

C# Windows Forms Applications

Trzy filary:

- OOP – programowanie obiektowe (ang. Object Oriented Programming)
- EDP – programowanie sterowane zdarzeniami (ang. Event Driven Programming)
- RAD – szybkie tworzenie interfejsu GUI (ang. Rapid Application Development)

OOP

Biblioteka komponentów .NET

- Gotowe komponenty (przyciski, pola edycyjne pola wyboru, listy, menu, paski narzędzi, okna dialogowe itd.)
- Stabilne działanie
- Znane użytkownikom
- Wyposażone w liczne właściwości, metody i zdarzenia, co umożliwia ich dostosowanie do potrzeb i reagowanie na działanie użytkownika

EDP

Aplikacje z GUI są sterowane zdarzeniami

- Zdarzenia generowane przez użytkownika (np. poruszenie myszką, kliknięcie, wciśnięcie klawisza, ...) są wykrywane przez system operacyjny; System wysyła do aplikacji komunikaty o tych zdarzeniach
- Aplikacja utrzymuje pętlę obsługi komunikatów systemowych; odbiera komunikaty i kolejno je obsługuje
- Poszczególne komponenty są informowane o zdarzeniach, które się do nich odnoszą;

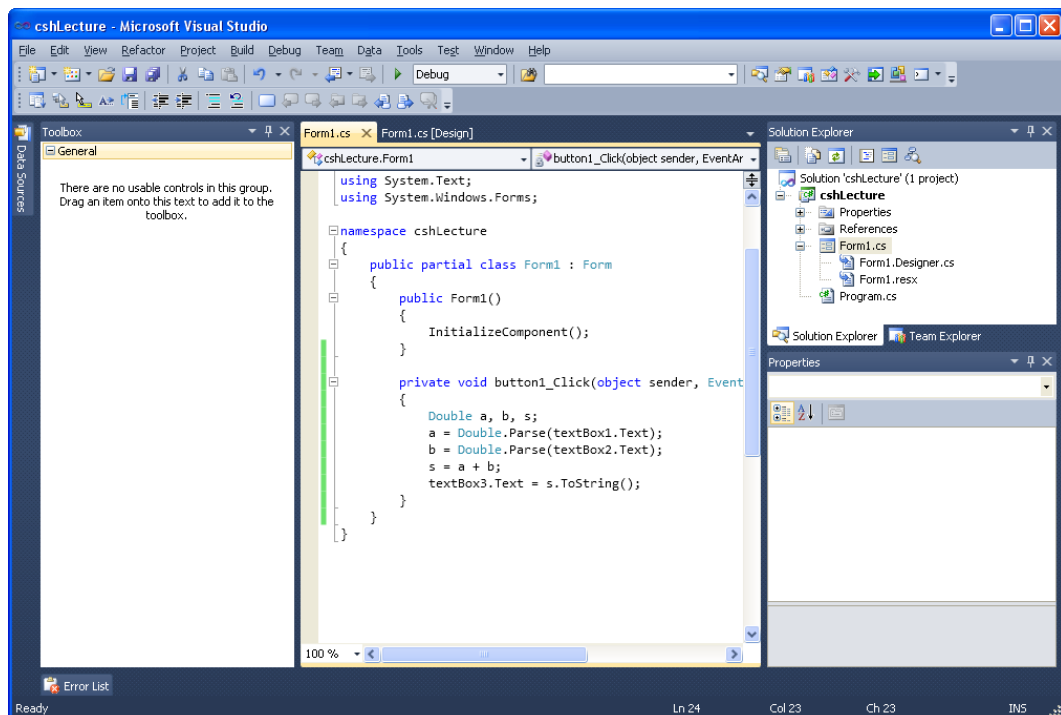
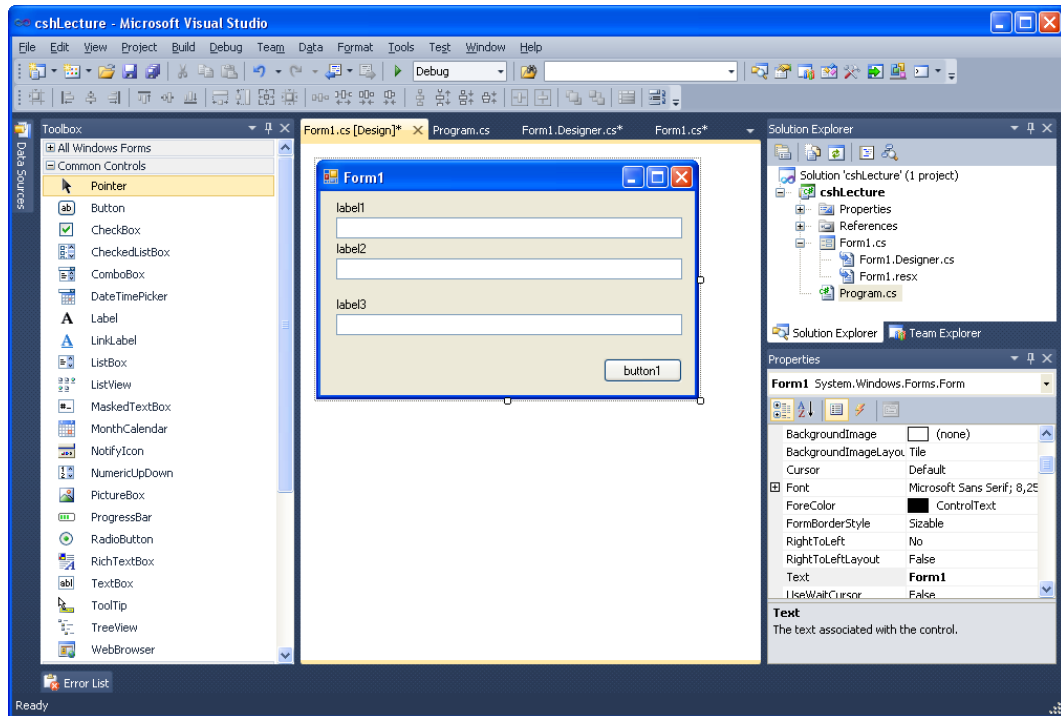
Rodzaje zdarzeń:

- Mysz:
MouseDown, MouseUp, MouseEnter, MouseMove, MouseHover, MouseLeave, MouseWheel
- Klawiatura:
KeyDown, KeyUp, KeyPress
- Akcje:
Click, MouseClick, DoubleClick, MouseDoubleClick, Scroll, ResizeBegin, ResizeEnd
- Fokus:
Enter, Leave, Validating, Validated, Activated, Deactivated
- Drag&Drop:
DragEnter, DragOver, DragLeave, DragDrop, GiveFeedback, QueryContinueDrag
- Cykl życia:
ControlAdded, ControlRemoved, Load, Shown, FormClosing, FormClosed, HelpRequested
- Zmiana właściwości:
BackColorChanged, ForeColorChanged, FontChanged, SizeChanged, TextChanged, ParentChanged, VisibleChanged, EnabledChanged

RAD

Środowisko programistyczne (IDE) wspomaga tworzenie GUI

- Umożliwia projektowanie GIU myszką (układanie komponentów na formularzu okna programu)
- Umożliwia zmianę właściwości komponentów (położenie, rozmiar, wyświetlany tekst, czcionka, kolory, ...)
- Tłumaczy obraz formularza na kod, który takie utworzy okno programu
- Tworzy funkcje obsługi zdarzeń i zapisuje powiązanie tych funkcji z komponentami



Podstawowe komponenty

Wspólne właściwości komponentów

Name

- Nazwa, pod jaką komponent jest widziany w programie
- Domyślnie: *nazwaKlasy* + numer, np. `button3`, `label1`, `textBox5` itp.
- Można zmienić – obowiązują zwykłe zasady dla identyfikatorów

Text (String)

- Tekst wyświetlany przez komponent, przeznaczenie zależne od komponentu:
 - Opis komponentu (np. `Button`, `CheckBox`)
 - Tekst edytowany przez użytkownika (np. `TextBox`, `RichTextBox`)
- Znak `&` (ampersand) tworzy skrót klawiaturowy `<Alt>+<Litera>`
 - np. `&Start` → Start → skrót `<Alt>+S`

TextAlign

- Wyrównanie tekstu
Tekst edytowalny: w poziomie
Tekst nieedytowalny: w poziomie i w pionie

TextImageRelation

- Położenie tekstu względem obrazka

Image

- Obrazek wyświetlany na komponentcie (obok lub zamiast tekstu)

ImageAlign

- Wyrównanie obrazka, w poziomie i w pionie

Visible (Boolean)

- Czy komponent jest widoczny

Enabled (Boolean)

- Czy użytkownik może z komponentu korzystać

Font

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

ForeColor

- Kolor tekstu

BackColor

- Kolor tła

Location (Point), Top, Left (Int32)

- Położenie - współrzędne lewego górnego narożnika względem komp. zawierającego

Size (Point), Width, Height (Int32)

- Rozmiar

Anchor

- Sposób zakotwiczenia – wskazuje krawędzie, które pozostaną nieruchome względem krawędzi komponentu zawierającego (domyślnie – krawędzie lewa i górna)

Zmiana właściwości komponentów

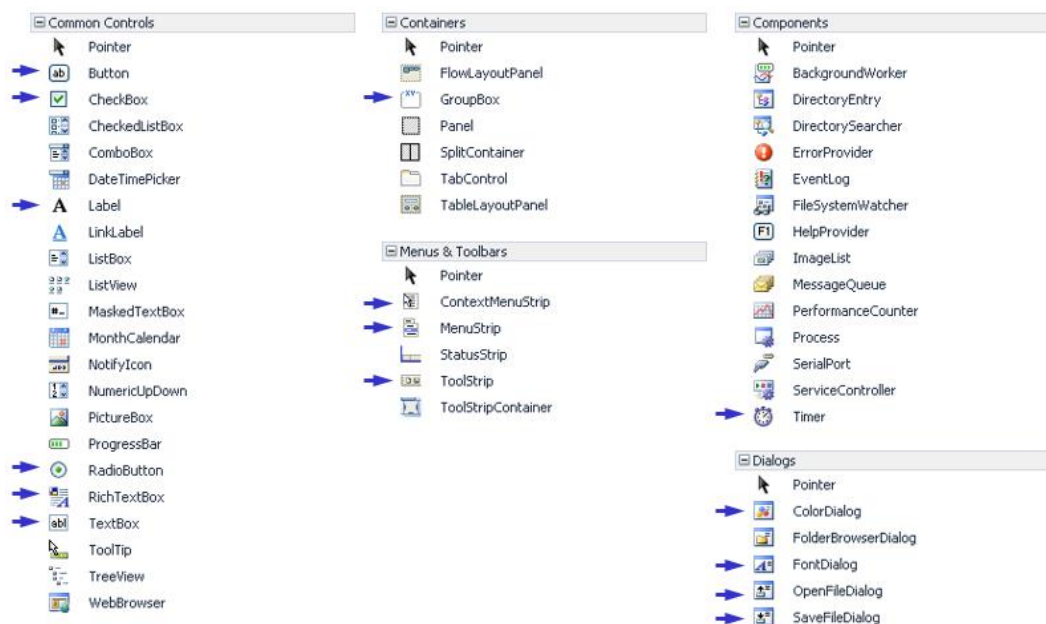
Właściwości można modyfikować:

- w inspektorze obiektów (właściwości na moment uruchomienia programu)
- programowo:
`textBox1.Text = "&Stop";`
`button1.Enabled = False;`

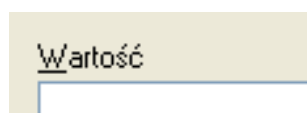
Właściwości można odczytywać

```
bool e = button1->Enabled;  
if (!label2.Visible)  
    label1.Visible = True;  
String s = textBox3.Text;
```

Komponenty



Label – etykieta



Statyczny tekst, opisuje inne komponenty

- Skróót <Alt>+<Litera> aktywuje komponent, którego TabIndex jest o jeden większy, niż etykiety (komponent dodany do formularza zaraz po etykiecie)
- Właściwości:
 - **AutoSize** – czy rozmiar ma zostać automatycznie dostosowany do zawartości

TextBox – pole edycyjne



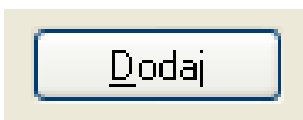
Do wczytywania i wyświetlania danych

- Właściwości:
 - **Multiline** – czy może zawierać więcej niż jedną linię tekstu
 - **MaxLength** – maksymalna długość tekstu
 - **ReadOnly** – tylko do odczytu
 - **WordWrap** – zawijanie tekstu
 - **ScrollBars** – paski przewijania (domyślnie brak)
 - **PasswordChar** – znak maskujący tekst
- Zdarzenia:
 - **TextChanged** – informuje o każdej zmianie tekstu
(nie można zmieniać właściwości Text – nieskończona pętla!)

Jak korzystać z właściwości Text

- Do wczytywania danych:
Double x;
Boolean ok;
x = Double.Parse(textBox.Text);
ok = Double.TryParse(textBox.Text, out x);
- Do wyświetlania wyników:
textBox.Text = x.ToString();
textBox.Text = "Dowolny tekst";
textBox.Text = "Suma:" + s;
textBox.Text = String.Format("Liczby: {0}, {1} oraz {2}");

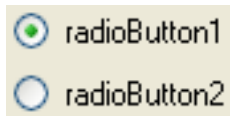
Button – przycisk



Inicjuje określone akcje programu

- Właściwości:
 - **DialogResult** – rezultat zwracany przez okno dialogowe
- Zdarzenia:
 - **Click** – kiedy przycisk zostanie kliknięty (kliknięcie myszką **lub** <Enter> **lub** <Alt>+<Litera> **lub** <Tab> <Spacja>)

RadioButton – przycisk radiowy

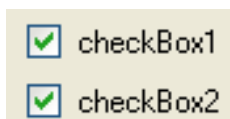


Do wyboru opcji wzajemnie wykluczających się

- Właściwości:
 - **Checked** – wskazuje, czy przycisk jest zaznaczony
 - **CheckAlign** – położenie symbolu zaznaczenia
- Zdarzenia:
 - **CheckedChanged** – kiedy właściwość Checked zmienia wartość

Jeżeli potrzebne są oddzielne grupy przycisków radiowych, należy umieścić je w kontenerze, np. `GroupBox`

CheckBox – pole wyboru



Do wyboru opcji niezależnych od siebie

- Właściwości:
 - **Checked** – wskazuje, czy przycisk jest zaznaczony
 - **CheckState** – Checked, Unchecked, Indeterminate
 - **ThreeState** – czy jest trzystanowy (dla użytkownika)
 - **CheckAlign** – położenie symbolu zaznaczenia
- Zdarzenia:
 - **CheckedChanged** – kiedy właściwość Checked zmienia wartość

Timer – czasomierz



Odmierza odstępy czasu

- Właściwości:
 - **Interval** – odstęp czasu między "tyknięciami" [ms]; rozdzielczość zależy od platformy sprzętowej, typowo $1/64 \text{ s} = 15,625 \text{ ms}$ albo $1/100 \text{ s} = 10 \text{ ms}$
 - **Enabled** – "tyka" / "nie tyka"
- zdarzenia:
 - **Tick** – po każdym "tyknięciu" (nieobsłużone tyknięcia nie kumulują się!)

Zadania

1. Komponenty i ich wbudowana obsługa zdarzeń
 - Utworzyć nowy projekt, język C#, typ projektu Windows Forms
 - Umieścić na formularzu po 2-3 komponenty: Label, TextBox, RichTextBox, Button, RadioButton, CheckBox Timer oraz inne dowolnie wybrane z przybornika;
Nie należy tworzyć funkcji obsługi zdarzeń ani zmieniać właściwości komponentów
 - Uruchomić aplikację, sprawdzić działanie komponentów: edycja tekstu i operacje kopiuj-wytnij-wklej TextBoxa, wybór opcji przez RadioButton i CheckBox, itp.
2. „Hello World!” – funkcje obsługi zdarzeń i zmiana właściwości komponentów



- Usunąć wszystkie komponenty (albo utworzyć nowy projekt)
 - Dodać Label, zmienić tekst na „Hello”, czcionkę na dużą i kolorową, Visible na false
 - Dodać Button, zmienić tekst na „Hello”, dodać skrót klawiaturowy, np. <Alt>+H
 - Utworzyć funkcję obsługi zdarzenia Click przycisku, wpisać kod, który spowoduje wyświetlenie (wcześniej ukrytej) etykiety
3. „Hello World!” v2
 - Zmienić program z zadania 2, aby kolejne kliknięcia przycisku na zmianę wyświetlały i ukrywały etykietę
 4. „Hello World!” v3
 - Do formularza z zadania 2 dodać trzy przyciski, opisane „Red”, „Green”, „Blue”, wyposażyć je w odpowiednie skróty klawiaturowe
 - Utworzyć funkcje obsługi zdarzenia Click przycisków, dodać kod który zmienia kolor tła formularza, np. `this.Color = Color.Red`
 - Sprawdzić działanie programu, w tym możliwość „kliknięcia” przycisków za pomocą klawiatury (<Alt>+litera oraz <Tab>, <Space>)

5. „Hello World!” v4

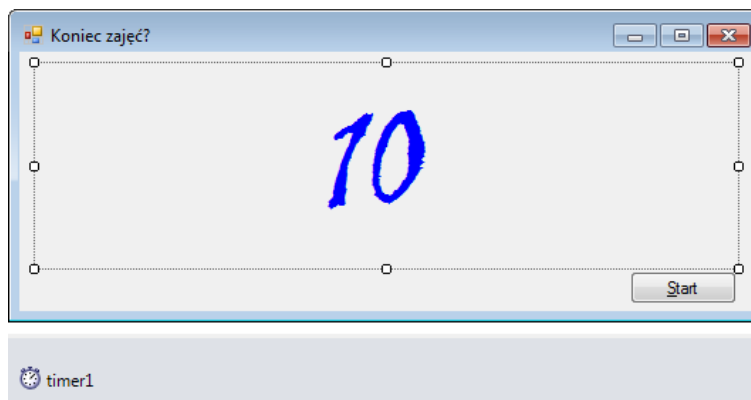


- Do formularza z zadania 4 dodać trzy komponenty RadioButton, opisane „Red”, „Green”, „Blue”, wyposażyć je w odpowiednie skróty klawiaturowe; Jeden z nich powinien być zaznaczony („wybrany”) – należy też adekwatnie zmienić kolor tekstu etykiety
- Dodać przycisk „Apply” (zastosuj), ze skrótem klawiaturowym i funkcją obsługi zdarzenia, która zmienia kolor tekstu etykiety „Hello” na wskazany przez aktualnie zaznaczony RadioButton

6. „Hello World!” v5

- Zmienić program z zadania 2, aby nie dało się ustawić tego samego koloru tekstu i tła; Należy dodać etykietę z komunikatem o błędzie, domyślnie niewidoczną, pokazywaną tylko kiedy kolor tekstu jest taki sam jak kolor tła
- Sprawdzić starannie działanie programu dla różnych scenariuszy zmiany kolorów, poprawić ewentualne błędy

7. „Koniec zajęć” – wykorzystanie komponentu Timer, zmienne „globalne”



- Utworzyć nowy projekt, język C#, typ projektu Windows Forms
- Dodać komponenty: Label (tekst „10”, czcionka duża i kolorowa), Button (tekst „Start”, odpowiedni skrót klawiaturowy) oraz Timer (Interval ustawić na 1 s)
- Dodać funkcje obsługi zdarzeń Click przycisku i Tick czasomierza; Kliknięcie przycisku ma uruchomić czasomierz, a każde tyknięcie czasomierza powinno zmniejszać wyświetlany przez etykietę czas pozostały do końca zajęć; Zamiast wartości 0 powinien pojawić się napis „Koniec” (i tak ma już zostać)
- Przetestować program (w szczególności drugie kliknięcie przycisku, po odliczeniu czasu do zera), poprawić ewentualne błędy