Programowanie obiektowe

Cel zajęć. Celem zajęć jest zapoznanie z praktycznymi aspektami tworzenia aplikacji okienkowych w C#.

Wprowadzenie teoretyczne. Rozważana w ramach niniejszych zajęć tematyka jest ważna, gdyż w praktyce większość komercyjnych aplikacji desktopowych jest utworzonych w oparciu o okna. Aby ze zrozumieniem zrealizować zadania, przewidziane do wykonania w ramach zajęć laboratoryjnych, należy znać znaczenie takich zagadnień jak: delegacje i zdarzenia, tworzenie aplikacji okienkowych w środowisku Visual Studio.

1. Delegacje i zdarzenia

Kliknięcie przycisku, przejechanie kursorem myszy ponad formularzem czy naciśnięcie klawisza Enter to **zdarzenia**, które mogą spowodować wywołanie jednej lub wielu określonych funkcji odpowiedzialnych za obsługę zdarzeń w ramach programu.

W .NET zdarzenia są pełnoprawnymi składowymi klas. Np. klasa „Control", będąca klasą bazową dla komponentów graficznego interfejsu użytkownika zawiera zdarzenia, których wystąpienie można obsłużyć. Za przykład może posłużyć zdarzanie „Click" oznaczające kliknięcie danej kontrolki (np. przycisku).

Za łączenie zdarzeń z odpowiednią metodą lub metodami je obsługującymi odpowiada obiekt **delegacji**. Obiekt ten musi utrzymywać listę metod wywoływanych w odpowiedzi na wystąpienie danego zdarzenia. Delegacja może wywołać tylko te metody, których sygnatura i zwracany typ odpowiadają sygnaturze i zwracanemu typowi określonemu w deklaracji delegacji.

Dla każdej napotkanej deklaracji delegacji kompilator tworzy klasę z modyfikatorem sealed, której nazwa jest identyczna jak nazwa delegacji. Klasa ta definiuje pojedynczy konstruktor, który jako parametr przyjmuje nazwę metody.

Pojedyncza delegacja może wywołać kilka metod obsługujących zdarzenia. Każda z tych metod musi zostać zarejestrowana w delegacji. Poniższy przykład ilustruje sposób działania delegacji:

Przykład 1 – Sposób działania delegacji

```
16      public delegate void MyString(string s);
17
18      public static void PrintLower(string s)
19      {
20          Console.WriteLine(s.ToLower());
21      }
22
23      public static void PrintUpper(string s)
24      {
25          Console.WriteLine(s.ToUpper());
26      }
27
28      static void Main(string[] args)
29      {
30          MyString myDel;
31          myDel = new MyString(PrintLower);
32          myDel += PrintUpper;
33
34          myDel("Programowanie obiektowe");
35
36          Console.ReadKey();
37      }
```

```
programowanie obiektowe
PROGRAMOWANIE OBIEKTOWE
```

Operator += dodaje metodę do listy wywołań. Metodę można usunąć z listy wywołań za pomocą operatora -=.

2. Obsługa zdarzeń z wykorzystaniem delegacji

Za obsługę zdarzenia kliknięcia kontrolki odpowiada predefiniowane zdarzenie „Click". Aby powiązać wystąpienie takiego zdarzenia z wywołaniem odpowiedniej metody, należy użyć predefiniowanej delegacji „EventHandler". Rozwiązanie to prezentuje poniższy przykład:

Przykład 2 – Obsługa zdarzenia

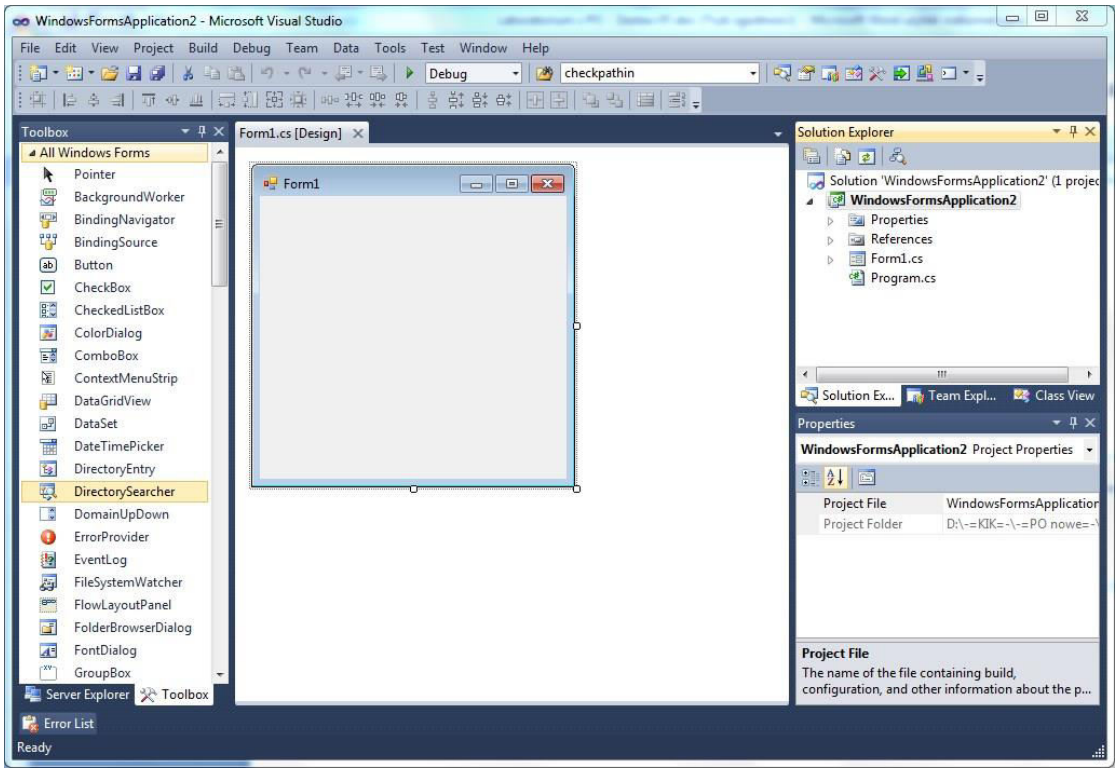
```
this.button1 = new System.Windows.Forms.Button();
ustawienie wlasciwosci przycisku
this.button1.Click += new System.EventHandler(this.button1_Click);

    //metoda wywołana na skutek wystąpienia zdarzenia
    private void button1_Click(object sender, EventArgs e)
    {
        //...
    }
```

Metoda „button1_Click" zostanie wywołana, gdy wystąpi zdarzenie „Click" znajdujące się w kontrolce „button1" (przycisk „button1" zostanie kliknięty).

3. Krótkie wprowadzenie do pracy z WindowsForms

Aby utworzyć projekt przeznaczony do implementacji aplikacji okienkowej, w środowisku Visual Studio należy utworzyć projekt typu „Windows Forms Application". Nowo utworzony projekt wygląda następująco:



Rys. 1 – Projekt okienkowy w Visual Studio

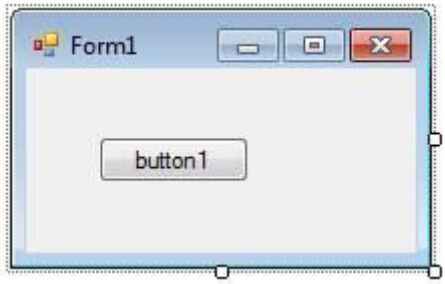
W głównym oknie projektu znajduje się domyślnie utworzona kontrolka formatki (obiekt typu „Form"), reprezentująca okno aplikacji. Kontrolka ta jest zdefiniowana w pliku „Form1.cs", który można zobaczyć w oknie „Solution Explorer".

Domyślnie formatka jest widoczna w trybie graficznego projektowania - „Design". Aby przejść do struktury kodu formatki, należy nacisnąć klawisz „F7" lub w menu „Solution Explorer" kliknąć na plik „Form1.cs" prawym klawiszem myszy i wybrać opcję „View Code".

Aby dodać do formatki kontrolkę korzystając z trybu „Design", należy wybrać odpowiednią kontrolkę z menu „Toolbox" i umieścić ją na powierzchni formatki.

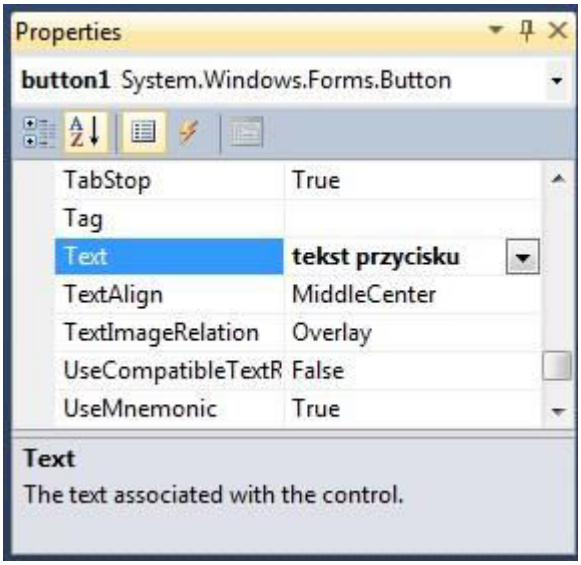
4. Ustawianie właściwości kontrolek

Jeśli umieścimy na formatce przycisk (kontrolka typu „Button"), formatka może wyglądać następująco:



Rys. 2 – Formatka z kontrolką „Button"

Aby zmienić właściwości przycisku, należy skorzystać z menu „Properties". Przykładowo aby zmienić tekst wyświetlany na przycisku, należy zaznaczyć przycisk i w menu „Properties" wpisać odpowiednią wartość właściwości „Text":

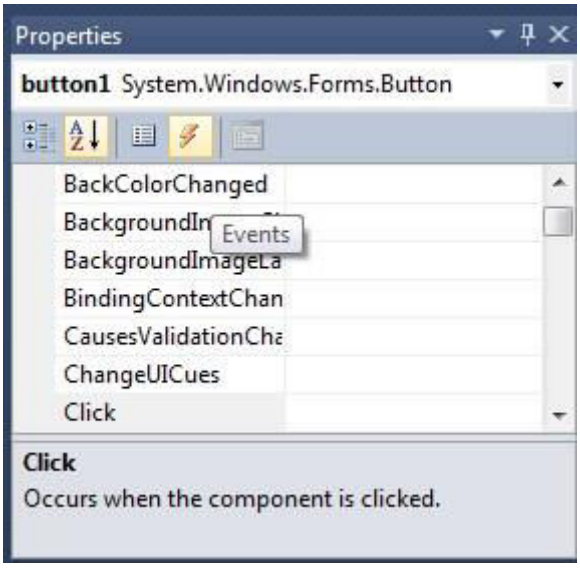


Rys. 3 – Zmiana właściwości „Text"

Tym sposobem można ustawiać także inne właściwości kontrolek.

5. Obsługa zdarzenia kontrolki

Aby obsłużyć zdarzenie „Click" wstawionego przycisku, należy zaznaczyć przycisk na formatce i w menu „Properties" wybrać ikonę z żółtym piorunem.

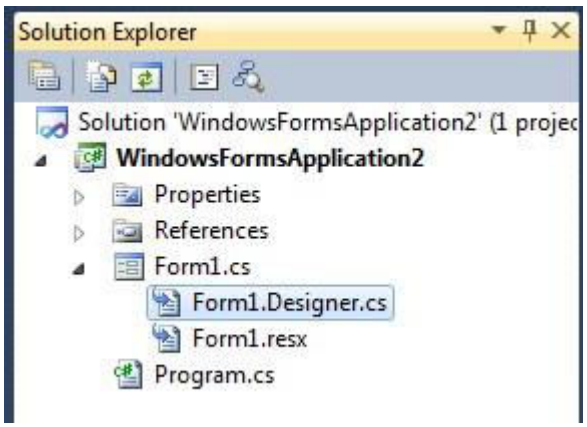


Rys. 4 – Okno właściwości – zdarzenia

Następnie należy kliknąć dwukrotnie interesujące nas zdarzenie, w tym wypadku będzie to zdarzenie „Click" (można także obok nazwy zdarzenia wpisać nazwę metody i nacisnąć przycisk „Enter"). Po wykonaniu tej operacji zostanie utworzona nowa metoda, która zostanie wywołana w momencie wystąpienia danego zdarzenia. Do programisty należy implementacja działania tej metody.

6. Plik *.Designer.cs

Czytając informacje zawarte w dwóch powyższych podpunktach można odnieść wrażenie, że operacje te nie przypominają pracy z poznanymi na poprzednich zajęciach laboratoryjnych właściwościami klas oraz pracy z delegacjami i zdarzeniami opisanymi powyżej. Okazuje się jednak, że każda operacja wykonana w trybie „Design" powoduje utworzenie przez środowisko Visual Studio odpowiedniego kodu aplikacji. Jeśli w oknie „Solution Explorer" zostaną rozwinięte składowe pliku „Form1.cs", okaże się, że w skład aplikacji wchodzi plik „Form1.Designer.cs":



Rys. 5 – Menu „Solution Explorer" – plik *.Designer.cs

Po dwukrotnym kliknięciu na ten plik zostanie otworzone okno zawierające jego treść, zawierającą kod odpowiadający wszystkim operacjom wykonanym w trybie „Design".

Można zauważyć, że klasa „Form1" zawarta w plikach „Form1.cs" i „Form1.Designer.cs" oznaczona jest modyfikatorem „partial" – informuje on o tym, że kod klasy może być rozdzielony do kilku plików. Klasa ta musi znajdować się w tej samej przestrzeni nazw.

7. Metoda Main

Metoda „Main" w projekcie okienkowym znajduje się domyślnie w pliku „Program.cs". Domyślnie powoduje ona uruchomienie głównego okna aplikacji.

Zadanie 1. Proszę zrealizować aplikację okienkową, która powinna odznaczać się następującymi cechami:

- Aplikacja ma stanowić kalkulator podobny do kalkulatora zainstalowanego domyślnie w systemie Windows.
- Przykładowy układ kontrolek na formatce przedstawia poniższy rysunek:

Wskazówki do realizacji:

- Na powyższej formatce umieszczono kontrolki typu „Button" i „TextBox".
- Do ustawiania i pobierania tekstu wyświetlanego w kontrolce „TextBox" służy właściwość „Text".
- Aby dokonać konwersji typu string do typu double należy skorzystać z konstrukcji „Convert.ToDouble(<ciąg_znaków>)".