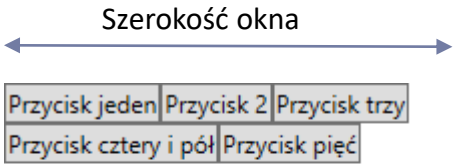
Grid (siatka) - może zawierać wiele rzędów i kolumn. Kolumnom można zdefiniować szerokość, a wierszom wysokość i to na kilka różnych sposobów: bezpośrednio w liczbie pikseli, w procencie dostępnego miejsca jak i w sposób automatyczny - gdzie rozmiar zostanie dostosowany w zależności od zawartości.

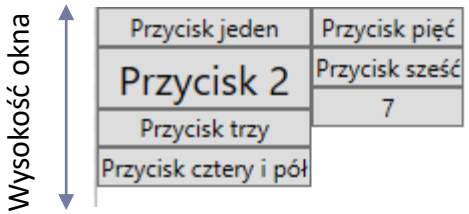
Canvas (płótno) - pozwala na dokładne określenie położenia każdego z elementów podrzędnych, daje pełną kontrolę nad układem.

Panele: WrapPanel

WrapPanel (panel zwijający) - ustawia elementy podrzędne jedno obok drugiego w poziomie (domyślnie) bądź w pionie. Gdy zabraknie miejsca przechodzi do nowej linii.

```
<WrapPanel>
    <Button>Przycisk jeden</Button>
    <Button>Przycisk 2</Button>
    <Button>Przycisk trzy</Button>
    <Button>Przycisk cztery i pół</Button>
    <Button>Przycisk pięć</Button>
</WrapPanel>
```



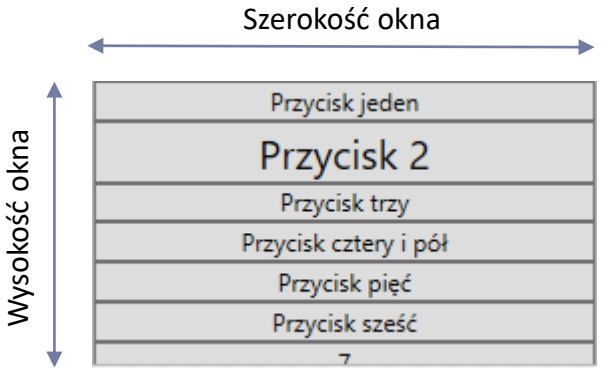


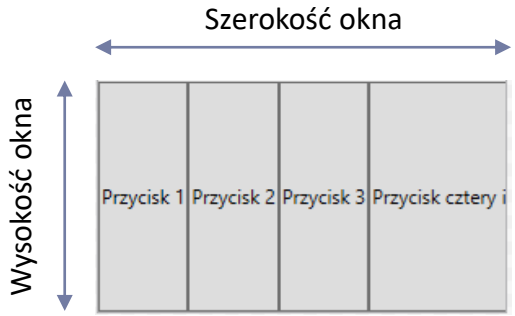

```
<WrapPanel Orientation="Vertical">
    <Button>Przycisk jeden</Button>
    <Button FontSize="15pt">Przycisk 2</Button>
    <Button>Przycisk trzy</Button>
    <Button>Przycisk cztery i pół</Button>
    <Button>Przycisk pięć</Button>
    <Button>Przycisk sześć</Button>
    <Button>7</Button>
</WrapPanel>
```

Panele: StackPanel

StackPanel (stos) - zamiast przenosić nowy element do kolejnej linii, sam się rozszerza tworząc miejsce na nowy element. Każdy z elementów jest rozciągany tak, aby stos zajął całą dostępną przestrzeń w pionie lub poziomie.

```
<StackPanel>
    <Button>Przycisk jeden</Button>
    <Button FontSize="15pt">Przycisk 2</Button>
    <Button>Przycisk trzy</Button>
    <Button>Przycisk cztery i pół</Button>
    <Button>Przycisk pięć</Button>
    <Button>Przycisk sześć</Button>
    <Button>7</Button>
</StackPanel>
```



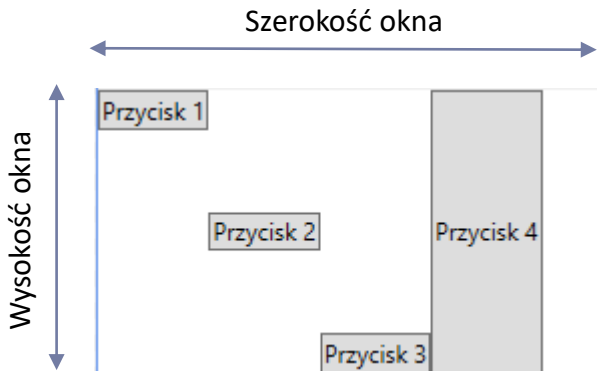


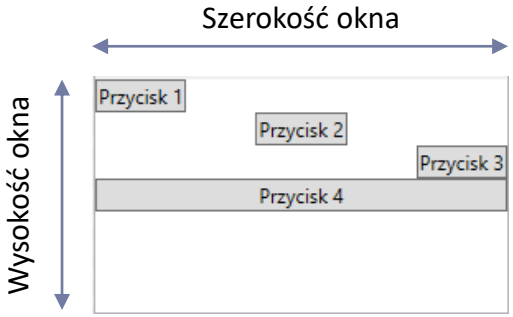

```
<StackPanel Orientation="Horizontal">
    <Button>Przycisk 1</Button>
    <Button>Przycisk 2</Button>
    <Button>Przycisk 3</Button>
    <Button>Przycisk cztery i pół</Button>
</StackPanel>
```

Panele: StackPanel

Wyrównanie wewnątrz komórek panelu.

```
<StackPanel Orientation="Horizontal">
    <Button VerticalAlignment="Top">Przycisk 1</Button>
    <Button VerticalAlignment="Center">Przycisk 2</Button>
    <Button VerticalAlignment="Bottom">Przycisk 3</Button>
    <Button VerticalAlignment="Stretch">Przycisk 4</Button>
</StackPanel>
```





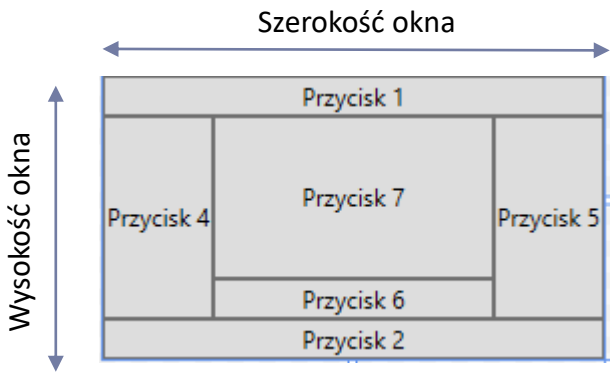

```
<StackPanel Orientation="Vertical">
    <Button HorizontalAlignment="Left">Przycisk 1</Button>
    <Button HorizontalAlignment="Center">Przycisk 2</Button>
    <Button HorizontalAlignment="Right">Przycisk 3</Button>
    <Button HorizontalAlignment="Stretch">Przycisk 4</Button>
</StackPanel>
```

Panele: DockPanel

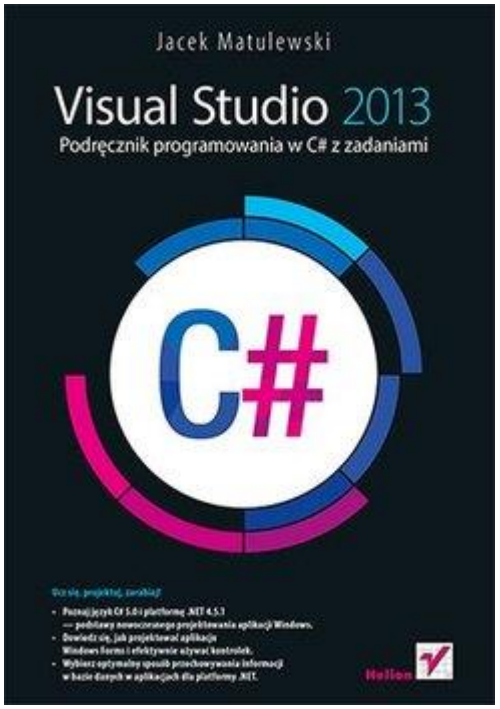
DockPanel (panel dokujący) - pozwala dokować element podrzędne do krawędzi panelu. Ostatni element wypełni cała pozostałą przestrzeń (jeśli nie określimy jego pozycji).

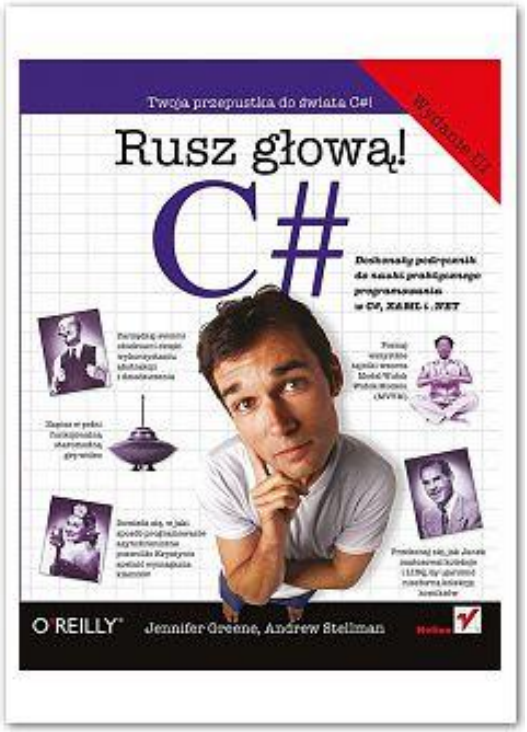
```
<DockPanel>
    <Button DockPanel.Dock="Top">Przycisk 1</Button>
    <Button DockPanel.Dock="Bottom">Przycisk 2</Button>
    <Button DockPanel.Dock="Left">Przycisk 4</Button>
    <Button DockPanel.Dock="Right">Przycisk 5</Button>
    <Button DockPanel.Dock="Bottom">Przycisk 6</Button>
    <Button>Przycisk 7</Button>
</DockPanel>
```

Ostatni z elementów podrzędnych zostanie automatycznie wycentrowany i rozciągnięty na całe pozostałe wolne miejsce.
Chyba, że dodamy atrybut:
<DockPanel LastChildFill="False">



Literatura:





Użyte w tej prezentacji tabelki pochodzą z książki:Visual Studio 2013. Podręcznik programowania w C# z zadaniami
Autor: Matulewski Jacek, Helion