

ГПИ. Други Windows Forms контроли

Лекция 5

СЪДЪРЖАНИЕ

- ▶ **Стандартни диалогови кутии**
- ▶ **Други Windows Forms контроли**
- ▶ **CheckBox**
- ▶ **RadioButton**
- ▶ **Panel**
- ▶ **TabControl и TabPage**
- ▶ **ListBox**
- ▶ **CheckedListBox**
- ▶ **ComboBox**
- ▶ **TreeView**
- ▶ **RichTextBox**
- ▶ **LinkLabel**
- ▶ **PictureBox**

Стандартни диалогови кутии

MessageBox Class

- ▶ Класът `MessageBox` позволява да извеждаме стандартни диалогови кутии, съдържащи текст, бутони и икони:
 - ▶ съобщения към потребителя
 - ▶ въпросителни диалози

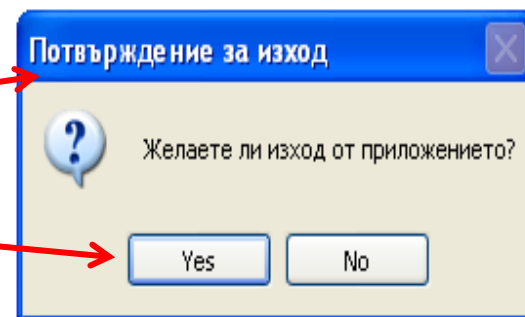
Примери:

```
MessageBox.Show("Hello World!");
```



```
MessageBox.Show("Hello World!", "A Message");
```

```
If (MessageBox.Show ("Желаете ли изход от приложението?",  
                    "Потвърждение за изход",  
                    MessageBoxButtons.YesNo,  
                    MessageBoxIcon.Question) == DialogResult.Yes)  
Application.Exit();
```

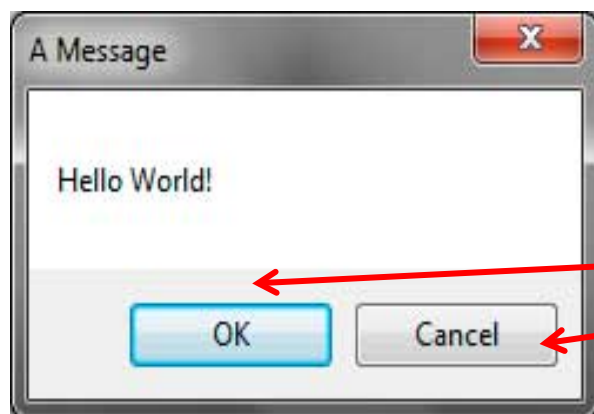


Стандартни диалогови кутии

MessageBox Class

- ▶ Таблицата показва членовете на **MessageBoxButtons**

| Метод | Бутони в съобщението |
|------------------|----------------------|
| AbortRetryIgnore | Abort, Retry, Ignore |
| OK | OK |
| OKCancel | OK, Cancel |
| RetryCancel | Retry, Cancel |
| YesNo | Yes, No |
| YesNoCancel | Yes, No, Cancel |



```
MessageBox.Show("Hello World!", "A Message",  
MessageBoxButtons.OKCancel);
```

Резултати от диалоговата кутия

- ▶ Методът **Show ()** връща стойност от класа **System.Windows.Forms.DialogResult**
- ▶ Използва се, за да се **определи избора** или **бутона, който сте натиснали** в полето за съобщения
- ▶ Например, ако щракнете върху бутона **"Да"** в полето за съобщения, след това методът **Show ()** ще върне стойността **DialogResult.Yes**





```
DialogResult result;  
result = MessageBox.Show("What is your choice?");  
if (result == DialogResult.Yes)  
{  
    //You pressed the Yes button }  
if (result == DialogResult.No)  
{  
    //You pressed the No button }
```

MessageBoxIcon

- ▶ Можете да добавяте също и **икона (icon)** в съобщението чрез класа **MessageBoxIcon**



```
MessageBox.Show("Hello World!", "A Message",  
MessageBoxButtons.OK, MessageBoxIcon.Information);
```

| Икона | Член | Употреба |
|---|------------------------|---|
|  | Information | Показва информация за потребителя |
|  | Error Hand Stop | Показва съобщение за грешка error messages |
|  | Exclamation Warning | Показва съобщение за внимание |
|  | Question | Задава се въпрос към потребителя |

Извикване на диалогови кутии

- ▶ Потребителските диалогови кутии се извикват по следният начин:

```
DialogResult result = dialog.ShowDialog();
```

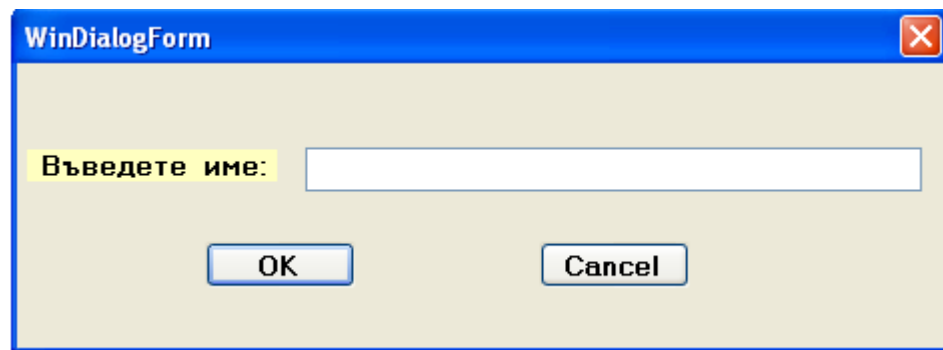
- ▶ Методът **ShowDialog()** показва формата като модална диалогова кутия
- ▶ Типът **DialogResult** съдържа резултата (**OK**, **Yes**, **No**, **Cancel** и др.)
- ▶ Задаване на **DialogResult**:
 - ▶ **Автоматично** – чрез свойството **DialogResult** на бутоните
 - ▶ **Ръчно** – преди затваряне на диалога чрез свойството му **DialogResult**

Пример: Извикване на диалогови кутии и предаване на данни между диалози

1. Създаваме нов **Windows Forms** проект
2. Създаваме нова форма (**File | Add New Item ... | Windows Form**). Сменяме името ѝ на **WinDialogForm**
3. Създаване на ГПИ на приложението – добавяне на етикет, бутони и текстова кутия
4. Настройка на някои свойства на елементите на ГПИ в режим на проектиране

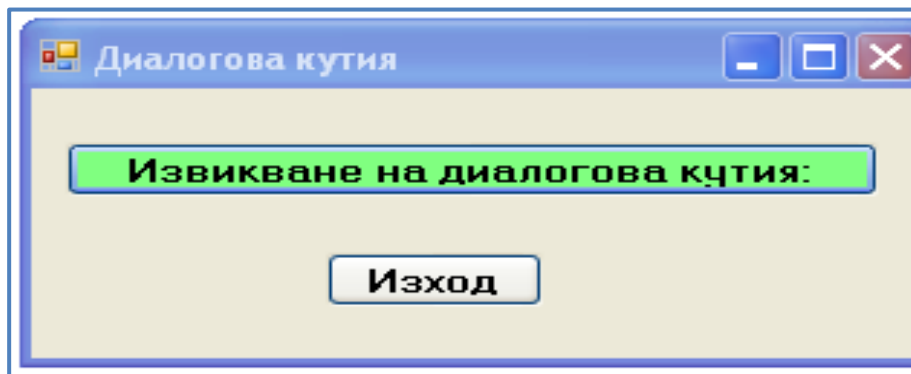
| Елементи на ГПИ | Име на елемент | Свойства |
|-----------------|----------------|---|
| Label | labelName | Text="Въведете име:" ; Font, Bold, 12, BackColor – по избор |
| Button | buttonOk | Text="ОК" ; Font, Bold, 12, BackColor – по избор, DialogResult = Ok |
| Button | buttonCancel | Text="Cancel" ; Font, Bold, 12, BackColor – по избор DialogResult = Cancel |
| Form | WinDialogForm | Text="WinDialogForm", FormBorderStyle = Fixed3D; MaximizeBox = False; MinimizeBox = False |
| TextBox | textBoxName | Text = " " |

5. Добавяне на програмен код към елементите:



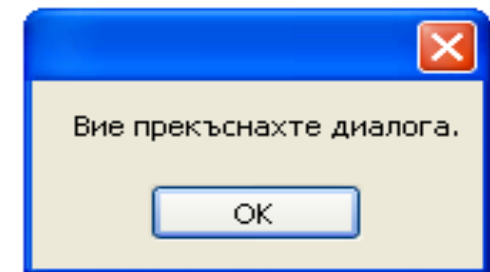
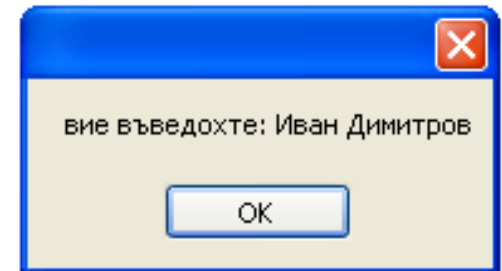
```
public string UserName
{
    get
    {
        return textBoxName.Text;
    }
}
```

6. Поставяме върху **главната форма** бутон със свойство **Name = buttonCallDialog** и задаваме на свойството му **Text = “Извикване на диалогова кутия”**, и бутон с **Name = buttonOut** **Text = “Изход”**

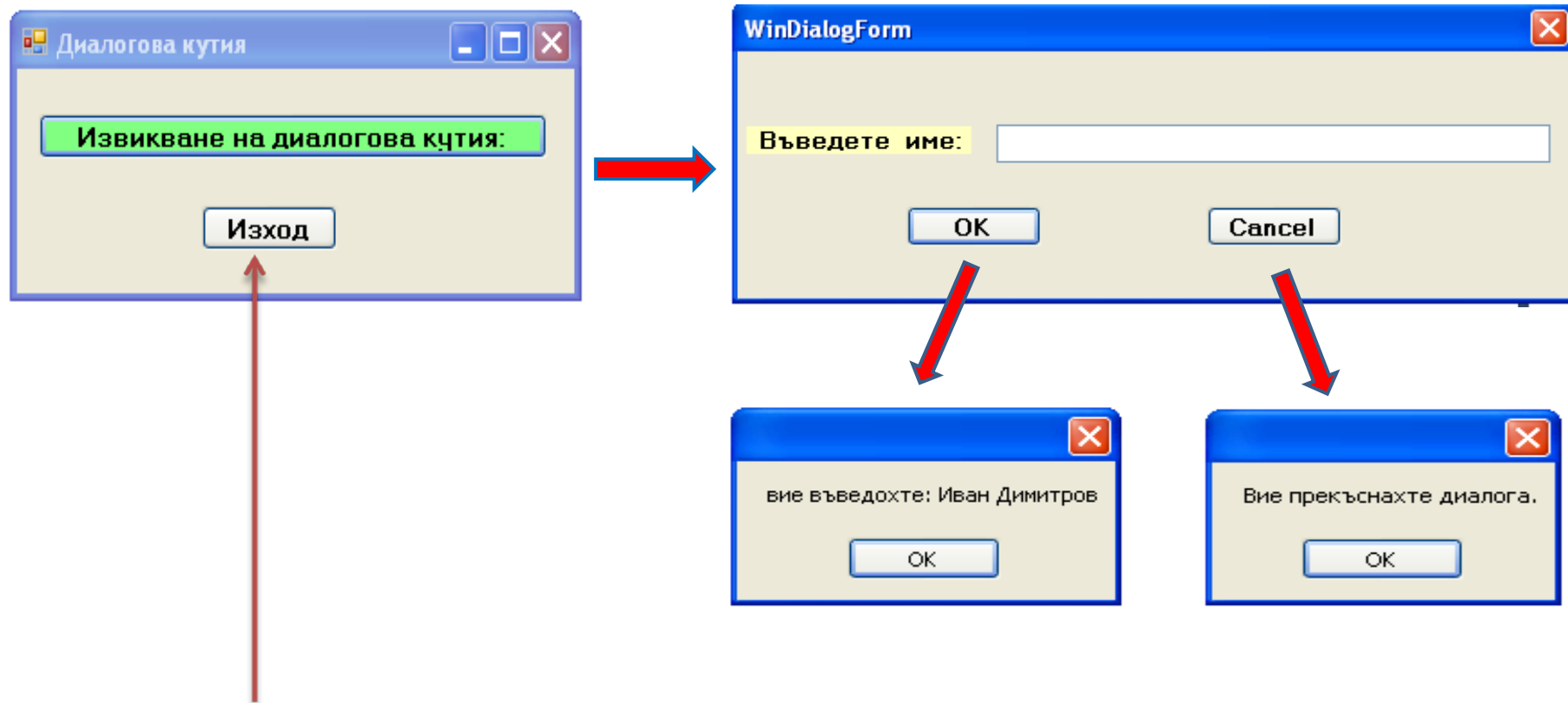


Добавяме обработчик на събитието Click на бутона в Главната форма:

```
private void buttonCallDialog_Click(object sender, EventArgs e)
{
    WinDialogForm dialog = new WinDialogForm();
    if (dialog.ShowDialog() == DialogResult.OK)
    {
        string userName = dialog.UserName;
        MessageBox.Show("Вие въведохте: " + userName);
    }
    else
    {
        MessageBox.Show("Вие прекъснахте диалога.");
    }
    dialog.Dispose();
}
```



Стартиране на приложението:



```
private void button1_Click(object sender, EventArgs e)
{
    WinDialogForm.ActiveForm.Close();
}
```

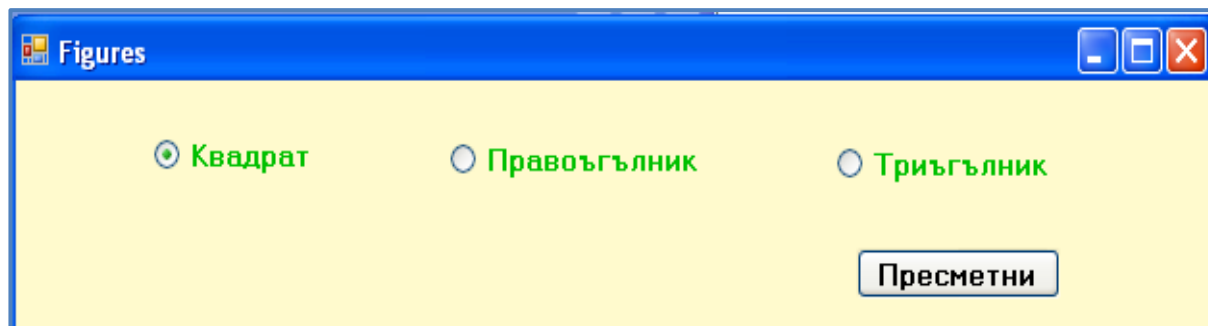
Други Windows Forms контроли

▶ Радио-бутон (RadioButton)



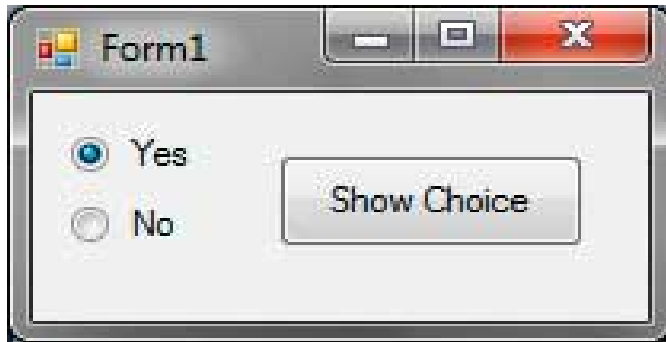
Радио-бутон е бутон, който има две състояния – **активно** и **неактивно**

- ▶ Изобразява се с **кръг**, като **в средата на активния бутон има точка**
- ▶ Обикновено се използват **група от радио-бутони**, като **във всеки момент е активен само един**.
- ▶ **Най-важното свойство** на радио-бутон е **Checked**. То е от булев тип и показва дали бутонът е **натиснат** или не.
- ▶ Друго негово свойство е **Text**.



Радио-букон (RadioButton)

► System.Windows.Forms.RadioButton



```
private void buttonShow_Click(object sender, EventArgs e)
{
    if (radioButtonYes.Checked)
        MessageBox.Show("You choosed yes!");
    else MessageBox.Show("You choosed no!"); } }
```

Радио-бутон (RadioButton)

- ▶ Радио бутона се използва в **групи**.
- ▶ Всички **RadioButton** контроли в даден контейнер (например форма) образуват една група и в нея **само един RadioButton е избран в даден момент**
- ▶ За да създадем няколко групи в една форма, трябва да поставим всяка група в свой собствен контейнер, като например **GroupBox**, **Panel** или **TabPage**.
- ▶ Свойството **Checked** задава дали контролата е **избрана**
- ▶ При **промяна** на свойството **Checked** се активира събитието **CheckedChanged**

Други Windows Forms контроли

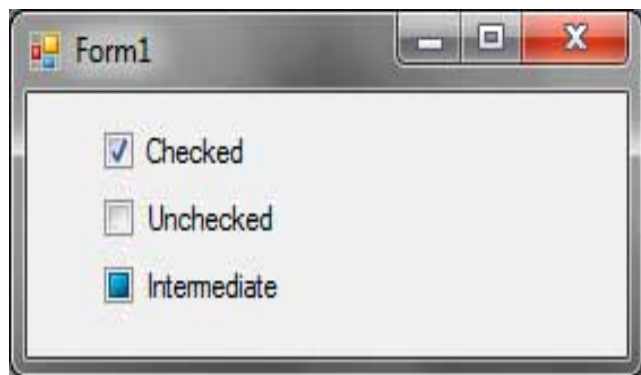
CheckBox

▶ CheckBox



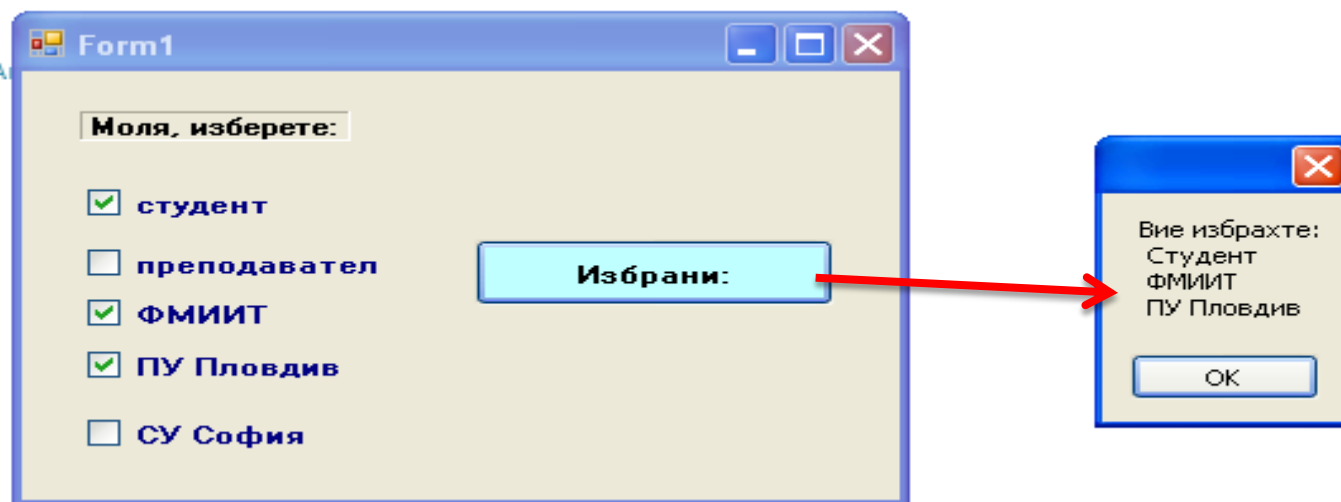
System.Windows.Forms.CheckBox

- ▶ **CheckBox** е кутия за избор в стил "да/не"
- ▶ Свойството **Checked**, задава **дали е избрана**
- ▶ **Има три състояния**



```
checkBox1.CheckState = CheckState.Checked;  
checkBox2.CheckState = CheckState.Unchecked;  
checkBox3.CheckState = CheckState.Intermediate;
```

Пример: Използване на CheckBox



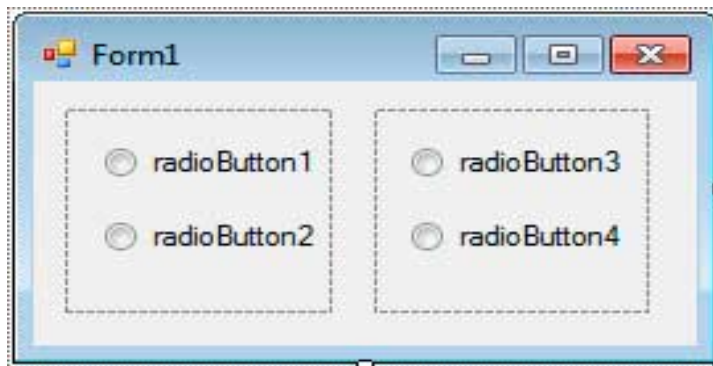
```
private void button1_Click(object sender, EventArgs e)
{
    string items = String.Empty;

    if (checkBox1.Checked)
        items += "\n Студент";
    if (checkBox2.Checked)
        items += "\n Преподавател";
    if (checkBox3.Checked)
        items += "\n ФМИИТ";
    if (checkBox4.Checked)
        items += "\n ПУ Пловдив";
    if (checkBox5.Checked)
        items += "\n СУ София";
    MessageBox.Show("Вие избрахте: " + items);
}
```


Контейнери - Panel

- ▶ **Panel** представлява **контейнер**, който съдържа група други контроли
- ▶ Служи за групиране на контроли
- ▶ Когато преместим даден панел на друга позиция, всички контроли, които са в него, също се преместват
- ▶ Ако стойността на свойството **Enabled** на **Panel** контролата има стойност **false**, то всички контроли, съдържащи се в нея, ще бъдат деактивирани

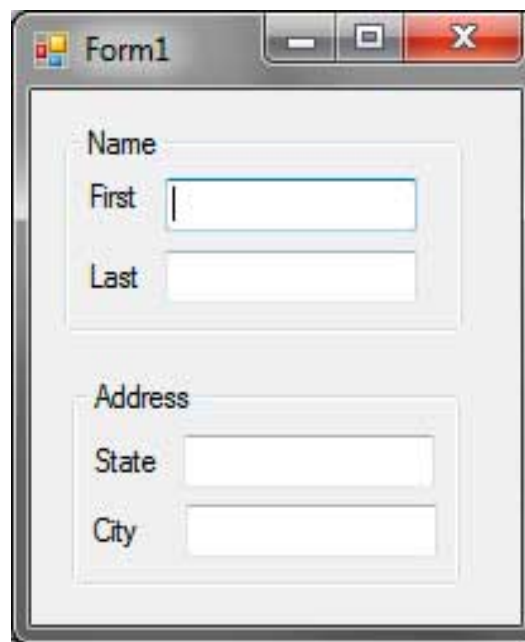
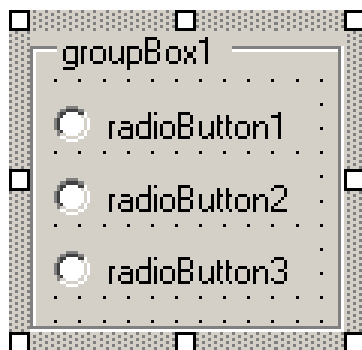
Други Windows Forms контроли - Panel



| СВОЙСТВО | Описание |
|-------------|--|
| BorderStyle | За избор на стила на линията |
| Controls | Колекция от контроли в панела |
| Enabled | Дава възможност за активиране или деактивиране на панела |

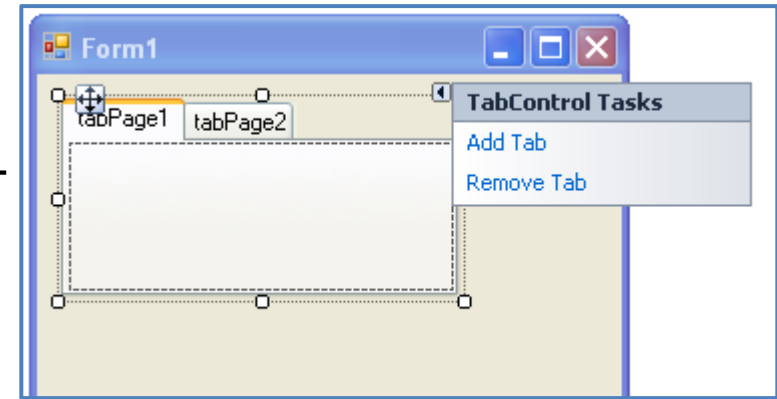
GroupBox

- ▶ **GroupBox** – тази контрола е подобна на Panel, но ви позволява да добавяте надписи на всяка група
- ▶ Това може да стане чрез свойството **Text** на **GroupBox**
- ▶ Контролата **GroupBox** по подразбиране има граници



TabControl и TabPage

- ▶ Контролите **TabControl** и **TabPage** осигуряват ползването на табовете със страници
- ▶ **TabControl** съдържа множество **TabPage** контроли, които се добавят в него чрез свойството **Controls**



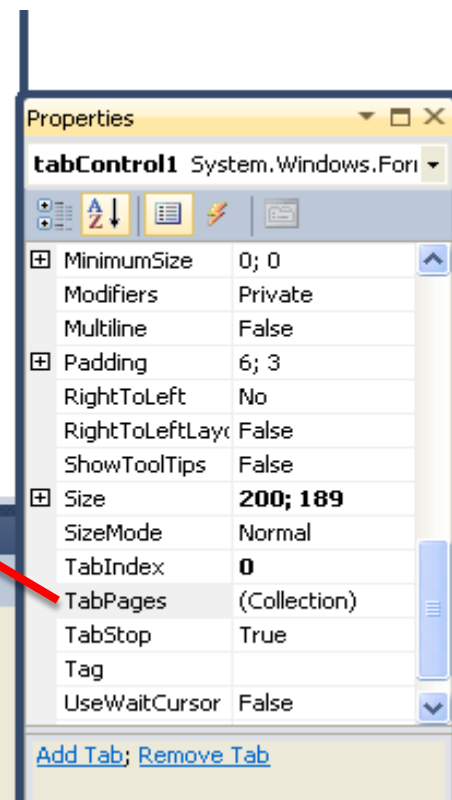
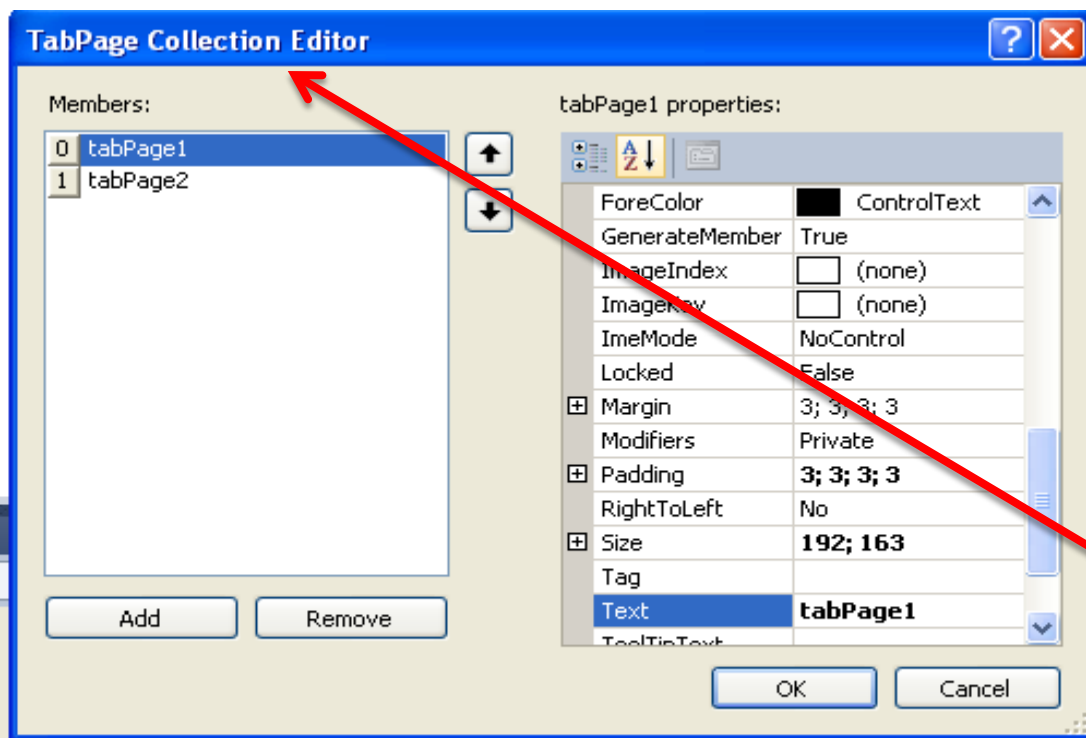
Важни свойства на контролата TabPage

| Свойство | Описание |
|------------|--|
| Controls | Добавя контроли в TabPage |
| ImageIndex | Извлича или задава индекса, показан в таба |
| ImageKey | Извлича или задава ключ за достъп до изображението в ImageList, свързано с TabControl. |
| Text | Текста, който ще се показва в tab бутона на TabPage. |

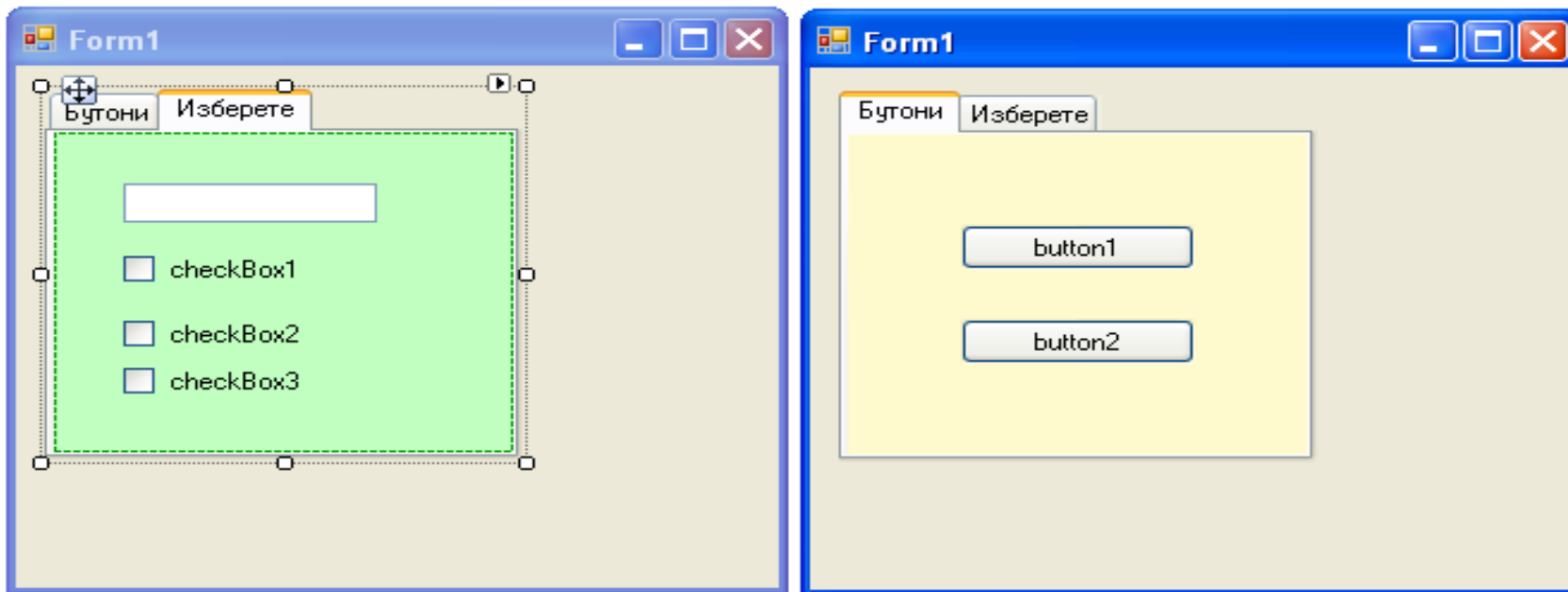
Други Windows Forms контроли

TabControl и TabPage

- ▶ Алтернативно може да изберем свойството **TabPages** на **TabControl**
- ▶ Отваря се следният **TabPages Colection Editor**.



Други Windows Forms контроли



- Свойството ImageList ви позволява да добавяте **картинки** към всеки таб
- За целта първо е необходимо да изберете контролата ImageList, след това да добавите исканите картинки от мястото, където са запазени

ListBox

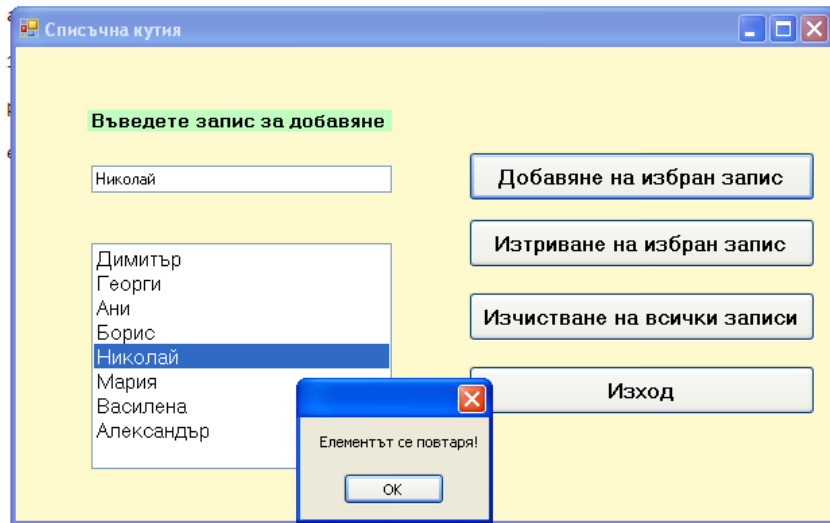
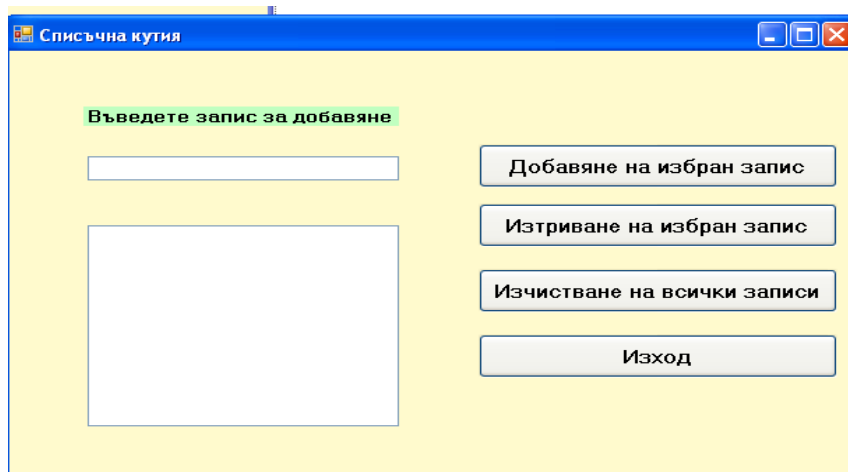
- ▶ Компонента **ListBox** - служи за **показване на списък от елементи (числа, текст и др.) върху екрана**
- ▶ Елементите на списъка се съхраняват в **колекцията Items** (елементи) на класа **ListBox**
- ▶ Потребителят може **да избира елементите** чрез **щракване** с мишката върху тях.
- ▶ По-важните свойства на тази контрола са:
 - ▶ **Items** – колекция, която задава **списъка от елементи** съдържащи се в контролата
 - ▶ **SelectionMode** – разрешава/забранява избирането на **няколко елемента едновременно**
 - ▶ **SelectedIndex, SelectedItem, SelectedIndices, SelectedItems** – връщат **избрания елемент** (или избраните елементи)



ListBox

- ▶ За да манипулираме с елементите от ListBox използваме следните **методи от тип ObjectCollection**
- ▶ **Add (<елемент>)** - **добавя** зададеният като аргумент **елемент в списъка** на кутията
- ▶ **Remove (<елемент>)** – **изтрива** зададеният като аргумент **елемент в списъка** на кутията.
- ▶ **Clear ()** – без аргументи, **изтрива всички добавени** до момента **елементи в списъка**, ако има такива. В резултат списъкът на кутията ще бъде **празен**

Пример за използване на списъчна кутия



Създайте приложение, което съдържа **следните елементи на ГПИ**: списъчна кутия, текстово поле, етикет и четири командни бутона, както е показано на фигурата. Действието на всеки един от бутоните е следното:

- ▶ При въвеждане на произволна (**недублираща се**) стойност в текстовото поле и натискане на бутон **Добавяне** да се извършва **добавяне на тази стойност** в списъчната кутия.
- ▶ Бутон **Изтриване на запис** извършва **изтриване на маркиран елемент** от списъчната кутия.
- ▶ Бутон **Изчистване на всички записи** **изтрива всички елементи** от списъчната кутия
- ▶ Бутон **Изход** **затваря** приложението

Пример за използване на списъчна кутия

```
private void cmdAdd_Click(object sender, EventArgs e)
{
    if (textInput.Text != "")
    {
        int index = listBox1.FindString(textInput.Text, -1);
        if (index < 0) { listBox1.Items.Add(textInput.Text); }
        else
        {
            MessageBox.Show ("Елементът се повтаря!");
            textInput.Text = "";
        }
        textInput.Focus();
    }
}

private void cmdDelete_Click(object sender, EventArgs e)
{
    listBox1.Items.Remove(listBox1.SelectedItem);
}

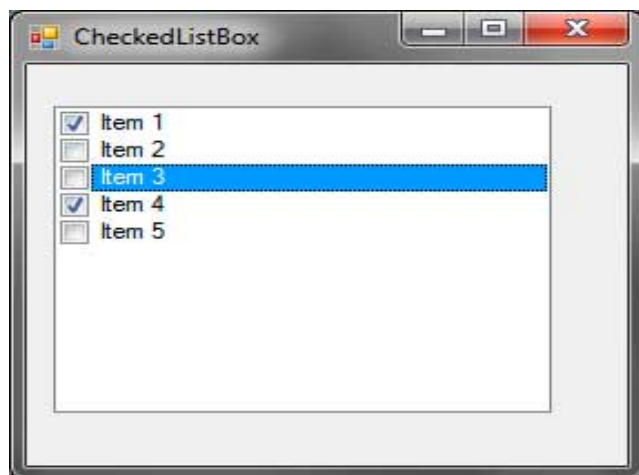
private void cmdExit_Click(object sender, EventArgs e)
{
    Close();
}

private void cmdClear_Click(object sender, EventArgs e)
{
    listBox1.Items.Clear();
}

private void textInput_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Return) cmdAdd_Click(sender, e);
}
```

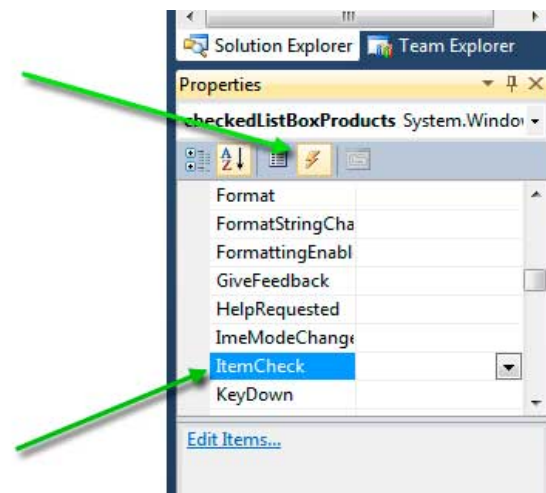
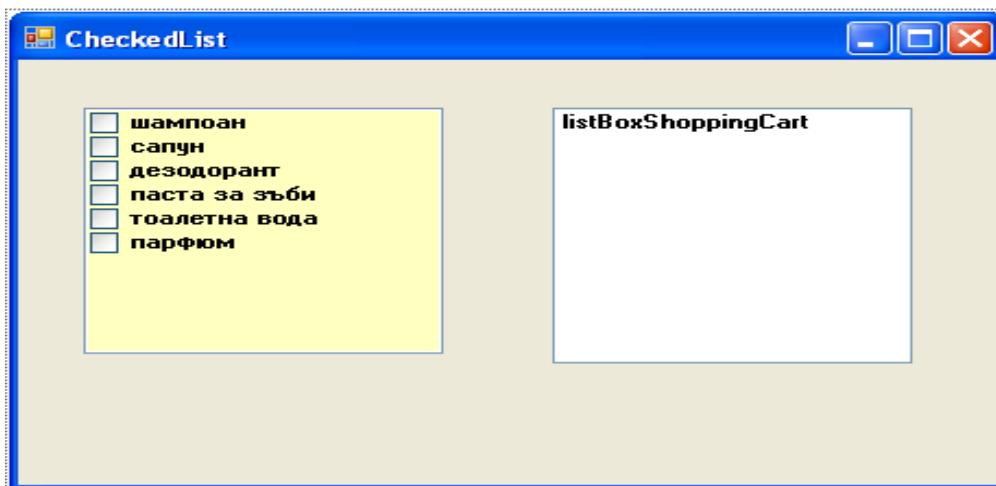
CheckedListBox

- ▶ **CheckedListBox** изобразява списък от възможности за избор "да/не".
- ▶ **По-важни свойства** са:
 - ▶ **Items** – задава **възможностите**, от които потребителят ще избира
 - ▶ **CheckedItems** – връща избраните елементи
 - ▶ **CheckOnClick** – указва дали да поставите отметка в квадратчето, ако е избран елементът



Пример за използване на **CheckedListBox**

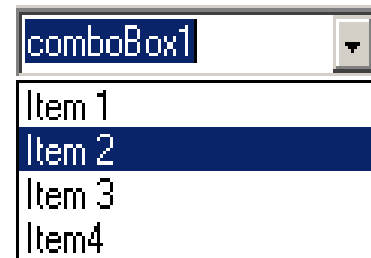
Пример: Създайте нова форма и добавете **CheckedListBox** с **Name = checkedListBoxProducts**. Добавете списъчна кутия **ListBox** със стойност на **Name = listBoxShoppingCart**



```
private void checkedListBoxProducts_ItemCheck(object sender, ItemCheckEventArgs e)
{
    if (e.NewValue == CheckState.Checked)
        listBoxShoppingCart.Items.Add(checkedListBoxProducts.Items[e.Index]);
    else if (e.NewValue == CheckState.Unchecked)
        listBoxShoppingCart.Items.Remove(checkedListBoxProducts.Items[e.Index]);
}
```

ComboBox

- ▶ **ComboBox** представлява кутия за **редакция на текст** с възможност за **drop-down алтернативен избор**

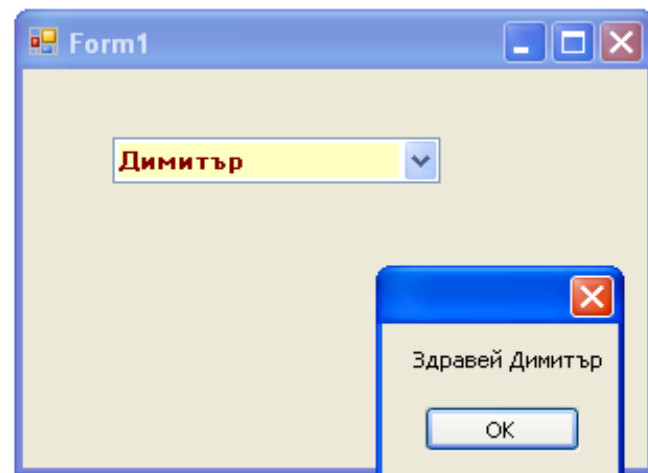
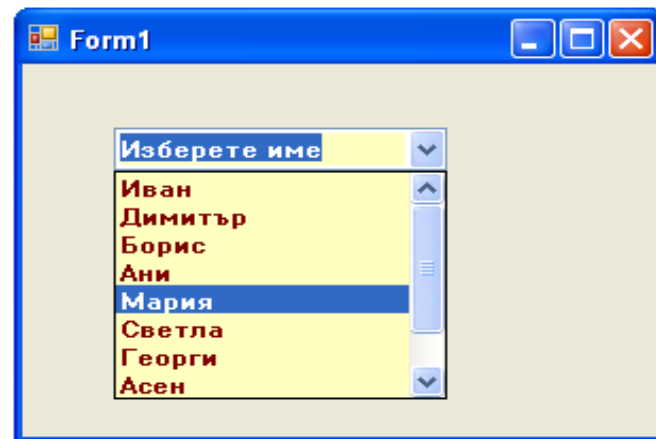


- ▶ **По-важни свойства**

- ▶ **Text** – съдържа **въведения текст**
- ▶ **Items** – задава възможните стойности, от които потребителят може да избира.
- ▶ **DropDownStyle** – задава стила на контролата – дали само се избира стойност от списъка или може да се въвежда ръчно и друга стойност
- ▶ **SelectedItem** – получава или задава стойността от избраната стойност на списъка

ComboBox

- ▶ **ComboBox** е контрола, която позволява на потребителя да избере от набор от опции
- ▶ **ComboBox** прилича на **текстово поле с един бутон** в дясната си страна
- ▶ Когато бутонът се натисне, **ComboBox** показва **падащ прозорец**, съдържащ **списък с опции/единици на разположение**
- ▶ Потребителят **може да избере** от тези опции, като **щракне една от тях**
- ▶ След това избраната опция ще бъде текстът вътре в **ComboBox**



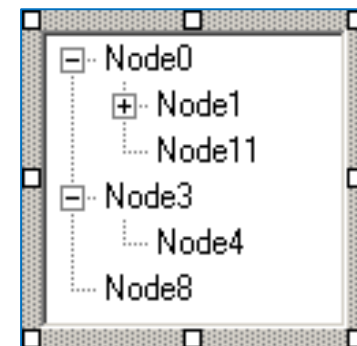
```
private void comboBoxName_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectedName = comboBoxName.SelectedItem.ToString();
    MessageBox.Show("Здравей " + selectedName);
}
```

TreeView и RichTextBox

- ▶ **TreeView** – изобразява дървовидни данни

- ▶ Важни **свойства**:

- ▶ **Nodes** – съдържа дървото
(списък от **TreeNode** обекти)
- ▶ **SelectedNode** – избраният възел



- ▶ **RichTextBox** е кутия за редакция на текст с форматиране (Rich Text Format)
- ▶ **LoadFile(...)** и **SaveFile(...)** **зареждат** или **записват** **текста от контролата** в RTF файл или в текстов файл
- ▶ Свойствата **SelectionStart** и **SelectionEnd** служат за **извличане и задаване на областта от текста, която е маркирана**
- ▶ Чрез свойствата **SelectionFont**, **SelectionColor** и **SelectionAlignment** могат да се задават **шрифт, цвят и подравняване** на текущия маркиран текст

RichTextBox

- ▶ **RichTextBox** е аналогична на контролата TextBox, но тя ви позволява да **форматирате различни части от текста вътре в нея**
- ▶ **TextBox** обикновено се използва **за въвеждане на текст** от потребителя
- ▶ **RichTextBox** се използва, **за да покаже форматиран текст** и да го **запишете в RTF**
- ▶ **TextBox** and **RichTextBox** са наследници на класа **TextBoxBase** и имат **общи свойства**



```
richTextBox1.SelectionColor = Color.Red;  
richTextBox1.SelectionBackColor =  
Color.Yellow;
```


RichTextBox

| Събития | Описание |
|------------------|--|
| LinkClicked | Когато се кликва върху връзка |
| Protected | Когато потребителя се опитва да модифицира защитен текст |
| TextChanged | Текстът вътре в RichTextBox е променен |
| SelectionChanged | Когато избраният текст е променен |

Често използвани методи:

- Текущият **избран текст** `richTextBox1.Select(10, 11);`
- Демаркиране на **избрания текст** `richTextBox1.DeselectAll();`
- Добавяне на **лента за превъртане** `Adding Scrollbars`

PictureBox



- ▶ **PictureBox** се използва за изобразяване на **картинки**
- ▶ Картинката, която **ще се изобразява**, се задава чрез свойството **Image**.
- ▶ Свойството **SizeMode** задава дали картинката да се **разшири/намали** или **центрира** при изобразяването в контролата
- ▶ Картинките, използвани в контролата **PictureBox**, се запазват като ресурси
- ▶ Те се записват в XML формат в **.resx** файла на съответната форма и при компилация се запазват като ресурси в асемблите на приложението

PictureBox

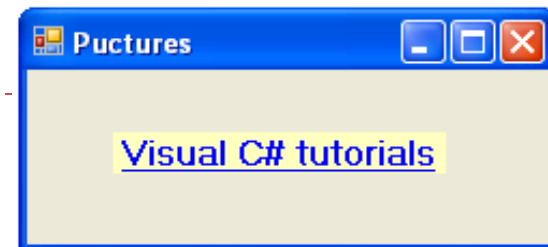
► Важни свойства:

| Свойство | Описание |
|---------------|---|
| Image | Картината, която ще се изобрази на контролата |
| ImageLocation | Пътя до картината, която ще се изобрази в PictureBox. |
| InitialImage | Картината, която ще се визуализира в началото |
| SizeMode | Указва как ще се изобрази картината. |
| WaitOnLoad | Ако е true, блокира всяко взаимодействие към формата, докато картината се зарежда |

| PictureBoxSizeMode | Описание |
|--------------------|---|
| Normal | Картината ще се позиционира от горния ляв ъгъл на PictureBox и ако картината е по-голяма от PictureBox, ще е отрязана |
| StretchImage | Оразмерява картината съответно на размера на PictureBox. |
| AutoSize | Оразмерява PictureBox съответно на размера на картината |
| CenterImage | Картината е центрирана в PictureBox. Ако картината е по-голяма от PictureBox,, ще бъде отрязана |
| Zoom | Разполага цялата картина в PictureBox като поддържа съотношението на нейните размери |

LinkLabel

- ▶ **LinkLabel** - препратка (hyperlink):
 - ▶ **Text** – съдържание на връзката
 - ▶ **LinkClicked** – активира се при щракване върху препратката
- ▶ **LinkLabel** е подобен на обикновен етикет, но е подчертан и **прилича на връзка в уеб страница**
- ▶ **LinkLabel** може да се използва за **връзка към файлове, директории или уеб страници**
- ▶ Когато поставите мишката върху **LinkLabel**, **курсорът** ще се **превърне в ръка**



Свойства на LinkLabel

| Свойство | Описание |
|------------------|--|
| BorderStyle | Стила на линиите около етикета |
| FlatStyle | Определя външния вид на LinkLabel. Когато е настроен на PopUp, бутонът е леко повдигнат, когато мишката е върху него. |
| LinkArea | Определя част от текста, която ще се появи като връзка |
| LinkColor | Цвета на непосетена връзка |
| Links | Определя колекция от връзки, които ще се визуализират. Това не са актуалните връзки, които ще бъдат посетени, а части от LinkLabel, които ще се визуализират като връзки |
| LinkVisited | Когато е настроен на true , цветът на връзката ще бъде разменен с цвета на свойството VisitedLinkColor |
| TextAlign | Задава местоположението на текста в рамките на контролата |
| VisitedLinkColor | Задава цвят на посетена връзка |

LinkLabel трябва да има манипулатор на събитие **LinkClicked**, което трябва да съдържа действителната навигация до желаното място за връзка.

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("http://visualcsharp tutorials.com");
    linkLabel1.LinkVisited = true;
}
```

Добавяне на методи за достъп до връзка

- ▶ Използваме метод **System.Diagnostics.Process.Start**, за да се отвори вашият браузер и да се намери зададеният уебсайт
`System.Diagnostics.Process.Start("http://visualcsharp tutorials.com");`
- ▶ След това задаваме **linkLabel1.LinkVisited = true;** за да покажем, че **връзката е посетена**
- ▶ **Забележете**, че тук може да зададете също **директория или път до файл**, за да ги отворите
- ▶ Ако желаете да посочите **повече от една връзка** в контролата **LinkLabel**, трябва да използвате свойството **LinkClickedEventArgs**

Добавяне на методи за достъп до повече от една връзка

```
private void FormLinkLabel_Load(object sender, EventArgs e)
{
    LinkLabel.Link googleLink = new LinkLabel.Link(0, 6);
    LinkLabel.Link fmiLink = new LinkLabel.Link(9, 5);
    LinkLabel.Link puLink = new LinkLabel.Link(17, 10);

    googleLink.Name = "Google";
    fmiLink.Name = "FMIIT";
    puLink.Name = "PU Plovdiv";

    linkLabel1.Links.Add(googleLink);
    linkLabel1.Links.Add(fmiLink);
    linkLabel1.Links.Add(puLink);
}
```



Добавяне на програмен код на събитие **LinkClicked**

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    switch (e.Link.Name)
    {
        case "Google":
            System.Diagnostics.Process.Start("http://google.com");
            break;
        case "FMIIT":
            System.Diagnostics.Process.Start("http://fmi-plovdiv.org");
            break;
        case "PU Plovdiv":
            System.Diagnostics.Process.Start("https://www.uni-plovdiv.bg");
            break;
    }
}
```


The MonthCalendar Control

System.Windows.Forms.MonthCalendar

- ▶ **MonthCalendar** прилича на календар и показва месеца и **ВСИЧКИ ДНИ**
- ▶ MonthCalendar ви позволява да **изберете** месец и дата
- ▶ За да изберете месец, **кликнете на лявата** или **дясната стрелка** за преместване на месец към предишен или следващ месец

Form1

Ноември 2013 г.

| понеделник | вторник | сряда | четвъртък | петък | събота | неделя |
|------------|---------|-------|-----------|-------|--------|--------|
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Днес: 21.11.2013 г.

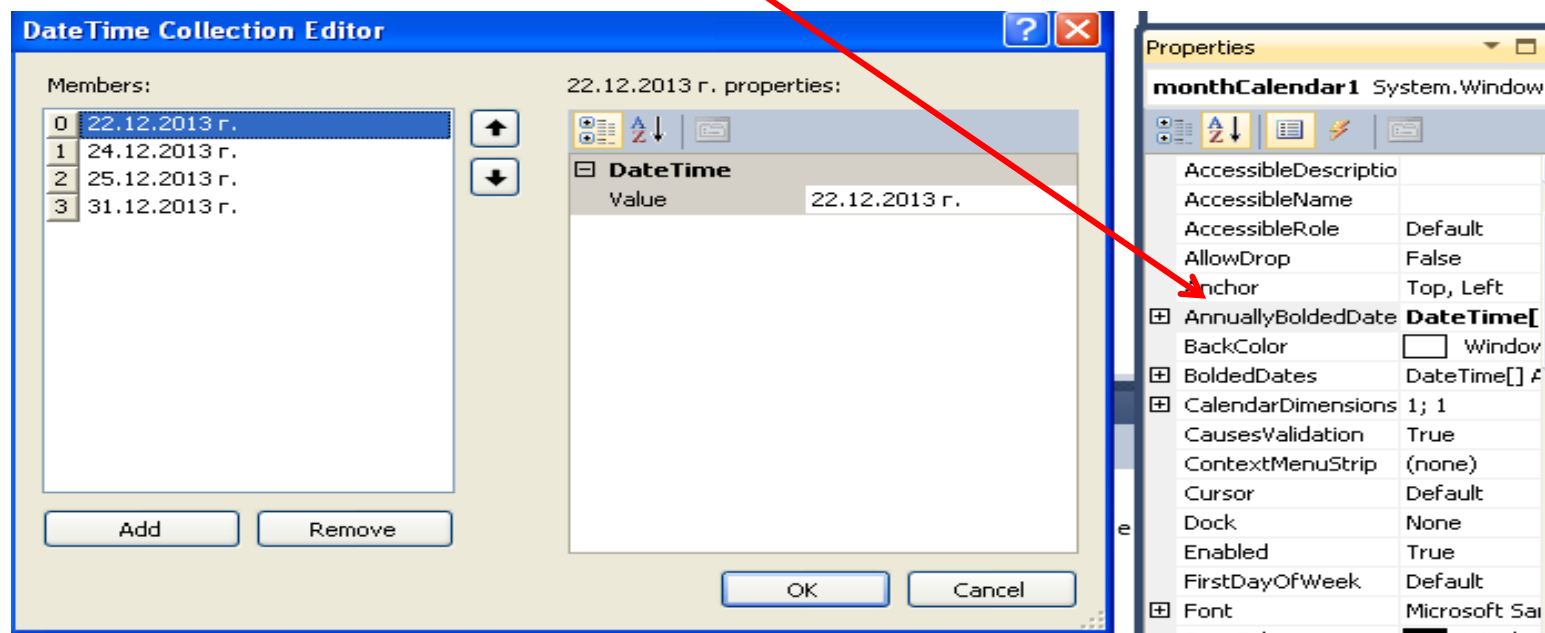
Свойства на контролата **MonthCalendar**

► Важни свойства

| Свойство | Описание |
|---------------------------|--|
| AnnualyBoldedDates | колекция от дати, които всяка година ще се визуализират като bold в контролата MonthCalendar |
| SelectionStart | If the user selects a range of date, this property indicates the first date in the range of dates. |
| ShowToday | Specifies whether to show the date today at the bottom of the control. |
| ShowTodayCircle | If set to true, the date today in the control will be enclosed with a square or a circle. |
| ShowWeekNumbers | Specifies whether the week number will be shown at the left of each row of the control. |
| TodayDate | The date used by the MonthCalendar as today's date. |

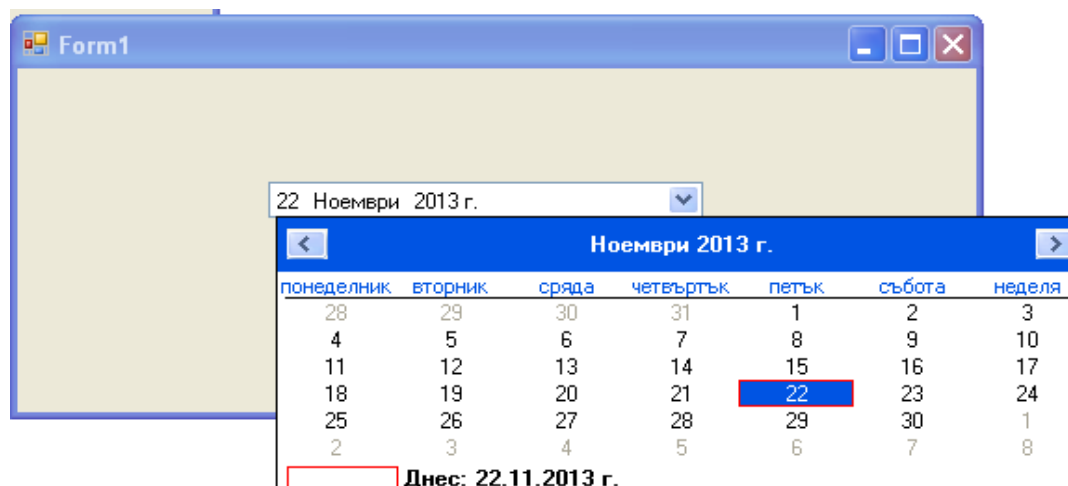
Настройка на свойства

▶ AnnuallyBoldedDates



The DateTimePicker Control

- ▶ The **DateTimePicker** се използва за **показване на единична дата**
- ▶ Контролата се появява по **подразбиране като combo box с икона** за календар в дясната част
- ▶ Вие може да изберете **всеки компонент на датата**, като използвате стрелка за избор на индивидуални компоненти



The ListView Control

System.Windows.Forms.ListView

- ▶ позволява да се покаже **списък с елементи с различни изгледи** и да добавите **икони за всеки от тях**
- ▶ състои се от **елементи** (ListViewItems), които образуват **таблична структура от ред и колони**
- ▶ всеки **ListViewItem** има **етикет** и в първата колона може да има **икона до тях**
- ▶ една **често срещана употреба** на контрола **ListView** е да се покаже **списък с файлове и папки на потребителя**

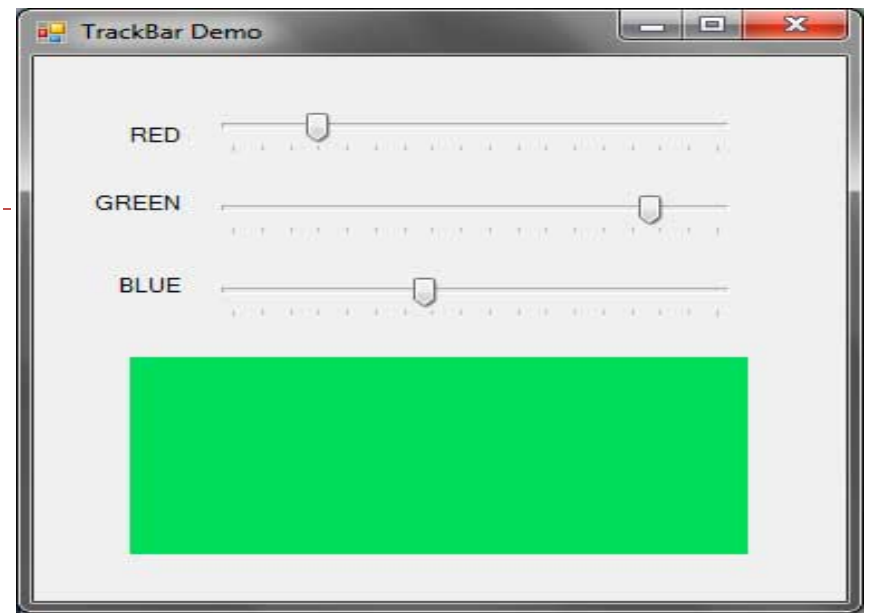
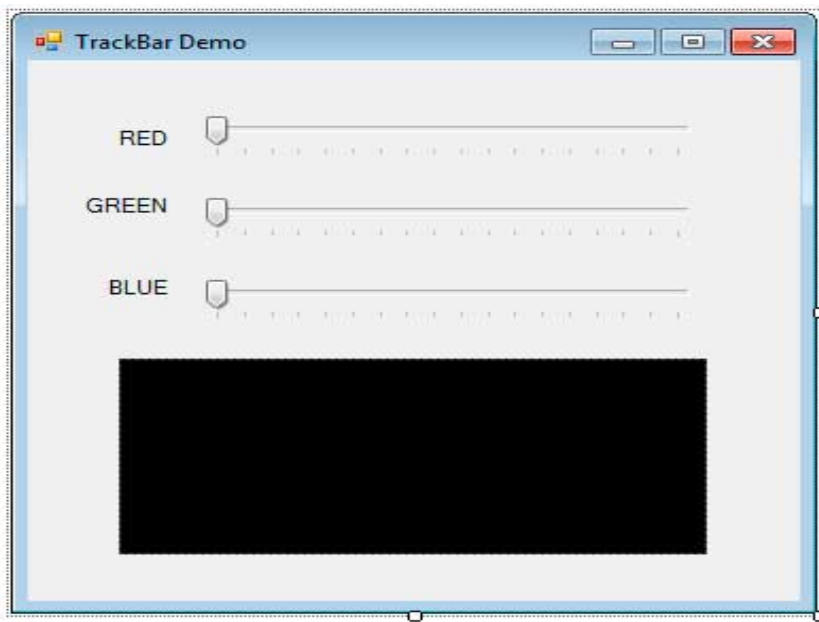
The TrackBar Control



System.Windows.Forms.TrackBar

- ▶ TrackBar изглежда като слайдер с палец, който може да се движи, за да се коригира стойността.
- ▶ Частта, в която се намира палеца представя текущата стойност.

| Свойство | Описание |
|---------------|--|
| LargeChange | Стойността , която добавяте или изваждате, когато потребителят натисне клавиша Page Up или Page Down или когато кликнете от двете страни на TrackBar |
| Maximum | Определя максималната стойност, която TrackBar може да има. Стойността на най-дясната или горна отметка |
| Minimum | Определя минималната стойност, която TrackBar може да има. Стойността на най-лявата или долна отметка |
| Orientation | Указва дали ориентацията на TrackBar е хоризонтален или вертикален. |
| SmallChange | Задава добавена или извадена стойност, когато потребителят натисне клавишите със стрелки, за да преместите палеца на TrackBar. |
| TickFrequency | Задава колко често показалеца ще се показват в зависимост от обхвата на стойността |
| TickStyle | Определя позицията, където показалеца ще се показва. Възможни стойности са: None: Don't show the ticks. Bottom: Show the ticks at the bottom. Top: Show the ticks at the top. Both: Show the ticks at both top and bottom. |
| Value | Съдържа текущата стойност, определена от TrackBar. |



```
private void trackBar_Scroll(object source, EventArgs e)
{
    int red = trackBarRed.Value;
    int green = trackBarGreen.Value;
    int blue = trackBarBlue.Value;

    Color color = Color.FromArgb(red, green, blue);

    panelColor.BackColor = color;
}
```

The Timer Control

System.Windows.Forms.Timer

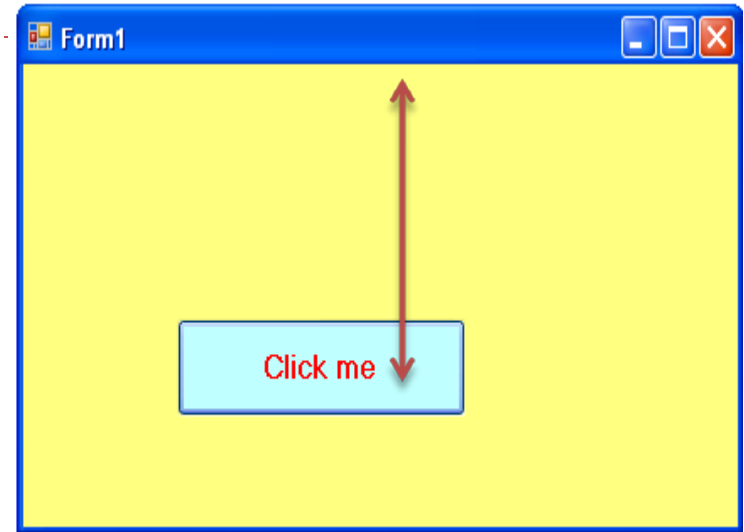
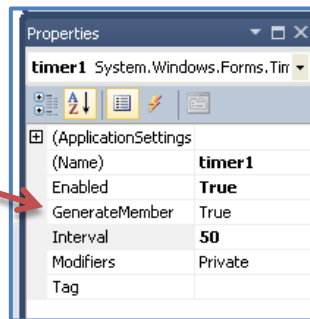
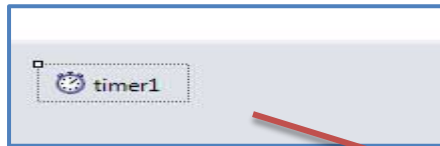
- ▶ Използва се, за да **изпълнява** команди **за всеки период от време**, който е определен с **интервал**
 - ▶ Например, ако искате да изписвате определено име на всеки 5 секунди, можете да използвате контрола Timer
- ▶ **Timer** е **невизуален контрол** и може да се види в частта за компоненти, след като го плъзнете към формата от кутията с инструменти

Важните свойства са **две** - **Enabled** и **Interval**.

- ▶ **Enabled** се задава с **true**, когато **започва да се брои времето** и **спира** при стойност **false**
- ▶ **Interval** определя **интервала от време между извикване на събитието Tick (event)**, което е **по подразбиране** за Timer
 - ▶ **Стойността**, която е приета от Interval е в **милисекунди**. Ако дадете стойност **5000** като **интервал**, след това събитието Tick ще бъде извикано на всеки 5000 милисекунди или 5 секунди

Пример: Движещ се обект

- ▶ Отворете нов файл и добавете **timer control** в частта за компоненти
- ▶ Задайте **Interval property = 50** и **Enabled = true**
 - ▶ Така анимацията ще започне в момента, в който потребителят стартира програмата
- ▶ Генерирайте манипулатор на събитие - **Tick**
 - ▶ Това събитие ще се извиква на всеки **50 милисекунди**
- ▶ Добавете следният програмен код към събитието Tick:



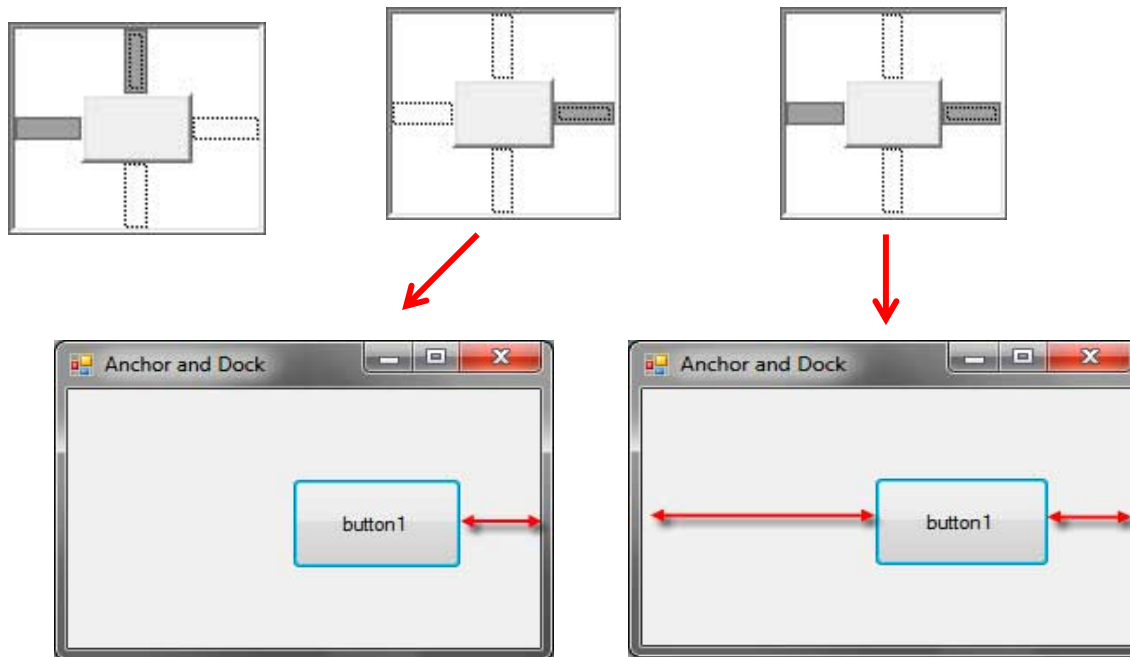
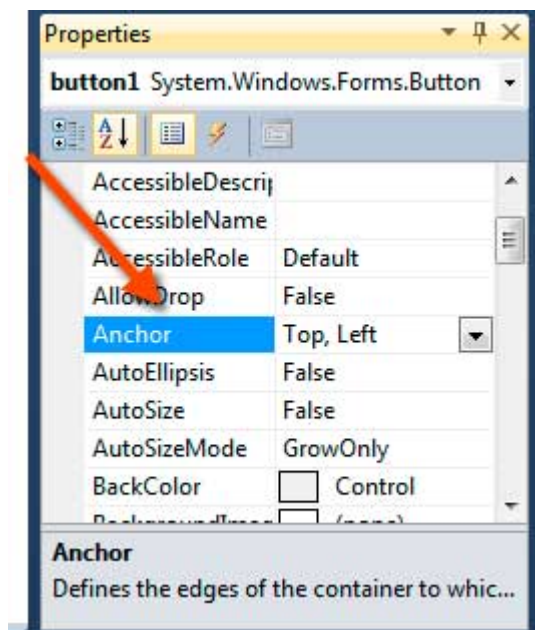
```
private int velocity = 5;
private void timer1_Tick(object sender, EventArgs e)
{
    if (button1.Top <= 0 || button1.Bottom > this.ClientSize.Height)
        velocity = -velocity;
    button1.Top += velocity;
}
```

Anchoring Controls

- ▶ Един от начините да се определи местоположението на контрола е с помощта на **свойство Anchor**
- ▶ Свойството **Anchor (котва)** указва как контролата се държи, когато променят размера на формата
- ▶ Може да се уточни дали контролата ще се **закотви към различните краища на формата**
- ▶ Можем също да се уточни как **контролите ще променят размера си**

Свойства на контрола Anchor

| AnchorStyle | Описание |
|-------------|--|
| Bottom | Контролата е закотвена към долната рамка на контейнера |
| Left | Контролата е закотвена към левия край на контейнера |
| Right | Контролата е закотвена към десния край на контейнера. |
| Top | Контролата е закотвен към горния край на контейнера |
| None | Контролата не е закотвена към нито един край на контейнера |

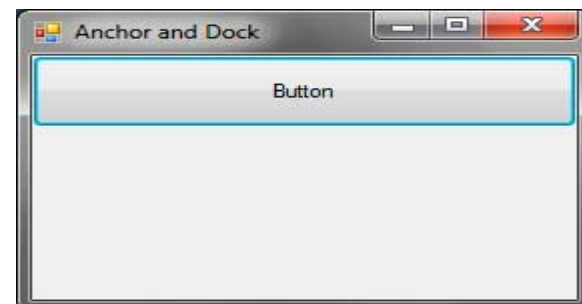
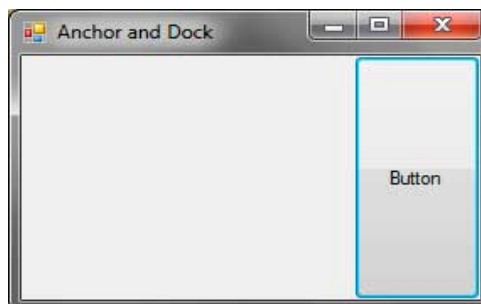
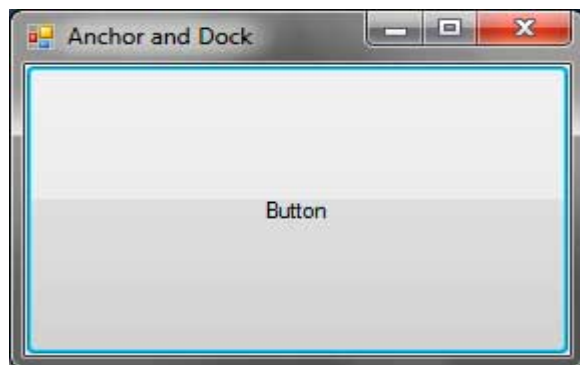
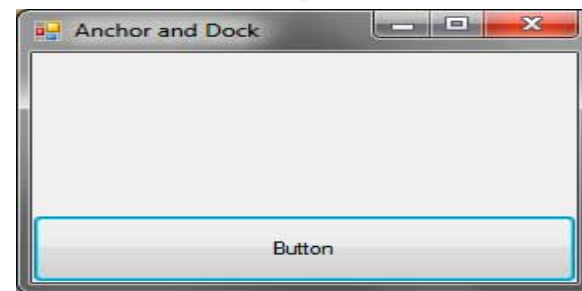
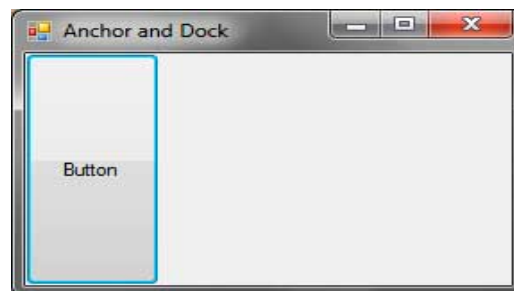
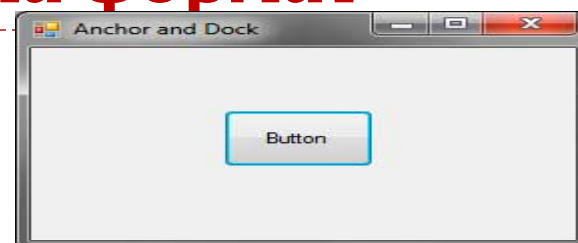
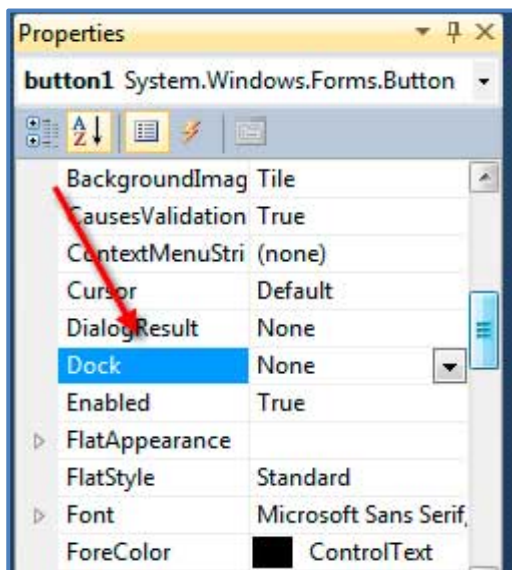


Docking Controls

- ▶ Свойство **Dock** ви позволява да **закачите контролата към някой от ръбовете на формата или контейнера**
- ▶ Това е друг начин да остане контрола непроменена, когато промените формата

| DockStyle | Описание |
|-----------|---|
| Bottom | Контролът е долепен към долната рамка на контейнера |
| Fill | Всички краища на контрола са фиксирани към всички ръбове на неговия контейнер и е оразмерен подходящо |
| Left | Левия край на контрола е закачен към левия ръб на контейнера |
| Right | Десния край на контролата е закачен към десния ръб на контейнера |
| None | Контролът не е свързан с контейнера |
| Top | Контролът е долепен към горната рамка на контейнера |

Различни положения на docking button control в различните рамки на формат

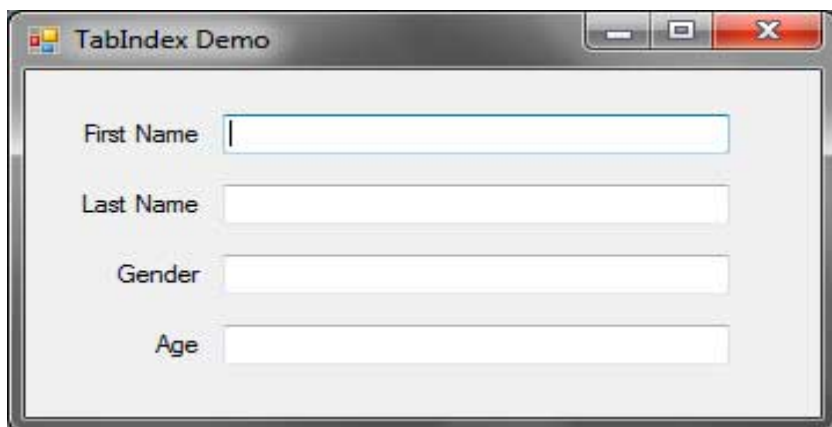


-
- ▶ Ако имате **няколко контроли** и използвате docking стойности, те ще бъдат поставени една до друга или една над друга



Използване на свойство **TabIndex**

- ▶ Можете да се придвижвате по **всяка контрола** чрез натискане на **tab** от клавиатурата
- ▶ Всяка контрола има свойство **TabIndex**, което определя **реда на фокусиране** върху всяка контрола (focus) при натискане на tab
- ▶ Пример с четири текстови кутии



| TextBox | TabIndex |
|--------------|----------|
| txtFirstName | 1 |
| txtLastName | 2 |
| txtGender | 3 |
| txtAge | 4 |

Пример

- ▶ Когато стартирате програмата, фокусът ще бъде върху контрола, която има най-малката стойност на индекса – **txtFirstName**
- ▶ Ако натиснете **tab**, фокусът ще отиде на контрола със следващият TabIndex – **txtLastName**
- ▶ Искаме получаването на фокус за всяка контрола да бъде в **правилната последователност**, така че потребителят да не се дразни
- ▶ Определяне на правилната последователност зависи от позицията на контролите
- ▶ За формата в примера, контролите са разположени вертикално (от горе до долу), и съответния TabIndex трябва да се увеличава отгоре – надолу

Благодаря за вниманието!

