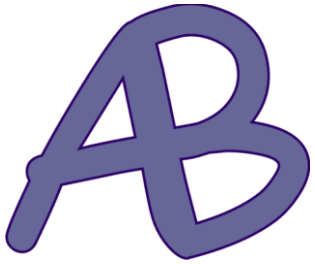
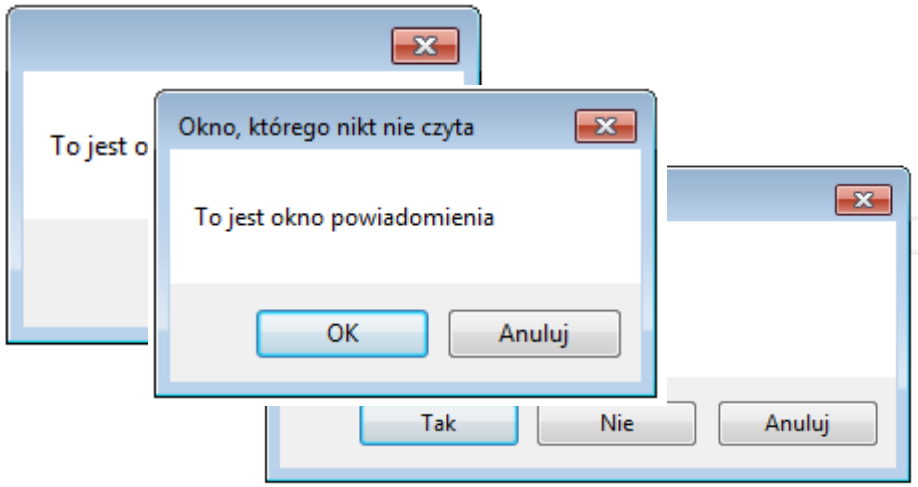


Wykład 7
Okna dialogowe

Okna dialogowe

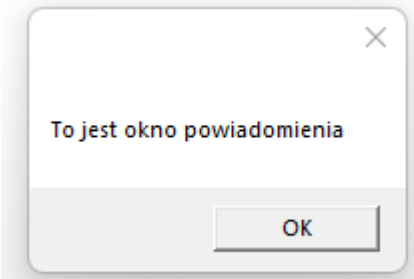


Okna dialogowe - komunikaty

Okno komunikatu to okno dialogowe, które może służyć do wyświetlania informacji tekstowych. Pozwala ono także użytkownikom na podejmowanie decyzji za pomocą przycisków.

```
MessageBox.Show("To jest okno powiadomienia");
```

W „standardzie” dostajemy tekst
i przycisk „OK.”



Okno dialogowe możemy uzupełniać o różne elementy.

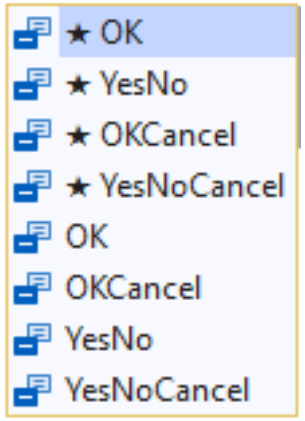
Okna dialogowe - komunikaty

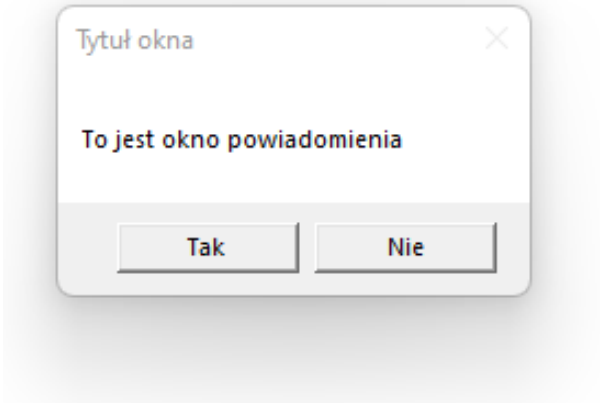
Okno dialogowe możemy uzupełniać o różne elementy.

```
MessageBox.Show("To jest okno powiadomienia",
                "Tytuł okna",
                MessageBoxButton.);
```

Nagłówek

Rodzaje przycisków



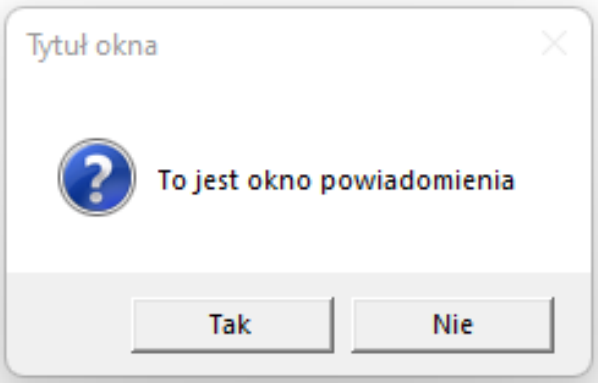


Okna dialogowe - komunikaty

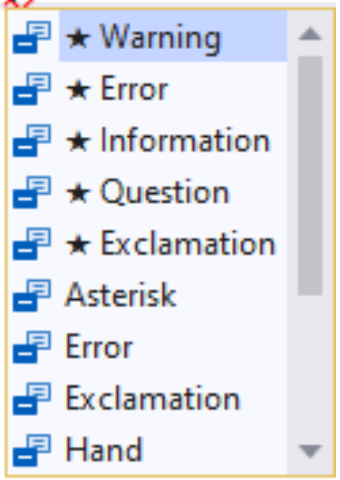
Okno dialogowe możemy uzupełniać o różne elementy.

```
MessageBox.Show("To jest okno powiadomienia"
                "Tytuł okna",
                MessageBoxButton.YesNo,
                MessageBoxImage.,
                MessageBoxResult
```

Defaltowy przycisk (zostanie wybrany, gdy użytkownik naciśnie Enter)



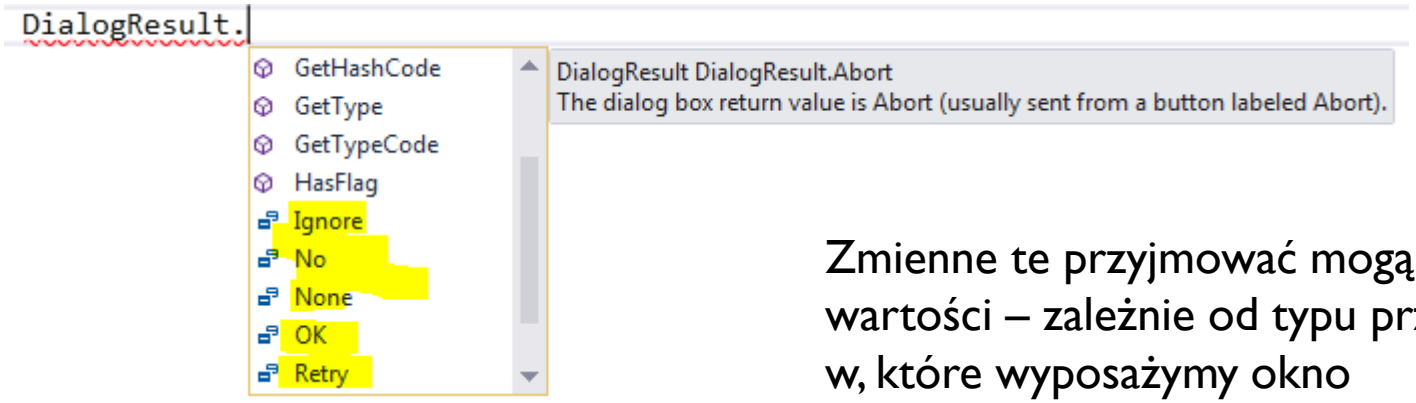
Wybór ikony



Okna dialogowe - komunikaty

Odpowiedź zwracaną przez okno (który przycisk naciśnięto) zapisać możemy w zmiennej typu DialogResoult.

```
DialogResult odpowiedz = MessageBox.Show("To jest okno z pytaniem","Pytanie",
                    MessageBoxButtons.YesNoCancel,
                    MessageBoxIcon.Question);
```



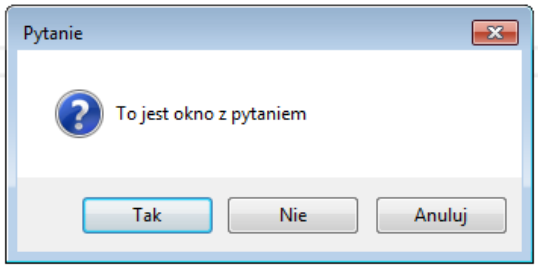
Zmienne te przyjmować mogą różne wartości – zależnie od typu przycisków w, które wyposażymy okno

Okna dialogowe - komunikaty

Okno o trzech możliwych odpowiedziach:

```
DialogResult odpowiedz = MessageBox.Show("To jest okno z pytaniem","Pytanie",
                MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question);

 switch (odpowiedz)
 {
     case DialogResult.Yes: //akja, jeżeli naciśnięto "TAK"
          break;
     case DialogResult.No: //akja, jeżeli naciśnięto "NIE"
          break;
     case DialogResult.Cancel: //akja, jeżeli naciśnięto "ANULUJ"
          break;
 }
```

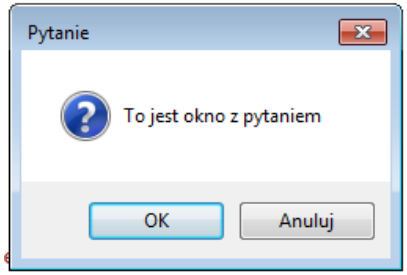


Okna dialogowe - komunikaty

Okno o dwóch odpowiedziach:

```
DialogResult odpowiedz2 = MessageBox.Show("To jest okno z pytaniem", "Pytanie",
     MessageBoxButtons.OKCancel,
     MessageBoxIcon.Question);

if (odpowiedz2==DialogResult.OK)
{
    //Akcja, jeżeli naciśnięto "OK"
} else
{
    //Akcja, jeżeli naciśnięto "ANULUJ"
}
```



Zapisywanie odpowiedzi okna w pośredniczącej zmiennej nie jest konieczne.

Zapis skrócony:

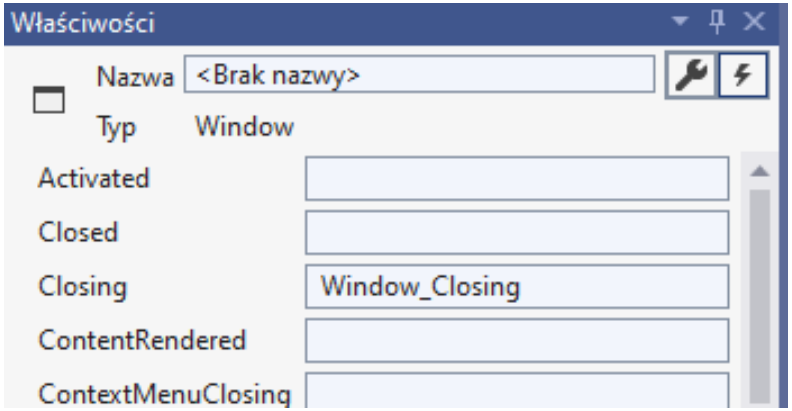
```
if (MessageBox.Show("To jest okno z pytaniem", "Pytanie",
     MessageBoxButtons.OKCancel,MessageBoxIcon.Question) == DialogResult.OK)
{
    //Akcja, jeżeli naciśnięto "OK"
} else
{
    //Akcja, jeżeli naciśnięto "ANULUJ"
}
```

Okna dialogowe - komunikaty

Przykład - potwierdzenie zamknięcia programu

```
private void button01_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

Do zamknięcia okna (w przypadku okna głównego równoznaczne z zamknięciem programu) służy polecenie Close( );



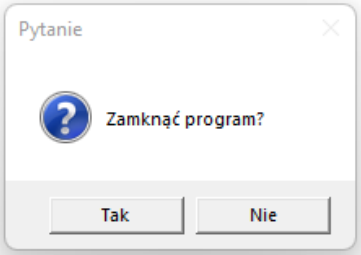
Aby przed zamknięciem okna wykonana została jakaś czynność (np. wyświetlenie okna dialogowego) oprogramować należy zdarzenie **Closing**

Okna dialogowe - komunikaty

Przykład - potwierdzenie zamknięcia programu

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{

    if (MessageBox.Show("Zamknąć program?", "Pytanie",
                        MessageBoxButton.YesNo, MessageBoxImage.Question,
                        MessageBoxResult.No) == MessageBoxResult.No)
        e.Cancel = true;
}
```



„e" to zdarzenie przesłane w parametrze.

Ustawienie jego pola „Cancel" na „true" kasuje polecenie zamknięcia okna.

Okna dialogowe systemu

Okna dialogowe – Okno otwierania pliku

Na początek ważna informacja: samo okno otwierania pliku żadnego pliku nie otworzy **!**

Okno (klasa OpenFileDialog) służy tylko do znalezienia nazwy i ścieżki do pliku.

Krok1: Utworzenie obiektu klasy „OpenFileDialog”

```
OpenFileDialog openFileDialog = new OpenFileDialog();
```

Krok2: Dodanie tytułu okna

```
openFileDialog.Title = "Otwieranie pliku";
```

Krok3: Zdefiniowanie filtru typów plików (rozszerzeń)

```
openFileDialog.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
```

Krok4: Wywołanie metody „.SchowDialog( )” okna dialogowego

```
openFileDialog.ShowDialog()
```

Okna dialogowe – Okno otwierania pliku

Metoda ShowDialog( ) okna dialogowego zwraca wartość prawda lub fałsz zależnie od tego jak zostało zamknięte okno (przyciskiem „Otwórz” lub „Anuluj”).

```
OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Title = "Otwieranie pliku";
openFileDialog.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
openFileDialog.Multiselect = true;
if (openFileDialog.ShowDialog() == true)
{
   text_01.Text = openFileDialog.FileName;
}
```

Po zamknięciu okna przyciskiem „Otwórz” w polu obiektu „openFileDialog.FileName” znajduje się ścieżka i nazwa wybranego pliku.

Okna dialogowe – Folder startowy okien

Folder startowy dla okien odczytu i zapisu do pliku możemy wybrać za pomocą parametru: `.InitialDirectory`

```
openFileDialog.InitialDirectory =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
```

Ponieważ w różnych instalacjach Windows foldery specjalne (Dokumenty, Obrazy, Pulpit itp.) mają różne adresy, posługujemy się zmienną środowiskową `Environment.SpecialFolder`

Okna dialogowe – Okno zapisu do pliku

Okno zapisu do pliku nies podobne do okno odczytu, stąd różnicą że nie wybieramy istniejącego pliku tylko katalog i podajemy nazwę pliku do utworzenia.

```
SaveFileDialog saveFileDialog = new SaveFileDialog();
saveFileDialog.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
saveFileDialog.DefaultExt = "txt";
saveFileDialog.Title = "Okno zapisywania do pliku";
saveFileDialog.InitialDirectory =
    Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
if (saveFileDialog.ShowDialog() == true)
{
    text_02.Text = saveFileDialog.FileName;
}
```

Po zamknięciu okna przyciskiem „Zapisz” w polu obiektu „openFileDialog.FileName” znajduje się wybrana ścieżka i podana nazwa pliku.
Rozszerzenie pliku definiowane jest przez wybrany filtr. Oraz (w drugiej kolejności) przez parametr DefoultExt.

Własne okna dialogowe (okna modalne)

Własne okna dialogowe

Własne okna dialogowe to zwykłe okna dziedziczące po klasie Window. Posiadają one wszystkie własności okien – możemy używać wszystkich kontrolek i tworzyć własny Layout. Różnica polega na sposobie ich uruchomienia.

Okno tworzymy w następujących etapach:

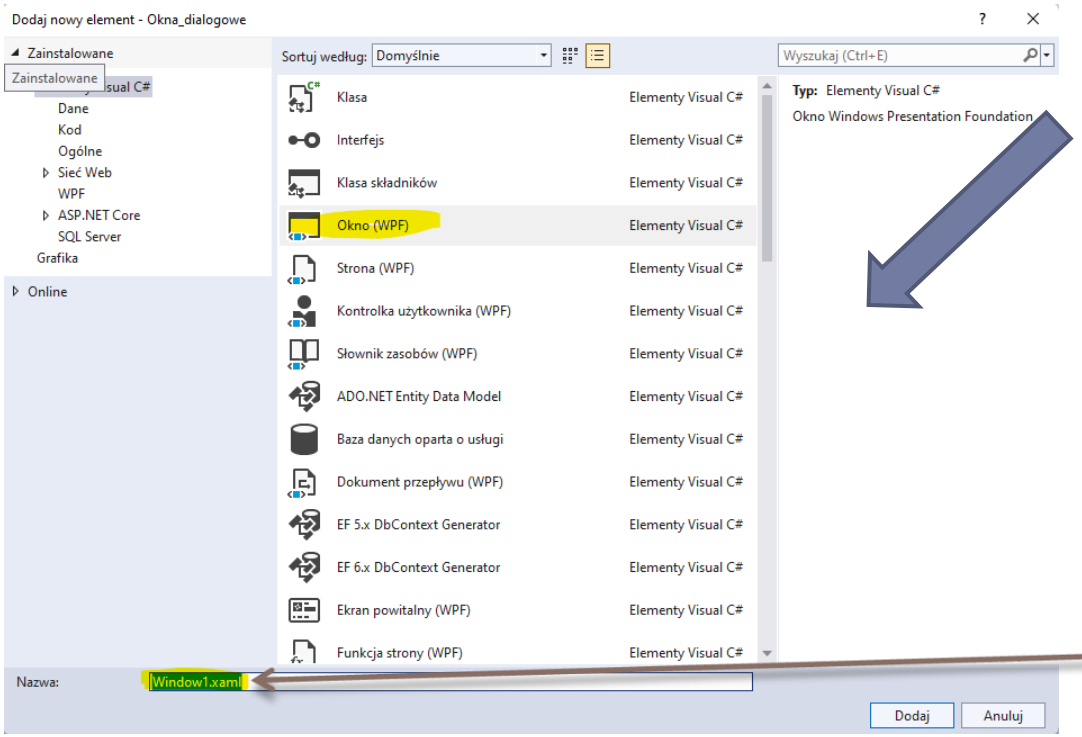
1. Przygotowanie okna (stworzenie klasy i przygotowanie layoutu (analogicznie jak przygotowanie okna głównego programu).
2. Zdefiniowanie odpowiedzi zwracanych przez okno (opcjonalnie).
3. W kodzie okna głównego – utworzenie nowego okna (referencji i obiektu za pomocą „new”) oraz wywołanie dla obiektu metody .ShowDialog( ).
4. Przyjęcie odpowiedzi okna (opcjonalnie).

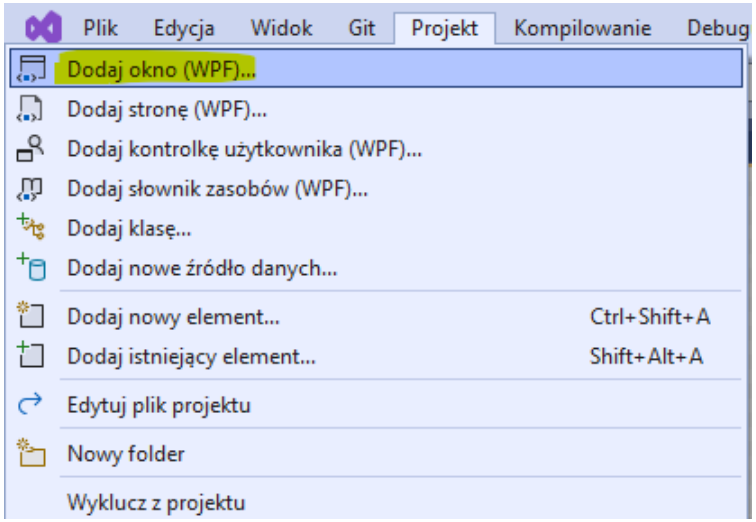
W chwili wywołania okna dialogowego, okno nadrzędne jest zatrzymywane. Okno nadrzędne kontynuuje pracę dopiero po zamknięciu dialogowego.

Własne okna dialogowe

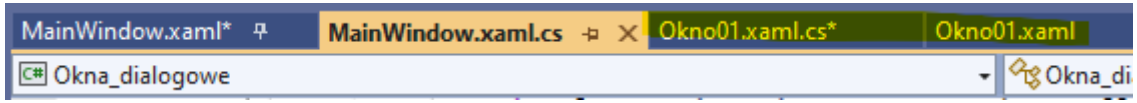
1. Przygotowanie okna (stworzenie klasy i przygotowanie layoutu

(analogicznie jak przygotowanie okna głównego programu).





Wybieramy nazwę nowego okna



Własne okna dialogowe

2. Zdefiniowanie odpowiedzi zwracanych przez okno (opcjonalnie).

Przyciskom przypisać należy role.

```
<Button IsDefault="True" Click="btn_OK_Click">Ok</Button>

<Button IsCancel="True">Cancel</Button>
```

Dodatkowo w zdarzeniu kliknięcia na przycisk potwierdzający (Ok) umieszczamy polecenie zwrócenia wartości `true` do okna nadrzędnego

```
private void btn_OK_Click(object sender, RoutedEventArgs e)
{
   this.DialogResult = true;
}
```

Nie musimy tego robić dla przycisku Cancel

Własne okna dialogowe

3. W kodzie okna głównego:

a) utworzenie nowego okna (referencji i obiektu za pomocą „new”)

b) wywołanie dla obiektu metody .ShowDialog( ).

```
Okno01 noweOkno = new Okno01();
noweOkno.Title = "Nowe okno";
if (noweOkno.ShowDialog() == true)
    text_03.Text ="Wybrano OK";
else
    text_03.Text = "Wybrano Cancel";
```

Tworzymy zmienną referencyjną dla nowego okna. Nazwa klasy zgodnie z nazwą jaką nadaliśmy oknu

Tworzymy obiekt - nowe okno. (obiekt powstaje tylko w pamięci, nie jest jeszcze wyświetlany na ekranie)

Wyświetlamy okno jako dialogowe (modalne), tzn., dopóki go nie zamkniemy nie możemy pracować w oknie głównym.

Własne okna dialogowe

4. Przyjęcie odpowiedzi okna (opcjonalnie).

```
Okno01 noweOkno = new Okno01();
noweOkno.Title = "Nowe okno";
if (noweOkno.ShowDialog() == true)
    text_03.Text = "Wybrano OK";
else
    text_03.Text = "Wybrano Cancel";
```

Metoda .ShowDialog( ) zwraca odpowiedź typu true lub false
Zależnie od tego którym przyciskiem zostało zamknięte.

Na tej podstawie wybrać można odpowiednią akcję
(w tym przykładzie wypisujemy odpowiedź w postaci tekstowej)

Odbieranie danych z okna dialogowego

Odbieranie danych z okna dialogowego

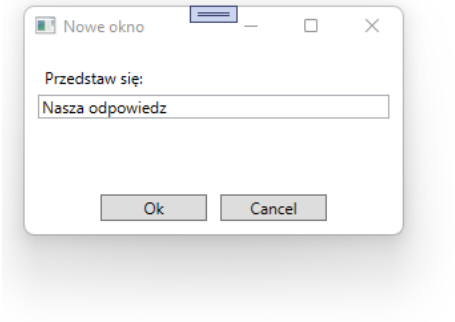
Trochę teorii

Ponieważ okna dialogowe (modalne) nie są usuwane z pamięci dopóki istnieje referencja prowadząca do okna po ich zamknięciu, odebranie danych z okna jest możliwe po zamknięciu okna i odebraniu jego odpowiedzi (DialogResult).

Wywołanie okna dialogowego metodą .ShowDialog() zatrzymuje pracę okna głównego – przetwarzanie kodu zatrzymuje się na poleceniu .ShowDialog()

Z tego powodu odczytanie danych z okna podrzędnego przez główne następuje po zamknięciu okna podrzędnego.

Komunikacja pomiędzy dwoma oknami



Załóżmy, że okno dialogowe posiada pole `TextBox` o nazwie `"odpowiedz"` do którego wpisać możemy dane

```
<TextBox Name="odpowiedz"></TextBox>
```

W kodzie okna głównego:

```
Okno01 noweOkno = new Okno01();
noweOkno.Title = "Nowe okno";
if (noweOkno.ShowDialog() == true)
    text_03.Text = noweOkno.odpowiedz.Text;
else
    text_03.Text = "Anulowano opercję";
```

W tym momencie program jest zatrzymany i czeka na zamknięcie okna dialogowego

Aby sięgnąć do kontrolek i zmiennych okna dialogowego odwołujemy się do jego nazwy (nazwy obiektu nie klasy)

Komunikacja pomiędzy dwoma oknami

UWAGA:
Wszystkie pola (zmienne) okna dialogowego do których sięgać będzie okno nadrzędne muszą być publiczne.

W kodzie okna dialogowego:

```
public int zmienna = 10;
```

Okna niemodalne

Własne okna (niemodalne)

Własne okna niemodalne to zwykłe okna dziedziczące po klasie Window. Posiadają one wszystkie własności okien – możemy używać wszystkich kontrolek i tworzyć własny Layout. W odróżnieniu od okien dialogowych (modalnych) okno nadrzędne pracuje równolegle z nim.

Okno tworzymy w następujących etapach:

1. Przygotowanie okna (stworzenie klasy i przygotowanie layoutu (analogicznie jak przygotowanie okna głównego programu).
2. ~~Zdefiniowanie odpowiedzi zwracanych przez okno (opcjonalnie).~~
3. W kodzie okna głównego – utworzenie nowego okna (referencji i obiektu za pomocą „new") oraz wywołanie dla obiektu metody .Show( ).
4. ~~Przyjęcie odpowiedzi okna (opcjonalnie).~~

Różnice pomiędzy omówionym wcześniej oknem dialogowym (modalnym), a niemodalnym widzimy głównie w puknie 3, czyli w sposobie jego otwierania.

Własne okna (niemodalne)

3. W kodzie okna głównego:
   a) utworzenie nowego okna (referencji i obiektu za pomocą „new”)
   b) wywołanie dla obiektu metody .Show( ).

```
Okno02 noweOkno = new Okno02();

noweOkno.Title = "Sterowanie";
noweOkno.suwak.Value = 20;

noweOkno.Show();
```

Tworzymy zmienną referencyjną dla nowego okna. Nazwa klasy zgodnie z nazwą jaką nadaliśmy przy dodawaniu nowego okna.
Następnie tworzymy obiekt - nowe okno

Przekazywanie danych do okna lub sterowanie jego kontrolkami z poziomu okna nadrzędnego, najczęściej odbywa się po utworzeniu okna ale jeszcze przed jego wyświetleniem (ale nie jest to jedyny sposób –patrz kolejne slajdy)

Wyświetlamy okno za pomocą metody .Show( )
Okno niemodalne nie zwraca wartości.

Dwustronna komunikacja pomiędzy dwoma oknami (niemodalnymi)

Komunikacja pomiędzy dwoma oknami

Trochę teorii

Ponieważ:

- okna niemodalne nie zwracają wartości i usuwane są z pamięci po ich zamknięciu,
- wywołanie okna niemodalnego metodą .Show() nie zatrzymuje pracy (przetwarzania kodu) okna głównego,

odebranie danych z okna modalnego jest możliwe gdy jest ono wyświetlane.

Wszelkie dane wprowadzone do okna dialogowego w trakcie jego pracy należy przesłać z okna podrzędnego do głównego.

Komunikacja pomiędzy dwoma oknami

PRZEKAZANIE DANYCH Z OKNA GŁÓWNEGO DO PODRZĘDNEGO

W tym momencie jest wywoływane okno podrzędne, ale program główny działa nadal

```
Okno02 noweOkno = new Okno02(this);
noweOkno.Title = "Sterowanie";
noweOkno.Show();

noweOkno.suwak.Value = 20;
```

Aby sięgnąć do kontrolek i zmiennych okna podrzędnego odwołujemy się do jego nazwy (obiektu nie klasy)

Komunikacja pomiędzy dwoma oknami

PRZEKAZANIE DANYCH Z OKNA PODRZĘDNEGO DO GŁOWNEGO

W kodzie okna głównego:

Otwierając okno podrzędne musimy przekazać do niego informację o tym, które okno je wywołało.

W tym celu, w parametrze konstruktora obiektu okno2, umieszczamy this, czyli referencję na obiekt, wewnątrz którego aktualnie jesteśmy

Komunikacja pomiędzy dwoma oknami

PRZEKAZANIE DANYCH Z OKNA PODRZĘDNEGO DO GŁOWNEGO

W kodzie okna podrzędnego:

```
public partial class Okno2 : Form
{
    Form1 okno1;
    public Okno2(Form1 ktoOtworzyl)
    {
        InitializeComponent();
        okno1 = ktoOtworzyl;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        okno1.label1.Text = "tekst przekazany z okna podrzędnego";
        Close();
    }
}
```

Tworzymy referencję na obiekt klasy odpowiadającej oknu nadrzędnemu

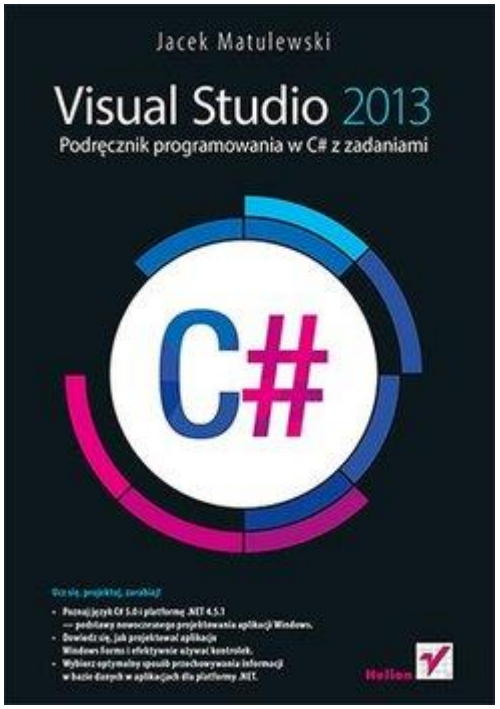
Jako parametr konstruktora dodajemy inną referencję na okno nadrzędne- tu trafi adres wysłany za pomocą this (poprzedni slajd)

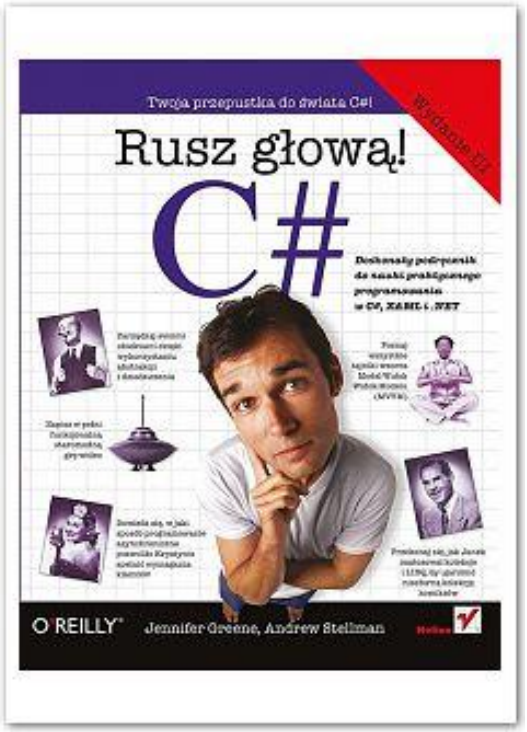
Zapamiętujemy przesłany adres w zmiennej globalnej, żeby móc używać go poza konstruktorem.

Teraz możemy sięgnąć do kontrolek i pól okna nadrzędnego – UWAGA: tylko tych publicznych

Opcjonalnie: zamykamy okno

Literatura:





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