Объектно-ориентированное программирование. Обзор среды разработки Visual Studio .NET

ОГУ

Кафедра ВТиЗИ

Галимов Р.Р.

Часто используемые свойства и событие TextBox	Описание
Часто используемые свойства	
`AcceptsReturn`	Если свойство равно true в многострочном поле TextBox, нажатие Enter в TextBox создает новую строку. Если свойство равно false (значение по умолчанию), нажатие Enter приводит к тому же эффекту, что и нажатие кнопки по умолчанию на форме (кнопкой по умолчанию называется кнопка, назначенная свойству AcceptButton формы)
`Multiline`	Если свойство равно true, содержимое элемента TextBox может занимать несколько строк (по умолчанию используется значение false)
`ReadOnly`	Если свойство равно true, поле TextBox имеет серый фон, а его содержимое не может редактироваться (по умолчанию используется значение false)
`ScrollBars`	Для многострочных текстовых полей это свойство определяет режим отображения полос прокрутки: None (по умолчанию), Horizontal, Vertical или Both
`Text`	Текстовое содержимое TextBox
`UseSystemPasswordChar`	Если свойство равно true, используется режим ввода пароля (вместо вводимых символов отображается системный символ-заполнитель)
Часто используемое событие	
`TextChanged`	Генерируется при изменении текста в TextBox (то есть при добавлении или удалении символов). При двойном щелчке на элементе управления TextBox в режиме конструктора создается пустой обработчик этого события

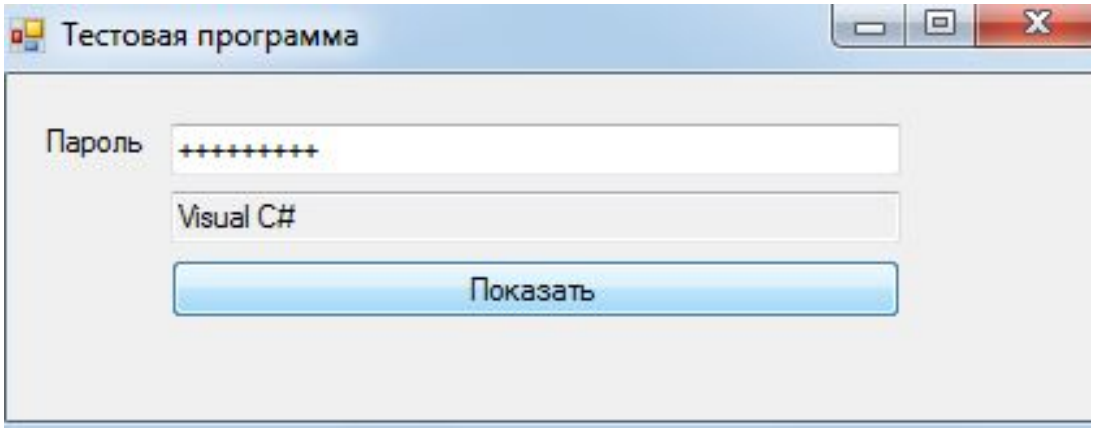
Кнопка (Button)

- *Кнопка (Button) - элемент управления, который нажимается пользователем для* выполнения конкретной операции или выбора варианта.
- в программах могут использоваться разные типы кнопок, включая флажки и переключатели.
- Все классы кнопок наследуют от класса ButtonBase (пространство имен System. Windows. Forms), определяющего общие возможности кнопок. приложению.

Кнопка (Button)

Часто используемые свойства и событие Button	Описание
Часто используемые свойства	
`Text`	Текст, выводимый на поверхности кнопки
`FlatStyle`	Изменяет внешний вид кнопки: Flat (кнопка выводится без имитации объема), Popup (кнопка выводится без имитации объема, пока пользователь не наведет на нее указатель мыши), Standard (трехмерная кнопка) или System (внешний вид кнопки определяется операционной системой)
Часто используемое событие	
`Click`	Генерируется при нажатии кнопки пользователем. При двойном щелчке на элементе управления Button в режиме конструктора создается пустой обработчик этого события

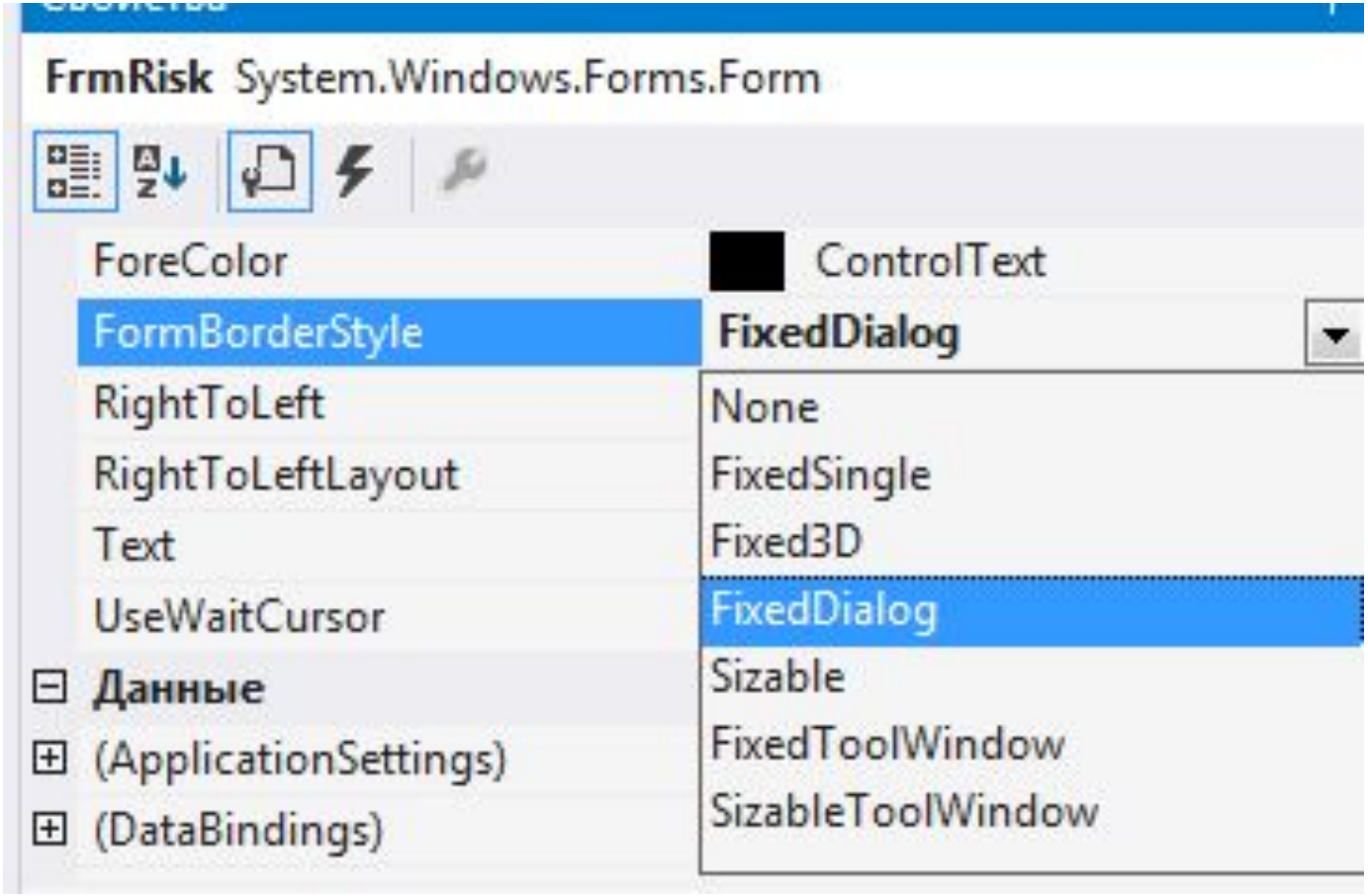
Кнопка (Button)



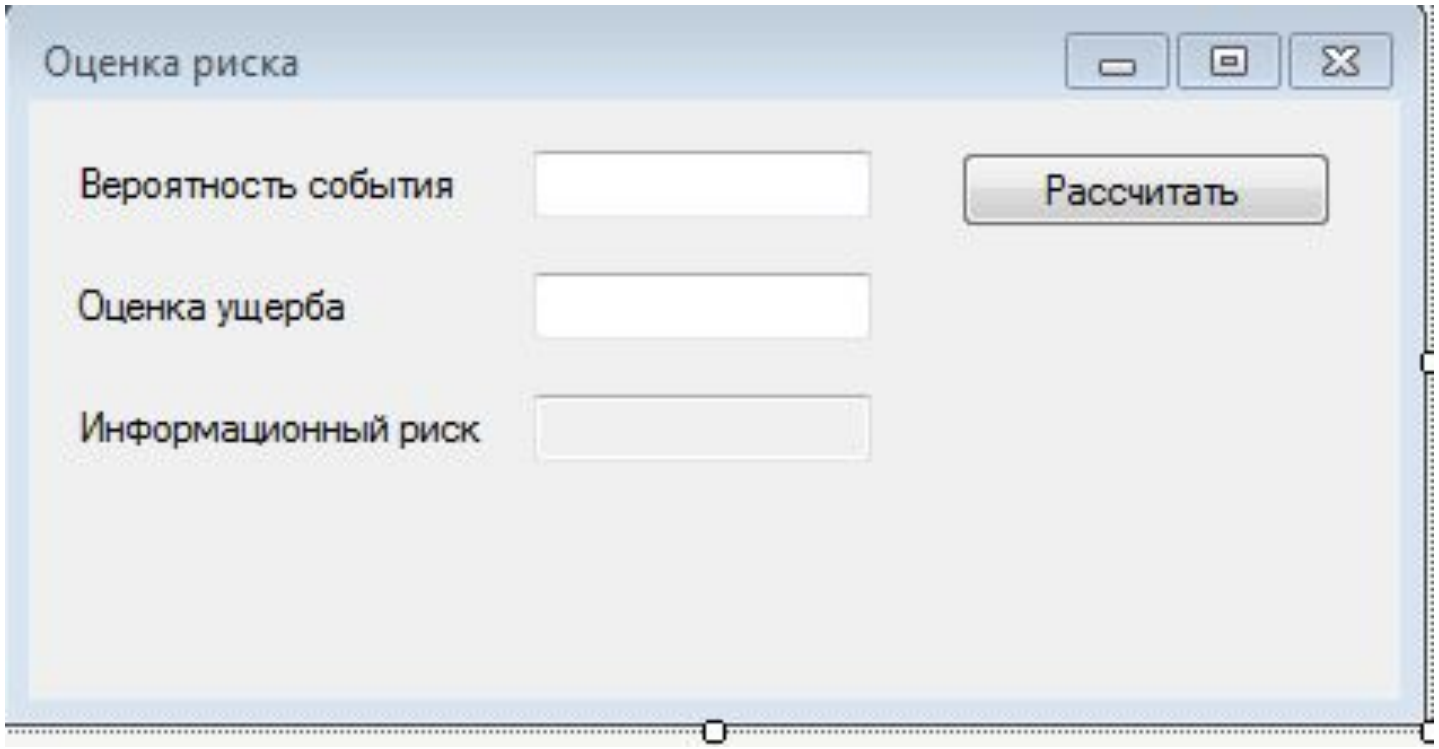
Кнопка (Button)

```
namespace PasswdTest
{
    public partial class frmTestPasswd : Form
    {
        public frmTestPasswd()
        {
            InitializeComponent();
        }

        private void buttonShow_Click(object sender,
   EventArgs e)
        {
            textBoxRes.Text = textBoxPasswd.Text;
        }
    }
}
```

Кнопка (Button)



Кнопка (Button)



```
private void btnCalc_Click(object sender, EventArgs e)
        {
            float P, U, R;
            try
            {
                P = Convert.ToSingle(txtBoxP.Text);             }
            catch (Exception ex)
            {
                MessageBox.Show("Не корректное значение оценки вероятности\n"+ex.Message);
                txtBoxP.Focus();
                return;
            }
            if (P < 0 || P > 1)             {
                MessageBox.Show("Не корректное значение оценки вероятности");
                txtBoxP.Focus();
                return;             }
            try
            {
                U = Convert.ToSingle(txtBoxU.Text);             }
            catch (Exception ex)
            {
                MessageBox.Show("Не корректное значение оценки ущерба\n" + ex.Message);
                txtBoxU.Focus();
                return;
            }
            if (U <= 0 )             {
                MessageBox.Show("Не корректное значение оценки вероятности");
                txtBoxP.Focus();
                return;             }
            R = P * U;
            txtBoxR.Text = Convert.ToString(R);         }
```

GroupBox и Panel

Контейнеры GroupBox и Panel предназначены для размещения элементов управления в графическом интерфейсе - обычно в них группируются элементы со сходной функциональностью или выполняющие общую функцию в графическом интерфейсе.

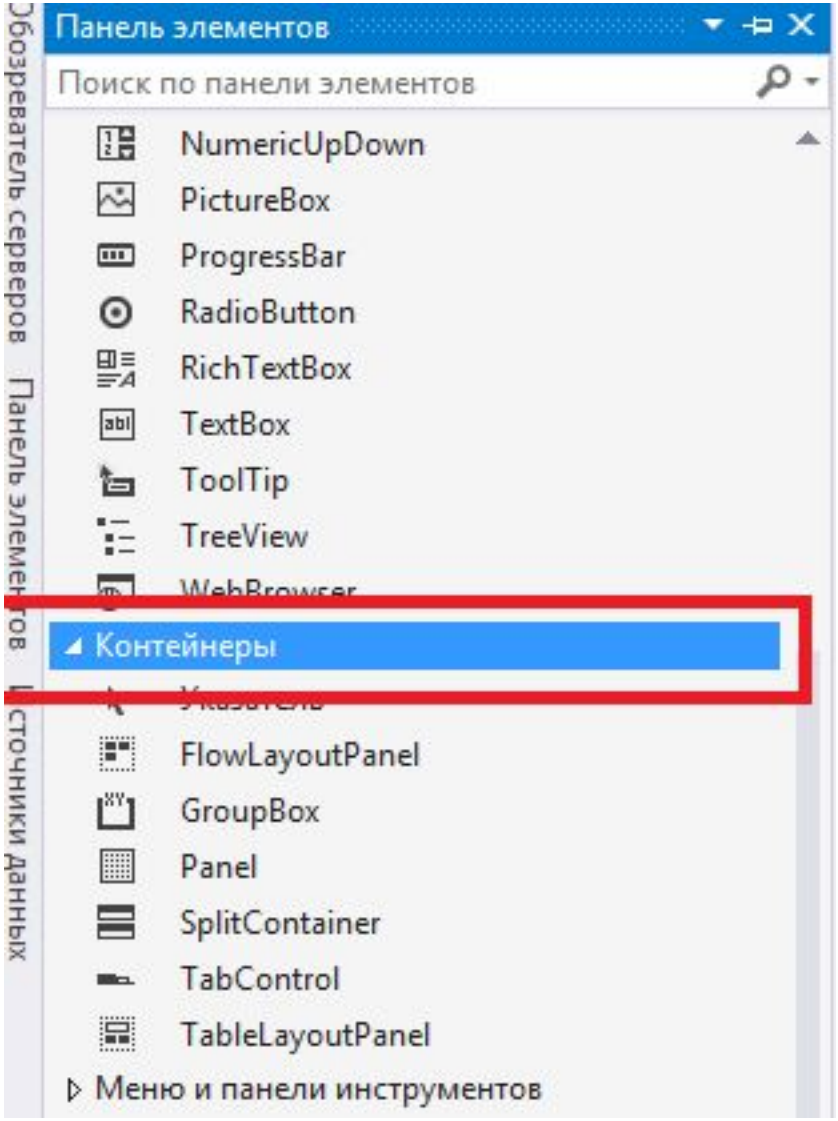
Все элементы управления в GroupBox или Panel перемещаются вместе с перемещаемым контейнером GroupBox или Panel. Кроме того, контейнеры GroupBox и Panel могут использоваться для одновременного отображения или сокрытия набора элементов.

Изменение свойства Visible контейнера изменяет видимость всех содержащихся в нем элементов управления.

Главное различие между этими двумя элементами управления заключается в том, что контейнер GroupBox может иметь заголовок и не поддерживает полосы прокрутки, а Panel может иметь полосы прокрутки и не содержит заголовка.

Контейнер GroupBox по умолчанию имеет тонкую границу; Panel тоже можно настроить подобным образом при помощи свойства BorderStyle.

GroupBox и Panel

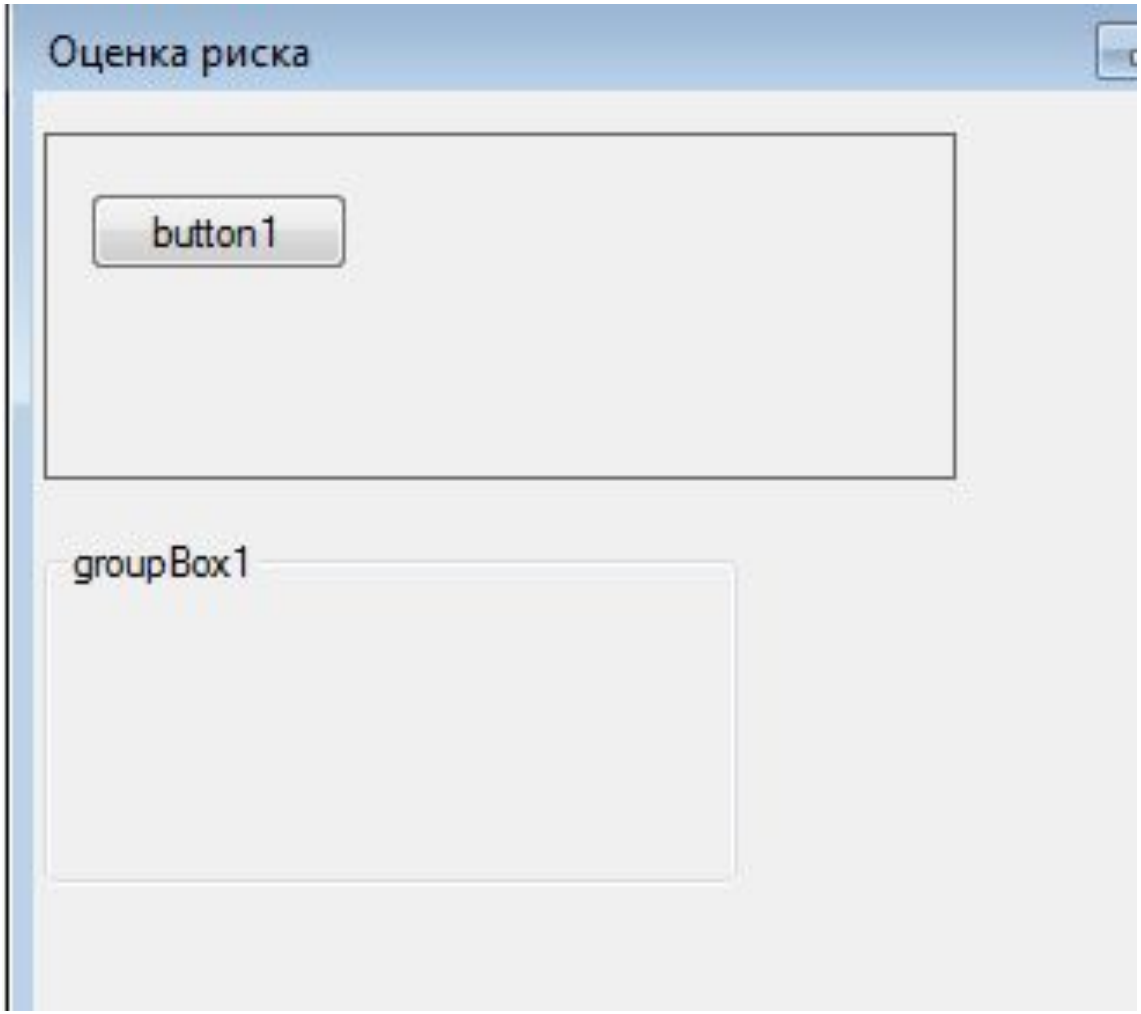


GroupBox и Panel

Часто используемые свойства GroupBox	Описание
Controls	Набор элементов управления, содержащихся в GroupBox
Text	Текст заголовка, отображаемого в верхней части GroupBox

Часто используемые свойства Panel	Описание
AutoScroll	Признак отображения полос прокрутки в том случае, если размеры Panel слишком малы для отображения всего содержимого (по умолчанию используется значение false)
BorderStyle	Тип обрамления Panel. По умолчанию используется значение None; другие допустимые значения — Fixed3D и FixedSingle
Controls	Набор элементов управления, содержащихся в Panel

GroupBox и Panel



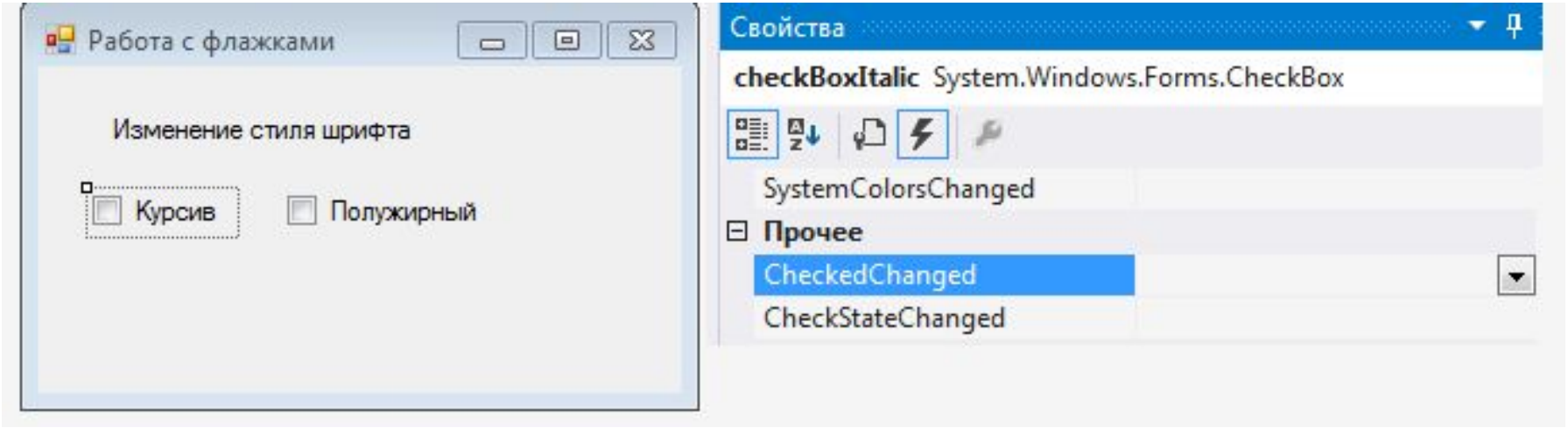
Флажки и переключатели

- C# поддерживаются две разновидности кнопок состояния, которые могут находиться *в состоянии <<вкл/выкл» или «истина/ложь» - флажки (CheckBox) и переключатели* (RadioButton).
- Классы CheckBox и RadioButton, как и класс Button, являются производными от класса ButtonBase.
- Флажки. Флажок (checkBox) представляет собой маленький квадратик, который может быть пустым или содержит пометку. Когда пользователь щелкает на флажке, чтобы установить его, *в квадратике появляется пометка. Повторный щелчок на установленном* флажке снимает пометку.
- Флажок также можно настроить для переключения между тремя состояниями (установленным, снятым и неопределенным); для этого его свойству ThreeState присваивается значение true.
- Любое количество флажков может одновременно находиться *в установленном состоянии.*

Флажки и переключатели

Свойства и события CheckBox	Описание
Checked	Свойство определяет, находится ли флажок в установленном (помеченном) или снятом (пустом) состоянии. Свойство возвращает тип bool. По умолчанию используется значение false (флажок не установлен)
CheckState	Свойство определяет текущее состояние флажка в виде значения из перечисления CheckState (Checked, Unchecked или Indeterminate). Значение Indeterminate используется для неопределенного состояния флажка (то есть не установленного и не снятого). Когда CheckState задается значение Indeterminate, флажок обычно окрашивается в серый цвет
Text	Текст, выводимый справа от флажка
ThreeState	Если это свойство равно true, флажок имеет три состояния — установленное, снятое и неопределенное. По умолчанию свойство равно false, а флажок имеет всего два состояния (снятое и установленное)
Часто используемые события	
CheckedChanged	Генерируется при изменении свойства Checked или CheckState; является событием по умолчанию класса CheckBox. При двойном щелчке на элементе управления CheckBox в режиме конструктора создается пустой обработчик этого события
CheckStateChanged	Генерируется при изменении свойства Checked или CheckState

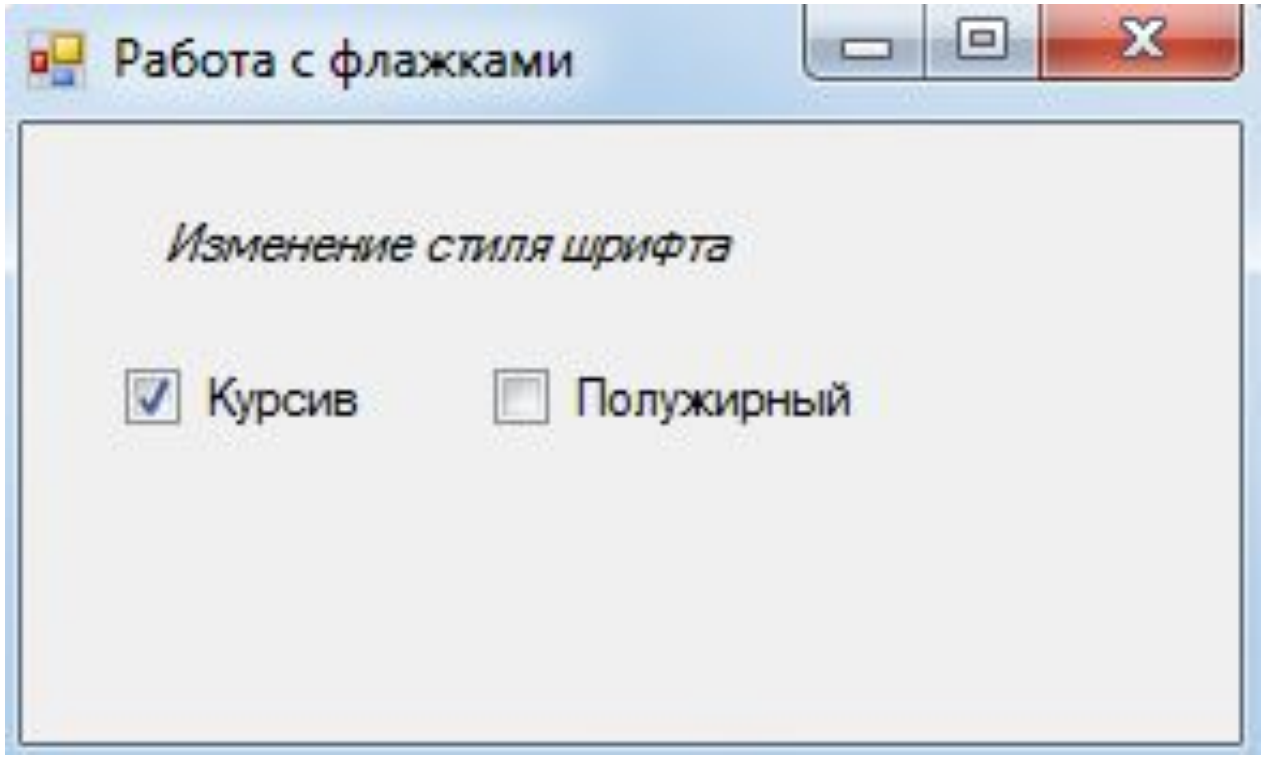
Флажки и переключатели



Флажки и переключатели

```
namespace CheckBox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void checkBoxItalic_CheckedChanged(object sender, EventArgs e)
        {
            labelSample.Font = new Font(labelSample.Font,
   labelSample.Font.Style ^ FontStyle.Italic);
        }
        private void checkBoxBold_CheckedChanged(object sender, EventArgs e)
        {
            labelSample.Font = new Font(labelSample.Font,
   labelSample.Font.Style ^ FontStyle.Bold);
        }
    }
}
```

Флажки и переключатели



Переключатели

- Переключатели (класс RadioButton), как и флажки, могут находиться *в одном из* двух состояний - установленном и снятом.
- Однако переключатели обычно объединяются *в группы, в которых в любой момент времени может быть установлен* только один переключатель.
- При установке одного переключателя *в группе остальные* переключатели снимаются. Таким образом, переключатели используются для представления наборов взаимоисключающих вариантов (то есть наборов, *в которых* одновременное выделение нескольких вариантов невозможно).

Переключатели

Свойства и событие RadioButton	Описание
Часто используемые свойства	
Checked	Свойство определяет, находится ли переключатель в установленном состоянии
Text	Текст, выводимый рядом с переключателем
Часто используемое событие	
CheckedChanged	Генерируется при установлении или снятии переключателя. При двойном щелчке на элементе управления RadioButton в режиме конструктора создается пустой обработчик этого события

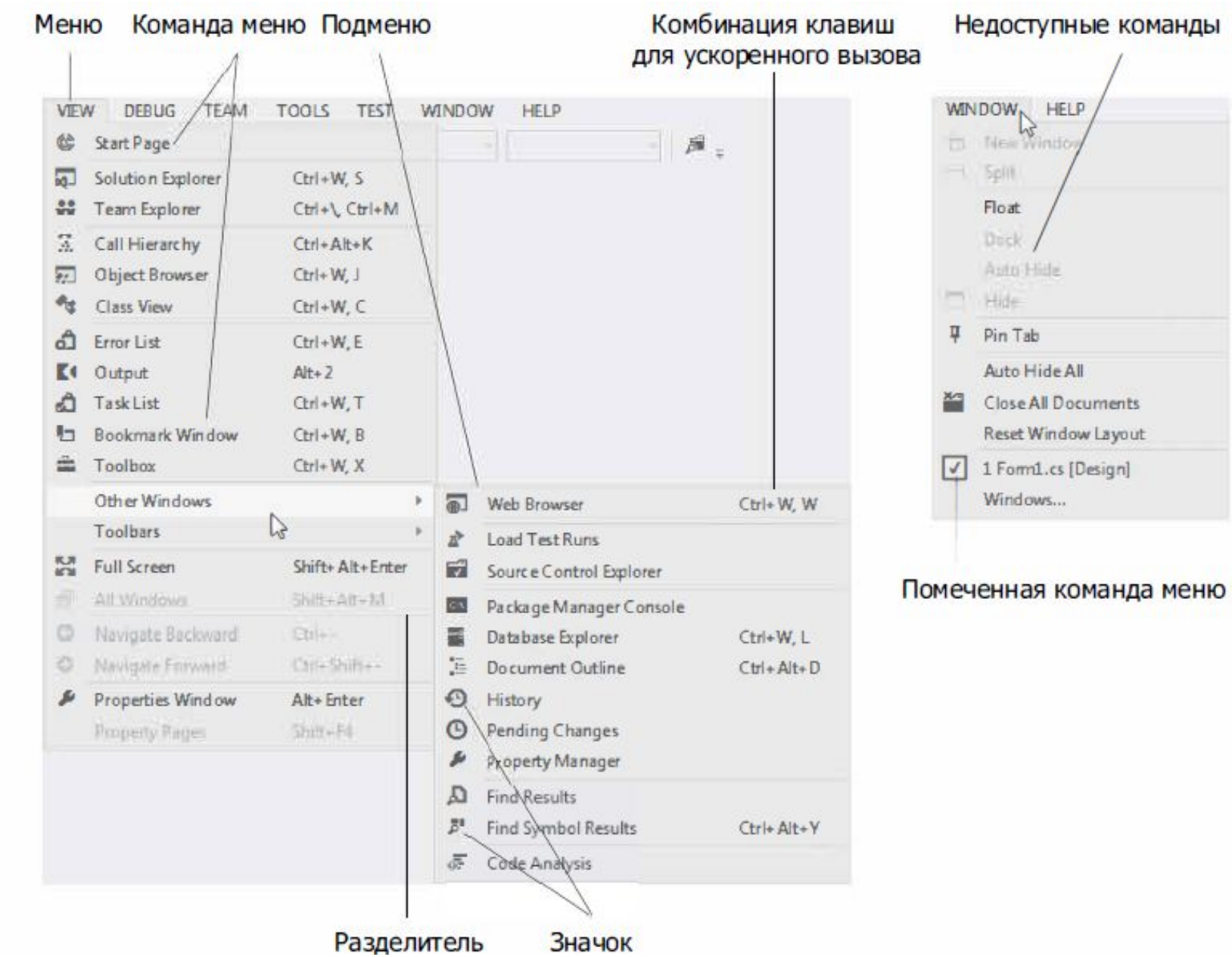
Переключатели

```
private void btnCalc_Click(object sender, EventArgs e)
        {
           float P, U, R;
           try
            {
         P = Convert.ToSingle(txtBoxP.Text);
            }
            catch (Exception ex)
            {

                MessageBox.Show("Не корректное значение оценки вероятности\n"+ex.Message);
                txtBoxP.Focus();
                return;
            }
            if (P < 0 || P > 1)
            {
                MessageBox.Show("Не корректное значение оценки вероятности");
                txtBoxP.Focus();
                return;
            }
            try
            {
                U = Convert.ToSingle(txtBoxU.Text);
            }
            catch (Exception ex)
            {

                MessageBox.Show("Не корректное значение оценки ущерба\n" + ex.Message);
                txtBoxU.Focus();
                return;
            }
            if (U <= 0 )
            {
```

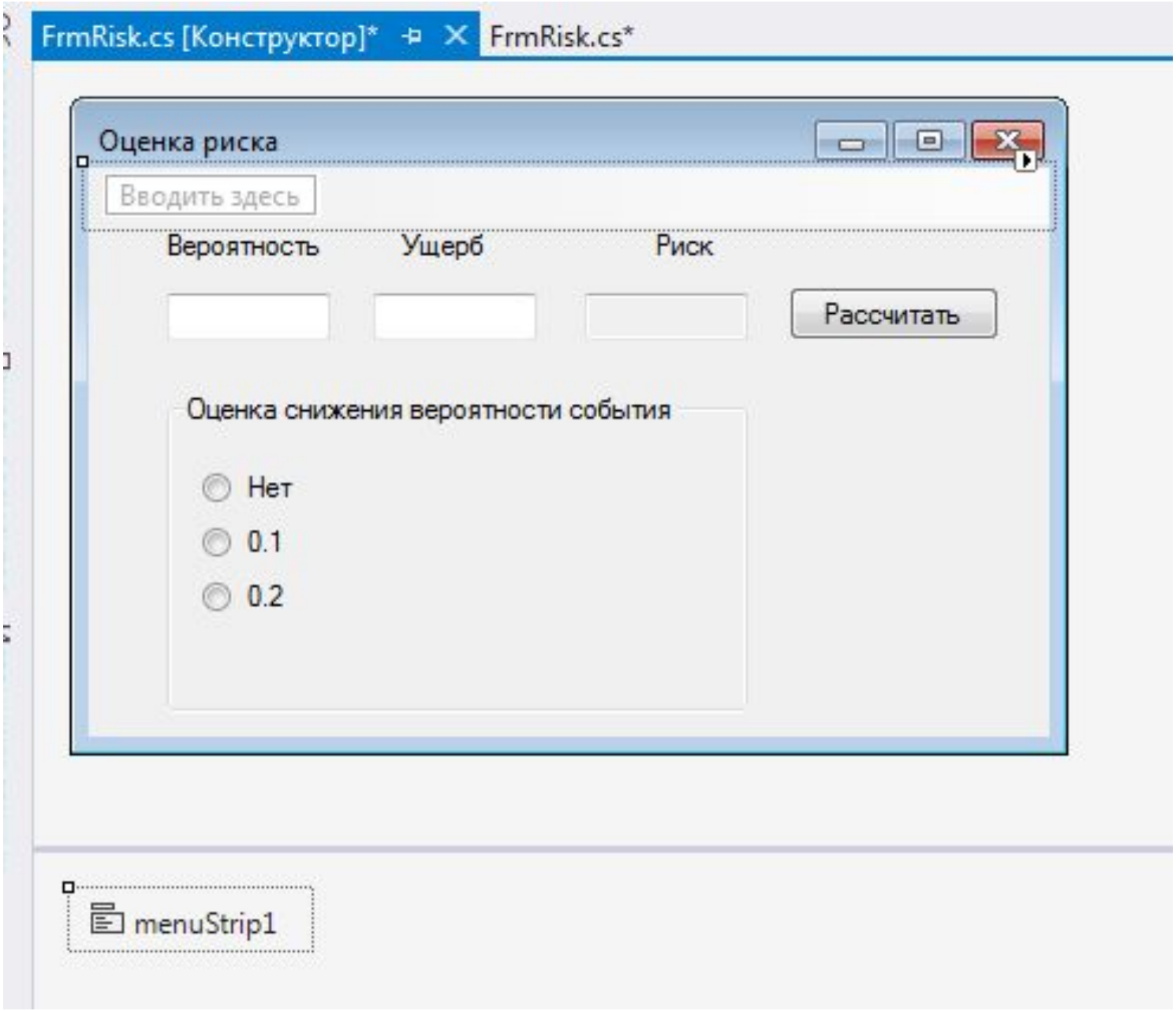
Меню

- *Меню предназначены для группировки взаимосвязанных команд в приложениях* Windows Forms. Хотя выбор команд зависит от конкретной программы, некоторые команды - такие, как Open и Save - присутствуют *во многих приложениях.*
- *Меню* являются неотъемлемой частью графических интерфейсов, потому что они позволяют группировать команды без загромождения интерфейса.
- Меню могут назначаться *Alt-комбинации клавиш, состоящие из клавиши Alt* и буквы, подчеркнутой *в названии меню, - например, комбинация Alt+ F обычно* открывает меню File.
- Командам меню тоже могут назначаться комбинации клавиш для ускоренного вызова (комбинации Ctrl, Shift, Alt, F1, F2, букв и т. д.).
- Некоторые команды меню выводятся с пометками, которые обычно означают, что несколько команд меню могут быть выбраны одновременно.

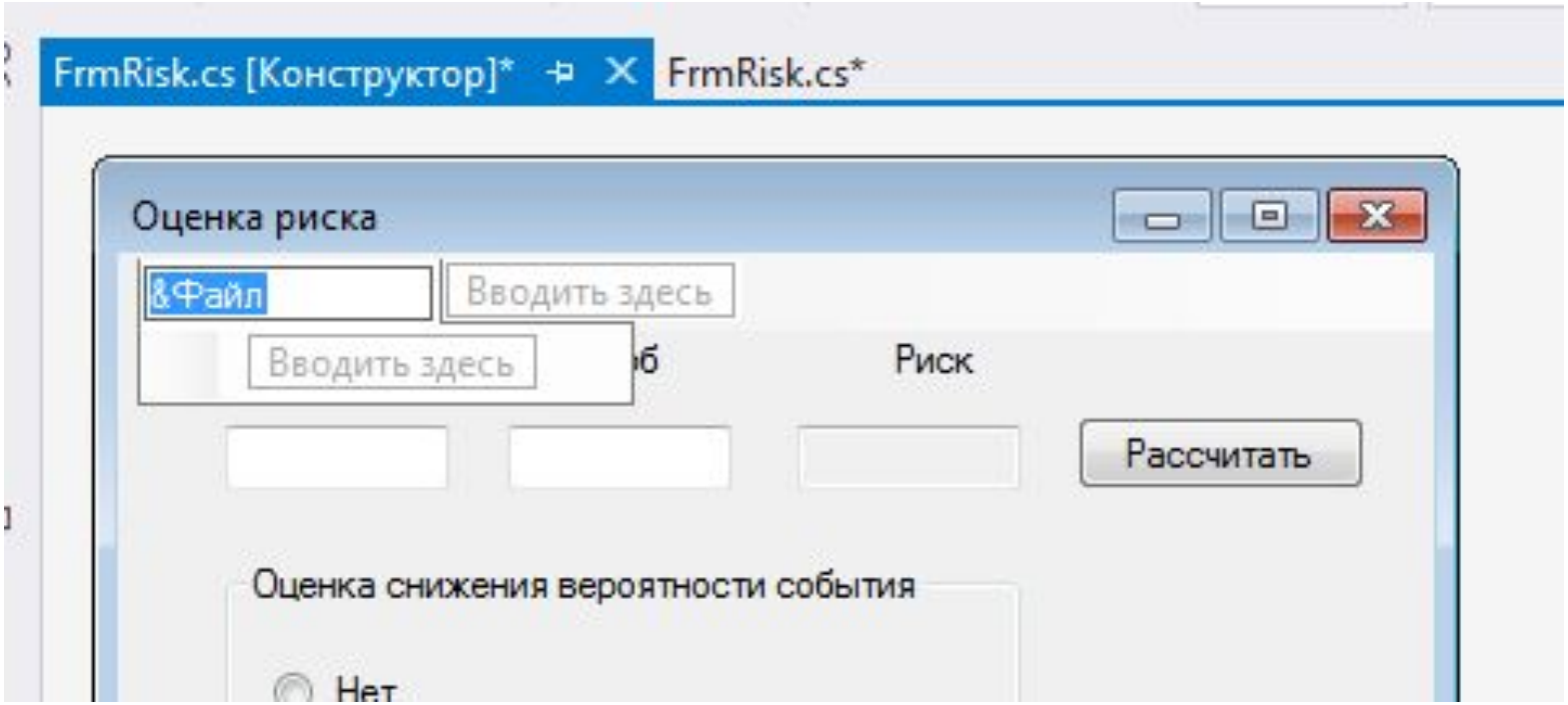
Меню


Меню
Команда меню
Подменю
Комбинация клавиш для ускоренного вызова
Недоступные команды
VIEW DEBUG TEAM TOOLS TEST WINDOW HELP
Start Page
Solution Explorer Ctrl+W, S
Team Explorer Ctrl+\, Ctrl+M
Call Hierarchy Ctrl+Alt+K
Object Browser Ctrl+W, J
Class View Ctrl+W, C
Error List Ctrl+W, E
Output Alt+2
Task List Ctrl+W, T
Bookmark Window Ctrl+W, B
Toolbox Ctrl+W, X
Other Windows
Toolbars
Full Screen Shift+Alt+Enter
All Windows Shift+Alt+M
Navigate Backward Ctrl+-
Navigate Forward Ctrl+Shift+-
Properties Window Alt+Enter
Property Pages Shift+F4
Web Browser Ctrl+W, W
Load Test Runs
Source Control Explorer
Package Manager Console
Database Explorer Ctrl+W, L
Document Outline Ctrl+Alt+D
History
Pending Changes
Property Manager
Find Results
Find Symbol Results Ctrl+Alt+Y
Code Analysis
WINDOW HELP
New Window
Split
Float
Dock
Auto Hide
Hide
Pin Tab
Auto Hide All
Close All Documents
Reset Window Layout
1 Form1.cs [Design]
Windows...
Помеченная команда меню
Разделитель
Значок

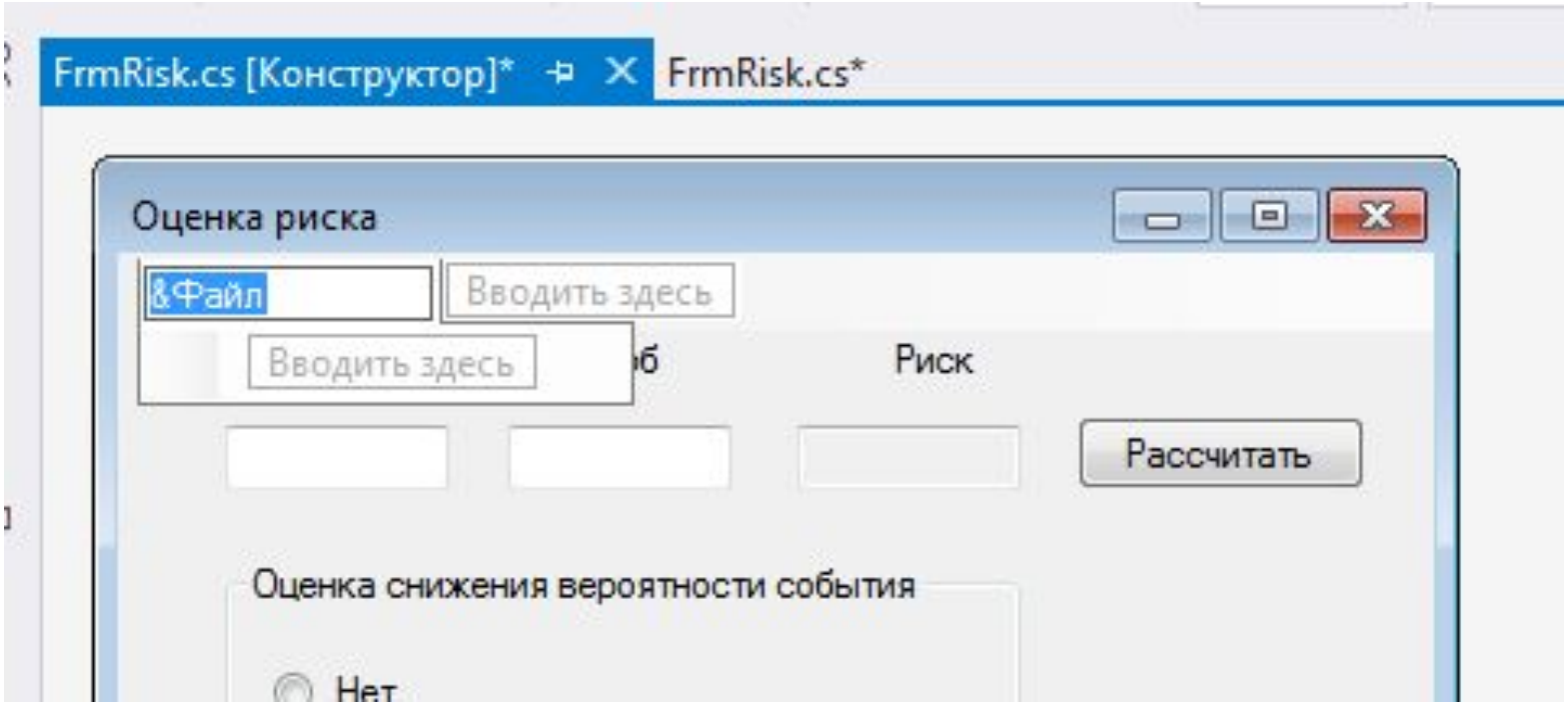
Меню


FrmRisk.cs [Конструктор]*
FrmRisk.cs*
Оценка риска
Вводить здесь
Вероятность
Ущерб
Риск
Рассчитать
Оценка снижения вероятности события
Нет
0.1
0.2
menuStrip1

Меню


FrmRisk.cs [Конструктор]*
FrmRisk.cs*
Оценка риска
&Файл
Вводить здесь
Вводить здесь
Риск
Рассчитать
Оценка снижения вероятности события
Нет

Меню


FrmRisk.cs [Конструктор]*
FrmRisk.cs*
Оценка риска
&Файл
Вводить здесь
Вводить здесь
Риск
Рассчитать
Оценка снижения вероятности события
Нет