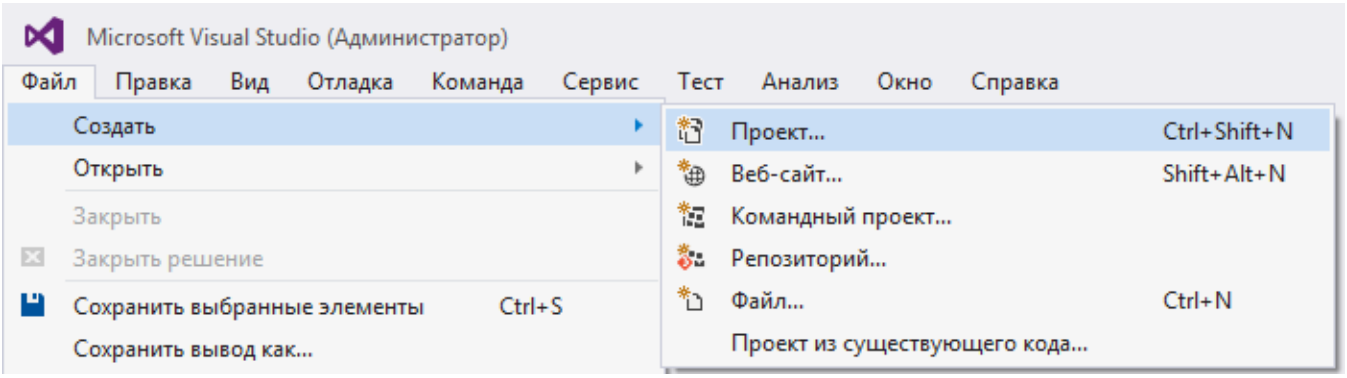
Лабораторная работа № 1
«Калькулятор»

1. Создаем проект



В открывшемся окне:

Шаблоны – **Visual C#**

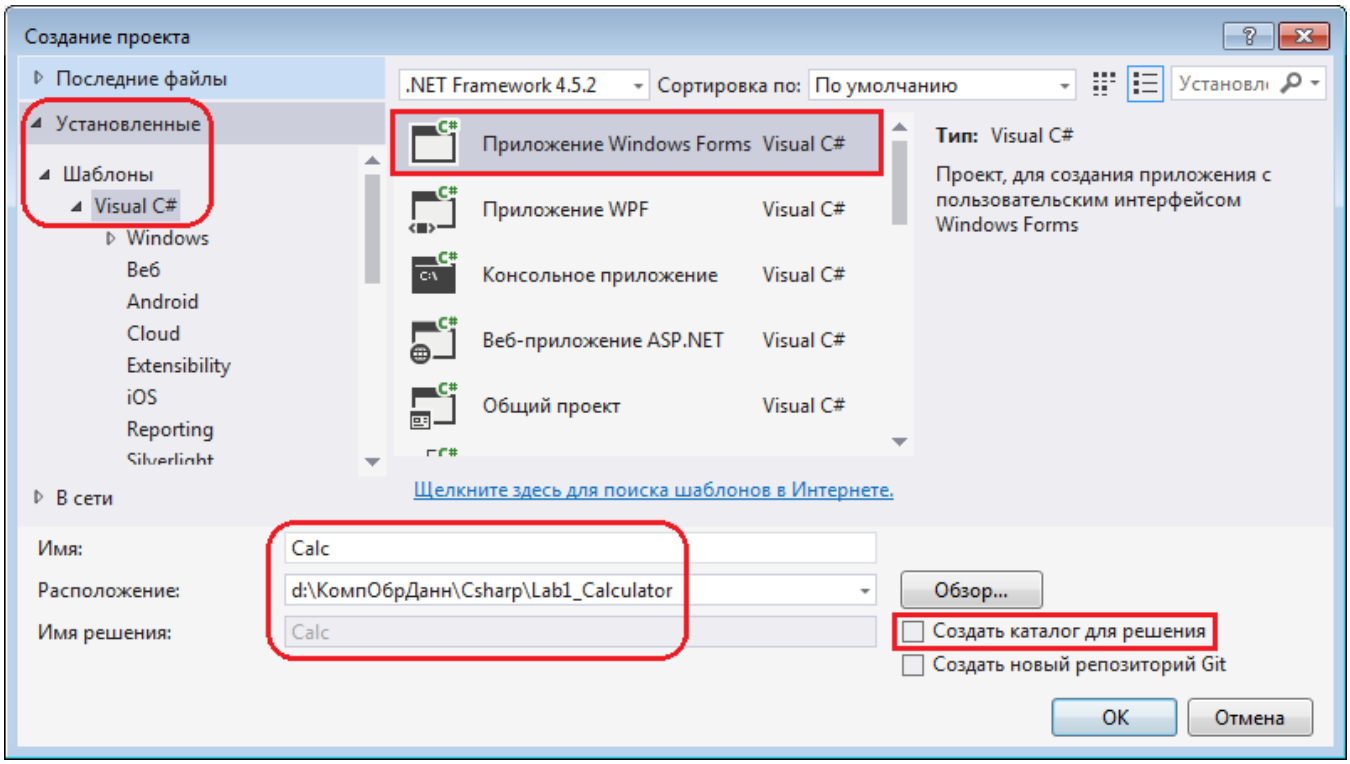
Шаблон – **Приложение Windows Forms**

Имя – **Calc**

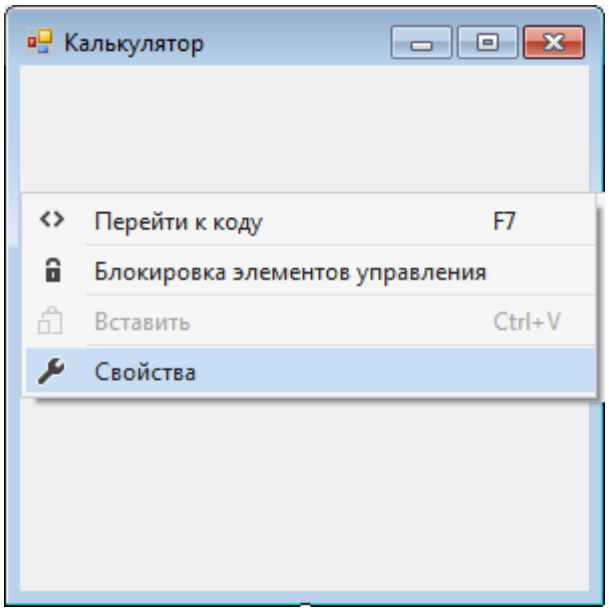
Расположение - **d:\КомпОбрДанн\Csharp\Lab1_Calculator**

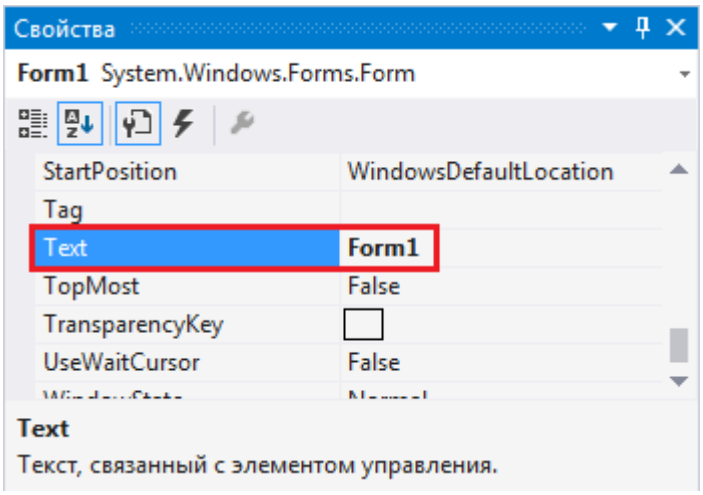
Создать каталог для решения – убрать галочку

Нажать **ОК**.



2. Изменим свойства формы: Правый клик по форме – **Свойства** (**Properties**)

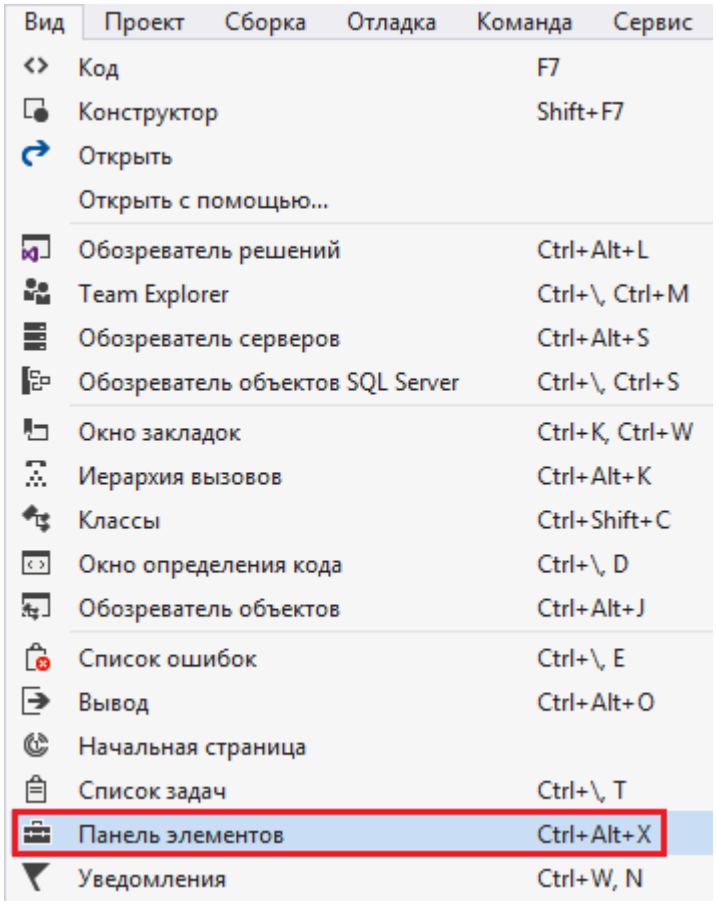


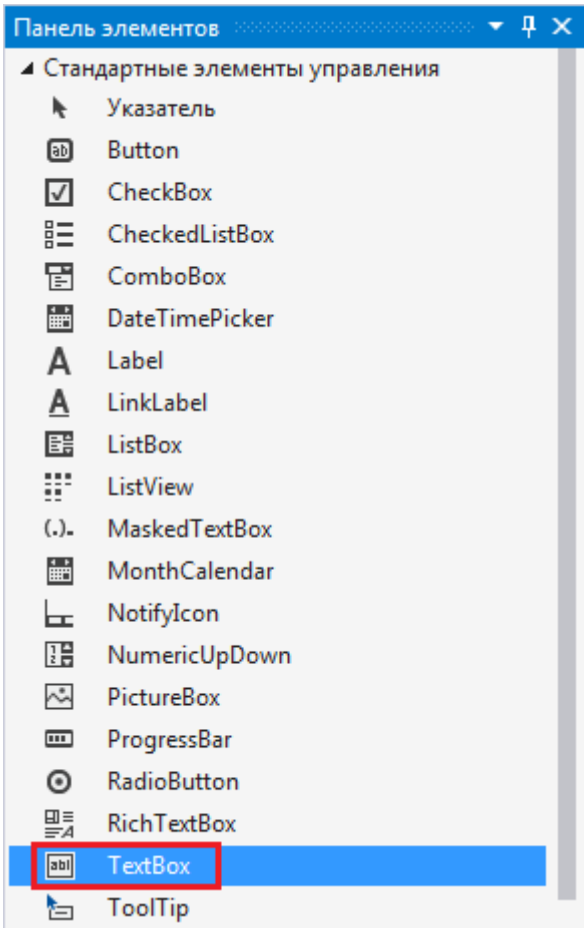


Меняем свойство **Текст** (**Text**) с **Form1** на **Калькулятор**, нажимаем Enter (поменялся заголовок окна на форме).

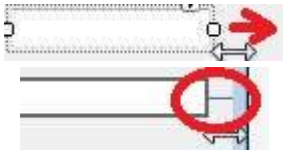
В этом же окне находим свойство **(Name)** и меняем на **frmCalc** (изменилось имя объекта, которое используется в коде программы).

3. Создаем Поле для ввода: **Вид** – **Панель элементов** (**Toolbox**) – на **Панели элементов** выбираем элемент **TextBox** – мышкой перетаскиваем этот элемент на форму.

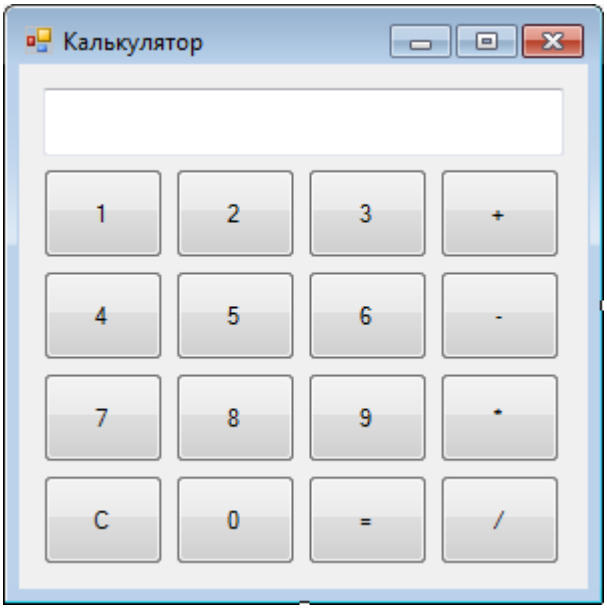




4. Так же как со свойствами формы, меняем свойства **TextBox**'а:
   (Name) – tbNumber
   Enabled - **False**
   textAlign – Center

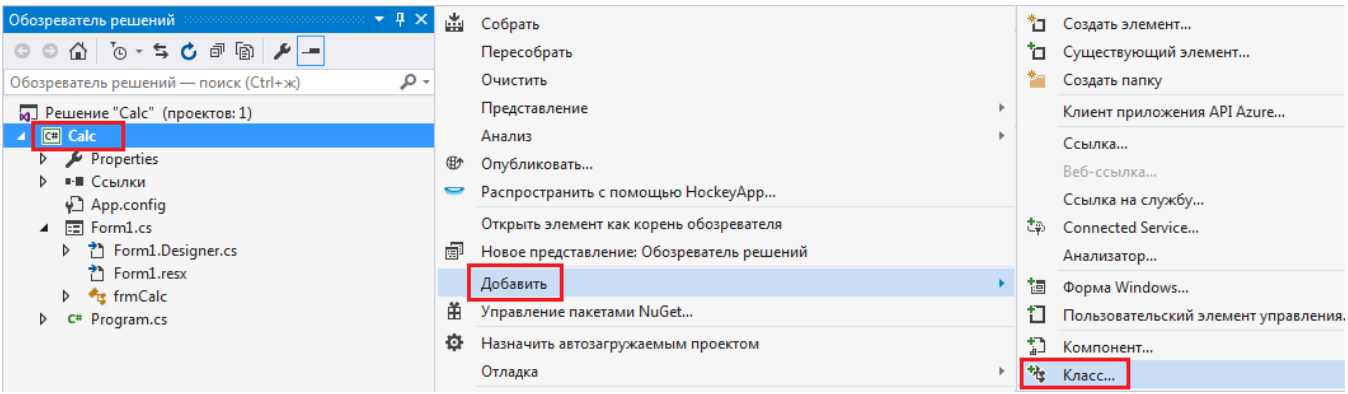
   Растягиваем текстбокс на всю ширину окна: 

5. Добавляем кнопки: в **Панели элементов** находим элемент **Button** и мышкой перетаскиваем его на форму.
   Чтобы изменить надпись на кнопке – меняем свойство **Text**, например на **+**.
   Добавляем кнопки **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **0**, **+**, **-**, **/**, *****, **C**.

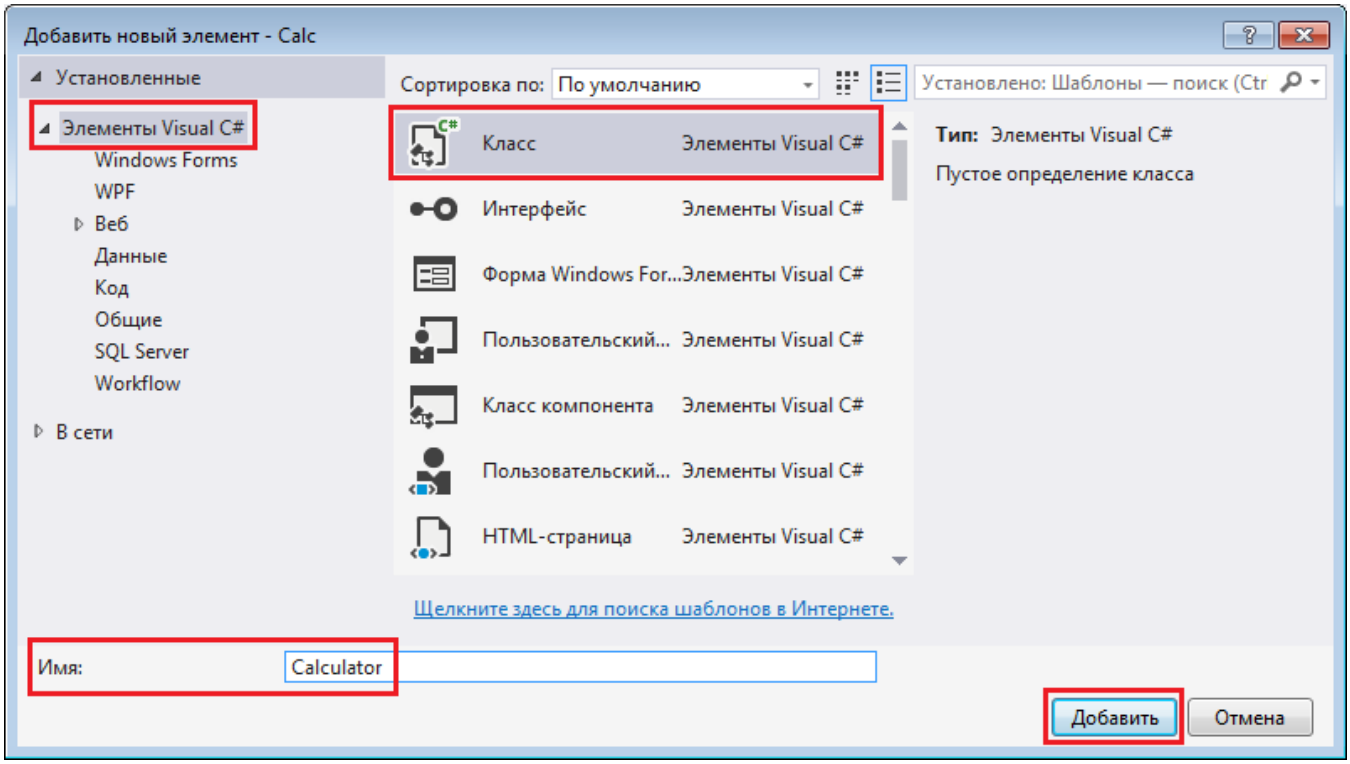


   Форма создана. Теперь надо написать код программы.

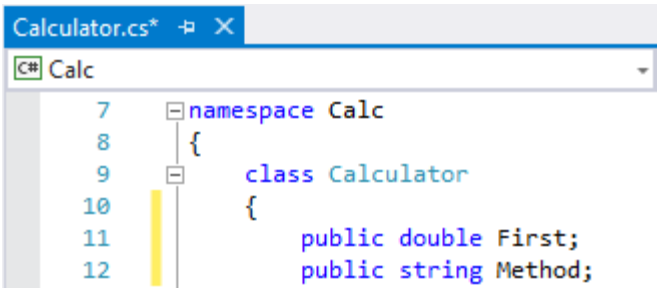
6. Создаем класс **Calculator**: в окне **Обозревателя решений** правый клик на названии проекта – **Добавить** – **Класс**



Зададим имя классу: **Calculator**



7. В этот класс **Calculator** добавим два свойства: число **First** и строку **Method**



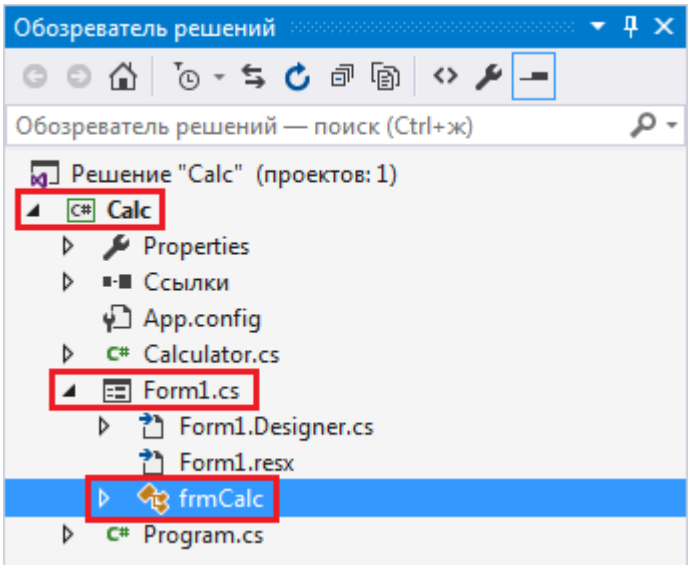
8. Научим калькулятор считать.

Добавим в класс **Calculator** метод **Action**.

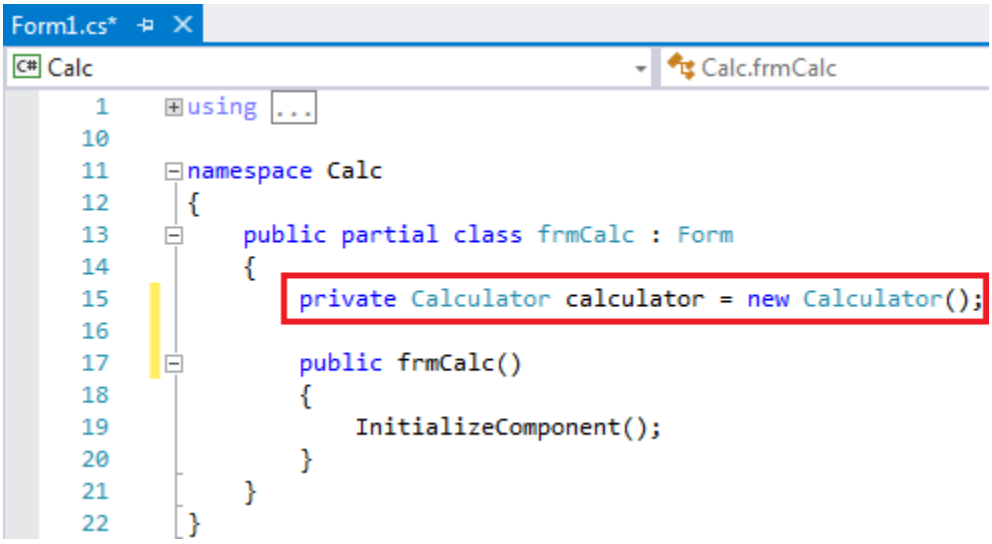
```
namespace Calc
{
    class Calculator
    {
        public double First;
        public string Method;

        public double Action(double second)
        {
            switch (Method)
            {
                case "+": return First + second;
                case "-": return First - second;
                case "*": return First * second;
                default: return First / second;
            }
        }
    }
}
```

9. Соединим форму и созданный класс **Calculator**:

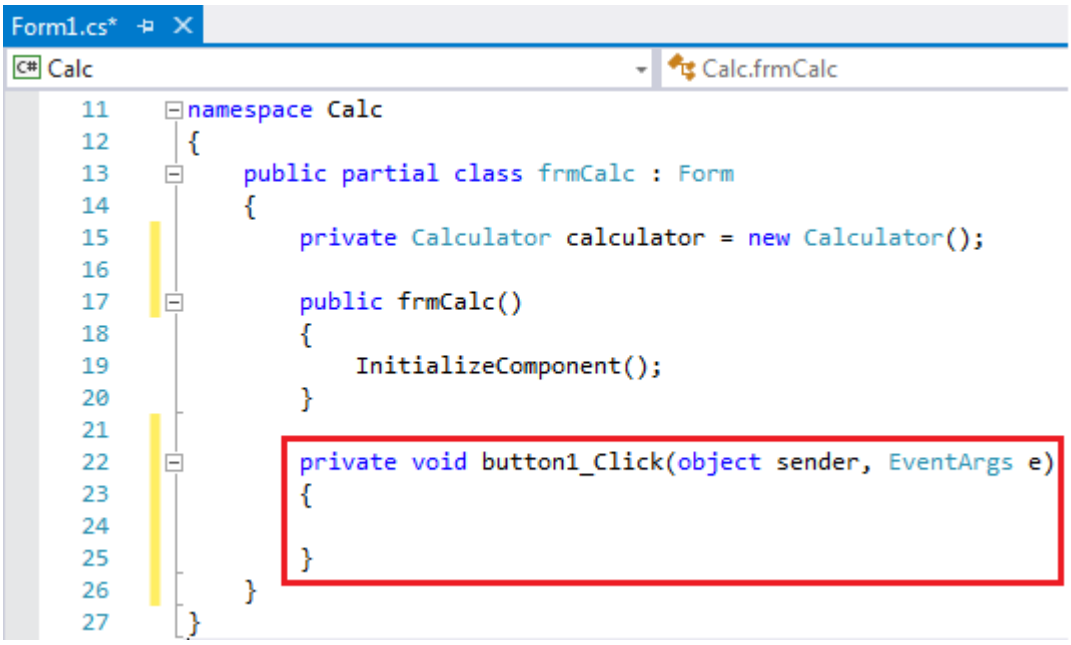
В **Обозревателе решений**: имя проекта **Calc** – **Form1.cs** – двойной клик мышью на классе **frmCalc**.



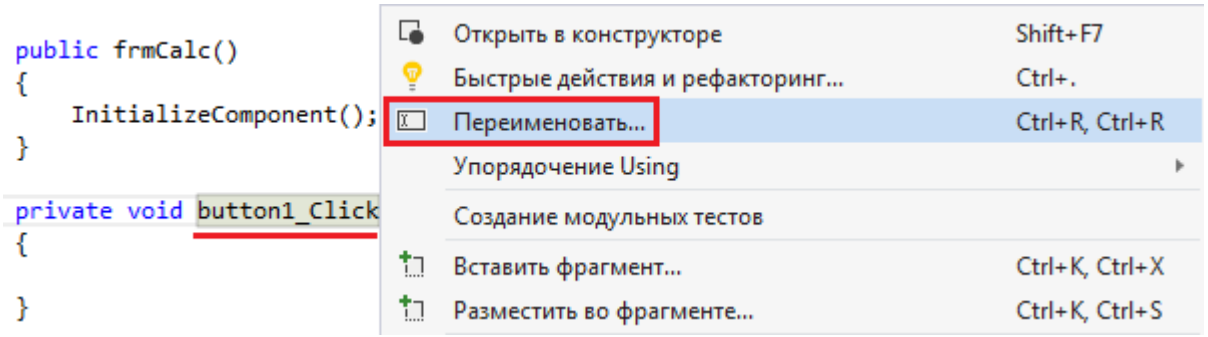
Добавляем экземпляр **Calculator** в класс **frmCalc**



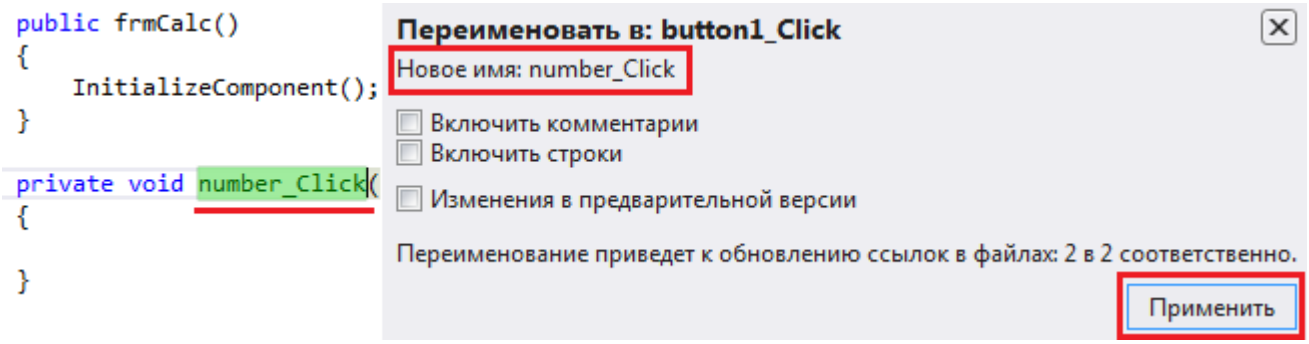
10.Обрабатываем нажатия на кнопки цифр: двойной клик по кнопке **1** на форме: в классе **frmCalc** автоматически добавится пустой метод **button1_Click**.



Изменим название метода: правый клик по названию – **Рефракторинг** – **Переименовать** (в новых версиях: правый клик – **Переименовать**):



Меняем название на **number_Click**:



Добавим логику в метод:

```
private void number_Click(object sender, EventArgs e)
{
    tbNumber.Text = tbNumber.Text + (sender as Button).Text;
}
```

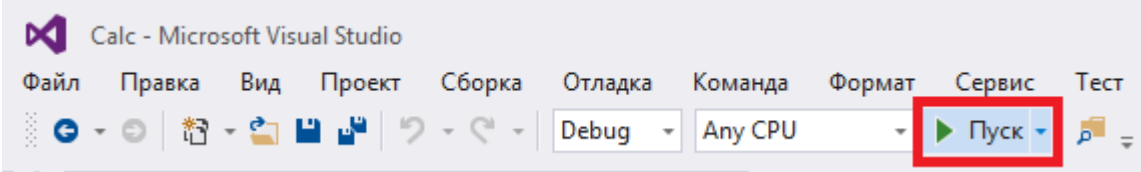
Пояснение:

tbNumber – тот самый элемент **TextBox**, который был добавлен на форму в самом начале, в него пользователь будет вводить цифры для подсчета, там же будет отображаться результат вычислений.

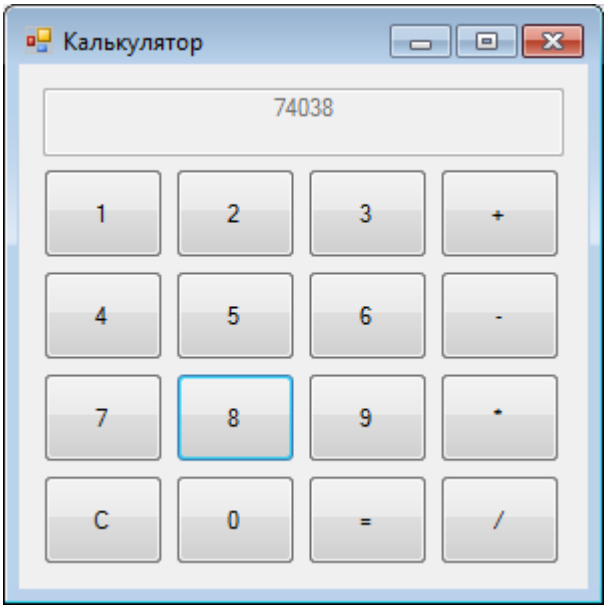
Свойство **Text** у элемента **tbNumber**: запись **tbNumber.Text =** означает, что происходит обращение к элементу **TextBox** с именем **tbNumber** и присваивание свойству **Text** некоторого значения.

В правой части выражения снова берем текст текстбокса и добавляем к нему текст с нажатой кнопки. Например, если ранее у нас уже была нажата цифра 4, а потом пользователь нажал еще цифру 5, то данное выражение будет выглядеть так: «4» + «5», т.е. «45».

Нажатием в Visual Studio на кнопку **Пуск** (или **F5** на клавиатуре) запускаем отладку и компиляцию проекта.

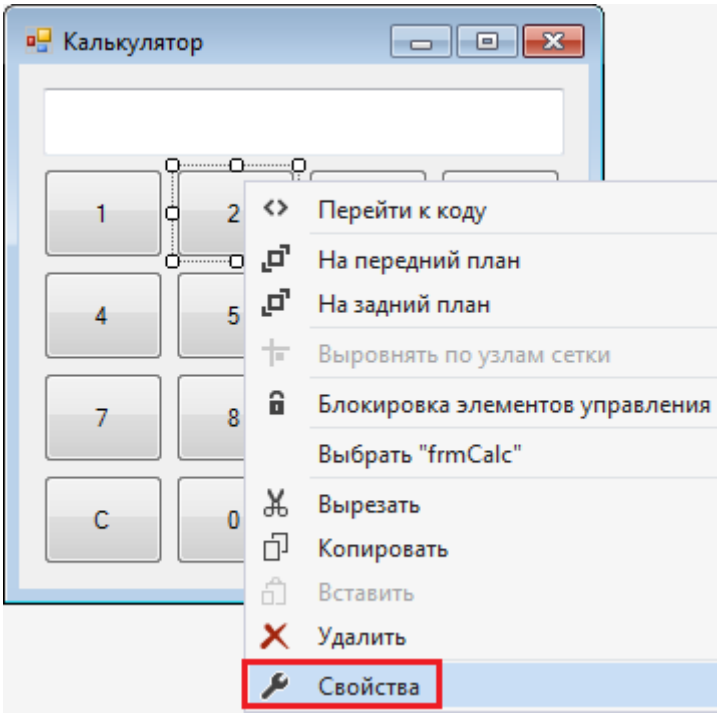


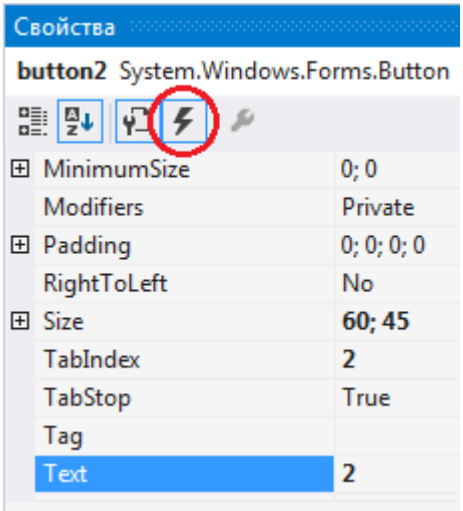
Проверяем, что работает ввод чисел в окне калькулятора.



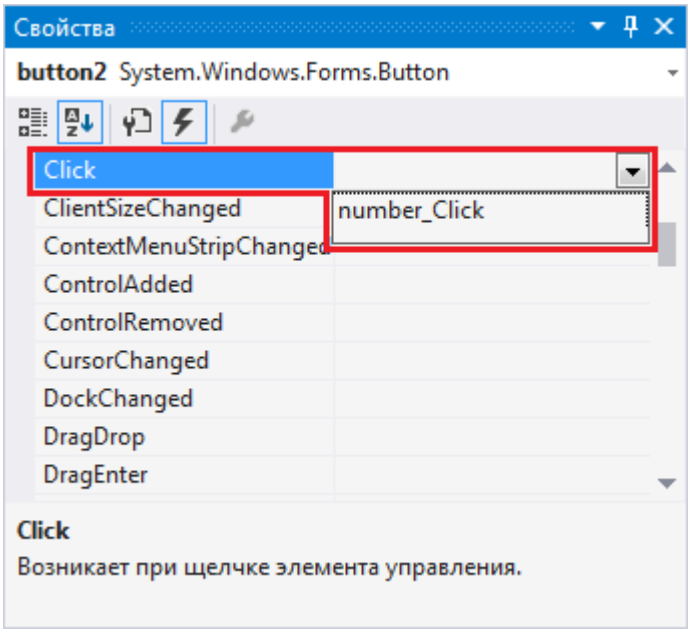
В принципе, можно было бы создать такой метод для каждой кнопки, но это, со всей очевидностью, долго и неэффективно.

Поэтому вернемся к редактированию формы и обратимся к свойствам следующей кнопки – **2**. В окне **Свойства** (**Properties**) нажимаем на кнопку **События** (с молнией)





В меню управления событиями, в левом столбце ищем событие **Click** (которое отвечает за клик по кнопке) и справа в выпадающем меню выбираем метод **number_Click**. Аналогичным образом устанавливаем этот метод для всех остальных кнопок с числами (0-9).



11. Обработчик для кнопки **Отмена** (**C**).

Двойной клик по кнопке **C**, переименовываем метод в **C_Click** и пишем:

```
private void C_Click(object sender, EventArgs e)
{
    calculator = new Calculator();
    tbNumber.Text = "";
}
```

Пояснение:

Здесь стираются все данные, которые хранятся в данный момент в памяти программы. Первая строчка обновляет (по сути, пересоздает) объект **Calculator**, можно было бы отдельно обратиться к полям **First** и **Method** и установить им пустые/нулевые значения, но в данном случае это произойдёт автоматически.

В строке с изменением объекта **tbNumber** просто стирается все, что написано в этом текстбоксе в данный момент, т.е. содержимое текстового поля (текстбокса) принимает значение «пусто».

Проверим: запускаем отладку программы, вводим несколько цифр и нажимаем кнопку **C**. Цифры в текстбоксе должны исчезнуть.

12.Теперь надо создать обработчики нажатий на кнопки **+**, **-**, ***** и **/**.

Двойной клик по кнопке **+**, переименовываем метод в **Action_Click** и пишем в теле данного метода:

```
private void Action_Click(object sender, EventArgs e)
{
    calculator.Method = (sender as Button).Text;
    calculator.First = Convert.ToDouble(tbNumber.Text);
    tbNumber.Text = "";
}
```

Пояснение:

В первой стоке передаем экземпляру класса **Calculator** (т.е. методу **Method** данного экземпляра) тип операции, которую будем выполнять. **(sender as Button)** – это обращение к объекту, который вызвал метод **Action_Click**. Мы понимаем, что это всегда будут кнопки (тип **Button**), но вот программа об этом не знает, поэтому явно указываем, что **sender** (кнопка, которая обращается к методу **Action_Click**) именно кнопка, а не текстбокс, например. Дальше уже знакомое обращение к полю **Text** нажатой кнопки, именно текст кнопки и будет передан объекту **calculator**, в данном случае это будет текст «**+**».

Второй строкой передаем объекту **calculator** первое число. Так как это число представлено в формате текста, то необходимо вызвать метод **Convert.ToDouble()**, чтобы перевести его в числовой формат.

Возвращаемся к редактированию формы и устанавливаем событие **Click** – **Action_Click** для кнопок **-**, ***** и **/**.

Запускаем отладку программы, проверяем, что все работает.

13.Наконец, создаем событие **Result_Click** для кнопки **=**. Двойной клик по ней и:

```
private void Result_Click(object sender, EventArgs e)
{
    double second = Convert.ToDouble(tbNumber.Text);
    double result = calculator.Action(second);
    tbNumber.Text = result.ToString();
}
```

Пояснение:

1) Получаем текст в строке ввода чисел, конвертируем его в числовой формат.
2) Обращаемся к методу **Action** объекта **calculator**, передаем ему второе число и получаем в ответ результат вычислений.
3) Выводим результат на форму приложения.

Алгоритм работы программы

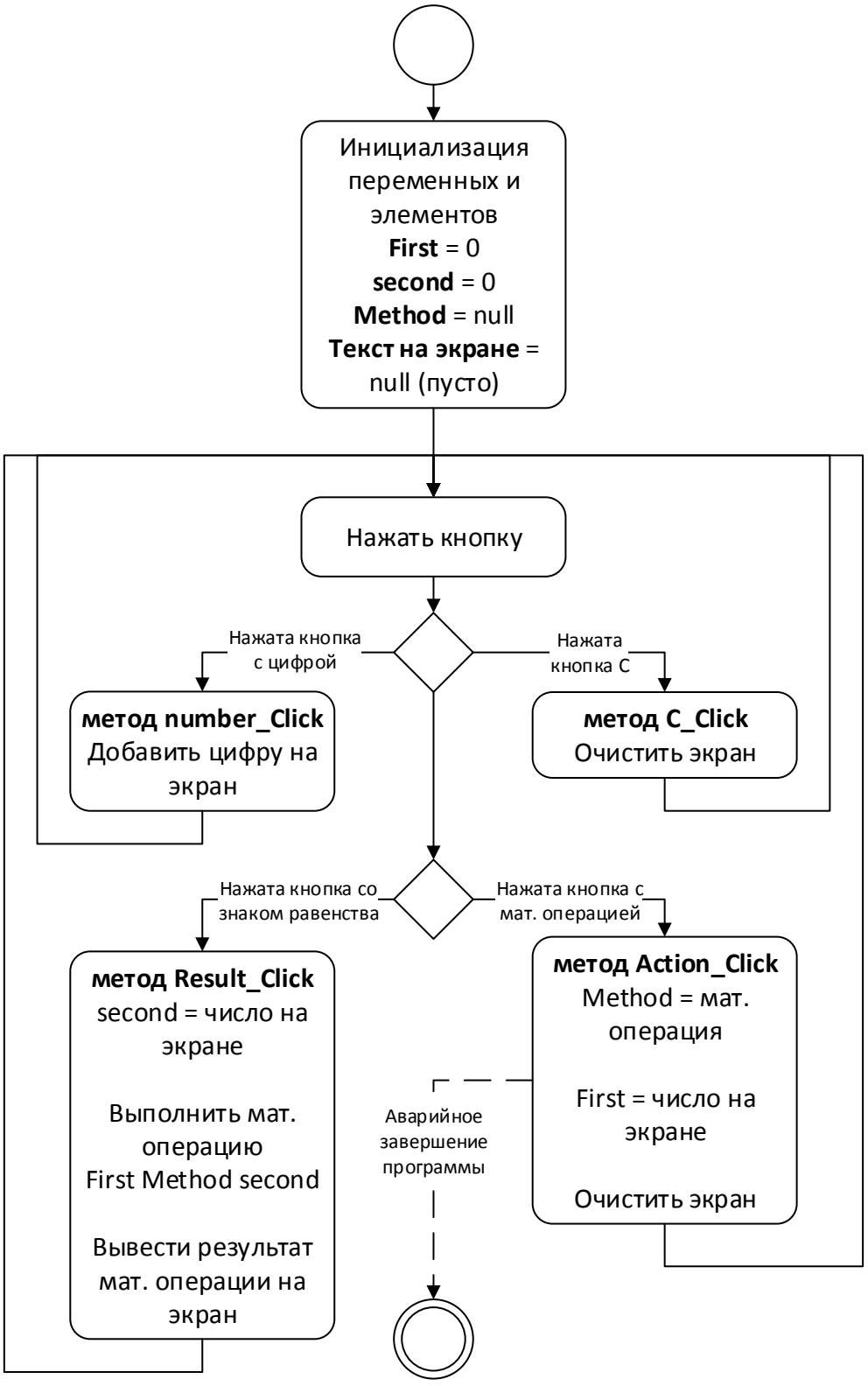
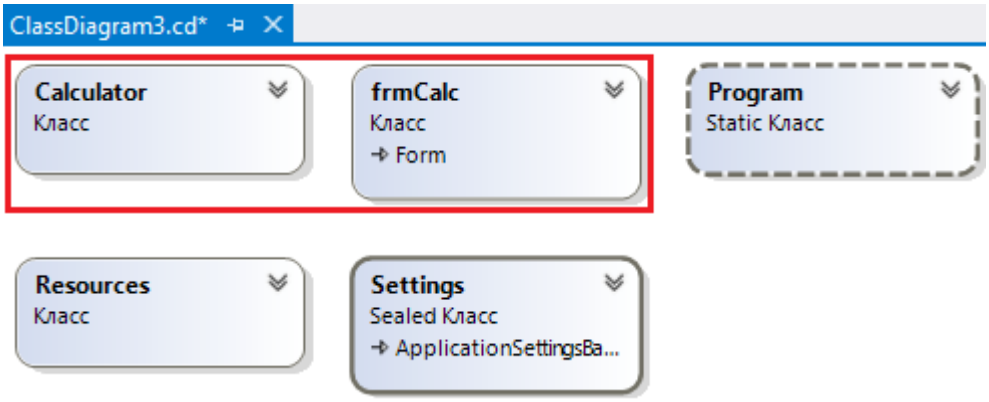
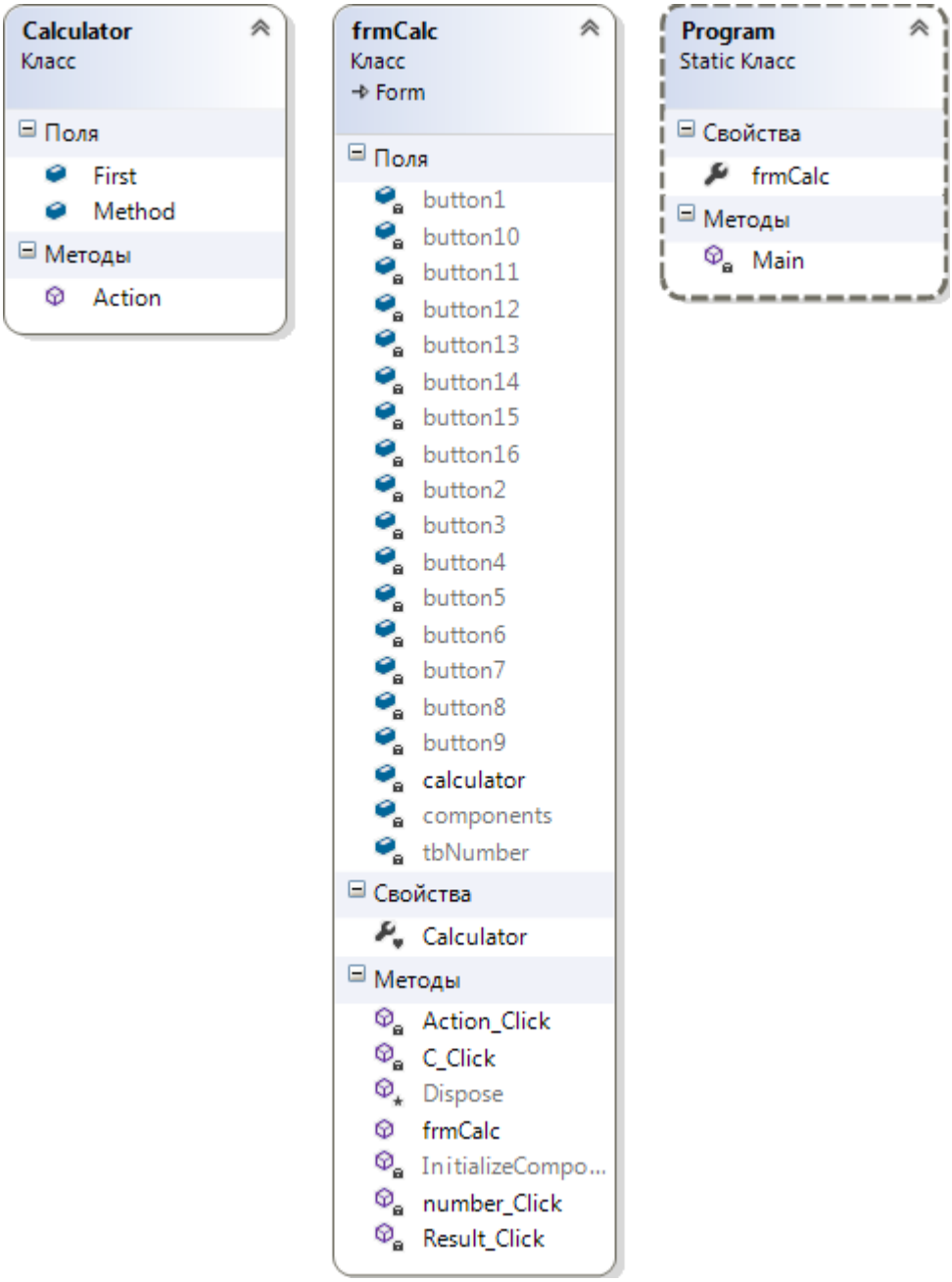


Диаграмма классов

Классы **Calculator** и **frmCalc** созданы пользователем. Остальные классы (**Program**, **Resources**, **Settings**) генерируются автоматически при создании проекта.



Структура классов



Улучшение программы

Чтобы не выполнять недопустимые операции (арифметические, конвертирование из текста в число, и т.д.), в том числе операции с несуществующими числами, можно предварительно проверить текстбокс **tbNumber** (текстовое поле) формы **frmCalc** на наличие или отсутствие в нём числа, и отсутствующее число заменить нулём.

В данном случае необходимо использовать метод **String.IsNullOrEmpty** из пространства имён **System** (`using System;` автоматически генерируется при создании новых классов). Метод проверяет, является ли указанная строка строкой **null** (несуществующая строка) или **Empty** (пустая, но существующая строка).

```
private void Action_Click(object sender, EventArgs e)
{
    calculator.Method = (sender as Button).Text;
    calculator.First = Convert.ToDouble(String.IsNullOrEmpty(tbNumber.Text) ? "0" :
        tbNumber.Text);
    tbNumber.Text = "";
}

private void Result_Click(object sender, EventArgs e)
{
    double second = Convert.ToDouble(String.IsNullOrEmpty(tbNumber.Text) ? "0" :
        tbNumber.Text);
    double result = calculator.Action(second);
    tbNumber.Text = result.ToString();
}
```

Данный код идентичен коду:

```
private void Action_Click(object sender, EventArgs e)
{
    calculator.Method = (sender as Button).Text;
    if (String.IsNullOrEmpty(tbNumber.Text) == true)
    {
        calculator.First = Convert.ToDouble("0");
    }
    else
    {
        calculator.First = Convert.ToDouble(tbNumber.Text);
    }
    tbNumber.Text = "";
}

private void Result_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(tbNumber.Text) == true)
    {
        double second = Convert.ToDouble("0");
    }
    else
    {
        double second = Convert.ToDouble(tbNumber.Text);
    }
    double result = calculator.Action(second);
    tbNumber.Text = result.ToString();
}
```