

Metodyka programowania

Visual Studio C#
Formularze

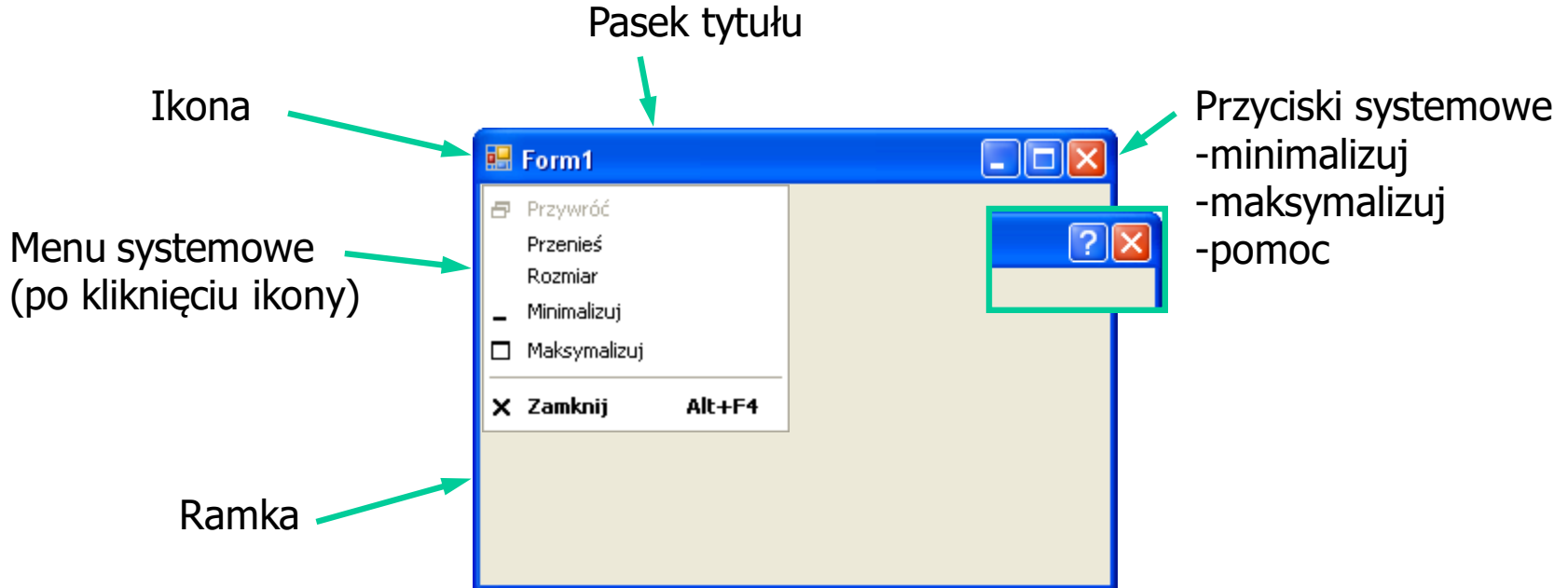
Formularz

Formularz (ang. Form) to przyszłe okno programu

- Nowo utworzony projekt C# typu Windows Forms ma jeden formularz;
- Jest to formularz główny (główne okno) aplikacji:
jego zamknięcie powoduje zakończenie działania programu

Formularz

- Pasek tytułu (wraz z ikoną i przyciskami) i ramka należą systemu operacyjnego, a nie do aplikacji; Nie można zmienić ich wyglądu ani umieścić tam komponentów
- Można je dostosować (np. zmienić ikonę lub rodzaj ramki) lub całkowicie wyłączyć

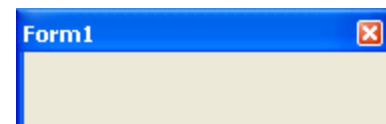
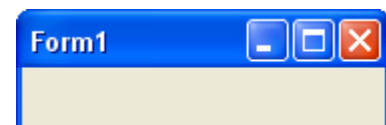
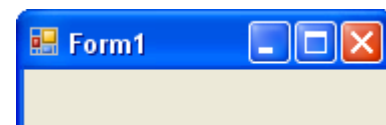


Właściwości formularza

FormBorderStyle

(enum System.Windows.Forms.FormBorderStyle)

- Rodzaj ramki
(kolor i grubość – w ustawieniach systemu!)
- Możliwe ustawienia:
 - **Sizable** - standardowe
 - Fixed3D – stały rozmiar
 - **FixedSingle** – stały rozmiar, ramka bez "3D"
 - **FixedDialog** – typowe dialogowe
 - SizableToolWindow – narzędziowe
 - FixedToolWindow – narzędziowe, stały rozmiar
 - None – brak ramki



Właściwości formularza

ControlBox (Boolean)

- Czy okno ma ikonę z menu systemowym i kontrolki (minimalizuj, maksymalizuj, zamknij)

HelpButton (Boolean)

- Czy wyświetlany jest przycisk pomocy (działa pod warunkiem, że nie ma minimalizuj, maksymalizuj)



Icon

- Ikona (.ico) wyświetlana na pasku tytułu i pasku zadań



MinimizeBox (Boolean)

MaximizeBox (Boolean)

- Czy mają być dostępne na pasku tytułu przyciski minimalizacji oraz maksymalizacji okna (jeżeli nie, to nie działa też klikanie w pasek tytułu)



Właściwości formularza

ShowIcon (Boolean)

- Czy okno ma ikonę z menu systemowym
(nie da się włączyć np. dla Dialog i ToolWindow)

ShowInTaskBar (Boolean)

- Czy okno ma reprezentujący je przycisk na pasku zadań

TopMost

- Zawsze na wierzchu, nawet jeżeli nie jest aktywne
(podobnie jak np. Menedżer Zadań)

Opacity

- Stopień przezroczystości okna
(można wpisywać w procentach)

Właściwości formularza

StartPosition

(enum System.Windows.Forms.FormStartPosition)

- Początkowe położenie okna
- Możliwe ustawienia:
 - Manual – określone przez właściwość Location
 - CenterScreen – środek okna
 - **WindowsDefaultLocation** – domyślne systemu
 - WindowsDefaultBounds – domyślne położenie i rozmiar
 - CenterParent – środek okna rodzica (tylko MDI)
- Alternatywnie można zapisywać położenie i wielkość okna w chwili zamykania aplikacji i odtwarzać te cechy przy następnym uruchomieniu

Właściwości formularza

Location (Point), **Top**, **Left** (Int32)

- Położenie - współrzędne lewego górnego narożnika względem komponentu zawierającego

Size (Point), **Width**, **Height** (Int32)

- Rozmiar

MinimumSize (Size), **MaximumSize** (Size)

- Minimalny i maksymalny rozmiar okna;

AutoSize

- Automatyczne dostosowanie rozmiaru do zawartości

AutoScroll

- Automatyczne wyświetlanie pasków przewijania

Właściwości formularza

Font

- Cechy czcionki, której używa komponent:
 - Name – krój (np. „Bookman Old Style”)
 - Size – rozmiar w punktach
 - Bold, Italic – pogrubienie, kursywa

BackColor

- Kolor tła

ForeColor

- Kolor tekstu
- Cechy te są "dziedziczone" przez komponenty umieszczane na formularzu (w różnym stopniu przez różne rodzaje komp.)

Właściwości formularza

BackgroundImage

- Obrazek wyświetlany jako tło

BackgroundImageLayout

- Sposób jego wyświetlenia
 - Tile - powielanie,
 - Stretch - rozciągnięcie,
 - Zoom - powiększenie

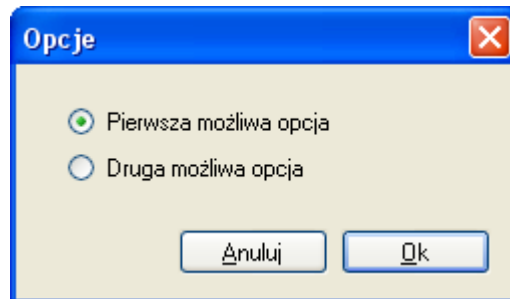
Właściwości formularza

AcceptButton (IButtonControl)

- Przycisk "klikany" przez naciśnięcie <Enter> tzw. domyślny przycisk okna, w oknach dialogowych przycisk "Ok"

CancelButton (IButtonControl)

- Przycisk "klikany" przez naciśnięcie <Esc> w oknach dialogowych przycisk "Anuluj"



Tworzenie formularzy

Nowo utworzony projekt C# typu Windows Forms ma jeden formularz, ale może być ich więcej

- Do projektu można dodawać kolejne formularze (menu: Project / Add Windows Form)
- Domyślnie wyświetlane jest tylko główne okno, dodatkowe okna trzeba utworzyć i wyświetlić programowo;
- Do wyświetlania okna można użyć jednej z dwóch metod:

```
Show();
```

```
ShowDialog();
```

Sposób zachowania okien i ich programowej obsługi jest zależny tego która z nich zostanie użyta

Tworzenie formularzy

Okno niemodalne

- Okno wyświetlone metodą *Show* jest niemodalne (użytkownik może przełączać się między oknem głównym a dodatkowymi oknami niemodalnymi);
- Program nie jest wstrzymywany
 - kolejne (po *Show*) instrukcje są natychmiast wykonywane
- Okno jest zamykane, a nie ukrywane (np. po kliknięciu w znak "x" w rogu okna)
 - zwalniane są wszystkie jego zasoby,
 - generowane są zdarzenia *FormClosing* i *FormClosed*
 - próba ponownego wyświetlenia lub odwołania się do jego komponentów lub właściwości kończy się błędem
 - aby ponownie wyświetlić okno, trzeba je na nowo utworzyć

Tworzenie formularzy

Okno niemodalne

- Sposób wyświetlenia

```
Form2 f2 = new Form2();  
f2.Show();
```

- Do komunikacji między formami trzeba napisać własne, dedykowane metody

Tworzenie formularzy

Okno modalne

- Okno wyświetlone metodą *ShowDialog* jest modalne (ang. modal dialog window; użytkownik nie może przełączać się między aktywnym oknem dialogowym a innymi oknami);
- Program jest wstrzymywany
 - kolejne (po *ShowDialog*) instrukcje są wykonywane dopiero po zamknięciu okna dialogowego
- Okno jest ukrywane, a nie zamykane (np. po kliknięciu w znak "x" w rogu okna)
 - jego zasoby nie są zwalniane,
 - można je wyświetlić ponownie (stan komponentów nie zmienia się w międzyczasie)
 - dostępna je właściwość *DialogResult*
 - należy upewnić się, że jest tworzone tylko raz (a nie przy każdym wyświetleniu)

Tworzenie formularzy

Okno modalne

- Sposób wyświetlenia

```
Form2 f2 = new Form2();  
f2.ShowDialog();
```

- Do komunikacji między formami można napisać własne, dedykowane metody
(np. aby sprawdzić jakie opcje zostały wybrane)
- Alternatywnie można odczytać stan komponentów
 - trzeba mieć pewność, że formularz istnieje
 - modyfikator dostępu komponentów
(na oknie dialogowym)
trzeba zmienić na *internal* lub *public*

Tworzenie formularzy

Rezultat okna modalnego

- Ważnym aspektem okien dialogowych jest ich rezultat (zwykle do wyboru są min. 2 możliwości, np. Ok/Anuluj); do określenia rezultatu służy właściwość DialogResult;
- Nadanie wartości właściwości DialogResult formularza powoduje "zamknięcie" (tak naprawdę ukrycie) okna modalnego

```
this.DialogResult = DialogResult.OK;
```

Tworzenie formularzy

Rezultat okna modalnego

- Jeżeli przycisk, umieszczony na formularzu modalnym, ma przypisaną wartość *DialogResult*, to jego kliknięcie powoduje przypisanie tej wartości do właściwości *DialogResult* formularza, a tym samym jego "zamknięcie"
- Zwykle przyciski *Ok* i *Anuluj* są przyciskami domyślnymi formularza:
 - Ok (albo Zapisz, Otwórz itp.)
AcceptButton okna
DialogResult.OK
 - Anuluj
CancelButton okna
DialogResult.Cancel

Tworzenie formularzy

Rezultat okna modalnego

- Ponieważ wykonanie funkcji wyświetlającej okno modalne zostaje wstrzymane do jego zamknięcia, to odczytanie wartości *DialogResult* zwykle umieszcza się w kolejnej instrukcji
- Niezależnie od wartości *DialogResult*, formularz nic sam nie robi - za to odpowiada program, np.:

```
f2.ShowDialog();  
if (f2.DialogResult == DialogResult.OK)  
{  
    // działania  
}
```

Tworzenie formularzy

Rezultat okna modalnego

- W przypadku okien z opcjami programu, jeżeli wybrano *Anuluj*, to zwykle odtwarzane są opcje sprzed wyświetlenia okna:

```
op1 = ...  
op2 = ...  
fOpt.ShowDialog();  
if (fOpt.DialogResult == DialogResult.Cancel)  
{  
    ... = op1;  
    ... = op2;  
}
```

Tworzenie formularzy

Rezultat okna modalnego

- Bardziej eleganckie rozwiązanie polega na utrzymywaniu pól z wartościami opcji

```
Boolean op = true;
Form2 fOpt = new Form2();

... btnOptions_Click ...
{
    fOpt.ShowDialog();
    if (fOpt.DialogResult == DialogResult.OK)
        op = ...
    else
        ... = op;
}
```

Tworzenie formularzy

Rezultat okna modalnego

- Można też zadbać o to, aby okno nie było tworzone bez potrzeby (aplikacja mniej obciąża system)

```
Form2 fOpt = null;
```

```
... btnOptions_Click ...
```

```
{
```

```
    if (fOpt == null)
```

```
        Form2 fOpt = new Form2();
```

```
    fOpt.ShowDialog();
```

```
    ...
```

```
}
```

Tworzenie formularzy

Dostęp do komponentów okna dialogowego

- Domyślnie komponenty mają modyfikator dostępu ***private***, co oznacza że nie są dostępne z innego formularza

```
Form2 fOpt;
```

```
... btnDoIt_Click ...
```

```
{
```

```
    if (fOpt.radioButton1.Checked)
```

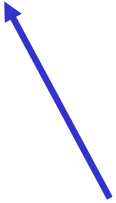
```
    {
```

```
        // ...
```

```
    }
```

```
    ...
```

```
}
```



Błąd kompilacji
(*Error: 'radioButton1' is inaccessible
due to its protection level*)

Tworzenie formularzy

Dostęp do komponentów okna dialogowego

- Najprostszym rozwiązaniem jest zmiana modyfikatora dostępu na ***internal*** (dostęp możliwy z innych klas należących do tego samego zestawu) lub ***public***
- Bardziej eleganckie rozwiązanie polega na zdefiniowaniu
 - metod (jednej do odczytu, drugiej do ustawienia)
 - właściwości (do odczytu i zapisu)

Tworzenie formularzy

Definicja metod:

```
internal Boolean getOption1()  
{  
    return radioButton1.Checked;  
}  
internal void setOption1(Boolean on)  
{  
    radioButton1.Checked = on;  
}
```

Metody są elementami klasy formularza, na równi z komponentami, mają zatem nieograniczony dostęp do wszystkich komponentów formularza i ich właściwości.

Tworzenie formularzy

Składnia metody

```
[Modyfikator] Typ Nazwa (Lista_argumentów)
{
    Ciało_metody
}
```

- *Modyfikator* – domyślnie modyfikatorem jest *private*; należy użyć *internal* lub *public*
- *Typ* – określa typ rezultatu; "void" jeżeli metoda nie dostarcza rezultatu
- *Lista_argumentów* – lista nazw i typów rozdzielonych przecinkami (np.: *Int32 i, Boolean b, String s*); może być pusta
- *Ciało_metody* – dowolne instrukcje; jeżeli *Typ* jest inny niż *void*, powinna kończyć się instrukcją **return**, zwracającą wartość zgodną z *Typem*

Tworzenie formularzy

Użycie metody

- Metoda jest wywoływana poprzez obiekt formularza, do którego należy

```
Form2 fOpt;
```

```
... btnDoIt_Click ...  
{  
    if ( fOpt.getOption1() )  
        // ...  
}  
... btnOptions_Click ...  
{  
    ...  
    fOpt.setOption1(opt1) )  
}
```

Tworzenie formularzy

Definicja właściwości :

```
internal Boolean Option1
{
    get
    {
        return radioButton1.Checked;
    }
    set
    {
        radioButton1.Checked = value;
    }
}
```

Tworzenie formularzy

Składnia właściwości

```
[Modyfikator] Typ Nazwa
{
    get { Sekcja_get }
    set { Sekcja_set }
}
```

- *Modyfikator* – domyślnie modyfikatorem jest *private*; należy użyć *internal* lub *public*
- *Typ* – określa typ właściwości
- *Sekcja_get* – musi zwrócić wartość właściwości zgodną z *Typem* (przy pomocy instrukcji `return`)
- *Sekcja_set* – otrzymuje niejawny argument o nazwie **value**, którego typ jest identyczny jak *Typ* właściwości, który powinna (choć nie musi) zapisać
- Jedną z sekcji można pominąć

Tworzenie formularzy

Składnia właściwości

```
[Modyfikator] Typ Nazwa  
{  
    get { Sekcja_get }  
}
```

- Jedną z sekcji można pominąć
jeżeli np. będzie tylko sekcja get, to uzyskamy
właściwość "tylko do odczytu"

Tworzenie formularzy

Użycie właściwości

- Właściwość jest używana poprzez obiekt formularza, do którego należy (podobnie, jak metoda)

```
Form2 fOpt;
```

```
... btnDoIt_Click ...
```

```
{
```

```
    if ( fOpt.Option1 )
```

```
        // ...
```

```
}
```

```
... btnOptions_Click ...
```

```
{
```

```
    ...
```

```
    fOpt.Option1 = opt1;
```

```
}
```