



ПОДГОТОВКА НА УЧИТЕЛИ ПО ИНФОРМАТИКА /СПЕЦИАЛИЗИРАНО ОБУЧЕНИЕ ПО C#/



д-р Красимир Манев



д-р Марияна Райкова



Пано Панов

С ПОДКРЕПАТА НА:





Програми с графичен интерфейс – част II

В тази тема ще стане дума за:

- Няколко съвета за дизайна на формата
- Още компоненти на графичния интерфейс.
- Още за дизайна на стартовата екранна форма.
- Още примери на програми с графичен интерфейс

За дизайна на формата

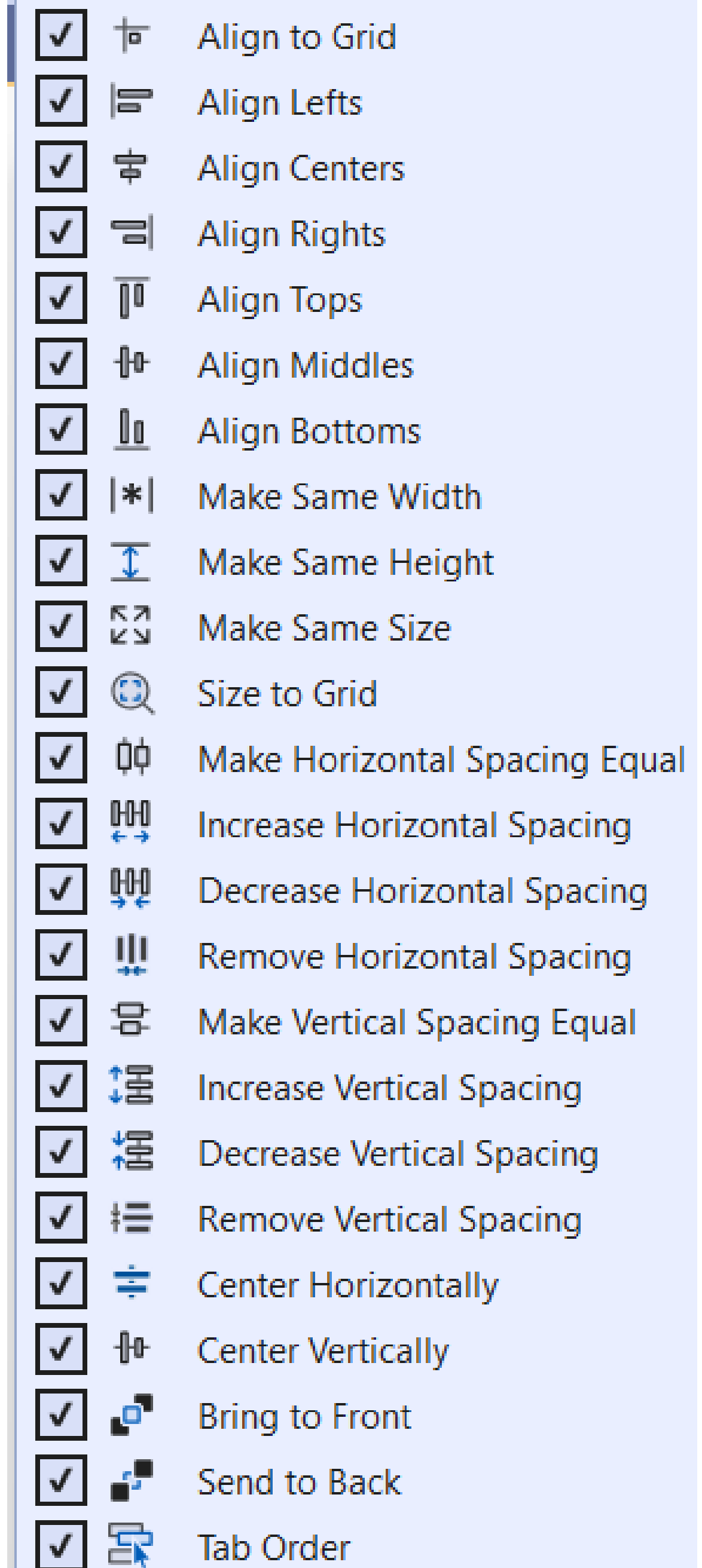
Изработването на дизайна на формата, по принцип, не е работа на програмиста. Най добре е той да се възложи на професионален дизайнер. Не е невъзможно, обаче, и е добре, програмистът да притежава определени дизайнерски умения. За целта ще предложим няколко съвета:

- Не правете формата нито много голяма, така че да остават много празни места, нито много малка и контролите да са притиснати една в друга.
- Ако ще използвате цветове, изберете малко и хармониращи един с друг.
- Използвайте подходящ шрифт. За дизайна на формата най-подходящи са безсерифните шрифтове (Arial). Добре е да използвате не повече от 2 големина на шрифта.
- Разполагайте контролите равномерно във формата а не струпани в някоя от страните.
- Поставайте близо едни до други контролите, които имат близко предназначение или като група са свързани с една и съща функционалност.
- Подравнявайте близките една до друга контроли – отгоре, отдолу, отляво или отдясно, като се стараете разстоянията – хоризонтално/вертикално – между редица контроли, разположени в ред/стълб да са еднакви (средата ви предоставя средства за да направите това с лекота).

За дизайна на формата

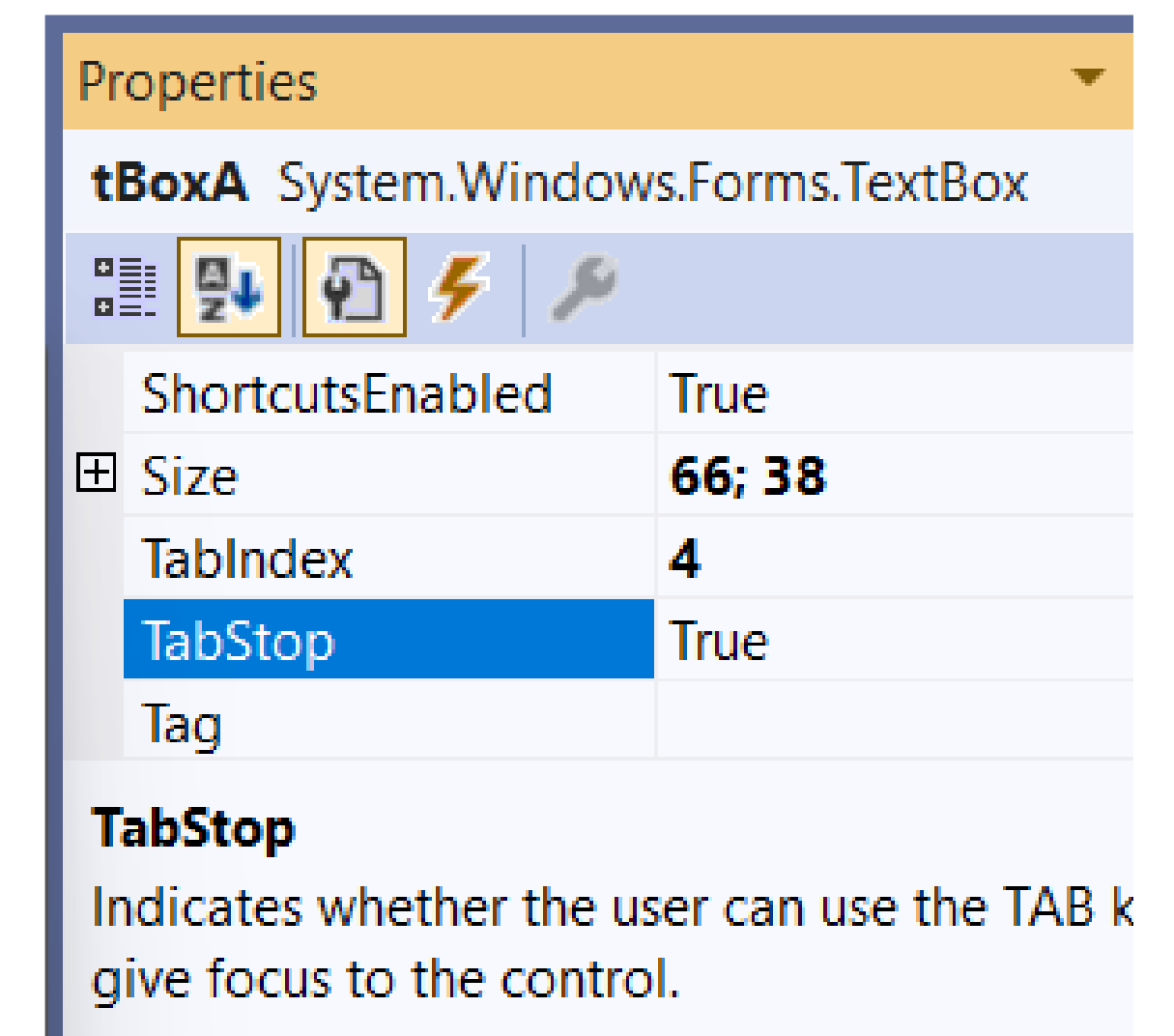
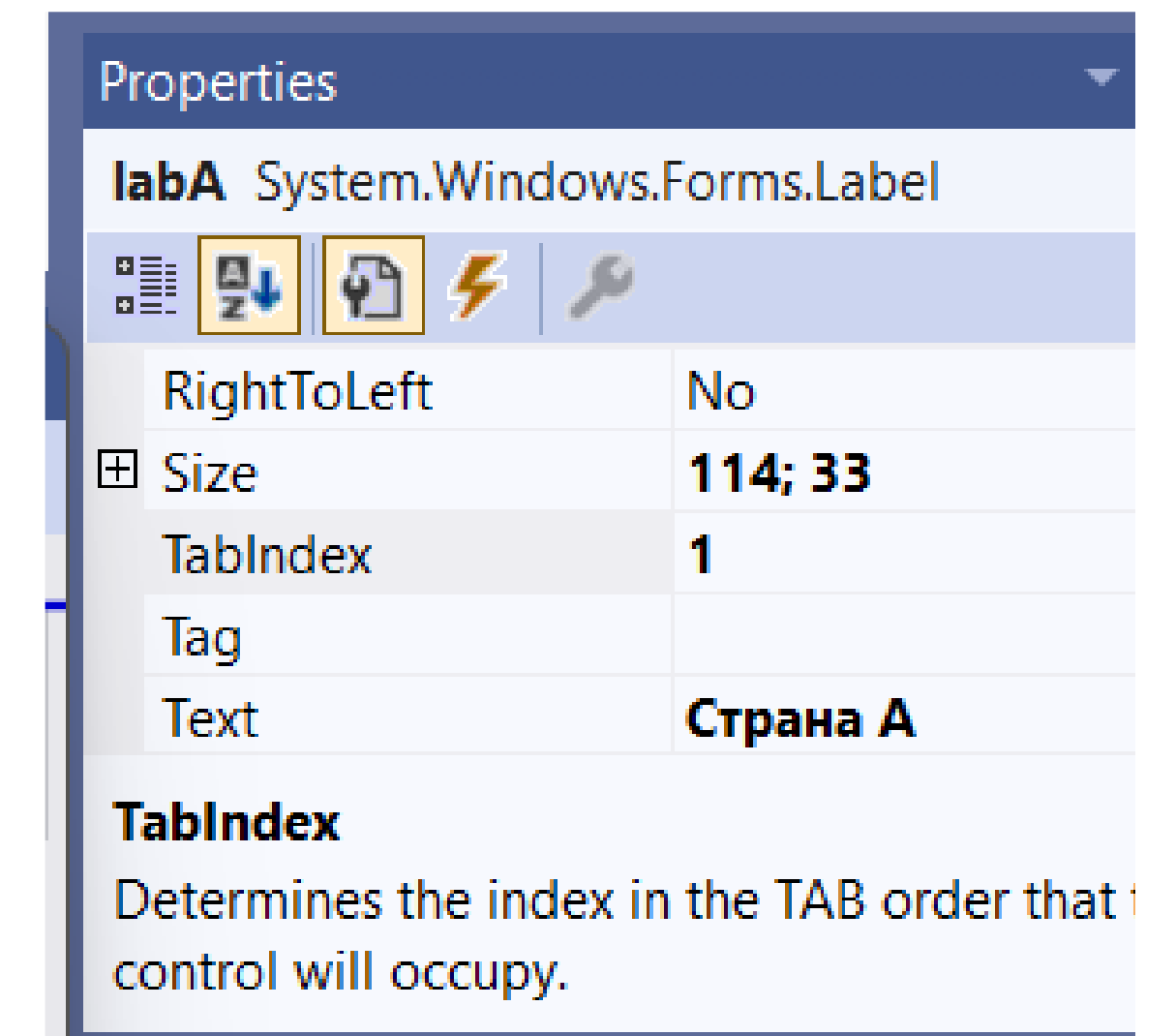
На Фигурата са показани бутоните на командите за оформяне на дизайн, които ще намерите в лентата с инструменти на средата (командите се изпълняват върху групи от маркирани контроли и стават достъпни саме след такова маркиране):

- Командите Align ... изравняват позициите на маркираните контроли в съответната посока до позицията на най-отдалечения в посоката;
- Командите Make Same ... изравняват размера на контролите – по ширина, по височина или и в двете посоки;
- Командите Make ... Spacing Equal изравняват разстоянията между маркираните контроли в съответната посока;
- Командите Increase/Decrease ... Spacing увеличават/намаляват разстоянията между маркираните контроли в съответната посока, а командите Delete ... Spacing премахва разстоянията между тях;
- Командите Center ... събират маркираните контроли в средата на формата – по вертикал или хоризонтал;
- Командите Bring to ... изтеглят маркирана контрола над/под всички, които се припокриват с нея;
- Командата Align to Grid измества контролата до най-близката линия на дизайнерската решетка.




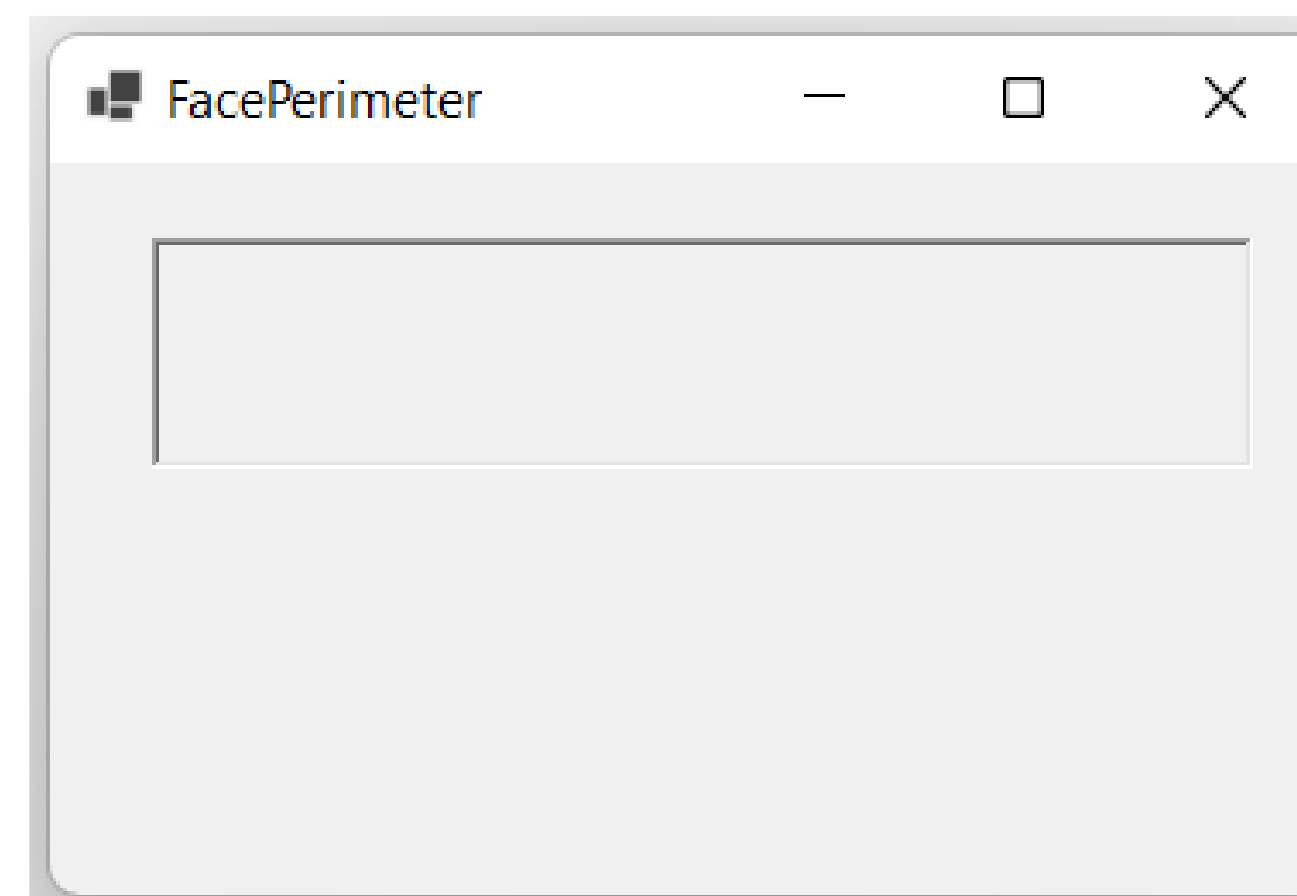
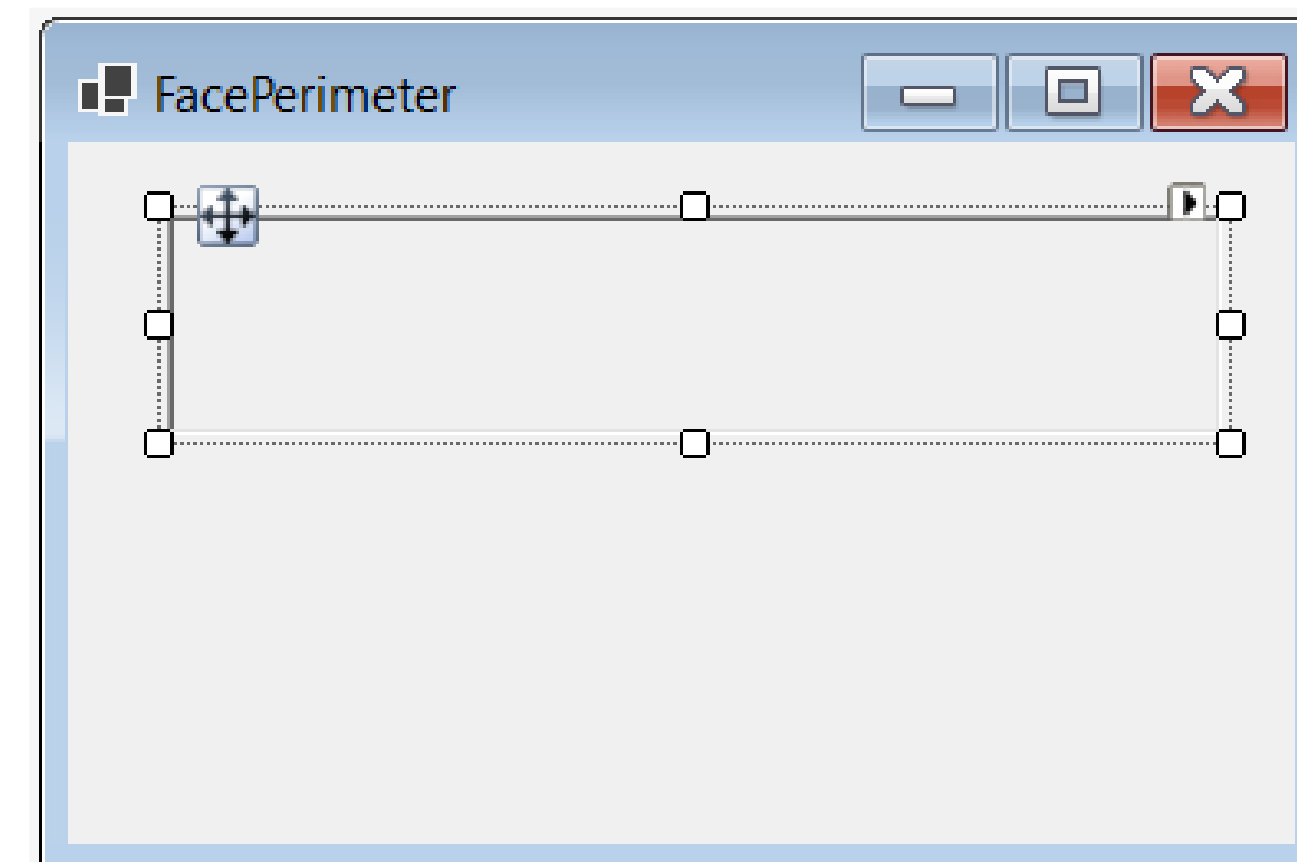
За управление на фокуса

- Както знаем от работа с други приложения с графичен интерфейс, форма с много контроли, една от **активните контроли** винаги е избрана като **текуща** – казваме че върху нея е „**поставен фокусът**“.
- С натискане на клавиша Tab, фокусът на формата се измества на следващата контрола, в избория от програмиста ред. А когато текущата контрола е последната в реда, с натискане на клавиша Tab фокусът се премества на първата в реда контрола.
- Когато създаваме форма, средата автоматично построява реда на контролите за поставяне на фокуса в реда, по който ги създаваме.
- Програмистът не винаги добавя във формата контролите в реда, по който би искал да се премества фокуса, затова след като приключи с дизайна на формата трябва да установи искания от него ред.
- Номерът в реда за преместване а фокуса се задава в свойството TabIndex. Неактивните контроли (горе) също имат такова свойство, макар че поставянето на фокуса върху тях не е възможно. При активните контроли пък може да се откажем от спирането на фокуса върху тях като поставим False в свойството TabStop (долу)




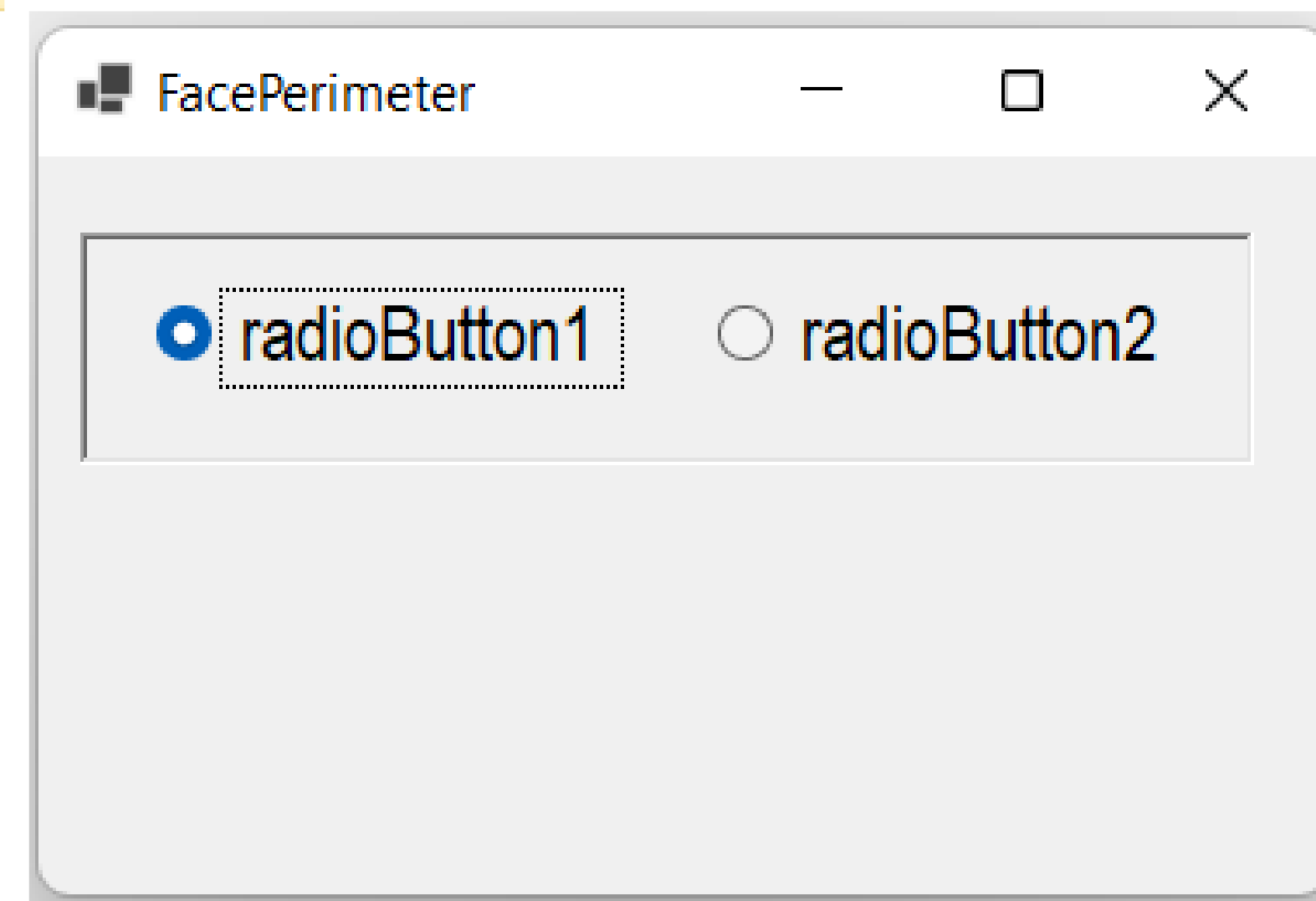
Контейнерът Panel (панел)

- Освен групата контроли с общо предназначение, по-горе споменахме и други групи компоненти на графичния интерфейс. За пример да разгледаме контейнера **Panel** (панел). Предназначението му е да се съберат в него група контроли с близко предназначение.
- В прозореца ToolBox той е в групата Containers и е обозначен с  Panel .
- Едно от немногото полезни свойства, освен за задаване размер и местоположение, които вече знаем да настройваме – от съответните полета на прозореца Properties или директно в дизайнерския изглед, е свойството BorderStyle (вида на очертаващата правоъгълна рамка).
- Възможните стойности са: None (панелът е без рамка, което не позволява да се почувства предназначението), FixedSingle, т.е. очертаване с проста линия или Fixed3D, т.е. имитиране на триизмерност (както на фигурата долу).



Контролата RadioButton (радио бутон)

- От работата с приложения с графичен интерфейс ни е позната контролата **Група радио бутони**. Предназначеното на тази контрола е да се даде възможност на потребителя да избира **точно една от няколко** алтернативи на функционалността на програмата.
- В C# няма такава контрола, но имаме контролата **RadioButton** (радио бутон), която както личи от името, съдържа само един радио бутон. Създаването на група радио бутони и синхронизирането на работата им е задължение на програмиста.
- Контролата има два елемента – етикет подсказващ ролята на бутона и кръгче, което е празно, когато бутонът не е натиснат и което се оцветява по характерен начин, когато бутонът е натиснат. Текстът, който ще се изпише в етикета, се задава в свойството **Text**.
- В прозореца Toolbox етикетът е представен с реда  **RadioButton**
- Най-важното свойство на радио-бутонът е свойството **Checked**. То е от булев тип и показва дали бутонът е натиснат (когато е със стойност True) или не е натиснат (когато е със стойност False).
- Не само за прегледност, а и за да работят синхронно като група, радиобутоните поставяме в панел. Панелът трябва да е поставен във формата предварително.

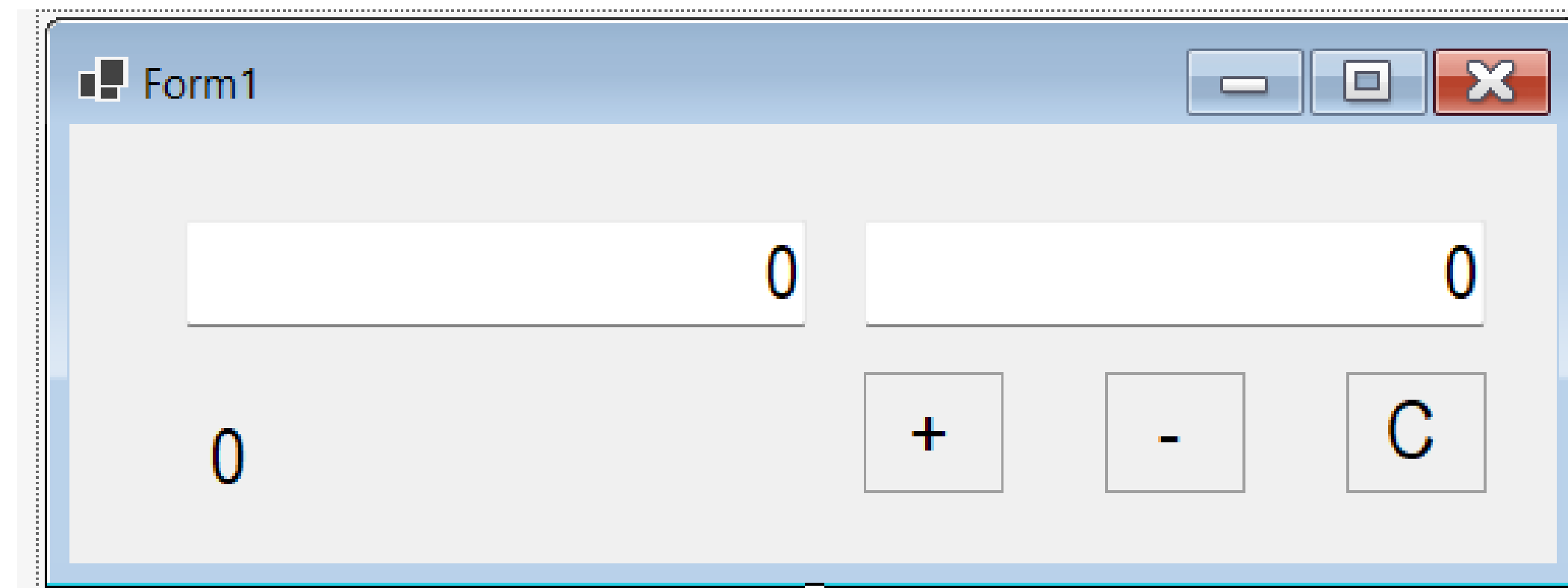


Работа с компютър

Задача 1. Напишете програма с графичен интерфейс Calc, която да събира и изважда две цели числа.

Проектиране на формата:

- Числата ще бъдат въвеждани от потребителя в текстови кутии textBox1 и textBox2.
- Изборът на операция ще става с натискане на един от двата бутона – button1, надписан с + и button2, надписан с –.
- Програмата трябва да изведе в етикета label1 сбора или разликата на числата, според натиснатия бутон.
- С бутона button3, надписан с C, потребителят ще изчиства съдържанието на двете кутии и етикета, когато поиска.
- На фигурата е показан един възможен изглед на прозореца на програмата. Направете необ. ходимото подравняване на контролите и поставете стойности по премълчаване 0 на текстовите кутии и етикета.
- Може да опитате свой дизайн на формата



Работа с компютър

- **Добавяне на функционалност:**

- За целите на пресмятането декларираме нужните променливи: `int a, b, c;`
- Програмата ще трябва да обработи събитието Click на всеки от трите бутона. При надписаните с + и – бутони ще трябва да се превърнат текстовете от двете кутии в числа a и b, да се извърши избраното аритметично действие, да се превърне полученият в числото c резултат в текст и да се запише в етикета:

```
a = int.Parse(textBox1.Text);  
b = int.Parse(textBox2.Text);  
c = a + b;  
label1.Text = c.ToString();
```

```
a = int.Parse(textBox1.Text);  
b = int.Parse(textBox2.Text);  
c = a - b;  
label1.Text = c.ToString();
```

- За бутона надписан с C(lean) остава само да се почистят старите стойности на етикета и текстовите кутии:

```
textBox1.Text = "0";  
textBox2.Text = "0";  
label1.Text = "0";
```

Работа с компютър

- За да демонстрираме използването на още едно събитие, нека направим следната промяна в програмата. Да създадем свободна функция

```
private void getData()  
{  
    a = int.Parse(textBox1.Text);  
    b = int.Parse(textBox2.Text);  
}
```

в която да поставим повтарящата се част от входа (това е добра практика – ако се наложи някаква промяна в повторения код, ще я правим само на 1 място).

- Сега можем да премахнем повтарящия се код и да го заменим с извикване на функцията getData()

```
private void button1_Click(object sender, EventArgs e)  
{  
    //a = int.Parse(textBox1.Text);  
    //b = int.Parse(textBox2.Text);  
    getData();  
    c = a + b;  
    label1.Text = c.ToString();  
}
```

и аналогично за другия бутон. Напишете двете версии на тази програма.

Работа с компютър

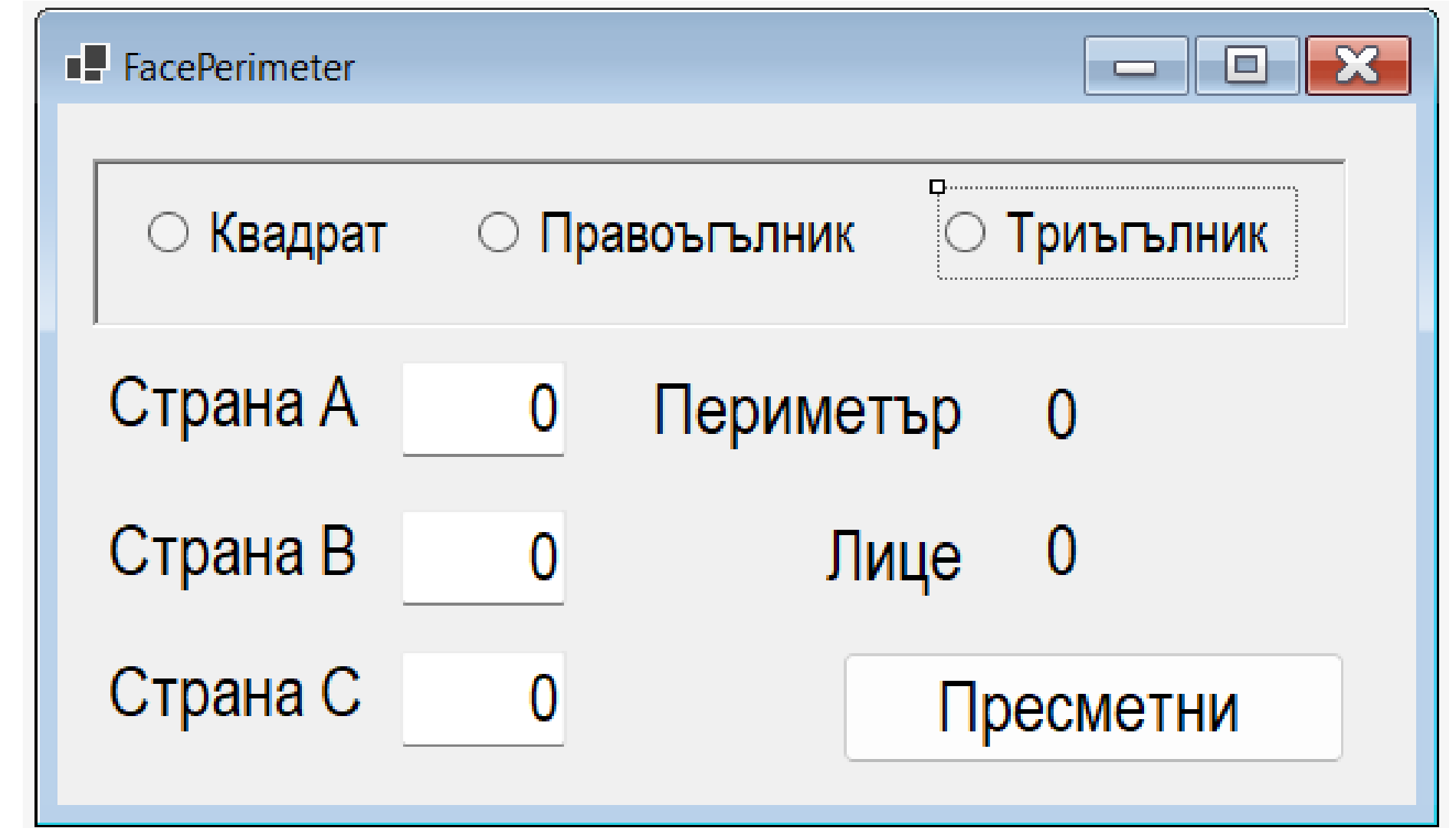
- **Задача 2.** Направете приложение с графичен интерфейс FacePerimeter , което по зададени характеристики на квадрат, правоъгълник или триъгълник, намира лицето и периметъра на зададената фигура. Квадратът ще бъде зададен с дължина на страната, правоъгълникът - с дължините на двете му страни, а триъгълникът – на трите му страни.

Анализ на формата:

- Изборът на фигурата, за която ще се извършват пресмятанията е естествено да направим с група от три радиобутона – Квадрат, Правоъгълник и Триъгълник.
- Пресмятаните резултати за трите фигури са едни и същи – Лице и Периметър и за това трябва да предвидим две двойки етикети „за надпис – за стойност“.
- За въвеждане на входните данни ще създадем три двойки „етикет – текстова кутия“ А, В и С, като за дължината на страната на квадрата ще използваме само първата, за дължините на двете страни на правоъгълника – първите две, а за дължините на страните на триъгълника – трите двойки.
- Трите двойки за въвеждане на данните ще са невидими докато потребителят избере за коя от фигурите ще се извършат пресмятанията. Тя трябва да е точно една!
- За стартиране на пресмятането ще добавим бутон Изчисли, след натискането на който ще показваме резултата.

Работа с компютър

- **Проектиране на формата.** Един примерен дизайн на екранната форма е показан на фигурата. Тъй като броят на контролите вече е доста голям, за да не бъркаме имената им, вместо даваните по премълчаване от средата имена сме поставили мнемонични:
- Трите радиобутона са с имена rBKv, rBPr и rBTr, като свойството Checked и на трите е False;
- Имената на етикетите за текстовите кутии, където се въвеждат страните са labA, labB и labC, а съответните им текстови кутии – tBoxA, tBoxB и tBoxC. Шестте контроли са проектирани като Visible = False и със стойности на текстовите кутии 0 по премълчаване;
- Имената на етикетите за имената на резултатите са labPer и labLice, а съответните им етикети за показване на резултатите – labPerVal и labLiceVal със стойността на Text 0 по премълчаване;
- Тъй като командният бутон Пресметни е само един ще запазим даденото му по премълчаване от средата име button1.
- При проектиране на формата сме приложили необходимите подравнявания и сме изравнили вътрешните разстояния между контролите.



Работа с компютър

- Функционалността на програмата ще се реализира с обработката на събитието Click на трите радио-бутона и на командния бутон button1.
- Започваме с радио бутоните: щракваме на събитието Click на радио-бутона rBKv в страницата Events на прозореца Properties. В прозореца с програмния код ще се покаже заготовката на метода за обработка и написваме в нея следния код::

```
private void rBKv_Click(object sender, EventArgs e)
{
    tBoxA.Visible = true; labA.Visible = true;
    tBoxB.Visible = false; labB.Visible = false;
    tBoxC.Visible = false; labC.Visible = false;
}
```

По същия начин написваме кода за обработка на събитието Click на радио бутона rBPr:

```
private void rBPr_Click(object sender, EventArgs e)
{
    tBoxA.Visible = true; labA.Visible = true;
    tBoxB.Visible = true; labB.Visible = true;
    tBoxC.Visible = false; labC.Visible = false;
}
```

Остава по същия начин да се напише кода кода за обработка на събитието Click на радио бутона rBTr.

Работа с компютър

- Пресмятанията извършваме при обработка на събитието Click на button1:

```
private void button1_Click(object sender, EventArgs e)
{
    int a, b, c, P, SI; double p, S;
    if(rBKv.Checked)
    {
        a = int.Parse(tBoxA.Text); P = 4 * a; SI = a * a;
        labPerVal.Text = P.ToString(); labLiceVal.Text = SI.ToString();
    }
    else if (rBPr.Checked)
    {
        a = int.Parse(tBoxA.Text); b = int.Parse(tBoxB.Text);
        P = 2 * (a + b); SI = a * b;
        labPerVal.Text = P.ToString(); labLiceVal.Text = SI.ToString();
    }
    else
    {
        a = int.Parse(tBoxA.Text); b = int.Parse(tBoxB.Text); c = int.Parse(tBoxC.Text);
        P = a + b + c; p = P / 2.0; S = Math.Sqrt(p * (p - a) * (p - b) * (p - c));
        labPerVal.Text = P.ToString(); labLiceVal.Text = String.Format("{0:0.#####}", S);
    }
}
```

Работа с компютър

За да съберем на едно място четенето на дължините на страните, и тук бихме могли да напишем свободна функция GetData():

```
int a, b, c;  
private void GetData()  
{  
    a = int.Parse(textBoxA.Text);  
    b = int.Parse(textBoxB.Text);  
    c = int.Parse(textBoxC.Text);  
}
```

Сега можем да заменим четенето във всеки от трите случая с извикване на функцията. Направете тези промени в програмата.

Работа с компютър

- Ако сте тествали внимателно така създаденото приложение може би сте установили някакви нередности:
- След като преминем от пресмятането за някоя фигура към пресмятане за друга фигура, в текстовите кутии за въвеждане на данни и в етикетите за резултата остават стойностите от предното пресмятане. Когато потребителят избере с коя фигура ще работи, нормално е да види празни текстови кутии, преди да започне да въвежда дължините на страните.
- Напишете функция `void CleanBoxes()`, която изчиства съдържанията на текстовите кутии за данните и етикетите за резултата. Не забравяйте, че стойностите на свойствата `Text` на текстовите кутии и етикетите са низове!
- Вмъкнете извикване на тази функция при обработката на събитието `Click` на всеки от трите радио бутона и отново тествайте програмата.

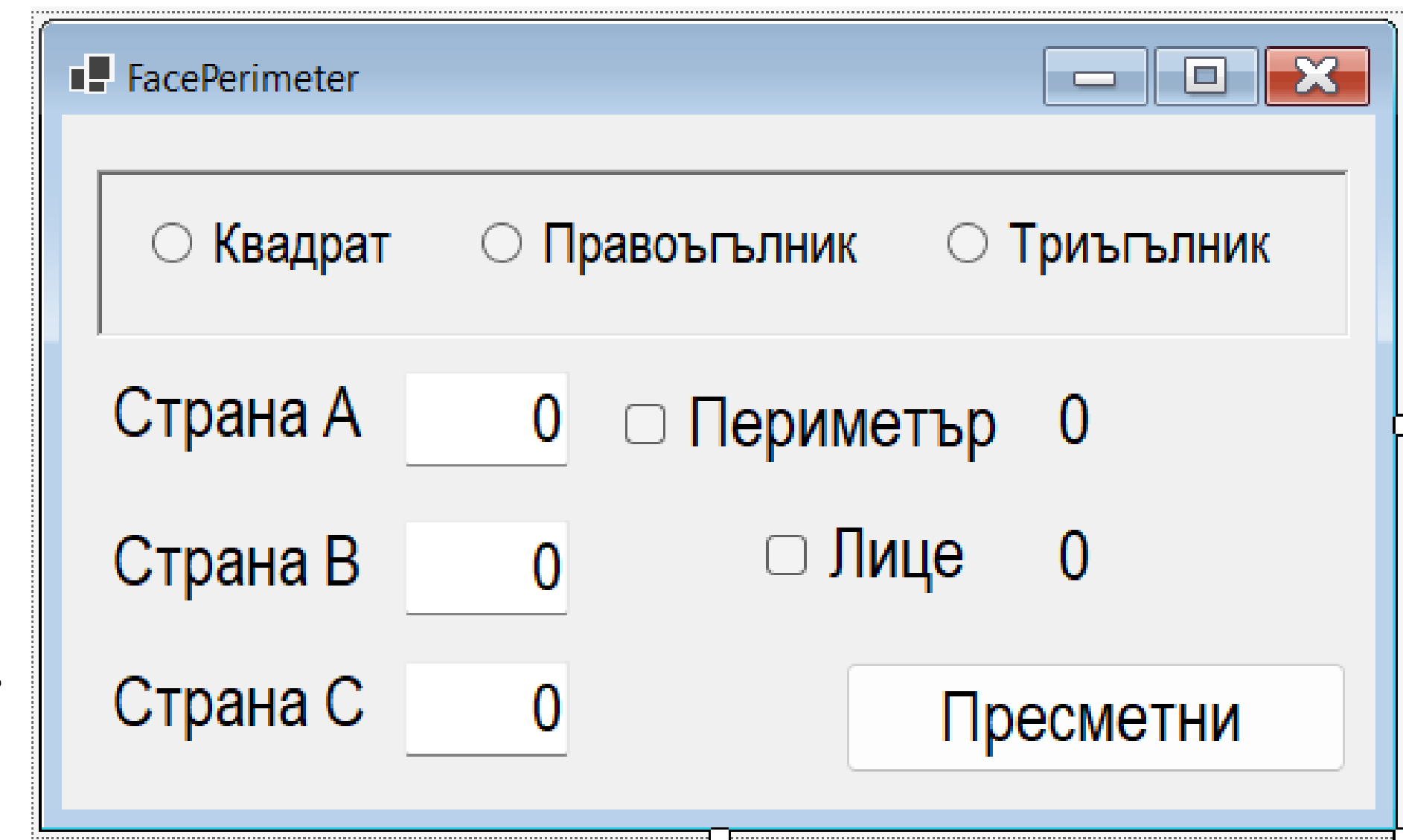
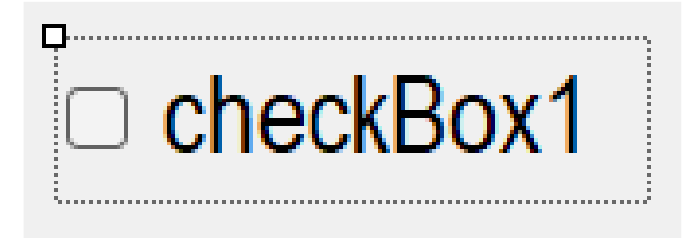
Работа с компютър

- Проверете как ще реагира програмата, ако поискате да се пресметне лицето на триъгълник със страни 4, 5 и 10. На какво се дължи това и как би трябвало да се предпазим от този недостатък на програмата?
- Проблемът при пресмятане за страни 4, 5 и 10 е очевиден. Не е възможно да има такъв триъгълник, тъй като трите стойности не изпълняват едно от Неравенствата на триъгълника. В резултат един от множителите на формулата на Херон става отрицателен и коренуването е невъзможно.
- Този недостатък можем лесно да отстраним, като добавим в съответния метод необходимата проверка и в случай, че някое от неравенствата на триъгълника не е изпълнено да прекратим изпълнението на метода, след като изведем подходящо съобщение за потребителя:

```
a = int.Parse(textBoxA.Text); b = int.Parse(textBoxB.Text);
c = int.Parse(textBoxC.Text);
if (a + b <= c || a + c <= b || b + c <= a)
{    MessageBox.Show("Не са страни на триъгълник"); }
else
{    ... Старият код ...    }
```

CheckBox (кутия за отметка)

- Друга полезна графична компонента е **CheckBox** (кутия за отметка). Тя също притежава свойствата `Checked` и `Text`, както при `RadioButton`. Разликата е, че кутиите за отметка не се събират в групи и може да имаме активни няколко кутии за отметка, или никоя кутия да не е активна.
- Тази контрола, също както `RadioButton` се изобразява с 2 елемента – самата кутия, в която при щракване с мишката се показва отметката и етикет показващ предназначението ѝ.
- За илюстрация на използването на тази контрола, да направим нов вариант на програмата `LicePerimeter`, като дадем възможност на потребителя да избира коя характеристика да пресметне – периметъра, лицето или и двете.
- За целта трябва да заменим двата етикета `Периметър` и `Лице` с две кутии за отметки, като поставим същите два надписа, като стойност на свойството `Text` на двете кутии
- Когато поставим отметка в кутията с надпис `Периметър`, програмата трябва да пресмята само параметъра на фигурата, когато отметка има само в кутията `Лице` – да пресмята само лице, а ако и двете отметки са поставени – да пресмята и двете характеристики.



CheckBox (кутия за отметка)

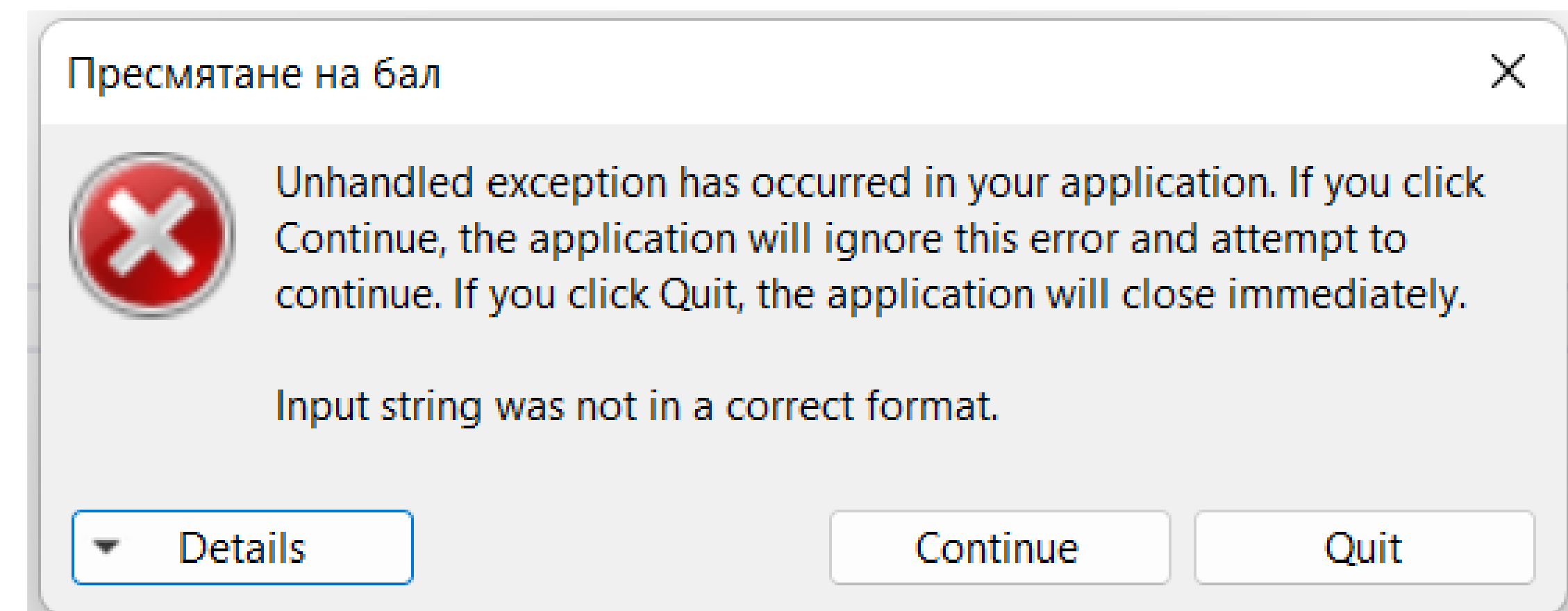
- За да се реализира функционалността трябва да отделим пресмятанията на двете характеристики в различни блокове. За квадрата кодът ще изглежда така:

```
int a, b, c, P, SI; double p=0.0, S;  
CleanResults();  
if (rBKv.Checked)  
{  
    GetData();  
    if (checkBox1.Checked == true)  
    { P = 4 * a; labPerVal.Text = P.ToString(); }  
    if (checkBox2.Checked == true)  
    { SI = a * a; labLiceVal.Text = SI.ToString(); }  
}
```

- Направете нужните поправки и за другите две фигури.
- А сега проверете как ще реагира програмата, ако зададете в някое от текстовите полета нечислова стойност?

Прихващане и обработка на изключения

- Събитийното програмиране има една неприятна страна. Когато потребителят предизвика някое събитие и стартира някаква обработка на данните, може да се окаже, че не всички данни са налични. За пример да стартираме програмата Val и стартираме пресмятане на бала, без да сме въвели някоя от оценките. В резултат програмата ще спре аварийно издавайки съответно съобщение
- В съобщението се казва, че по време на работата на програмата е възникнала аварийна ситуация, наричана **изключение** (exception), за която не е предвидено как да се продължи. Съобщението подсказва две възможности – да се прекрати изпълнението и да се потърси възможност да се предвиди продължение (бутона Quit) или да се продължи изпълнението (бутона Continue). В нашата програма продължаване на работата е възможно, но за други програми то може да се окаже невъзможно.
- Това, което трябва да направи програмистът в такъв случай е, да се опита да отстрани проблема с добавяне на код, както в предишна програма (ако това е възможно) или да използва механизма на средата за **прихващане и обработка на изключения**.



Прихващане и обработка на изключения

- В C# има дефинирани много изключения, някои от които се отнасят за много специфични ситуации. Ето някои изключения, които начинаещият програмист е полезно да знае:
- `ArithmeticException` – типична ситуация, при която възниква това изключение е например опит за деление на 0;
- `FileNotFoundException` – това изключение възниква, когато програмата безконтролно се опитва да отвори файл със зададено име, а такъв файл няма във файловата система там, където програмата очаква;
- `IndexOutOfRangeException` – това изключение възниква, когато при обработване на елементите на масив програмата се опита да използва елемент с несъществуващ индекс (често срещана грешка при начинаещите програмисти, и т.н.
- Пълен списък на изключенията в C# може да намерите в Интернет:
[Complete List of Exception Class in C# \(completecsharp tutorial.com\)](http://completecsharp tutorial.com)
- Важно е да се отбележи, че програмистът сам може да определи ситуации, в които да спре програмата аварийно, като „*хвърли exception*“ (това е програмисткият жаргон за възникване на аварийна ситуация) с оператора: `throw new` <изключение>(<описващ текст>). Например

```
throw new FileNotFoundException("Нямате достъп до този файл!");
```

Прихващане и обработка на изключения

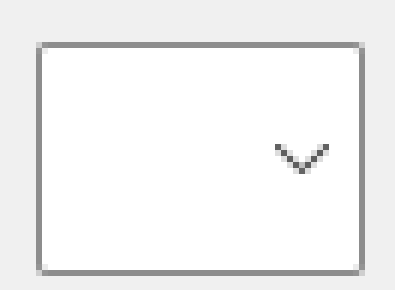

- В езика C#, с оператора `try...catch... finally`, може да се обработват изключения (exception), т.е. ситуации, които ако не бъдат обработени, ще предизвикат аварийно спиране на програмите.
- Синтаксисът на оператора `try ... catch ... finally` е:
`try` { < оператори, при изпълнение на които очакваме изключение > }
`catch` (< вид на изключението >)
{ < оператори, които ще се изпълнят при възникване на изключение > }
`finally` { < оператори, които ще се изпълнят винаги > }
- Когато програмата достигне този оператор, се опитва да изпълни операторите в блока `try` (опитвам). В частта си `catch ... finally` операторът прилича на `if ... else`, но със следните разлики. Ролята на условие тук играе възникването на някакво **очаквано изключение**, като видът на изключението се поставя в скоби както условие.
- Ако изключението се случи, тогава се изпълнява кодът след `catch` (хващам, улавям), а **след това** – кодът след `finally` (накрая). Ако очакваното изключение не се случи, се изпълнява само кодът след `finally`.
- Както кодът в клаузата `catch`, така и кодът в клаузата `finally`, може да липсва.

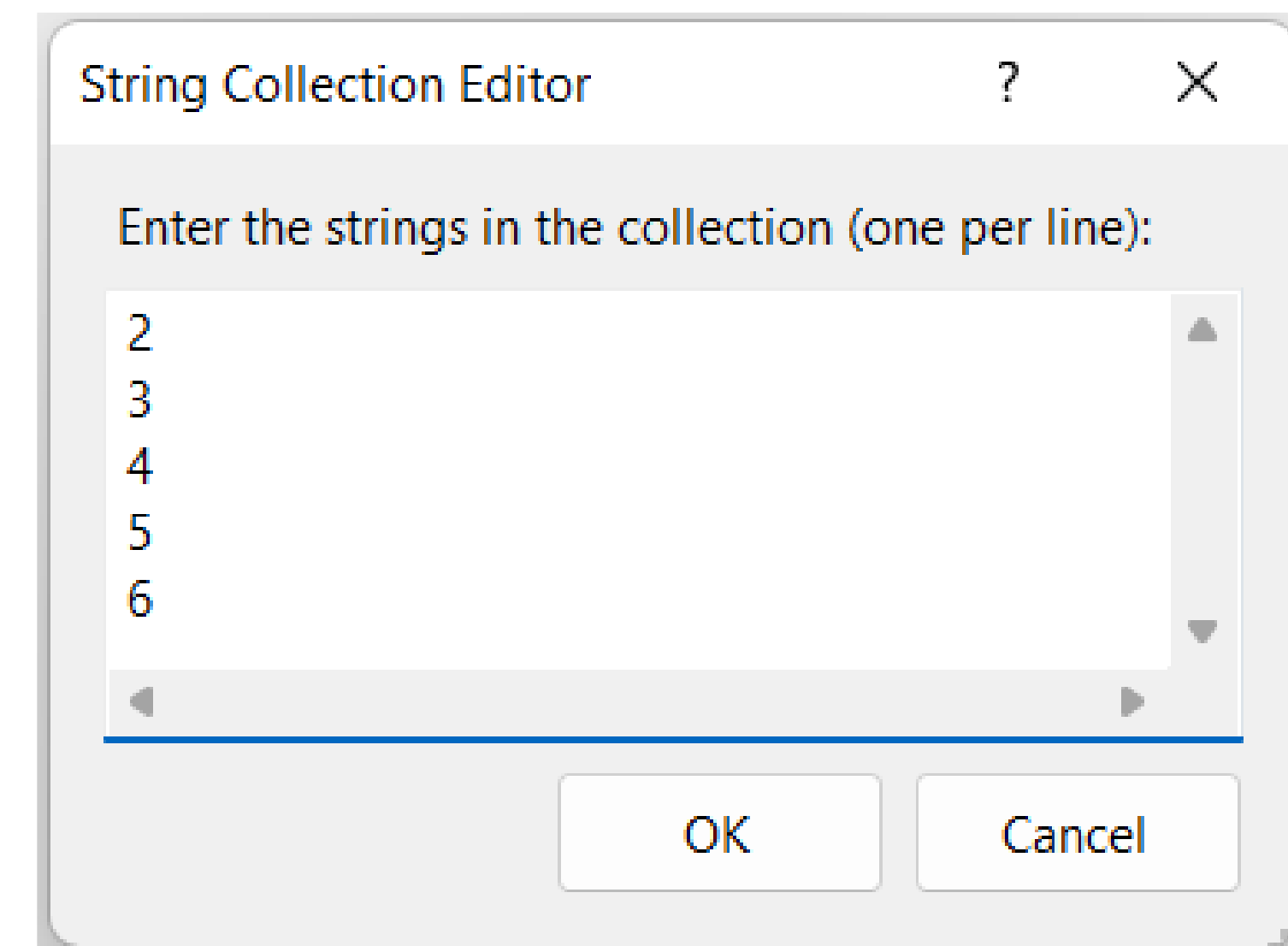
Прихващане и обработка на изключения

- Изключенията в C# са оформени в класове. Класът `FormatException` обхваща всички грешки, които възникват при опит за обработка на данни с неправилен формат. Такава грешка е, например, изписването на нецифрови знаци, там където се очакват цифри. Или пък оставяне празно на поле, в което непременно трябва да има данни.
- За да поправим неправилното поведение на нашата програма трябва задължително да поставим кода, който трябва да се изпълни в клаузата `try`, като изключение да поставим `FormatException`, а в клаузата `catch` да изведем подходящо съобщение:
- Синтаксисът на оператора `try ... catch ... finally` е:

```
int Sum;  
try  
{  
    Sum = int.Parse(comboBox1.Text) + int.Parse(comboBox2.Text)  
        int.Parse(comboBox3.Text) + int.Parse(comboBox4.Text);  
    label16.Text = String.Format("{0:0.00}", Sum / 4.0);  
}  
catch (FormatException)  
{  
    MessageBox.Show("Не всички оценки са нанесени");  
}
```
- Методът `MessageBox.Show` извежда на екрана диалогов прозорец, в който изписва зададения като аргумент низ. След прочитането му, можем да го зтворим с бутона OK и да продължим работа с приложението.

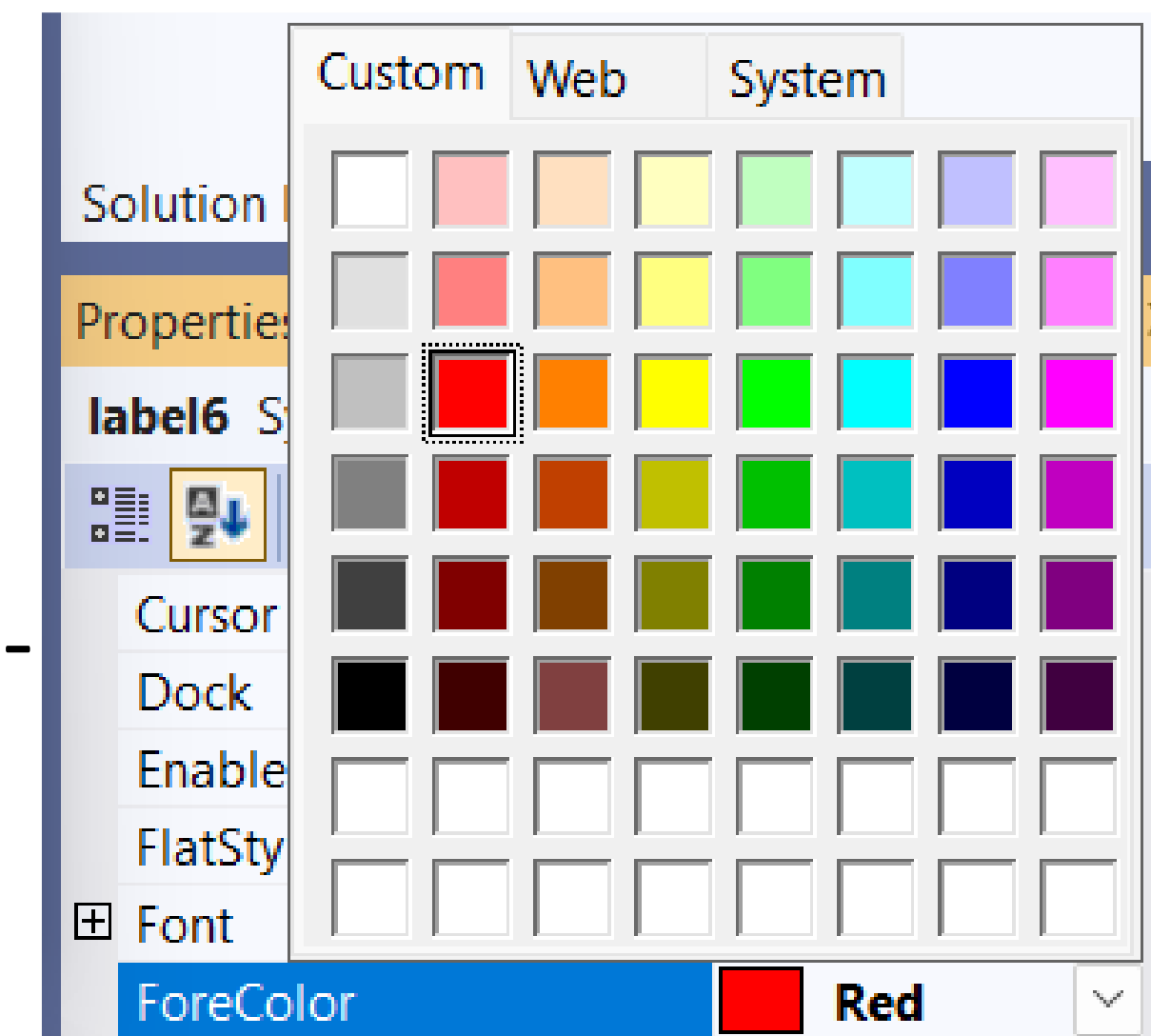
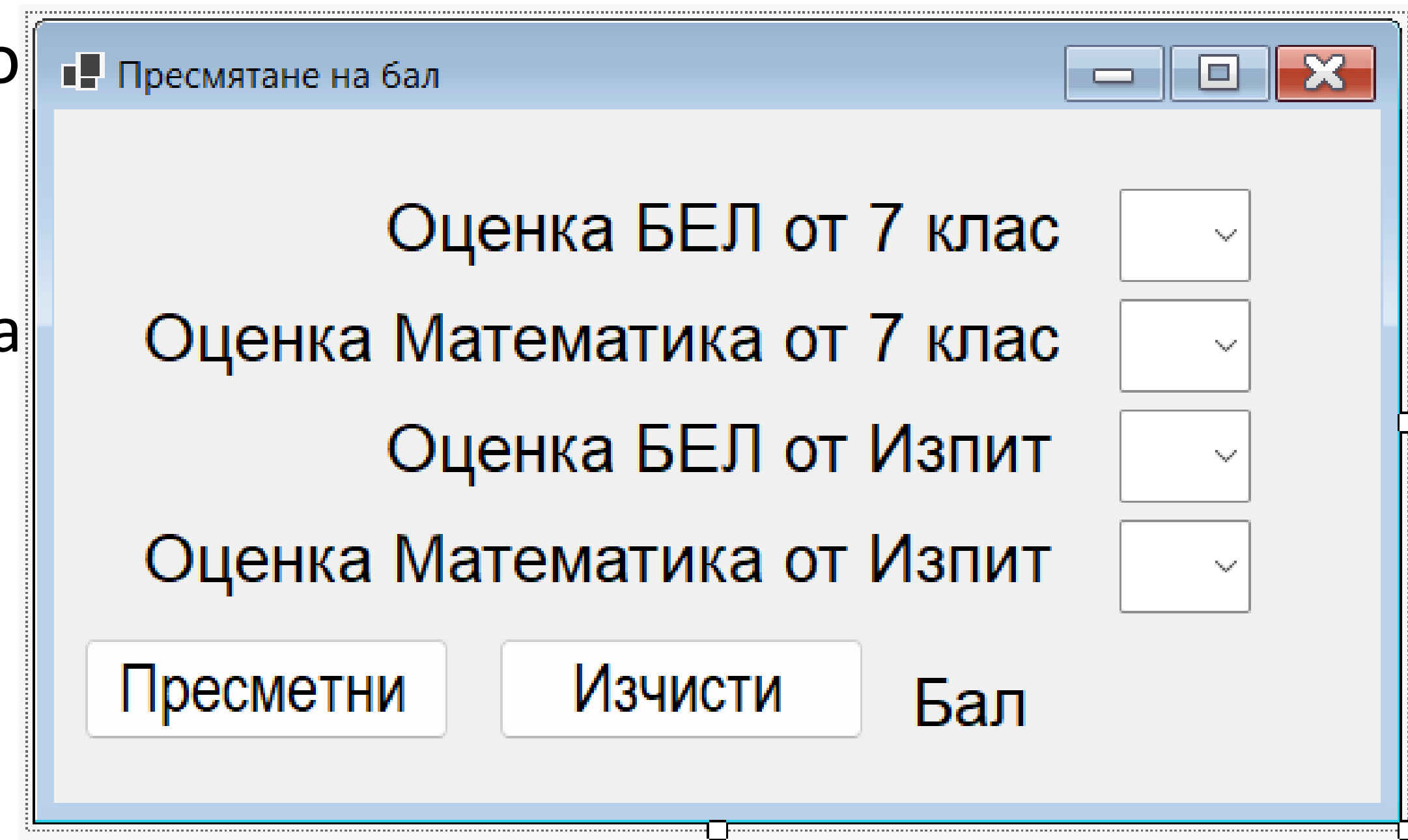
ComboBox (комбинирана кутия)

- От работата с приложения с графичен интерфейс ни е позната и контролата **ComboBox** (Комбинирана текстова кутия). Предназначението на тази контрола е да облекчи потребителя в случай, че възможните данни, които ще се въвеждат от нея са измежду ограничен *списък от стойности*.
- Контролата има две части – списъкът от допустими стойности, който се отваря с характерния бутон стрелка, и текстова кутия, в която се изобразява избраната от потребителя стойност. Текстът, който ще се изпише в кутията, става стойност на свойството **Text**.
- В прозореца Toolbox етикетът е представен с реда  **ComboBox**
- Най-важното свойство на комбинираната текстова кутия е **Items**. Стойността му е инстанция на клас, който представлява *колекция* от низове, които съставят списъка. При щракване с мишката в полето за стойност на свойството се отваря диалогов прозорец, в който програмистът да изпише стойностите от Списъка – по една стойност на ред.
- Контролата предоставя много други възможности. Например, вместо фиксиран списък да ползва външен източник.



Работа с компютър

- **Задача 3.** Напишете конзолно приложение Val, което По зададени годишните оценки по БЕЛ и Математика на ученичка/ученик от 7 клас и оценките по БЕЛ и Математика от проведен изпит, пресмята средната на четирите за прием в някакво специализирано обучение.
- **Планиране на формата.**
- Съществена част на формата трябва да бъдат четирите комбинирани кутии comboBox1, comboBox2, comboBox3 и comboBox4, в които ще се въвеждат четирите оценки и съответните четири етикета, сочещи коя комбинирана кутия за коя оценка е. С бутона button1 стартираме пресмятането, а с бутона button2 почистваме съдържанието на кутиите за следващо пресмятане.
- Получената средна оценка ще изписваме в етикета label6, който не се вижда в дизайна на формата. За да подчертаем получения резултат може да го изпишем в червен цвят. Изборът на цвят на текста на една контрола става от свойството ForeColor. При щракване в полето за стойност се отваря прозорец с три цвятови палитри. Палитрата Custom ни е позната от уроците по ИТ. Затова от този палитра избираме необходимия ни цвят.



Работа с компютър

- **Функционалност.** Остава да добавим нужната ни функционалност към събитието Click на двата бутона:

```
private void button1_Click(object sender, EventArgs e)
{
    int Sum;
    Sum = int.Parse(comboBox1.Text) + int.Parse(comboBox2.Text) +
        int.Parse(comboBox3.Text) + int.Parse(comboBox4.Text);
    label6.Text = String.Format("{0:0.00}", Sum/4.0);
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    comboBox1.Text = ""; comboBox2.Text = "";
    comboBox3.Text = ""; comboBox4.Text = "";
    label6.Text = "";
}
```

Въпроси и задачи

1. Проверете всяко от написаните до момента приложения с графичен интерфейс за възможно възникване на изключения от класа `FormatException` и направете нужното за прихващане и обработване на тези изключения.
2. (Домашно) Променете програмата `FacePerimeter` така, че да може да приема като дължини на страните дробни числа.
3. (Домашно) Променете програмата `Calc` така, че да може да пресмята произведението, частното при целочислено деление и остатъка при целочислено деление на двете зададени числа.

Изгревът е близо

Благодаря за вниманието!

Готов съм да отговарям на вашите
въпроси

