



UNIWERSYTET
EKONOMICZNY
we Wrocławiu



TECHNOLOGII .NET– WYKŁAD 8

OBSŁUGA WYJĄTKÓW.

TWORZENIE APLIKACJI Z INTERFEJSEM GRAFICZNYM

DR RADOSŁAW WÓJTOWICZ

OBSŁUGA WYJĄTKÓW (1/9)

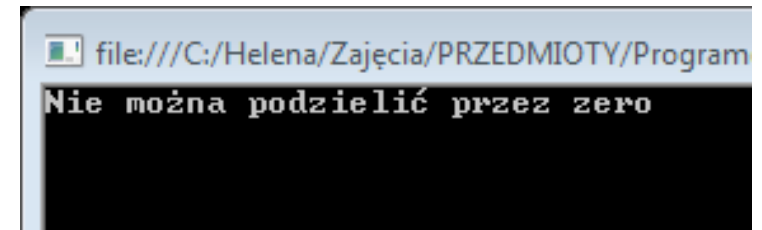
```
try
{
    blok instrukcji do wykonania;
}
catch (typWyjätku [identyfikatorWyjätku])
{
    blok instrukcji obsługujący wyjątek;
}
```

[...] - parametr opcjonalny

OBSŁUGA WYJĄTKÓW (2/9)

- Typ wyjątku: `DivideByZeroException`

```
int z, y, x;  
x = 7;  
y = 0;  
try  
{  
    z = x / y;  
}  
catch (DivideByZeroException)  
{  
    Console.WriteLine ("Nie można podzielić przez zero");  
}  
Console.ReadKey();
```



OBSŁUGA WYJĄTKÓW (3/9)

- Identyfikator wyjątku – wyświetlenie znaczenia błędu

```
int z, y, x;  
x = 7;  
y = 0;  
try  
{  
    z = x / y;  
}  
catch (Exception e)      //e – nazwa zmiennej obiektowej  
{  
    Console.Write ("Komunikat systemu: ");  
    Console.WriteLine (e.Message);  //Console.WriteLine (e.ToString);  
}  
Console.ReadKey();
```

```
Komunikat systemu: Nastąpiła próba podzielenia przez zero.
```

OBSŁUGA WYJĄTKÓW (4/9)

- Typ wyjątku oraz identyfikator wyjątku

```
int z, y, x;  
x = 7;  
y = 0;  
try  
{  
    z = x / y;  
}  
catch (DivideByZeroException e)  
{  
    Console.WriteLine ("Nie można podzielić przez zero");  
    Console.Write ("Komunikat systemu: ");  
    Console.WriteLine (e.Message)  
}  
Console.ReadKey();
```

```
Nie można podzielić przez zero  
Komunikat systemu: Nastąpiła próba podzielenia przez zero.
```

PRZYKŁAD 1

```
string dane = Console.ReadLine();
int liczba;
try
{
    liczba = Int32.Parse(dane);
    // wprowadzone liczbę
}
catch (FormatException) //typ błędu wynikający z niezgodności
typów danych
{
    Console.WriteLine ("Nie wprowadzono poprawnie oceny");
}
```

PRZYKŁAD 2

```
string dane = Console.ReadLine();  
int liczba;  
bool wynikKonwersji = int.TryParse(dane, out liczba);  
if (wynikKonwersji == true) // if (wynikKonwersji  
    // wprowadzone liczbę  
else  
    // wprowadzone dowolny znak niebędący cyfrą (wartością  
    // liczbową)
```

OBSŁUGA WYJĄTKÓW (5/9)

Przykłady wyjątków:

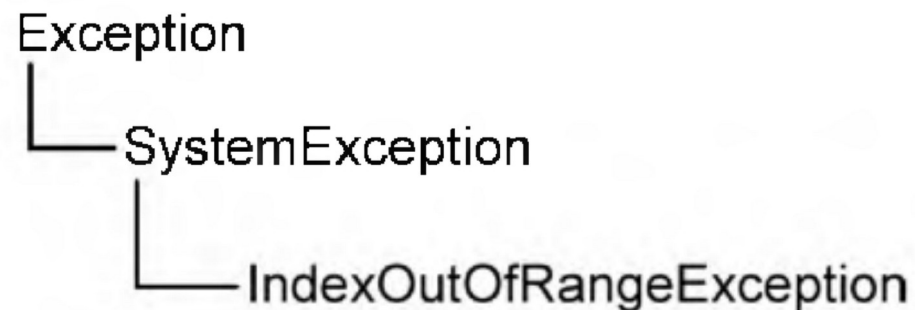
- `System.IO.IOException` - obsługa wyjątków związanych z odczytawaniem/zapisywaniem plików
- `System.IndexOutOfRangeException` - obsługa wyjątków związanych z tablicami, a ściślej z wyjściem indeksu poza dopuszczalny zakres
- `System.ArrayTypeMismatchException` - obsługa wyjątków związanych z niezgodnością typu z typem danej tablicy
- `System.NullReferenceException` - obsługa wyjątków związanych z typem pustym
- `System.DivideByZeroException` - obsługa wyjątków związanych z dzieleniem przez zero
- `System.InvalidCastException` - obsługa wyjątków związanych z niepoprawnym rzutowaniem
- `System.OutOfMemoryException` - obsługa wyjątków związanych z niewystarczającą ilością pozostałej pamięci
- `System.StackOverflowException` - obsługa wyjątków związanych z przepełnieniem stosu

OBSŁUGA WYJĄTKÓW (6/9)

```
try
{
    blok instrukcji do wykonania;
}
catch (typWyjätku_1 [identyfikatorWyjätku_1])
{
    blok instrukcji obsługujący wyjątek nr 1;
}
catch (typWyjätku_2 [identyfikatorWyjätku_2])
{
    blok instrukcji obsługujący wyjątek nr 2;
}
....
catch (typWyjätku_N [identyfikatorWyjätku_N])
{
    blok instrukcji obsługujący wyjątek nr N;
}
```

OBSŁUGA WYJĄTKÓW(7/9)

- Wyjątki to obiekty, tworzą hierarchiczną strukturę.
- Na przykład:
 - *Exception*, do której należy:
 - *SystemException*, zaś do tej ostatniej należą m.in.
 - *FormatException*
 - *OverflowException*
 - *IndexOutOfRangeException*
 - ...



OBSŁUGA WYJĄTKÓW (8/9)

Nie zostanie wykonany wyjątek:

OverflowException

```
try
{
    instrukcja ;
}
catch (SystemException)
{
    instrukcja ;
}
catch (OverflowException)
{
    instrukcja ;
}
```

Dwa wyjątki zostaną wykonane

```
try
{
    instrukcja ;
}
catch (OverflowException)
{
    instrukcja ;
}
catch (SystemException)
{
    instrukcja ;
}
```

OBSŁUGA WYJĄTKÓW (9/9)

```
try
{
    blok instrukcji do wykonania;
}
catch (typWyjätku [identyfikatorWyjätku])
{
    blok instrukcji obsługujący wyjątek;
}
finally
{
    blok instrukcji, który zawsze zostanie wykonany;
}
```

Przykład (1/3)

```
int z, y, x = 7;
Console.WriteLine ("");
try
{
    y = System.Int32.Parse (Console.ReadLine());
    z = x / y;
    Console.WriteLine ("z = " + z.ToString());
}
catch (Exception e)
{
    Console.Write ("Komunikat systemu: ");
    Console.WriteLine (e.Message);
}
finally
{
    Console.WriteLine ("Naciśnij jakiś klawisz");
    Console.ReadKey();
}
```

```
x
Komunikat systemu:
Nieprawidłowy format ciągu wejściowego.
Naciśnij jakiś klawisz
```

```
7
z = 1
Naciśnij jakiś klawisz
```

```
0
Komunikat systemu:
Nastąpiła próba podzielenia przez zero.
Naciśnij jakiś klawisz
```

Przykład (2/3)

```
int z, y, x =7;
```

```
Console.WriteLine ("");
```

```
try
```

```
{
```

```
    y = System.Int32.Parse (Console.ReadLine());
```

```
    z = x / y;
```

```
    Console.WriteLine ("z = " + z.ToString());
```

```
}
```

```
catch (DivideByZeroException)
```

```
{
```

```
    Console.WriteLine ("Nie można podzielić przez zero");
```

```
}
```

```
catch (Exception e)
```

```
{
```

```
    Console.Write ("Komunikat systemu: ");
```

```
    Console.WriteLine (e.Message);
```

```
}
```

```
finally
```

```
{
```

```
    Console.WriteLine ("Naciśnij jakiś klawisz");
```

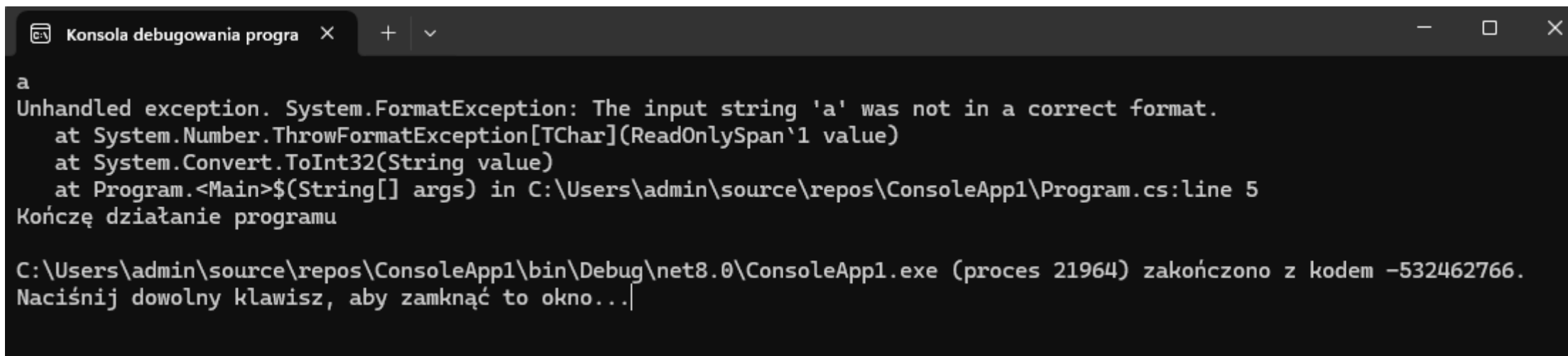
```
    Console.ReadKey();
```

```
}
```

```
0  
Nie można podzielić przez zero  
Naciśnij jakiś klawisz
```

Przykład (3/3)

```
1.// See https://aka.ms/new-console-template for more information
2.int a, b, wynik;
3.    try
4.    {
5.        a=Convert.ToInt32(Console.ReadLine());
6.        b=Convert.ToInt32(Console.ReadLine());
7.        wynik = a / b;
8.        Console.WriteLine(wynik.ToString());
9.    }
10.   catch (DivideByZeroException)
11.   {
12.       Console.WriteLine("Wystąpił błąd dzielenia przez 0");
13.   }
14.   //catch (Exception)
15.   //{
16.       //Console.WriteLine("Wystąpił jakiś inny błąd");
17.   //}
18.   finally
19.   {
20.       Console.WriteLine("Kończę działanie programu");
21.   }
22Console.WriteLine("Czy to się wyświetli zawsze?");
```



The screenshot shows a Windows command prompt window titled "Konsola debugowania progra". The window contains the following text:

```
a
Unhandled exception. System.FormatException: The input string 'a' was not in a correct format.
   at System.Number.ThrowFormatException[TChar](ReadOnlySpan`1 value)
   at System.Convert.ToInt32(String value)
   at Program.<Main>$(String[] args) in C:\Users\admin\source\repos\ConsoleApp1\Program.cs:line 5
Kończę działanie programu

C:\Users\admin\source\repos\ConsoleApp1\bin\Debug\net8.0\ConsoleApp1.exe (proces 21964) zakończono z kodem -532462766.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

ZAGNIEŻDŻENI E TRY ... CATCH

```
try
{
    instrukcje mogące spowodować wyjątek nr 1 ;
    try
    {
        instrukcje mogące spowodować wyjątek nr 2 ;
    }
    catch (typWyjätku_2 [identyfikatorWyjätku_2])
    {
        blok instrukcji obsługujący wyjątek nr 2;
    }

}
catch (typWyjätku_1 [identyfikatorWyjätku_1])
{
    blok instrukcji obsługujący wyjątek nr 1;
}
```


ZGŁASZANIE WYJĄTKÓW

- Wyjątek możemy zgłosić samodzielnie
- Służy do tego instrukcja `throw`

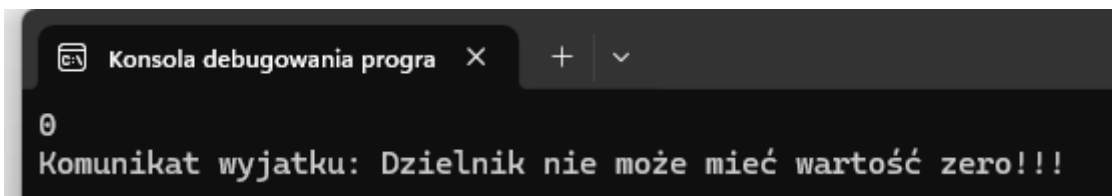
```
throw new TypWyjatkku()
```

- Z wyjątkiem możemy skojarzyć również pewien komunikat, informujący o źródłach błędu. Robimy to w następujący sposób:

```
throw new TypWyjatkku("Treść komunikatu")
```

PRZYKŁAD ZGŁOSZENIA WYJĄTKU

```
double dzielnik, iloraz;
try
{
    dzielnik = Convert.ToDouble(Console.ReadLine());
    if(dzielnik == 0)
        throw new DivideByZeroException("Dzielnik nie
            może mieć wartość zero!!!");
    iloraz = 10.2 / dzielnik;
    Console.WriteLine(iloraz);
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Komunikat wyjątku: {0}",
        ex.Message);
}
```



RODZAJE APLIKACJI DESKTOPOWYCH Z INTERFEJSEM GRAFICZNYM W .NET

- Aplikacje formularzowe (Windows Forms)
- Aplikacje WPF (Windows Presentation Foundation)
- Aplikacje MAUI (Multi-platform App UI)

TWORZENIE APLIKACJI FORMULARZOWYCH (WINDOWS FORMS)



Windows Forms to sposób na łatwe tworzenie aplikacji zawierających graficzny interfejs użytkownika powstały w ramach .NET Framework. Daje także możliwość tworzenia wieloplatformowych aplikacji.

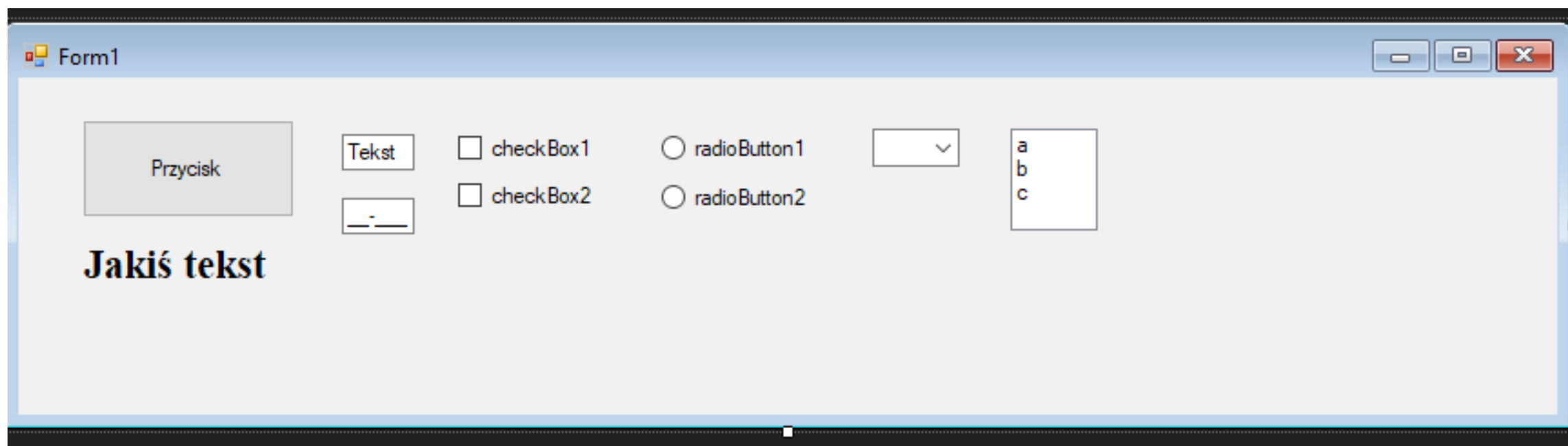


.NET Framework – platforma programistyczna opracowana przez Microsoft, obejmująca środowisko uruchomieniowe (Common Language Runtime – CLR) oraz biblioteki klas dostarczające standardowej funkcjonalności dla aplikacji.

NAJWAŻNIEJSZE KOMPONENTY

- Label (etykieta),
- Button (przycisk),
- TextBox (pole tekstowe),
- CheckBox, RadioButton (pole wyboru),
- ComboBox (lista rozwijana),
- ListBox (lista zwykła).

WYGLĄD PODSTAWOWYCH KOMPONENTÓW APLIKACJI WINDOWS FORMS

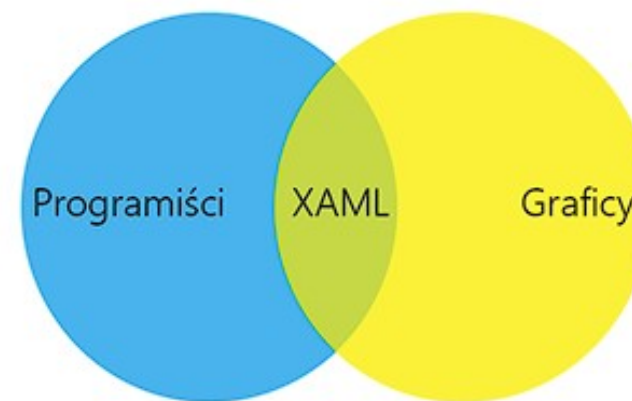


TWORZENIE APLIKACJI WPF (WINDOWS PRESENTATION FOUNDATION)

- Windows Presentation Foundation (WPF), to platforma służąca do budowania aplikacji wizualnych. Opiera się na XML, skalowalnych kontrolkach, całkowicie nowych kontrolkach systemowych, animacji 2D i 3D, rozkładzie tekstu i formatowaniu dokumentów.
- WPF jest stosunkowo nową technologią. Zapewnia wszelakie wymagane funkcjonalności do stworzenia bardzo atrakcyjnych interfejsów. Rozdzielono pracę programisty oraz osoby tworzącej GUI.
- Największą zaletą aplikacji używających WPF jest fakt, iż są oparte o grafikę wektorową.

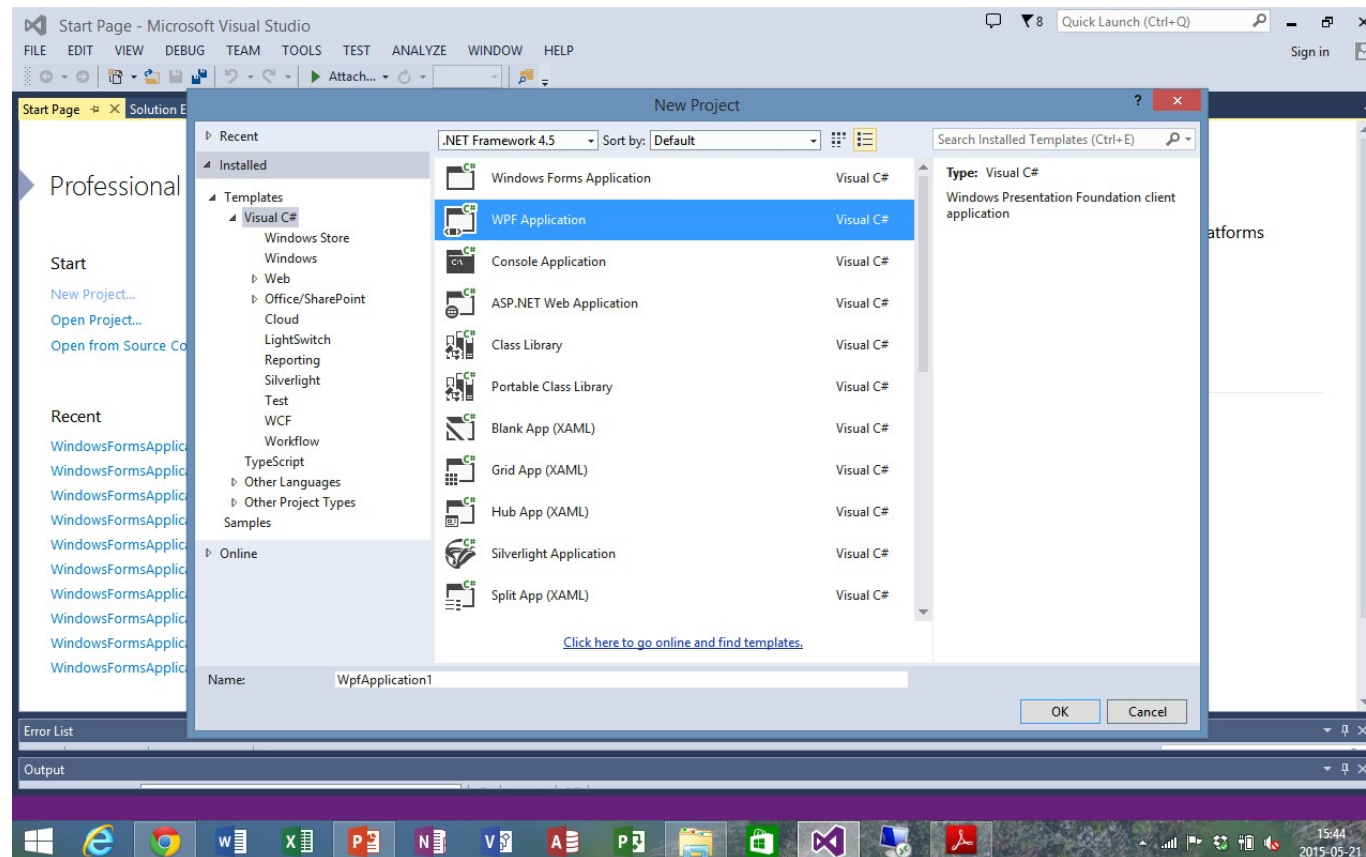
WPF - XAML

- Extensible Application Markup Language (XAML – język opisu interfejsu użytkownika wykorzystywany m.in. w technologii WPF, która jest elementem platformy .NET Framework począwszy od wersji 3.0.
- XAML jest językiem opartym na języku XML zoptymalizowanym do opisu wizualnych interfejsów.

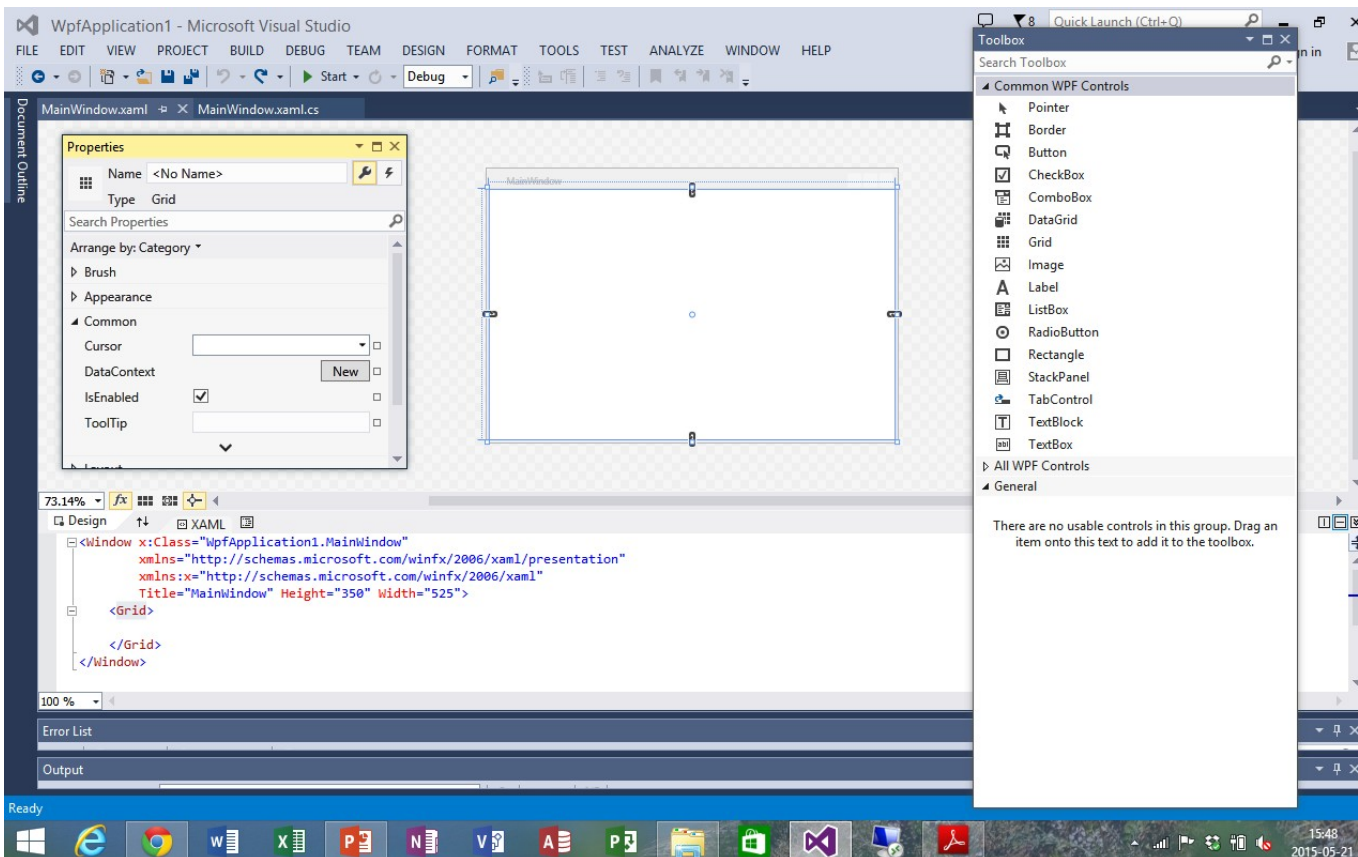


PRZYKŁAD KODU XAML

```
<Button>  
    <Button.FontWeight>Bold</  
Button.FontWeight>  
<Button.Content>Przycisk</Button.Content>  
</Button>
```



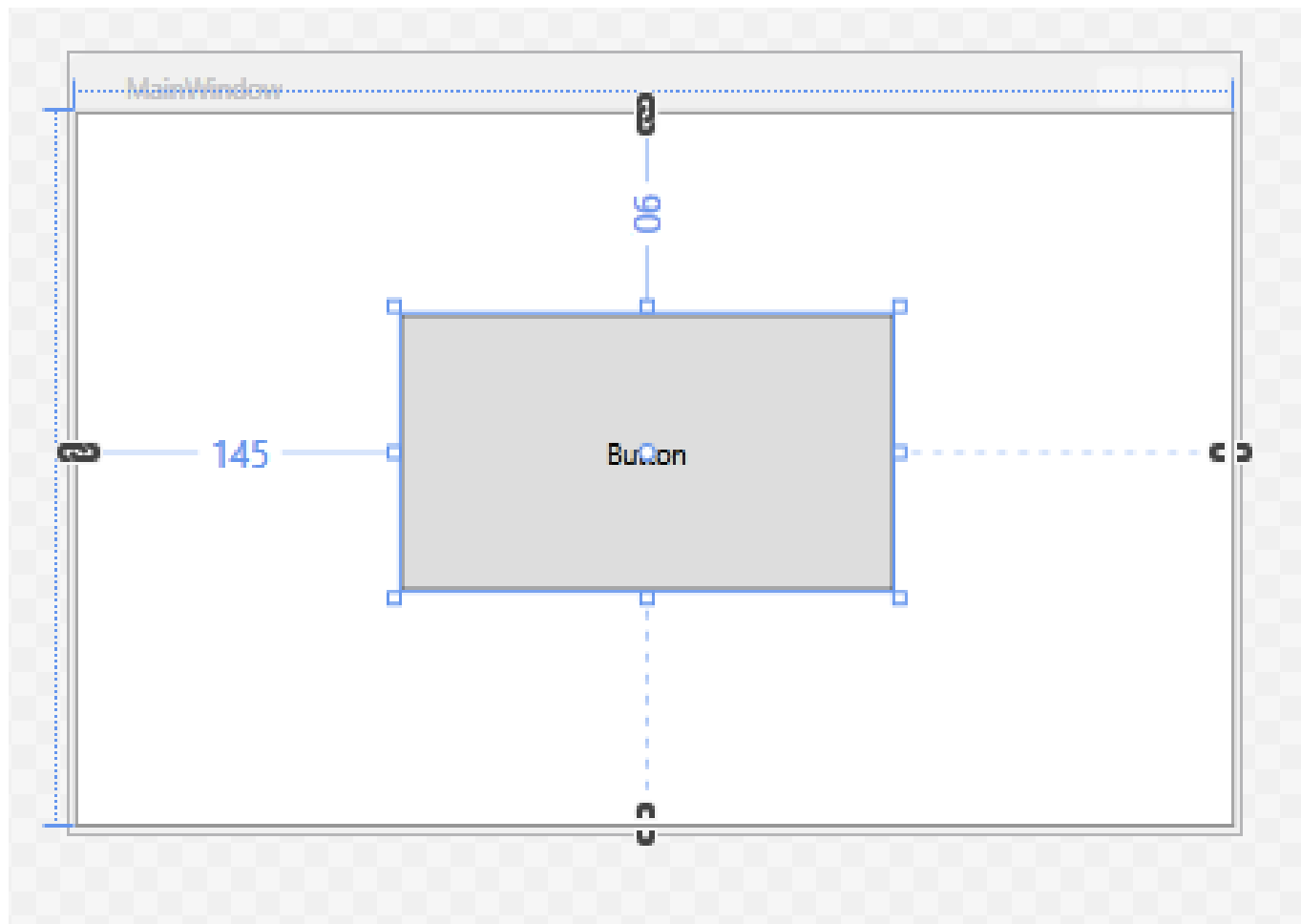
TWORZENIE APLIKACJI GRAFICZNEJ WPF (1)



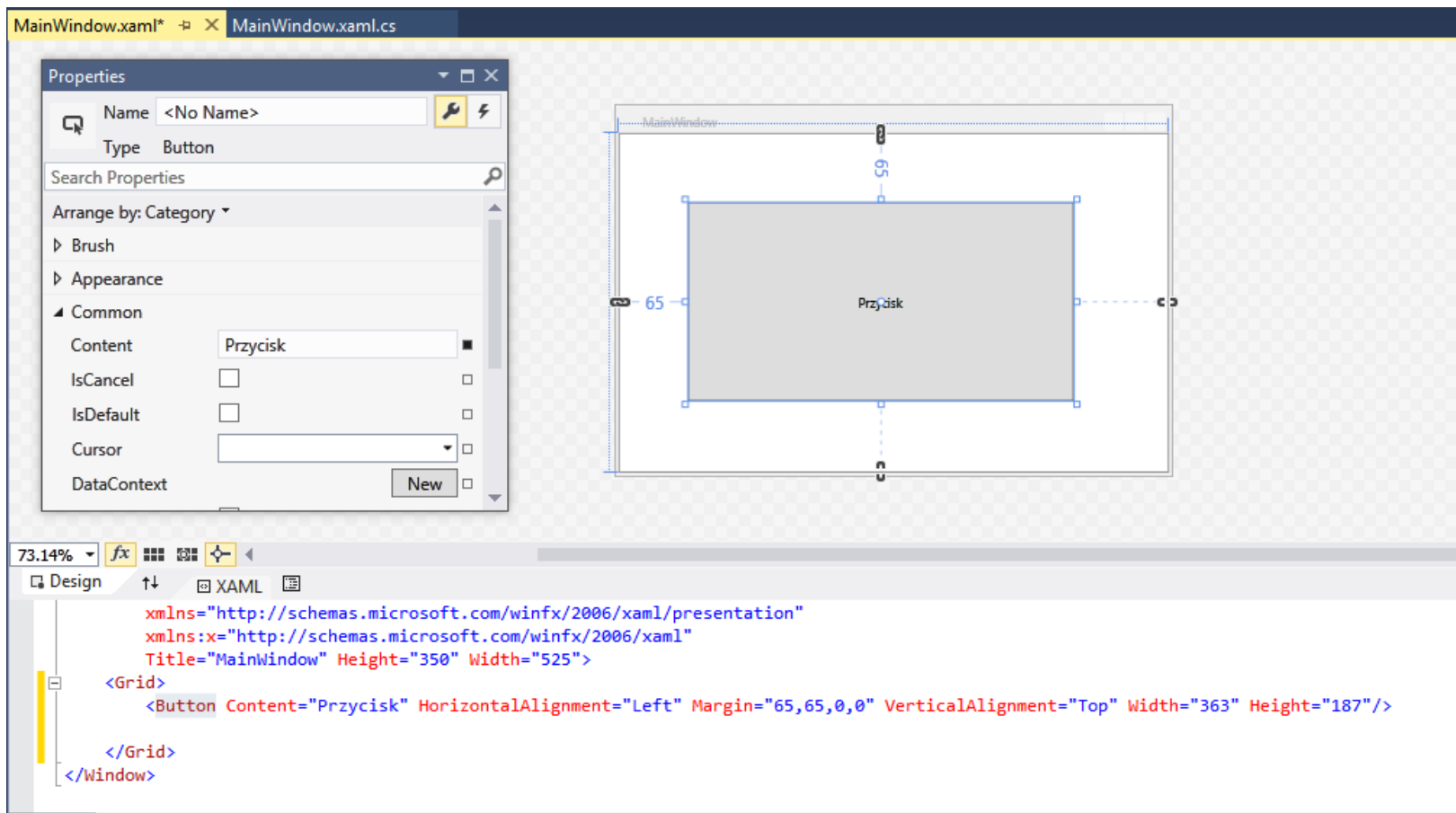
TWORZENIE APLIKACJI GRAFICZNEJ WPF (2)

```
MainWindow.xaml  MainWindow.xaml.cs  -  X
WpfApplication1.MainWindow
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 WpfApplication1
17 {
18     /// <summary>
19     /// Interaction logic for MainWindow.xaml
20     /// </summary>
21     public partial class MainWindow : Window
22     {
23         public MainWindow()
24         {
25             InitializeComponent();
26         }
27     }
28 }
29
```

DEKLARACJA KLASY MAINWINDOW



DODAWANIE KONTROLEK

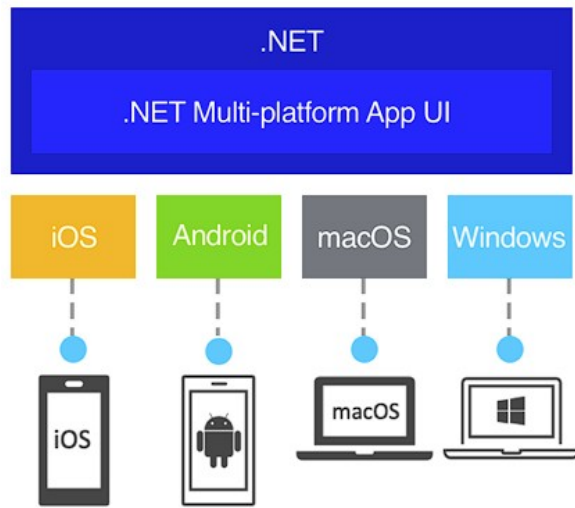


OKNO WŁAŚCIWOŚ CI I XAML

```
MainWindow.xaml*   MainWindow.xaml.cs* -p X
WpfApplication1.MainWindow
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 WpfApplication1
17 {
18     /// <summary>
19     /// Interaction logic for MainWindow.xaml
20     /// </summary>
21     public partial class MainWindow : Window
22     {
23         public MainWindow()
24         {
25             InitializeComponent();
26         }
27
28         private void Button_Click(object sender, RoutedEventArgs e)
29         {
30
31         }
32     }
100 %
```

DODAWANIE ZDARZEŃ

TWORZENIE APLIKACJI MAUI (MULTI-PLATFORM APP UI)



- Wykorzystywana jest platforma .NET (wcześniej .NET Core).
- W ramach wersji .NET 6, 23 maja 2022 premierę miał wieloplatformowy framework umożliwiający tworzenie aplikacji z interfejsem graficznym użytkownika – MAUI.
- Jest to międzyplatformowa struktura do tworzenia natywnych aplikacji mobilnych i klasycznych przy użyciu języków C# i XAML.

DO OBEJRZENIA – TRY CATCH

- <https://www.youtube.com/watch?v=qp08qIT2fwM&t=1065s>

DO OBEJRZENIA – TWORZENIE APLIKACJI WPF

- <https://www.youtube.com/watch?v=RSTfqnQIKdM>



DZIĘKUJĘ
ZA UWAGĘ!

RADOSLAW.WOJTOWICZ@UE.WROC.PL