

# Windows Forms

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

# Form1.cs

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
}
```

# Form1.Designer.cs

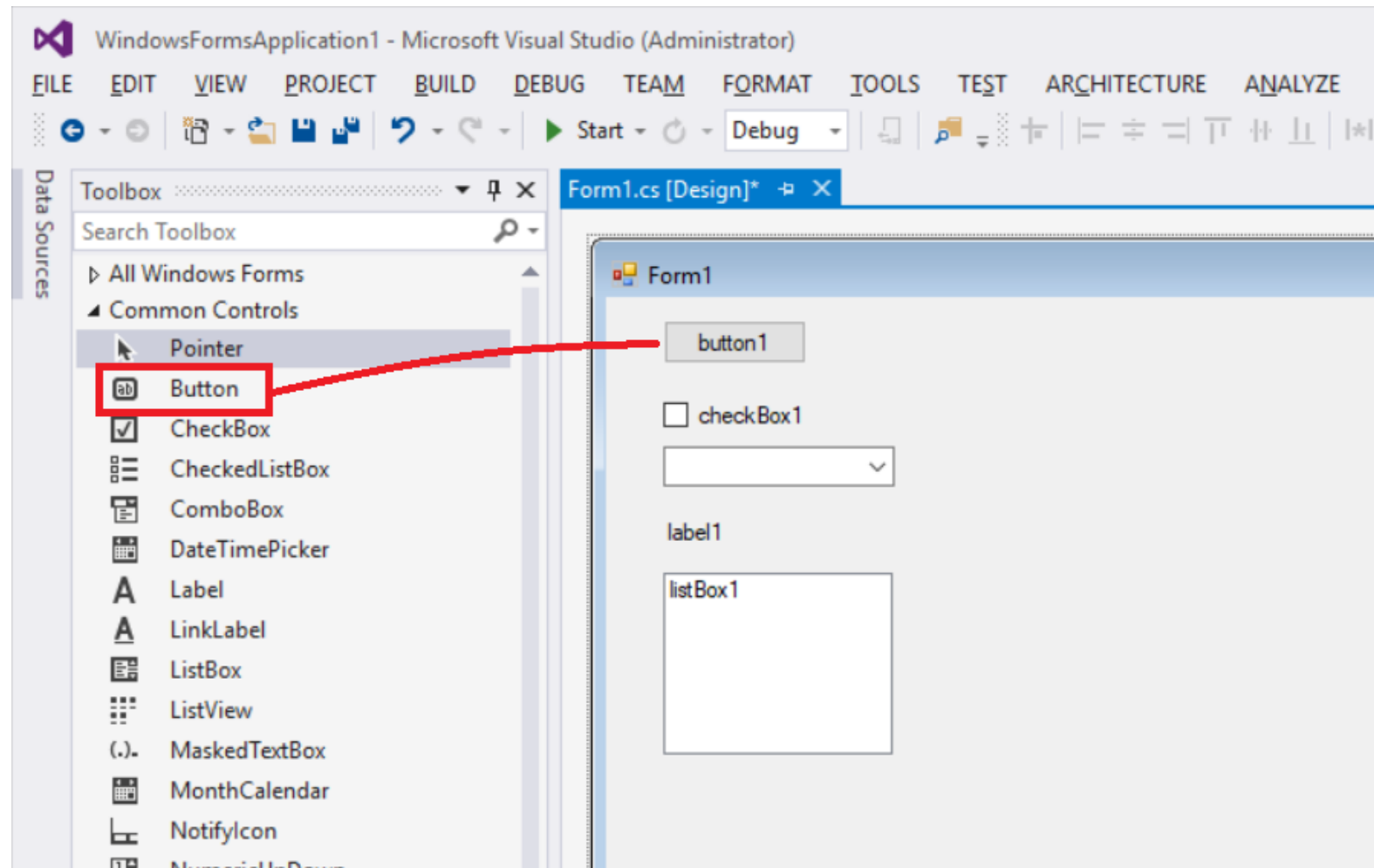
```
partial class Form1
{
    private System.ComponentModel.IContainer components = null;

    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);
    }
}
```

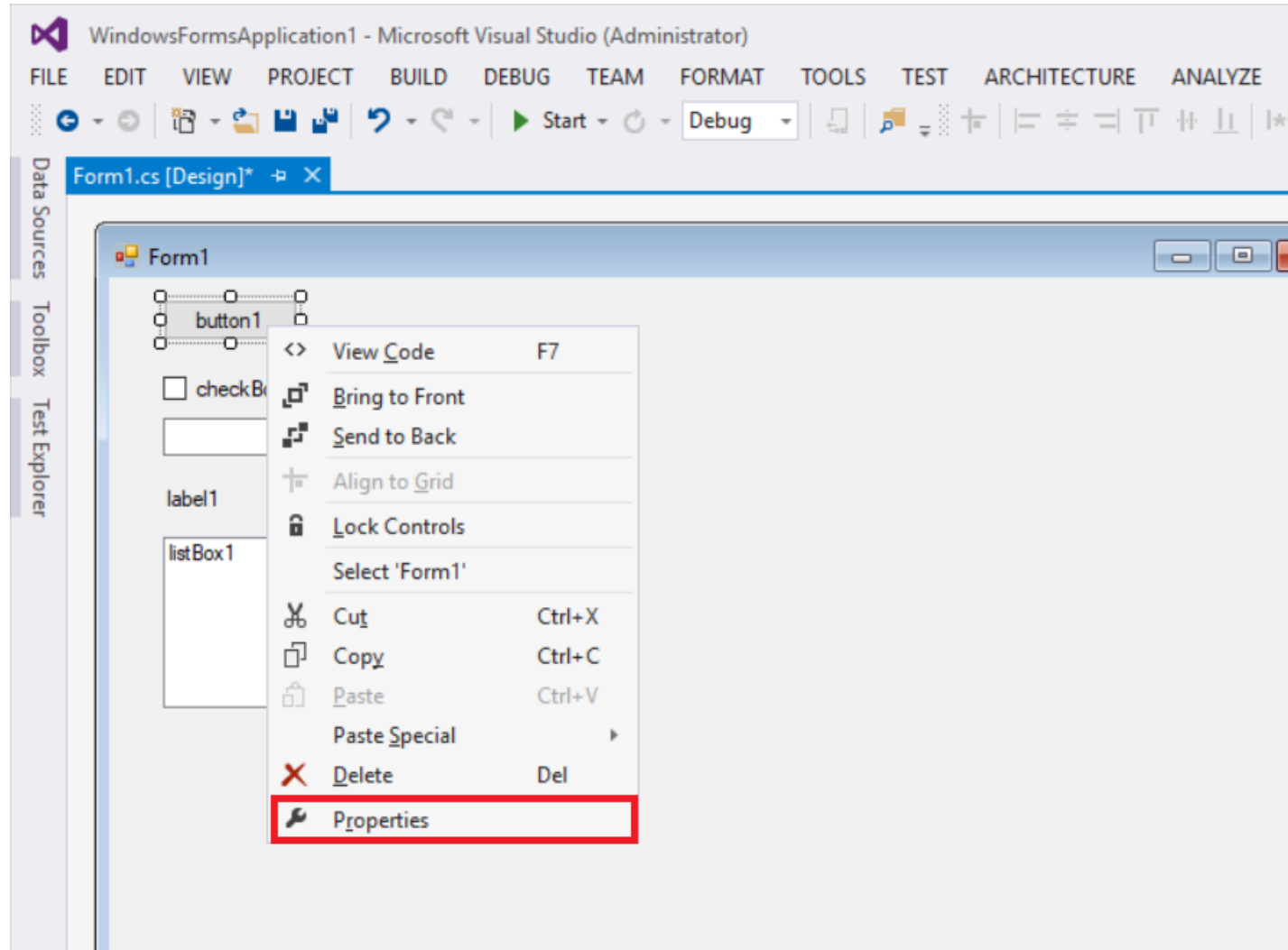
# Common Controls



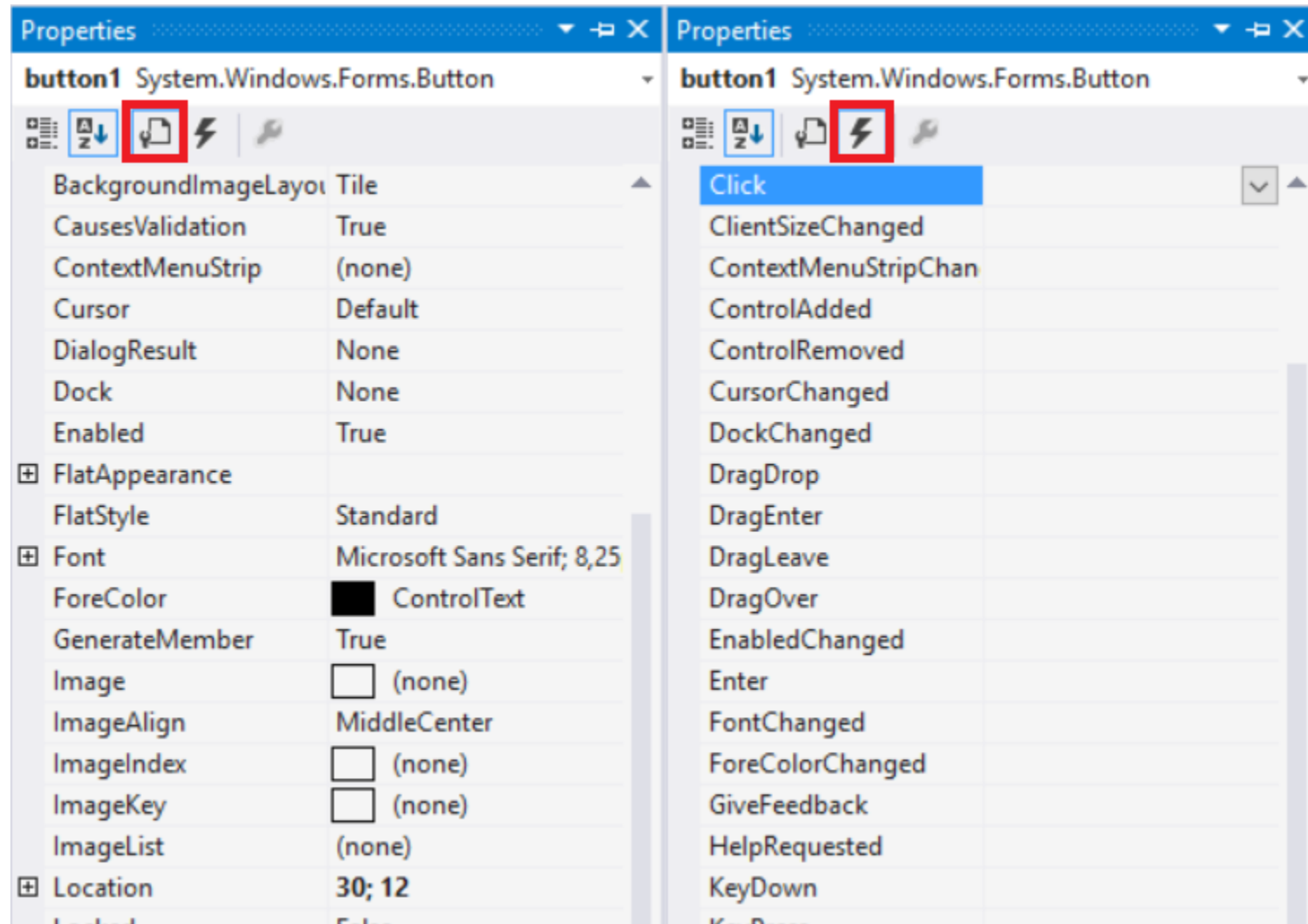
# Form1.Designer.cd

```
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



# Properties



# 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!");
    }
}
```

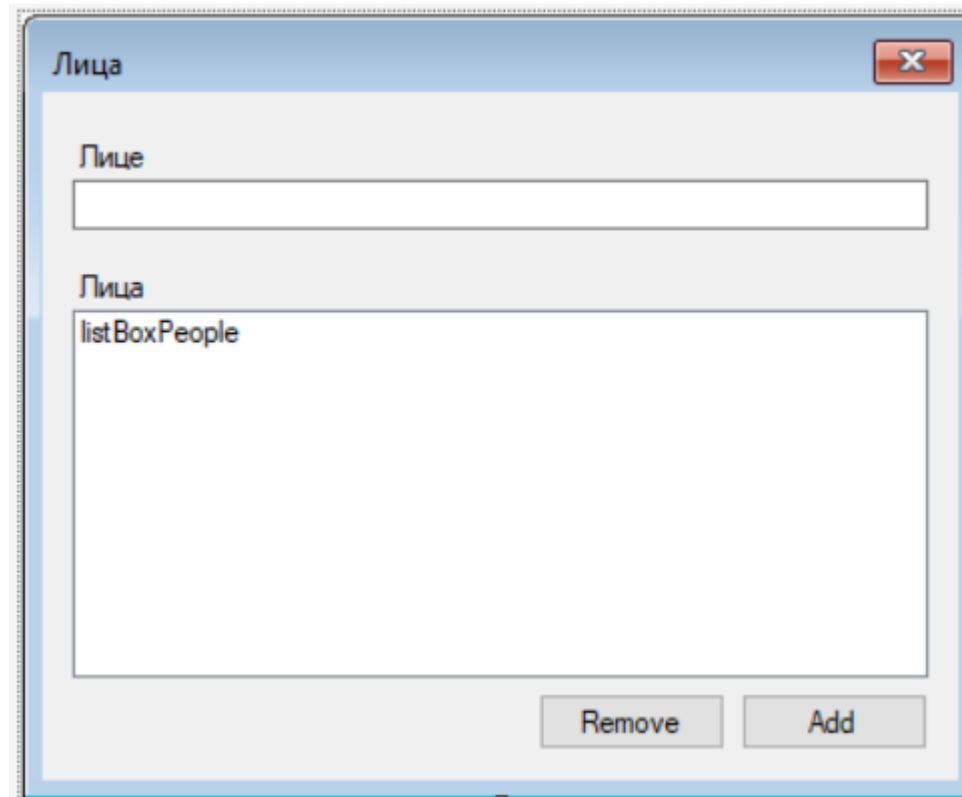
# Добавяне на елементи в ListBox

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

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

# Задача

- Да се направи програма добавяща лица в списък



# Задача

```
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

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

# Задача: FormPeople.cs

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    Person person = new Person();
    person.Name = textBoxName.Text;

    listBoxPeople.Items.Add(person);
}
```

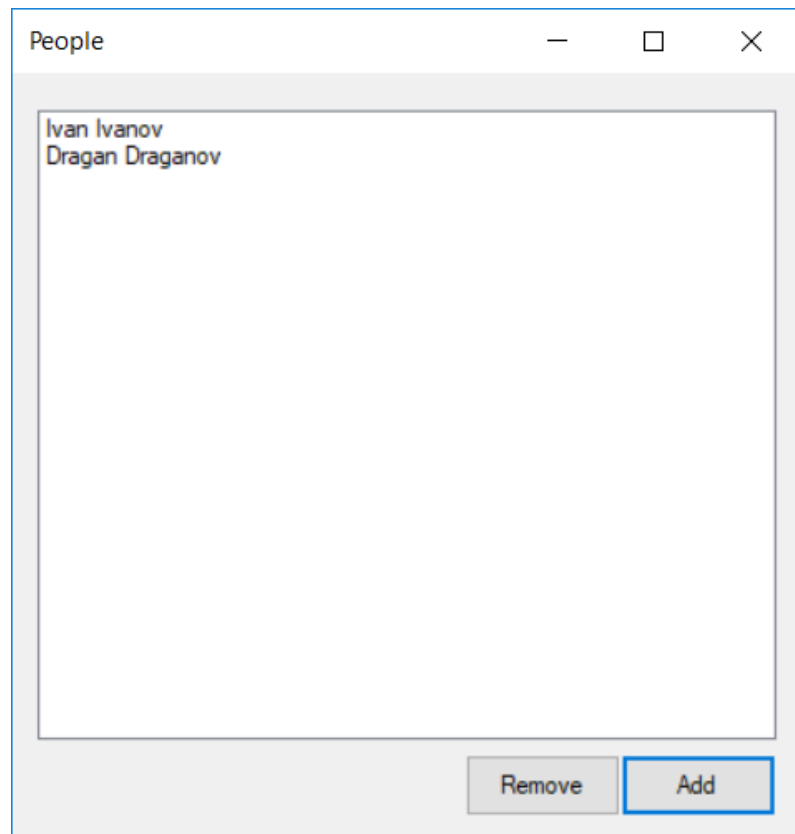
```
private void buttonRemove_Click(object sender, EventArgs e)
{
    listBoxPeople.Items.Remove(listBoxPeople.SelectedItem);
}
```

# Задача: Person.cs

```
class Person
{
    public string Name { get; set; }

    public override string ToString()
    {
        return Name;
    }
}
```

# Задача

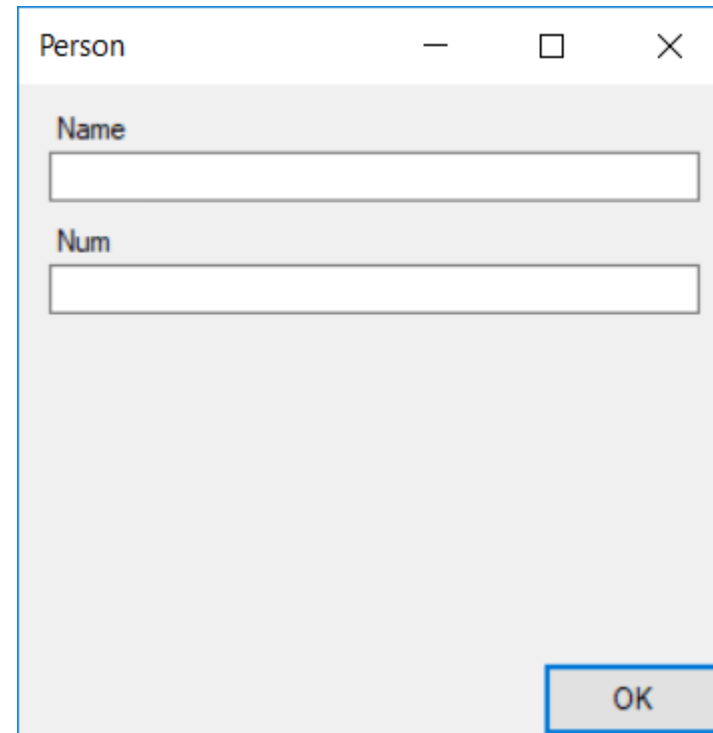


A window titled "People" with standard Windows window controls (minimize, maximize, close). It contains a list box with two entries: "Ivan Ivanov" and "Dragan Draganov". At the bottom right, there are two buttons: "Remove" and "Add". The "Add" button is highlighted with a blue border.

People

Ivan Ivanov  
Dragan Draganov

Remove Add



A window titled "Person" with standard Windows window controls (minimize, maximize, close). It contains two text input fields: "Name" and "Num". At the bottom right, there is an "OK" button. The "OK" button is highlighted with a blue border.

Person

Name

Num

OK



# Метод ShowDialog

- Показва формата в модален режим;
- Връща стойност от изброимия тип DialogResult;
- Ако показаната форма присвои стойност на свойството DialogResult автоматично се извиква метода Close(), който затваря формата.

# Метод ShowDialog

```
var formPerson = new FormPerson();  
  
if (formPerson.ShowDialog() == DialogResult.OK)  
{  
}
```

# Задача

- Да се разшири функционалността на приложението така че:
  - Лицето има име и егн;
  - Главната форма има ListBox с лица и бутони за добавяне и изтриване на лице;
  - Бутон за добавяне отваря нов диалогов прозорец FormPerson с полета за име и егн и бутон „ОК“. При натискане на „ОК“ лицето се добавя в списъка.
  - При двойно щракване (събитие OnDoubleClick) на елемент в списъка се отваря диалоговия прозорец FormPerson за редакция на избраното лице.

# FormPeople.cs

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    var formPerson = new FormPerson();
    var person = new Person();
    formPerson.Person = person;

    if (formPerson.ShowDialog() == DialogResult.OK)
    {
        listBoxPeople.Items.Add(person);
    }
}
```

# FormPerson.cs

```
public partial class FormPerson : Form
{
    private Person _person ;

    public Person Person
    {
        get
        {
            return _person;
        }
        set
        {
            _person = value;

            textBoxName.Text = _person.Name;
            textBoxNum.Text = _person.Num;
        }
    }
}
```

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    Person.Name = textBoxName.Text;
    Person.Num = textBoxNum.Text;

    DialogResult = DialogResult.OK;
}
```

# FormPeople.cs

```
private void listBoxPeople_DoubleClick(object sender, EventArgs e)
{
    if (listBoxPeople.SelectedItem == null)
    {
        return;
    }

    var formPerson = new FormPerson();

    var person = (Person)listBoxPeople.SelectedItem;
    formPerson.Person = person;

    if (formPerson.ShowDialog() == DialogResult.OK)
    {
        int index = listBoxPeople.SelectedIndex;
        listBoxPeople.Items.RemoveAt(index);
        listBoxPeople.Items.Insert(index, person);
    }
}
```

# FormPeople.cs

```
private void buttonRemove_Click(object sender, EventArgs e)
{
    if (listBoxPeople.SelectedItem == null)
    {
        return;
    }

    listBoxPeople.Items.Remove(listBoxPeople.SelectedItem);
}
```

# Задача

A window titled "People" with standard window controls (minimize, maximize, close). The window contains a search bar with a "Search" button to its right. Below the search bar is a large, empty rectangular area, likely for displaying a list of people. At the bottom right of the window are two buttons: "Remove" and "Add".



# Задача

- Лицата да се съхраняват в списък (ArrayList)
- Да се добави поле за търсене `textBoxSearch` и бутон `buttonSearch`.  
При натискане на бутона в `listBoxPeople` се зареждат лицата, чието име съдържа текста в `textBoxSearch`. За изчистване на `ListBox` може да се ползва метода `listBox.Items.Clear()`

# FormPeople.cs

```
private ArrayList people = new ArrayList();
```

# FormPeople.cs

```
private void buttonSearch_Click(object sender, EventArgs e)
{
    listBoxPeople.Items.Clear();

    foreach (var person in people)
    {
        if (((Person)person).Name.Contains(textBoxSearch.Text))
        {
            listBoxPeople.Items.Add(person);
        }
    }
}
```

# ArrayList

```
...public class ArrayList : IList, ICollection, IEnumerable, ICloneable
{
    ...public ArrayList();
    ...public ArrayList(ICollection c);
    ...public ArrayList(int capacity);

    ...public virtual int Capacity { get; set; }
    ...public virtual int Count { get; }
    ...public virtual bool IsFixedSize { get; }
    ...public virtual bool IsReadOnly { get; }
    ...public virtual bool IsSynchronized { get; }
    ...public virtual object SyncRoot { get; }

    ...public virtual object this[int index] { get; set; }

    ...public static ArrayList Adapter(IList list);
    ...public virtual int Add(object value);
    ...public virtual void AddRange(ICollection c);
}
```

# ArrayList

```
var list = new ArrayList();  
  
list.Add((Object)person);  
  
Person person = (Person)list[0];
```

# Шаблонни класове и методи (Generics)

- Определение: Функционалност позволяваща типово параметризиране на класове и методи.

# Шаблонни методи (Generics)

```
public class List
{
    public void Add<T>(T item);
    public T Get<T>(int index);
}
```

# Шаблонни методи (Generics)

```
List list = new List();
```

```
list.Add<Person>(new Person());
```

```
Person person = list.Get<Person>(0);
```



# Шаблонни класове (Generics)

```
public class List<T>
{
    public T this[int index] { get; set; }

    public void Add(T item);
}
```

# Шаблонни класове (Generics)

```
List<Person> list = new List<Person>();
```

```
list.Add(new Person());
```

```
Person person = list[0];
```

# Класът List<>

```
...public class List<T> : IList<T>, ICollection<T>, IList, ICollection, IReadOnlyList<T>,
{
    ...public List();
    ...public List(IEnumerable<T> collection);
    ...public List(int capacity);

    ...public int Capacity { get; set; }
    ...public int Count { get; }

    ...public T this[int index] { get; set; }

    ...public void Add(T item);
    ...public void AddRange(IEnumerable<T> collection);
    ...public ReadOnlyCollection<T> AsReadOnly();
    ...public int BinarySearch(T item);
    ...public int BinarySearch(T item, IComparer<T> comparer);
    ...public int BinarySearch(int index, int count, T item, IComparer<T> comparer);
    ...public void Clear();
    ...public bool Contains(T item);
    ...public List<TOutput> ConvertAll<TOutput>(Converter<T, TOutput> converter);
    ...public void CopyTo(T[] array);
```

# Задача

People

Search

Remove Add

Person

Name

Num

Products

Remove product Add product

OK

Product

Name

Price

Serial

OK

# Задача

- Да се добави нов клас Продукт със свойства Име, Цена, Сериен номер;
- Към класа за Лице да се добави поле за списък от продукти;
- Да се добави форма за добавяне/редактиране на продукти за всяко лице;

# Person.cs

```
public class Person
{
    public string Name { get; set; }
    public string Num { get; set; }
    public List<Product> products = new List<Product>();
    public override string ToString()
    {
        return Name;
    }
}
```

# FormPerson.cs

```
public partial class FormPerson : Form
{
    private Person _person = new Person();

    public Person Person
    {
        get
        {
            return _person;
        }
        set
        {
            _person = value;

            textBoxName.Text = _person.Name;
            textBoxNum.Text = _person.Num;

            listBoxProducts.Items.Clear();
            foreach (var product in Person.products)
            {
                listBoxProducts.Items.Add(product);
            }
        }
    }
}
```

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    Person.Name = textBoxName.Text;
    Person.Num = textBoxNum.Text;

    DialogResult = DialogResult.OK;
}

private void buttonAddProduct_Click(object sender, EventArgs e)
{
    var formProduct = new FormProduct();
    var product = new Product();

    formProduct.Product = product;

    if (formProduct.ShowDialog() == DialogResult.OK)
    {
        Person.products.Add(product);
        listBoxProducts.Items.Add(product);
    }
}
```

# FormPeople.cs

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    var formPerson = new FormPerson();
    var person = new Person();
    formPerson.Person = person;

    if (formPerson.ShowDialog() == DialogResult.OK)
    {
        people.Add(person);
    }
}
```

```
private void listBoxPeople_DoubleClick(object sender, EventArgs e)
{
    if (listBoxPeople.SelectedItem == null)
    {
        return;
    }

    var formPerson = new FormPerson();

    var person = (Person)listBoxPeople.SelectedItem;
    formPerson.Person = person;

    if (formPerson.ShowDialog() == DialogResult.OK)
    {
        int index = listBoxPeople.SelectedIndex;
        listBoxPeople.Items.RemoveAt(index);
        listBoxPeople.Items.Insert(index, person);
    }
}
```