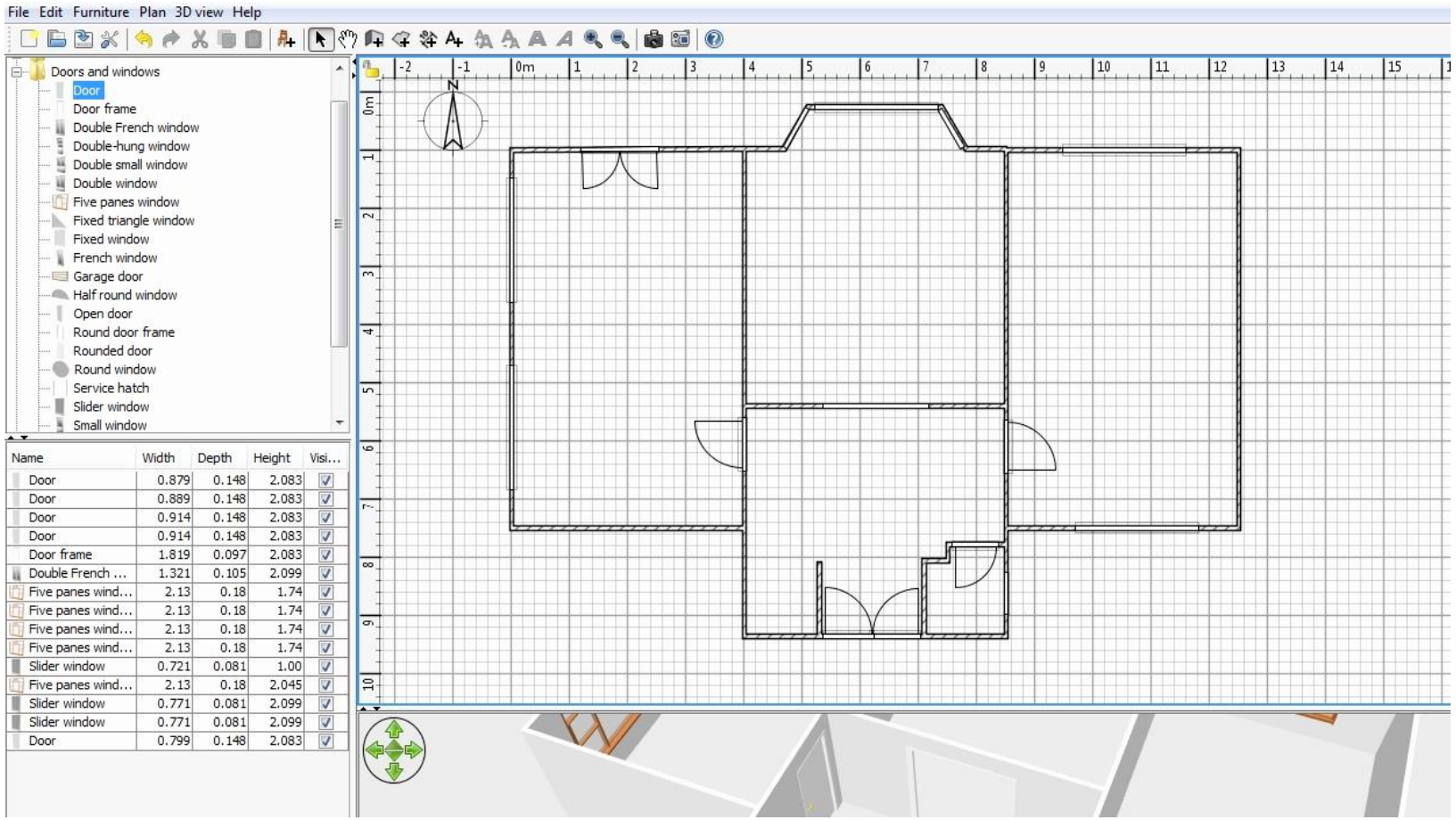
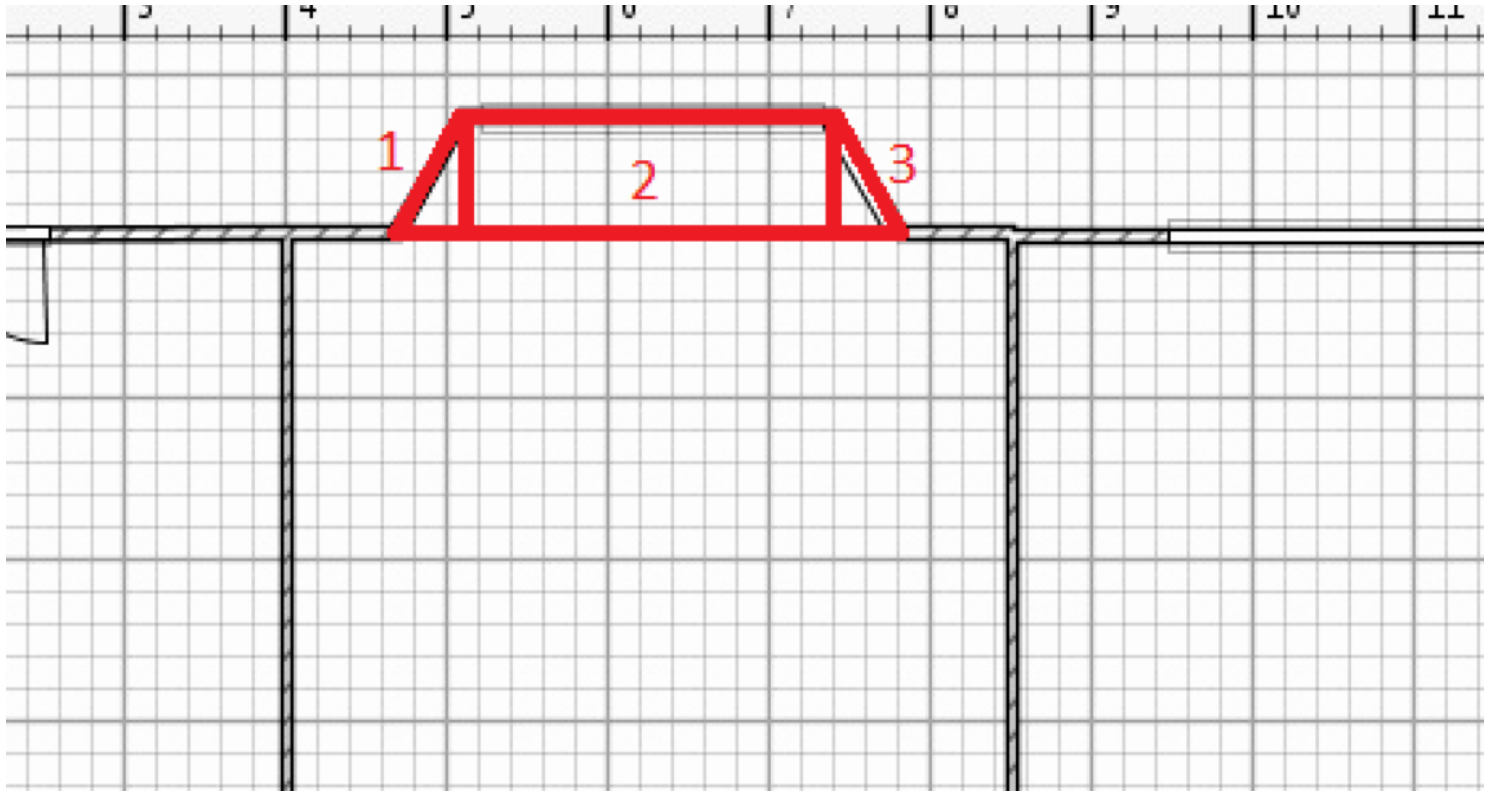
House design software



Сцена



Класове за основни фигури

```
class Triangle
{
    int x, h;

    public double CalculateSurface()
    {
        return x * h / 2;
    }
}
```

```
class Rectangle
{
    int x, y;

    public double CalculateSurface()
    {
        return x * y;
    }
}
```

Клас за сцена

```
class Scene
{
    private List<Triangle> _triangles =
        new List<Triangle>();

    private List<Rectangle> _rectangles =
        new List<Rectangle>();


    public void AddTriangle(Triangle triangle)
    {
        _triangles.Add(triangle);
    }
```

```
    public void AddRectangle(Rectangle rectangle)
    {
        _rectangles.Add(rectangle);
    }


    public double CalculateSurface()
    {
        var surface = 0.0;

        foreach (var triangle in _triangles)
            surface += triangle.CalculateSurface();

        foreach (var rectangle in _rectangles)
            surface += rectangle.CalculateSurface();

        return surface;
    }
```

Виртуални методи

- Определение: Методи които могат да се пренапишат в клас наследник

Базов клас за фигури

```
class Shape
{
    protected int x, y;


    public virtual double CalculateSurface()
    {
        throw new NotImplementedException();
    }
}
```

Класове за основни фигури

```
class Triangle : Shape
{

    public override double CalculateSurface()
    {
        return x * y / 2;
    }
}
```

```
class Rectangle : Shape
{

    public override double CalculateSurface()
    {
        return x * y;
    }
}
```

Клас за сцена

```
class Scene
{
    private List<Shape> _shapes = new List<Shape>();


    public void AddShape(Shape shape)
    {
        _shapes.Add(shape);
    }


    public double CalculateSurface()
    {
        var surface = 0.0;
        foreach (var shape in _shapes)
            surface += shape.CalculateSurface();

        return surface;
    }
}
```

Полиморфизъм

```
Scene scene = new Scene();

scene.AddShape(new Triangle());
scene.AddShape(new Triangle());
scene.AddShape(new Rectangle());

var surface = scene.CalculateSurface();
```

Абстрактни методи

- Нямат имплементация
- Декларират се с ключовата дума abstract
- Напрактика са виртуални методи

Абстрактен клас

- Определение: Клас съдържащ методи без имплементация (абстрактни)
- Декларира се с ключовата дума abstract
- Не може да има инстанции

Абстрактен базов клас за фигури

```
abstract class Shape
{
    protected int x, y;


    public abstract double CalculateSurface();
}
```

Интерфейс

- Съдържа само методи без имплементация
- Декларира се с ключовата дума interface
- Не може да има инстанции

Интерфейси

```
interface ISurfaceShape
{

    double CalculateSurface();
}
```

Интерфейси

```
class Circle : Shape, ISurfaceShape
{

    public double CalculateSurface()
    {
        return Math.PI * Math.Pow(x, 2) / 2;
    }
}
```

```
class Triangle : Shape, ISurfaceShape
{

    public double CalculateSurface()
    {
        return x * y / 2;
    }
}
```

Интерфейси

```
private List<ISurfaceShape> _shapes =
    new List<ISurfaceShape>();


public double CalculateSurface()
{
    var surface = 0.0;

    foreach (var shape in _shapes)
        surface += shape.CalculateSurface();

    return surface;
}
```

Типът Object

- Всеки тип в C# наследява Object

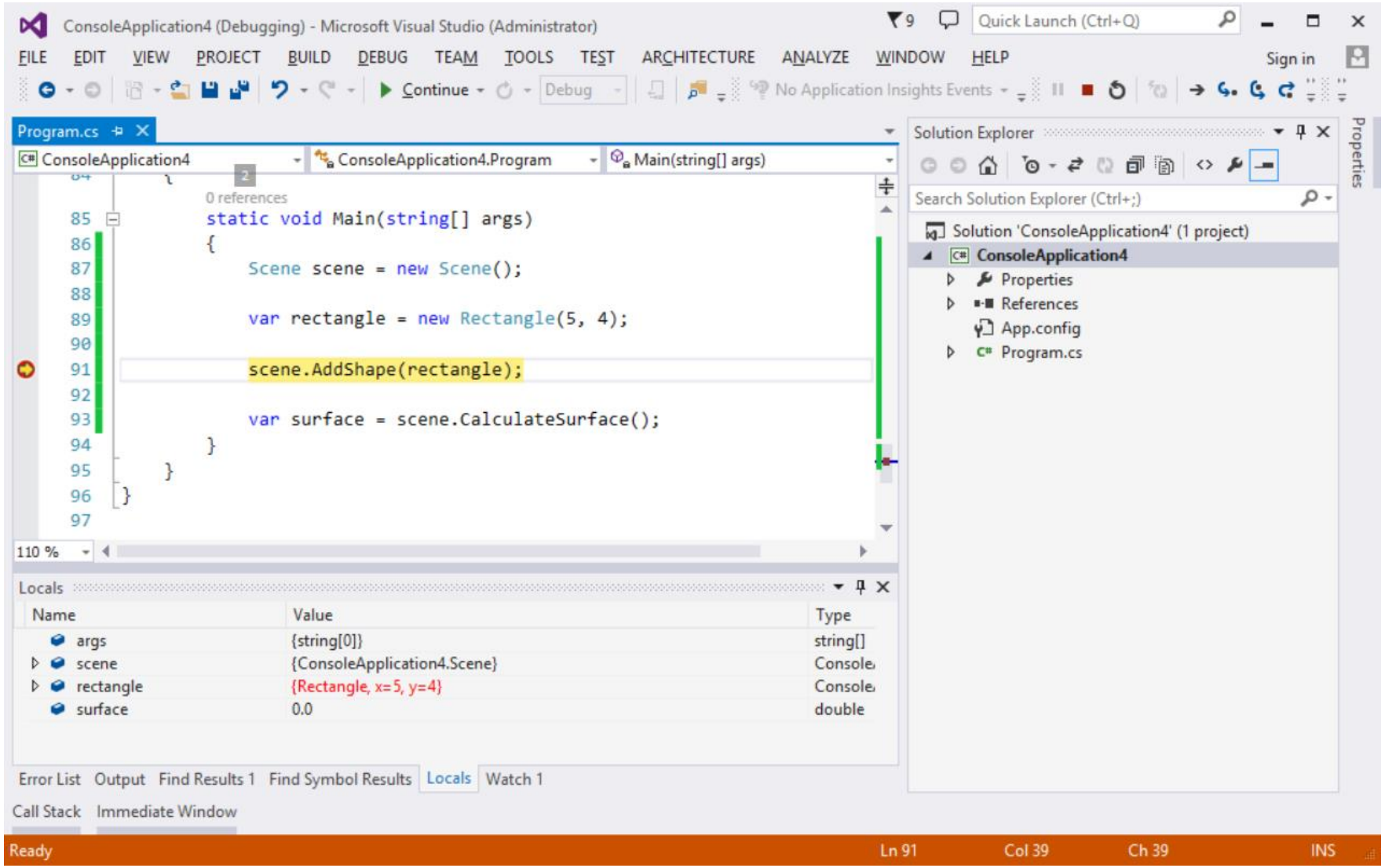
Типът Object

```
public class Object
{
        public Object();

        public virtual bool Equals(object obj);
        public static bool Equals(object objA, object objB);
        public virtual int GetHashCode();
        public Type GetType();
        protected object MemberwiseClone();
        public static bool ReferenceEquals(object objA, object objB);
        public virtual string ToString();
}
```

Типът Object

```
class Rectangle : Shape, ISurfaceShape
{

    public double CalculateSurface()
    {
        return x * y;
    }


    public override string ToString()
    {
        return "Rectangle, x=" + x + ", y=" + y;
    }
}
```

Пренаписване на метода ToString



Свойства

- Определение: Методи за достъп до полета на клас

Свойства

- Декларация:

  <име на тип> <име на свойство>

  { [get{ }] [set{ }] }

Свойства

```
class Triangle : Shape, ISurfaceShape
{

    public int H
    {
        get
        {
            return x;
        }
        set
        {
            x = value;
        }
    }
```

Делегати

- Определение: Делегатите представляват .NET типове, които описват сигнатурата на даден метод (броя, типа и последователността на параметрите му) и връщания от него тип

Делегати

- Декларация:

delegate <method type> <method name>(
[<param type> <param name>, …])

Делегати

```
public delegate void Click(Shape shape);


abstract class Shape
{
    protected int x, y;

    public Click OnClick;
}
```

Делегати

```
public static void OnClickShape(Shape shape)
{
}

static void Main(string[] args)
{
    Scene scene = new Scene();

    var rectangle = new Rectangle(5, 4);
    rectangle.OnClick = OnClickShape;
```

Избиране на форма

- С мишката се щраква върху сцената
- Извиква се метод Click на сцената
- Метода проверява за всяка форма, дали мишката е била в нея
- Ако щракването е в някоя форма се извиква делегата Click

Делегати

```
class Scene
{
    private List<Shape> _shapes =
        new List<Shape>();
    public void Click(int x, int y)
    {
        foreach (var shape in _shapes)
        {
            if (shape is ISurfaceShape &&
                (shape as ISurfaceShape).PointIn(x, y))
            {
                shape.OnClick(shape);
            }
        }
    }
```

Събития

- Определение: Свойство на обекта позволяващо той да извести предварително регистрирани за събитието получатели

Събития

- Декларация: <event> <тип делегат> <име на събитието>;

Събития

```
public delegate void Click(Shape shape);

abstract class Shape
{
    protected int x, y;

    public event Click OnClick;

    public void FireClick()
    {
        OnClick(this);
    }
}
```

Събития

```
public static void OnClickShape(Shape shape)
{
}
static void Main(string[] args)
{
    Scene scene = new Scene();

    var rectangle = new Rectangle(5, 4);
    rectangle.OnClick += OnClickShape;
```

Избиране на форма

- С мишката се щраква върху сцената
- Извиква се метод Click на сцената
- Метода проверява за всяка форма, дали мишката е била в нея
- Ако щракването е в някоя форма се извиква FireClick на обекта, който извиква делегата на събитието OnClick

Събития

```
class Scene
{
    private List<Shape> _shapes =
        new List<Shape>();

    public void Click(int x, int y)
    {
        foreach (var shape in _shapes)
        {
            if (shape is ISurfaceShape &&
                (shape as ISurfaceShape).PointIn(x, y))
            {
                shape.FireClick();
            }
        }
    }
```

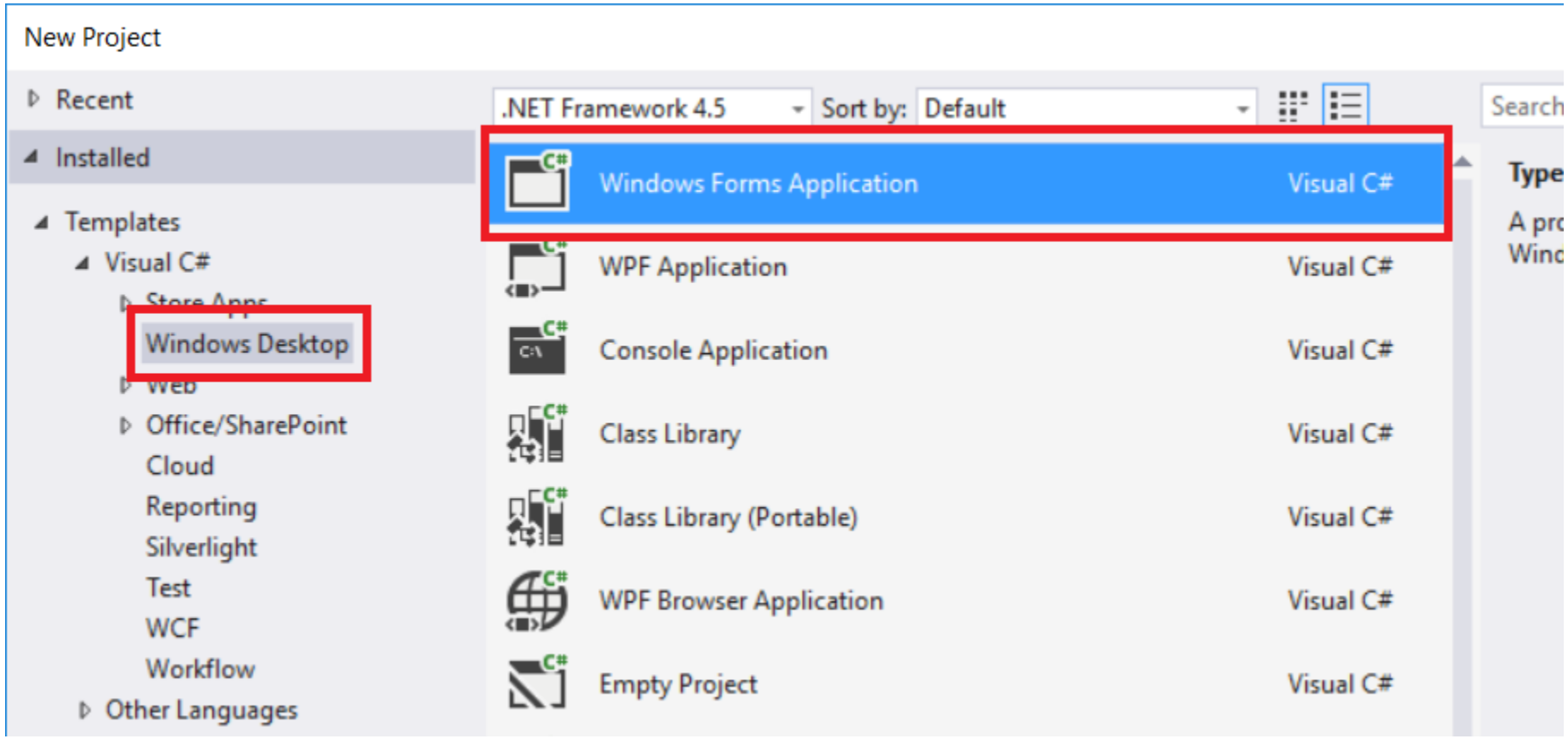
Windows Forms

- Колекция от класове даващи възможност за изграждане на графично ориентиран интерфейс

Windows Forms във Visual Studio

- New Project -> Windows Desktop ->Windows Forms Application

Windows Forms във Visual Studio



Program.cs

```
static class Program
{
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }
}
```

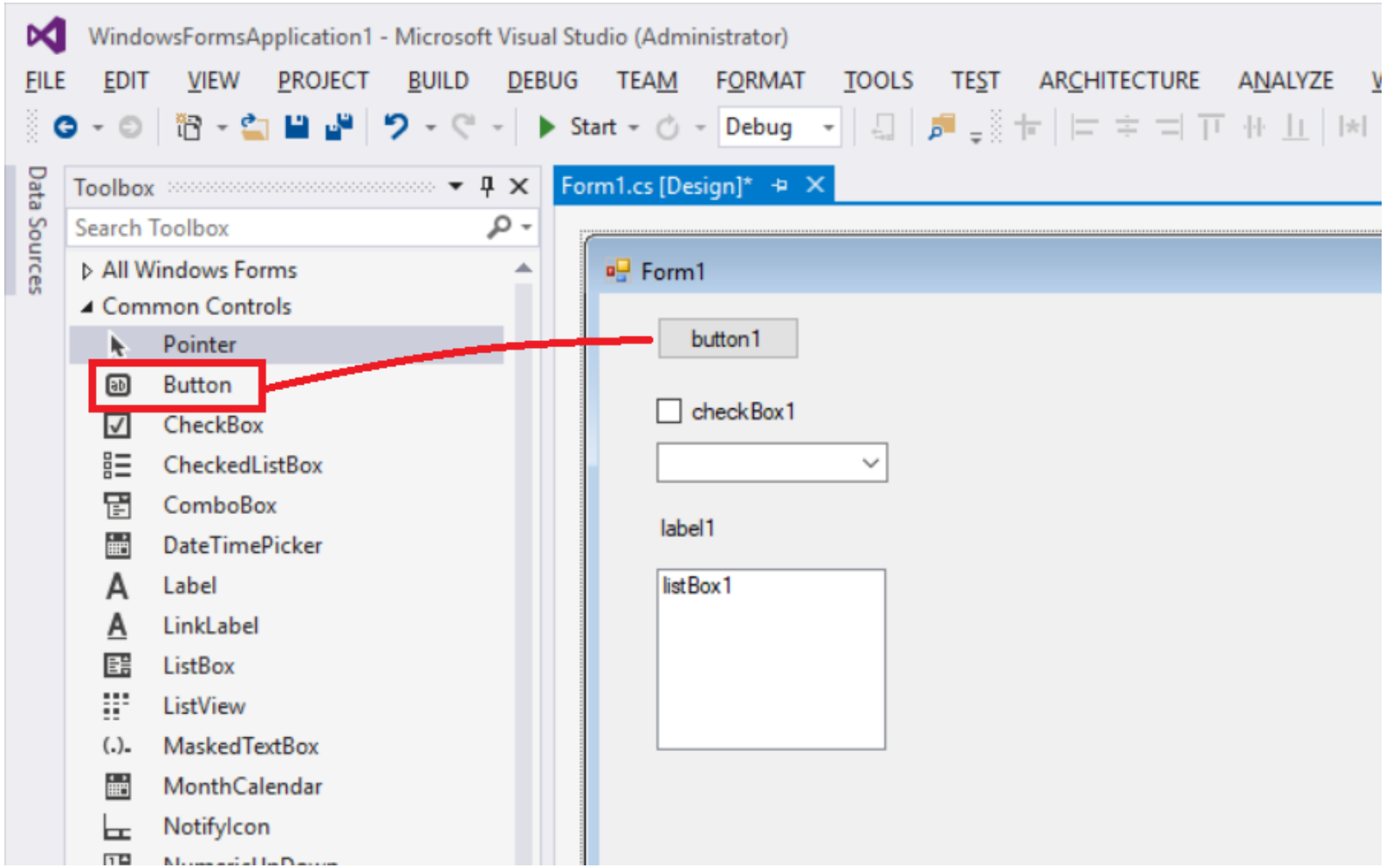
Form1.cs

```
public partial class Form1 : Form
{

    public Form1()
    {
        InitializeComponent();
    }
}
```

Form1.Designer.cs

```
partial class Form1
{
    private System.ComponentModel.IContainer components = null;

    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    private void InitializeComponent()
    {
        this.SuspendLayout();

        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(682, 459);
        this.Name = "Form1";
        this.Text = "Form1";
        this.ResumeLayout(false);

    }

}
```

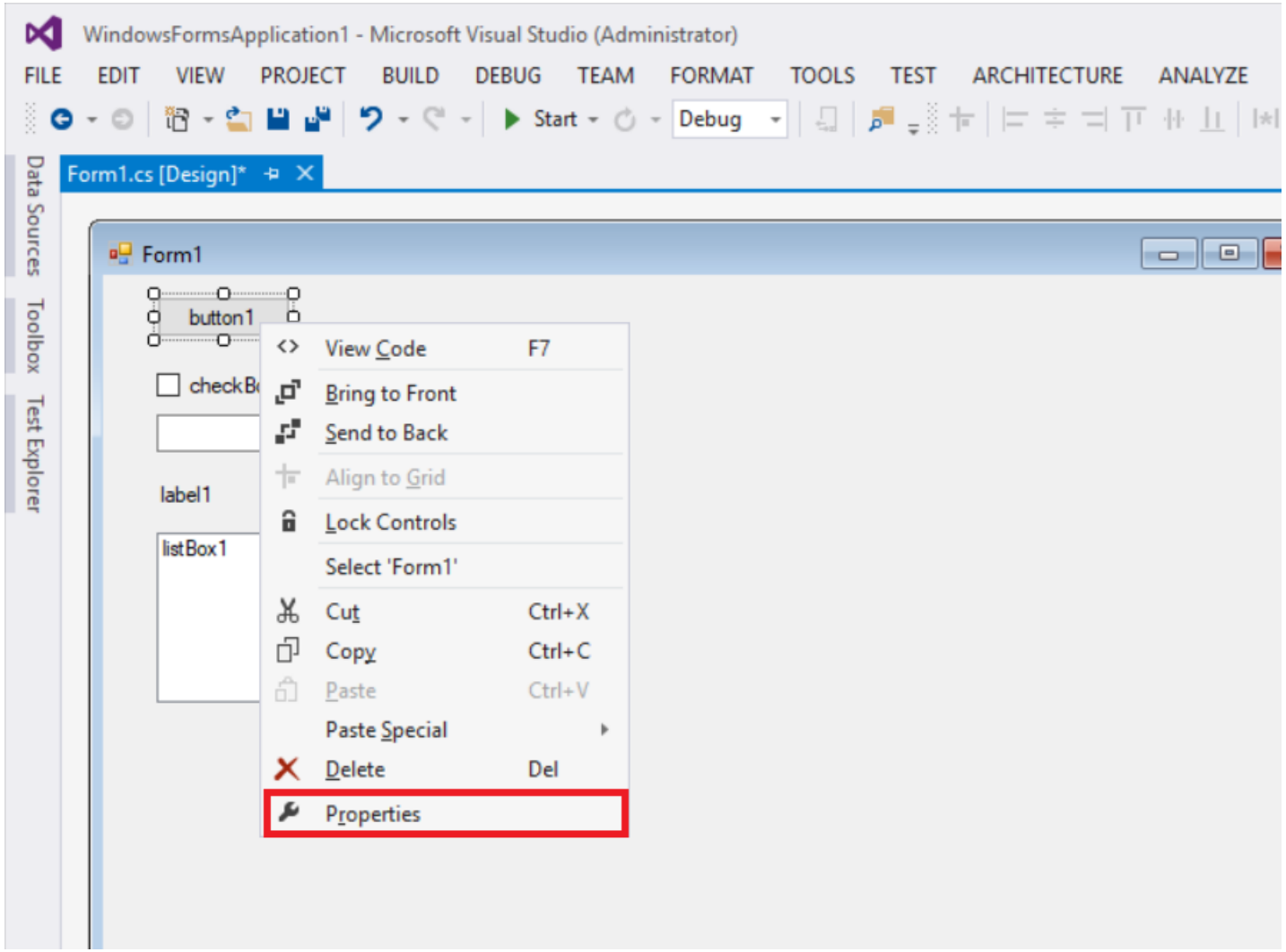
Common Controls



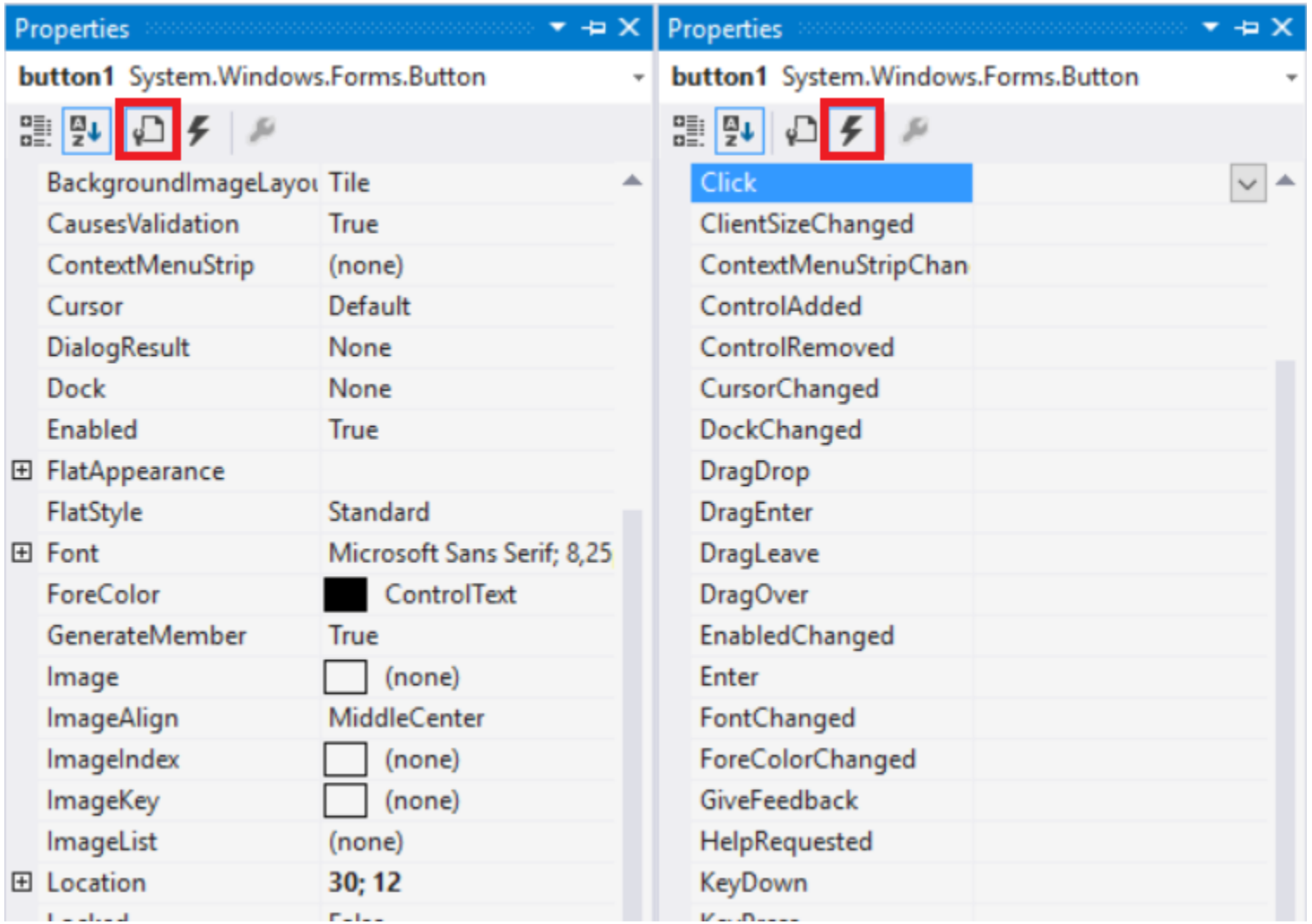
Form1.Designer.cd

```
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.checkBox1 = new System.Windows.Forms.CheckBox();
    this.comboBox1 = new System.Windows.Forms.ComboBox();
    this.label1 = new System.Windows.Forms.Label();
    this.listBox1 = new System.Windows.Forms.ListBox();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(30, 12);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
    //
    // checkBox1
```

Properties


WindowsFormsApplication1 - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ARCHITECTURE ANALYZE
Start
Debug
Form1.cs [Design]*
Data Sources
Toolbox
Test Explorer
Form1
button1
checkB
label1
listBox1
View Code F7
Bring to Front
Send to Back
Align to Grid
Lock Controls
Select 'Form1'
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Paste Special
Delete Del
Properties

Properties


Properties
button1 System.Windows.Forms.Button
BackgroundImageLayou Tile
CausesValidation True
ContextMenuStrip (none)
Cursor Default
DialogResult None
Dock None
Enabled True
FlatAppearance
FlatStyle Standard
Font Microsoft Sans Serif; 8,25
ForeColor ControlText
GenerateMember True
Image (none)
ImageAlign MiddleCenter
ImageIndex (none)
ImageKey (none)
ImageList (none)
Location 30; 12
Properties
button1 System.Windows.Forms.Button
Click
ClientSizeChanged
ContextMenuStripChan
ControlAdded
ControlRemoved
CursorChanged
DockChanged
DragDrop
DragEnter
DragLeave
DragOver
EnabledChanged
Enter
FontChanged
ForeColorChanged
GiveFeedback
HelpRequested
KeyDown

OnClick

```
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(30, 12);
    this.button1.Name = "button1";
    this.button1.Click += new System.EventHandler(this.button1_Click);
```

OnClick

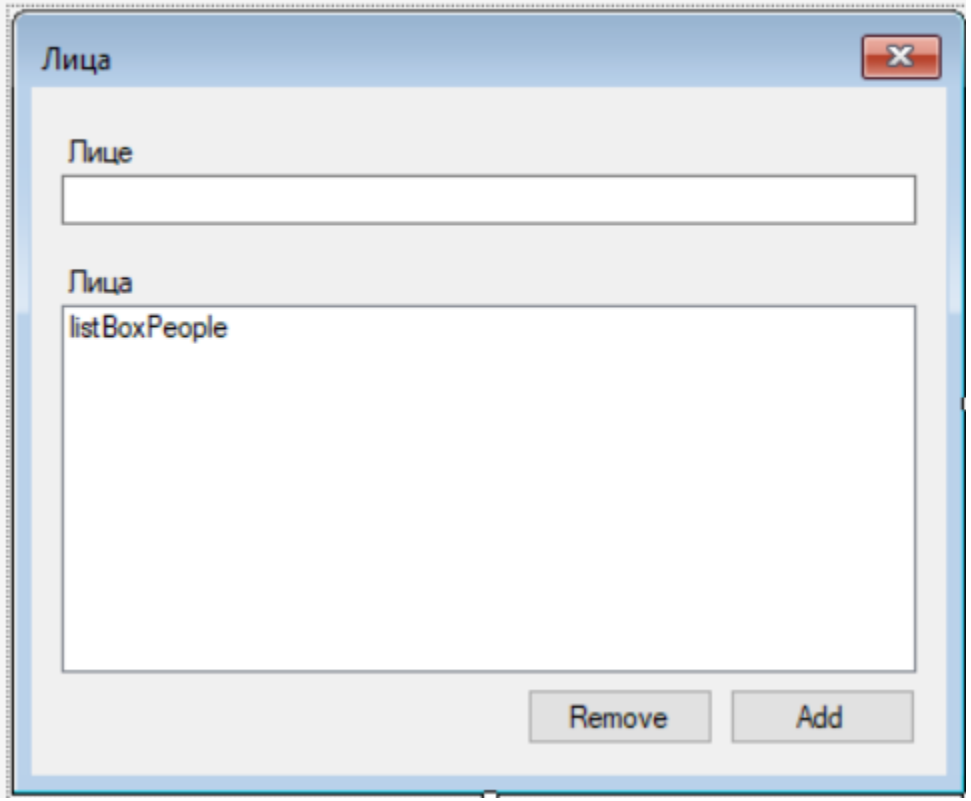
```
public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("hello world!");
    }
}
```

Добавяне на елементи в ListBox

```
public partial class Form1 : Form
{

    public Form1()
    {
        InitializeComponent();
    }


    private void button1_Click(object sender, EventArgs e)
    {
        listBox1.Items.Add("Click");
    }
}
```

Задача 1

- Да се направи програма добавяща лица в списък



Задача 1

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    listBoxPeople.Items.Add(
        textBoxName.Text);
}


private void buttonRemove_Click(object sender, EventArgs e)
{
    listBoxPeople.Items.Remove(
        listBoxPeople.SelectedItem);
}
```

Задача 2

- Да се направи програма добавяща лица в списък (ListBox)
- Лицата се представят с клас Person
- Да се реализира изтриване на лица от списъка