Zadanie 0

Otwórz projekt gry w kółko i krzyżyk napisany na poprzednich zajęciach. Do projektu dodaj nowy projekt - aplikację graficzną (Windows Form Application). Zrób to w sposób pokazany na poniższym rysunku.

| | So | lution Explorer | | | | | | | |
|-----------|---|-----------------|-------------------------|----------------------------|---------------------|-------------|----------------------------|---------------------|--------|
| | n na statistica de la constatistica de la constatistica de la constatistica de la constatistica de la constatis | | | Recent Templates | | .NET Fr | amework 4 🛛 🔽 Sort b | /: Default | |
| | | | Installed Templates | | | | | ^ T | |
| | | Solution 'kółko | i krzyżyk' (2 projects) | Visual C# | ^ | ⊂¢≉ | Windows Forms Application | Visual C# | |
| ** | Build Solution | F6 | | Windows | | | | | A W |
| | Rebuild Solution | | BS | Web | | | WPF Application | Visual C# | |
| | | | ces | Office | | 64 | | | |
| | Clean Solution Project Build Order | | ~~ | Cloud | | - di | Console Application | Visual C# | |
| | | | | 🗈 Database | - | ¥- | | | |
| | Add | • | New Project | Online Templates | × | C# | Silverlight Class Library | Visual C# | _ |
| | Call Charles Durate the | | | Online reinplaces | | | · · | | ~ |
| | Set Startup Projects | | Existing Project | Name: | kikGUI _l | | | | |
| 1 | Add Solution to Source Control | | New Web Site | Location: | C:\Documents and | d Settings' | Adam\Moje dokumenty\Visual | Studio 2010\Proje 🗸 | Brc |
| 199 | Dache | CERTER | Existing Web Site | | | | | | |

Ustaw nowy projekt, jako ten domyślny do uruchomienia.



Dodaj assemblację projektu KolkoiKrzyzyk (nazwa wcześniejszego projektu). Zrób to poprzez dodanie w pliku Form1.cs projektu WindowsFormApplication1 poniższej linii:

using kikCore;

Nie zapomnij o dodaniu referencji w projekcie **WindowsFormApplication1** do projektu **KolkoiKrzyzyk**.



Dodatkowo przy niektórych elementach klas będzie trzeba zmienić identyfikator dostępu na public.

Zadanie 1

Na formatce umieść 9 przycisków. Nazwij je odpowiednio **button1**, **button2**, ...,**button2**. Możesz to zrobić za pomocą kreatora, tak jak przedstawiono na rysunku.

| 🛐 Form1.cs [Design]* 🗙 | - | Solution Explorer 🗾 🔻 🕂 🗙 | |
|--|-------------------------------------|---|--|
| Deter Explorer Colors | | Solution 'kółko i krzyżyk' (3 projects) KikCore Kikoore KikCore KikCore KikCore KikC | |
| | button3 System.Windows.Forms.Button | | |
| Error List | ∄ 2↓ 🗉 🗲 🖻 | | |
| 😮 0 Errors 🛛 🔔 0 Warnings 🔹 0 Messages | (ApplicationSettings) | | |
| Description V F | Line Column Project | E (DataBindings) | |
| | | (Name) buttonPole3 | |

Zadanie 2

Dodaj obsługę zdarzenia wciśnięcia jednego z 9 przycisków. W celu dodanie uchwytu obsługującego zdarzenie wciśnięcia pierwszego przycisku skorzystaj z kreatora. Ten sam uchwyt będzie wykorzystywany do obsługi wciśnięcia pozostałych przycisków. Najpierw zacznij od obsługi uchwytu dla przycisku nr 1 (buttonPole0). Za pomocą kreatora, przejdź do właściwości kontrolki. Następnie przejdź do zakładki z symbolem błyskawicy. W zakładce tej są wszystkie możliwe zdarzenia związane z daną kontrolką. Metodę, która zostanie wykonana po wciśnięciu przycisku nazwij **button_Click**



Klikając w kreatorze na pole z metodą **button_Click** lub na pierwszy przycisk, zostajemy przeniesienie do pliku z kodem metody. Teraz metoda jest pusta. Po wciśnięciu prawego przycisku (patrz rysunek poniżej) rozwija się menu, z którego możemy wybrać opcję **Find All References**. Po jej wybraniu widzimy wszystkie fragmenty kodu, w których metoda ta jest wykonywana. Możemy dzięki temu otworzyć plik generowany przez kreator i widzimy jak dla obiektu **button1** zostaje dokonana subskrypcja zdarzenia wciśnięcia przycisku.

| ι Γ | private void buttonPol | <pre>private void buttonPole Click(object sender. EventArgs e)</pre> | | | | | |
|--------|---|--|---------------------------|----------------|-----|--|--|
| ÿrve | { | == | View Designer | Shift+F7 | | | |
| r Exp | 3 | | Refactor | • | | | |
| olorei | • [} ' | | Organize Usings | • | | | |
| 8 | L} | 8 | Create Unit Tests | | | | |
| d. | | | Generate Sequence Diagram | | | | |
| ollox | 100 % 👻 < | E), | Insert Snippet | Ctrl+K, X | | | |
| | Error List | 9 | Surround With | Ctrl+K, S | | | |
| | 🔕 0 Errors 🛛 🔔 0 Warnings 🗐 🚺 0 Message | <u>a</u> | Go To Definition | F12 | | | |
| | Description | | Find All Refer | Ctrl+K, R | Lin | | |
| | | . | View Call Hierarchy | Ctrl+K, Ctrl+T | | | |
| | 💫 Error List 🥻 Find Symbol Results | | Breakpoint | • | | | |
| Ready | | ≯≣ | Run To Cursor | Ctrl+F10 | | | |
| | | | Cut | Ctrl+X | | | |

Taką samą subskrypcję należy dodać dla pozostałych przycisków. Wystarczy skopiować linijkę tekstu:

this.button1.Click += new System.EventHandler(this.button_Click);

i wkleić ją w odpowiednim miejscu dla pozostałych obiektów. Linijkę należy dodatkowo zmodyfikować. Podobny efekt można osiągnąć za pomocą kreatora, wpisując nazwę metody **buttonPole_Click** dla zdarzenia wciśnięto przycisk. Ręczna edycja pliku kreatora jest jednak szybsza i pozwoli na zrozumienie jego działania.

```
// buttonPole0
//
this.buttonPole0.Location = new System.Drawing.Point(12, 12);
this.buttonPole0.Name = "buttonPole0";
this.buttonPole0.Size = new System.Drawing.Size(75, 49);
this.buttonPole0.TabIndex = 0;
this.buttonPole0.Text = "button1";
this.buttonPole0.UseVisualStyleBackColor = true;
this. System.Windows.Forms.Button Form1.buttonPole0 er(this.buttonPole_Click);
//
// buttonPole1
//
this.buttonPole1.Location = new System.Drawing.Point(93, 12);
```

Uruchom i sprawdź, czy program działa.

Zadanie 3

W klasie Form1 (klasie okna z grą) dodaj 2 pola:

- Obiekt klas plansza o nazwie szachownica
- Obiekt klasy gracz o nazwie aktGracz

W konstruktorze klasy Form1 utwórz oba obiekty.

Do klasy Form1 dodaj metodę **PokazPlansze**. Metoda ta odczytuje z obiektu szachownica (klasy plansza) zaznaczone ruchy i w odpowiedni sposób prezentuje je na przyciskach.

Czynność taką trzeba wykonać dla wszystkich przycisków. Dodatkowo należy ulepszyć kod, np zmieniając kolor przycisku, wstawiając odpowiednią grafikę lub zmieniając opis przycisku na ' ', 'X', 'O'.

W tym celu najlepiej napisać metodę:

void prezentujPole(Button przycisk, ruch wartosc)

Metoda ta następnie będzie wywoływana dla każdego przycisku, co pozwala uniknąć powielania kodu.

Zadanie 4

Zaimplementuj metodę button_Click tak by spełniają następujące funkcje:

- zaznaczała na szachownicy odpowiedni ruch. W tym celu musimy rozpoznać jaki klawisz został wciśnięty. Można do tego "dojść" wykonując następujące działanie:
 - 1. Argument metody (object sender) rzutujemy na obiekt klasy Button

- Klasa Button ma pole typu string o nazwie Name. Pole to można porównywać odpowiednio z "buttonPole0", "buttonPole1", ... Najładniej cel ten można zrealizować przy pomocy konstrukcji switch
- Zmieniała gracza,
- Sprawdzała, czy gra została już zakończona,
- Prezentowała zamieszczone ruchy.

Zadanie 5

Zmodyfikuj program.

- 1. Dodaj kontrolkę (np RadioButton), która pozwoli na wybór gracza.
- 2. Po zakończeniu gry, program pyta się czy rozpocząć nową rozgrywkę. Zapytanie można wyświetlić w następujący sposób:

if (MessageBox.Show("Czy rozpocząć nową grę ?", "Nowa gra", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)