

# **ПРОЕКТИРАНЕ НА МЕНЮТА**

**Лекция 7**

# ТЕМИ

---

- ▶ Проектиране на менюта
- ▶ Стандартно и йерархично меню
- ▶ Контекстно меню
- ▶ Добавяне и премахване на елемент от меню
- ▶ Ленти с инструменти
- ▶ Диалози за избор на файл
- ▶ Диалози за избор на шрифт и цвят

# МЕНЮТА

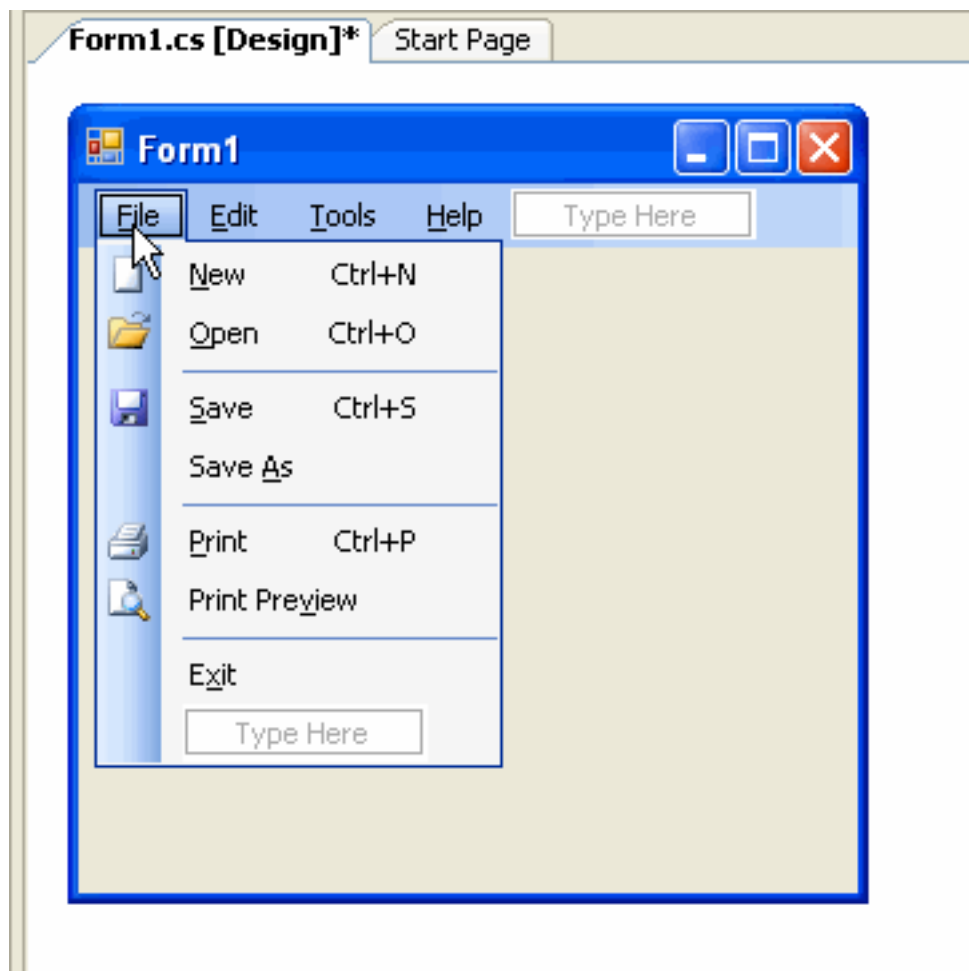
---

## Менюто

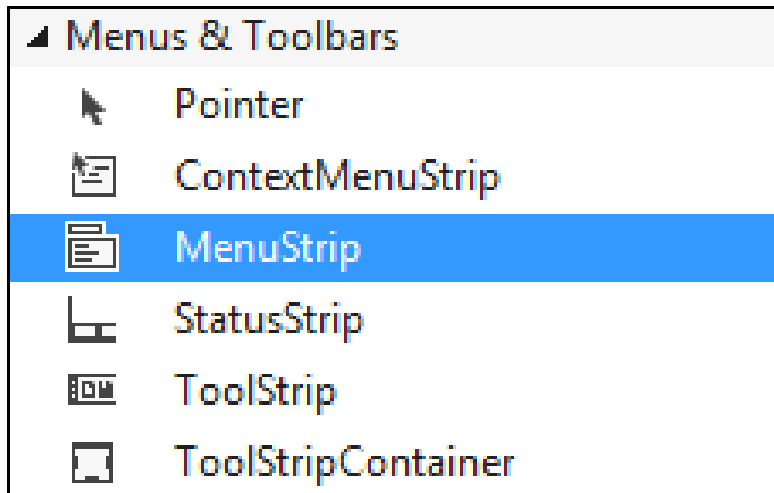
- ▶ средство, чрез което потребителят по удобен начин указва **започването на дадена операция**
- ▶ съдържа **различни команди (операции)**, които потребителят може да използва
- ▶ предоставя възможност за **бърз достъп** при изпълнение на **избрана операция** чрез директен избор на съответната команда

# МЕНЮТА

---



# МЕНЮТА

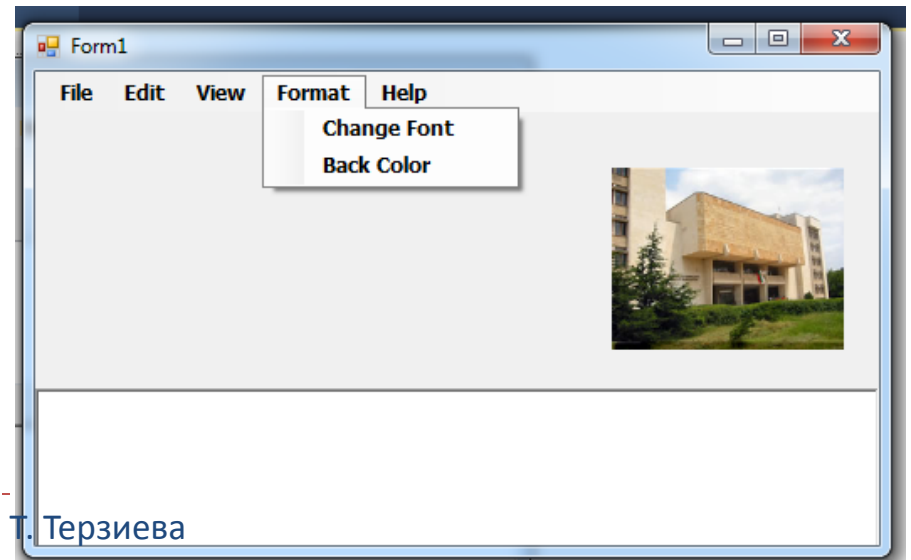
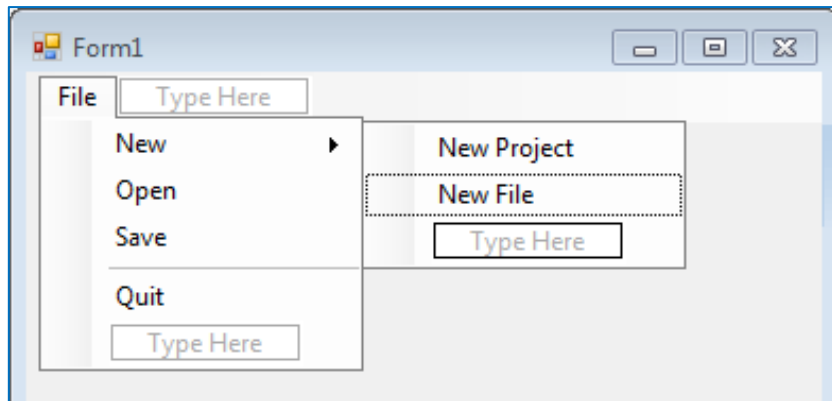


В Windows Forms за работа с менюта се използват **класовете**

- ▶ **MainMenu, MenuItem**
- ▶ **MenuStrip**
- ▶ **ContextMenuStrip**
- ▶ **StatusStrip**
- ▶ **ToolStrip**

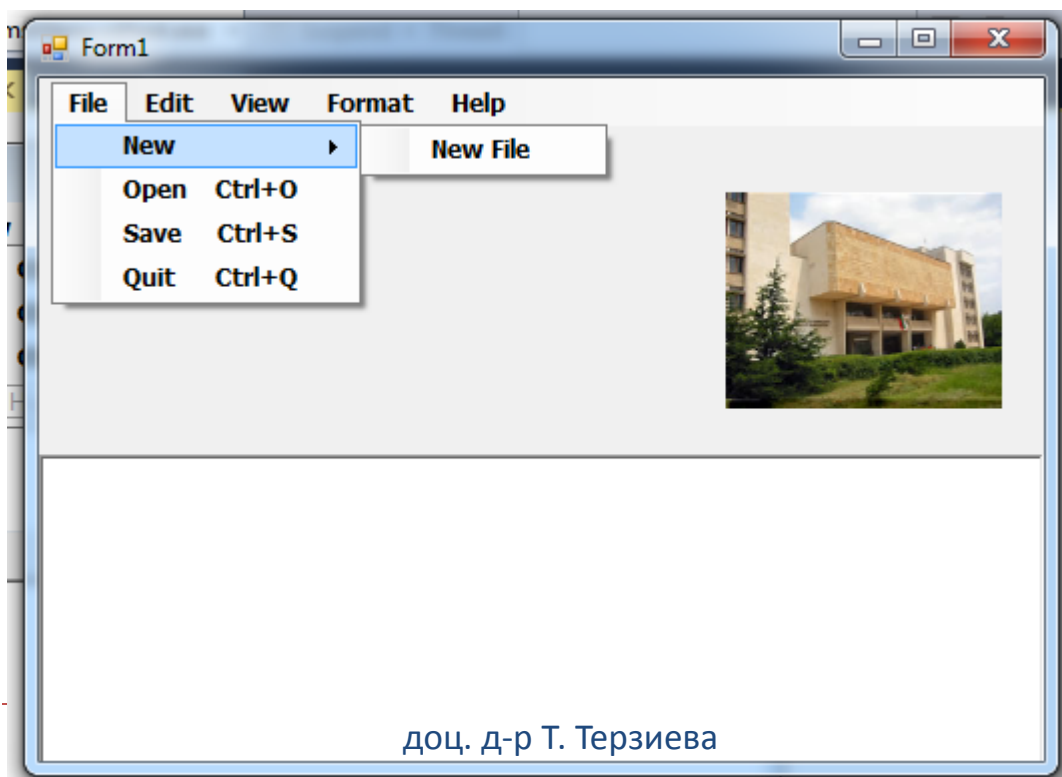
# Менюта

- ▶ Менютата са организирани в **йерархична структура**
- ▶ **Даден елемент от меню**, сам по себе си може да **представява меню**, когато се използва за визуализиране на елементи от подменю
- ▶ Менютата от **първо ниво** представляват стандартни **падащи менюта** и образуват **лентата с основни действия** за дадено приложение



# Йерархична структура на менюто

- Всяко меню може да съдържа в себе си списък от **MenuItem** елементи, които представят **отделните възможности за избор (команди)** от избраното меню
- **MainMenu** представлява **стандартно падащо меню**



# РЕАЛИЗИРАНЕ НА МЕНЮТА

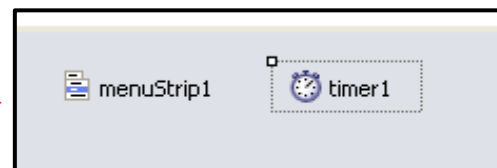
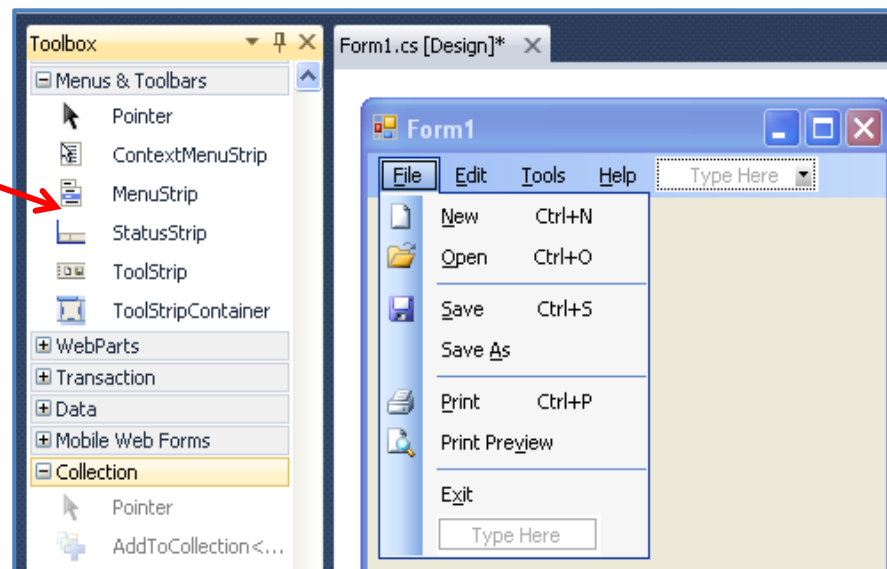
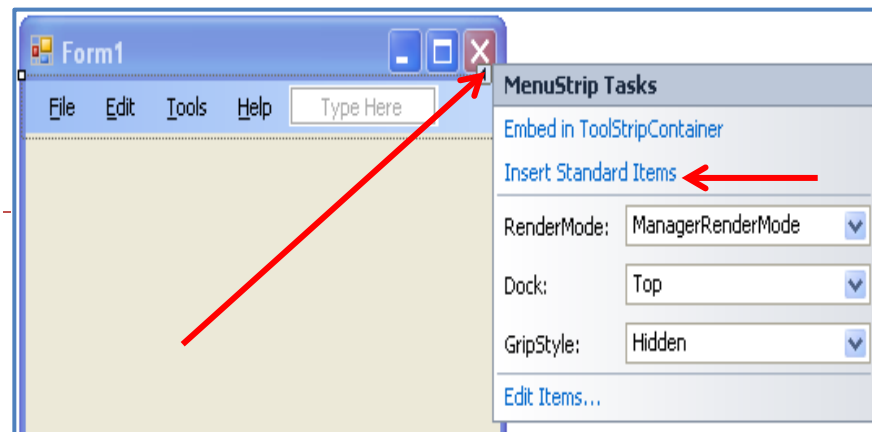
---

- ▶ За реализиране на **менюта** в средата Visual C# се използват компонентите **MenuStrip** и **ContextMenuStrip**
- ▶ Тези компоненти **нямат графичен образ** върху формата на приложението (т.е. добавят се в специално поле под нея) и представляват **списък от MenuItem елементи за избор**
- ▶ Всеки **MenuItem** елемент вече има **реален графичен образ** върху формата и може да бъде, както **команда в приложението**, така и **родителско меню** за други елементи



# Контрола MenuStrip

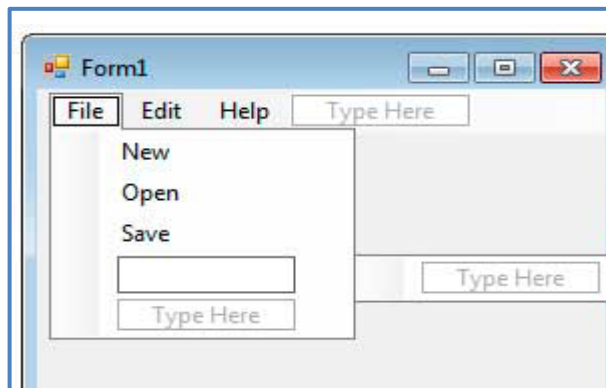
- ▶ Може да добавите ленти с менюта във форма с помощта на контрола **MenuStrip** от Toolbox
- ▶ Може да видите контрола **MenuStrip** в раздела, **разположен в долната част на дизайнера**
- ▶ Тук се намират някои **неграфични компоненти** като контролен таймер и много други



# Основни свойства на контролата MenuStrip

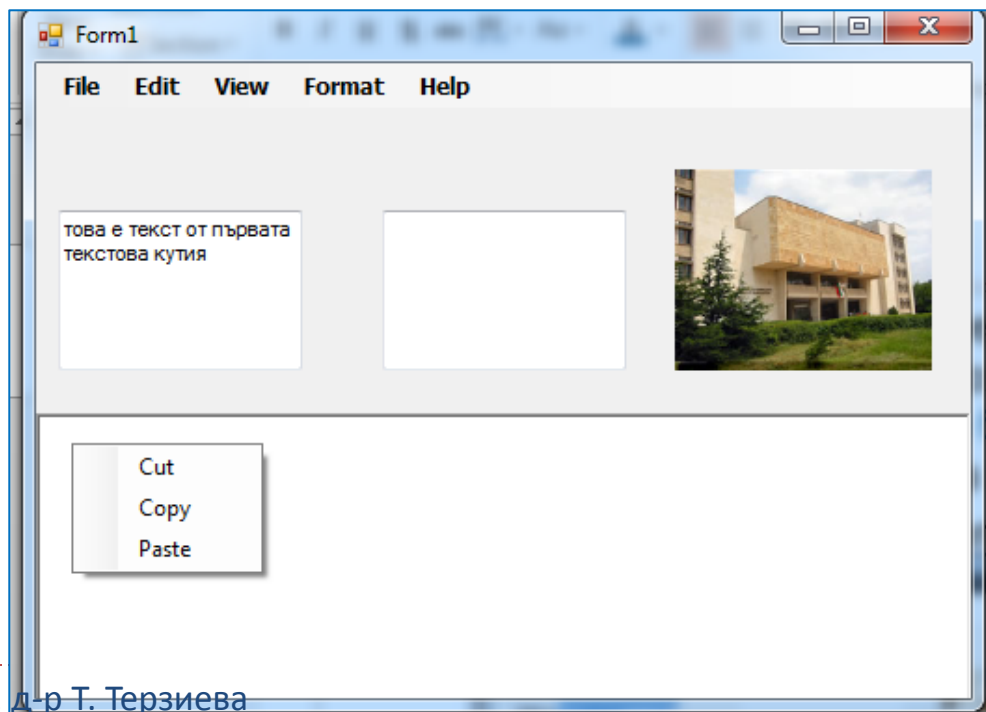
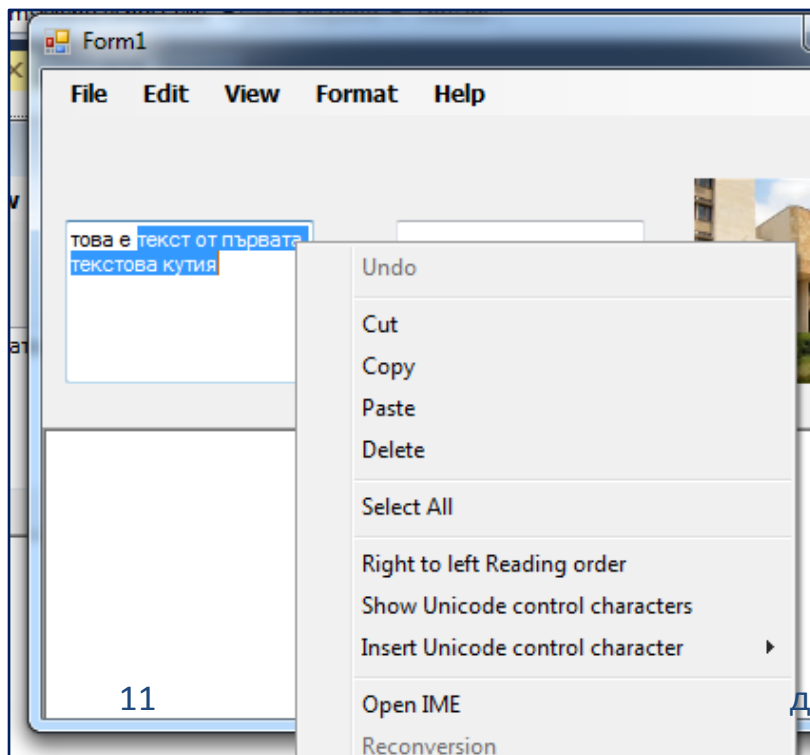
Свойство	Описание
Dock	Определя на кое място ще се постави лентата с менюта. По подразбиране е <b>Top</b>
GripStyle	Позволява ви да покажете връзката, която използвате за съхранение на елементи от менюто
Items	Колекция от <b>top-level</b> меню
Stretch	Задава дали MenuStrip се простира от край до край в неговия контейнер

➤ За да **добавите подменюта**, изберете елемент от менюто, ще се отвори контейнер за **избор на подменюта**



# МЕНЮТА

- ▶ **ContextMenuStrip** представлява **КОНТЕКСТНО МЕНЮ** (popup меню), което се появява, когато потребителят щракне с десния бутон на мишката върху контрола или някъде във формата.
- ▶ Съдържа списък от **MenuItem** елементи



# MenuItem

- ▶ **MenuItem** елементите представляват **отделните възможности за избор (команди)**, показвани в **MainMenu** или **ContextMenu**
- ▶ Всеки **MenuItem** елемент може да бъде **команда в приложението** или **родителско меню за други елементи**, (**менютата могат да се влагат**)
- ▶ По-важни **събития и свойства** на класа **MenuItem** са:
  - ▶ **Text** – задава заглавие на елемента, например “&New” или “Op&en...” или “–”. Символът “&” задава **горещ клавиш за бързо избиране** на съответния елемент. Поставя се **преди дадена буква** от текста на елемента. Елемент от менюто с текст “–” задава **разделител**
  - ▶ **Shortcut** – **кратък клавиш**, асоцииран с този елемент
  - ▶ **Click** – **събитие “избиране”** – активира се при избиране на елемента от менюто

# Клавиши за достъп

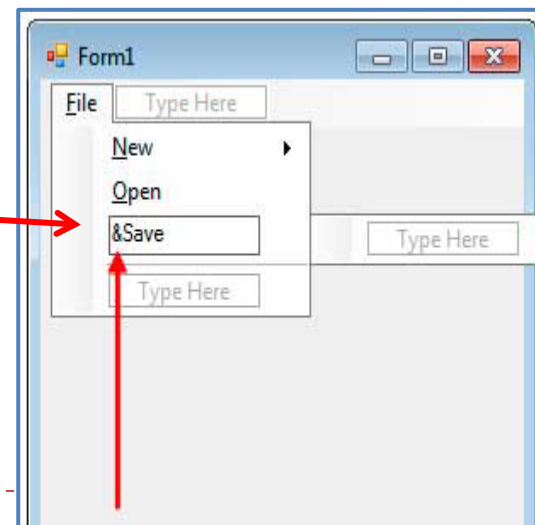
---

- ▶ **Клавишите за достъп** се използват **за обхождане на йерархията от менюта** в комбинация с **клавиша Alt**
- ▶ Това се постига чрез поставяне на знака **амперсанд (&)** **пред буквата**, която ще се използва като **клавиш за достъп**, в заглавието на менюто или елемента от менюто
- ▶ **Клавишите за достъп** трябва да бъдат **уникални** на всяко ниво от йерархията на менюта

# Добавяне на клавиши за бърз достъп

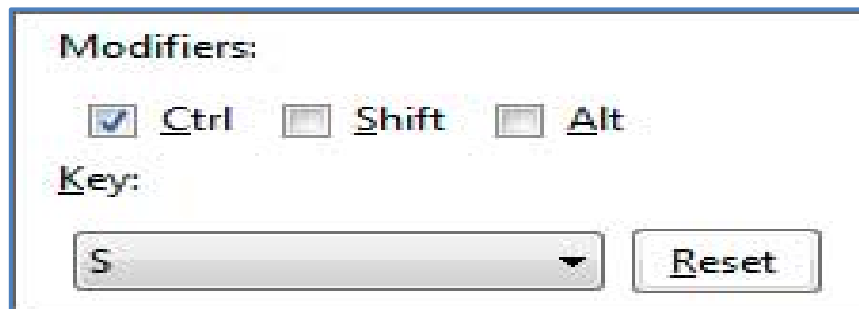
## Shortcut Keys to a Menu Item

- ▶ Буквата следваща знака **амперсанд “&”** ще бъде показана като **“shortcut key”**
- ▶ Например **&Save** ще покаже **S** като “shortcut key”
- ▶ **E&xit** ще покаже **X** като “shortcut key”
- ▶ В дизайнера клавиша за бърз достъп ще бъде подчертан
- ▶ Използването на този тип клавиш **изисква използването на клавиша Alt**
- ▶ За да активирате елемент от менюто, трябва да използвате ~~Alt + клавиш~~  
**Например: S + Alt**  
или **F + Alt**



# Клавиши за бърз достъп

- ▶ Клавишите за бърз достъп се различават от клавишите за достъп, по това, че **предизвикват мигновено отваряне** на елемент от менюто
- ▶ За да се **достигне до елемент** от менюто, използвайки клавиши за достъп трябва да се **обходи съответната йерархия на менюта**
- ▶ **Клавишите за бърз достъп** се свързват към елементи от меню посредством свойството **ShortcutKeys** за конкретното меню



Modifiers:

☒ Ctrl ☐ Shift ☐ Alt

Key:

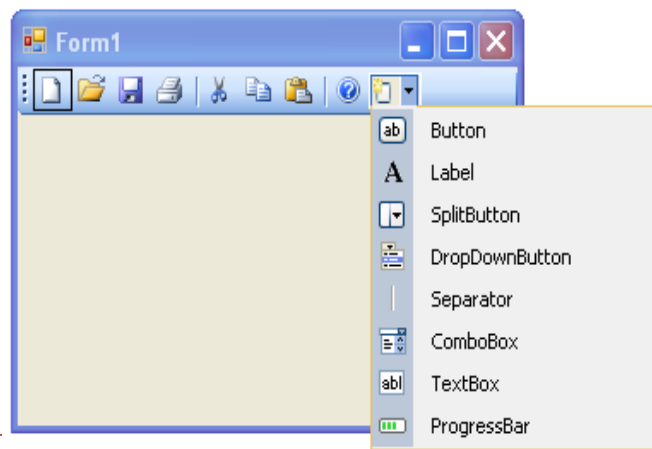
S

Reset

# СЪЗДАВАНЕ НА ЛЕНТА С ИНСТРУМЕНТИ

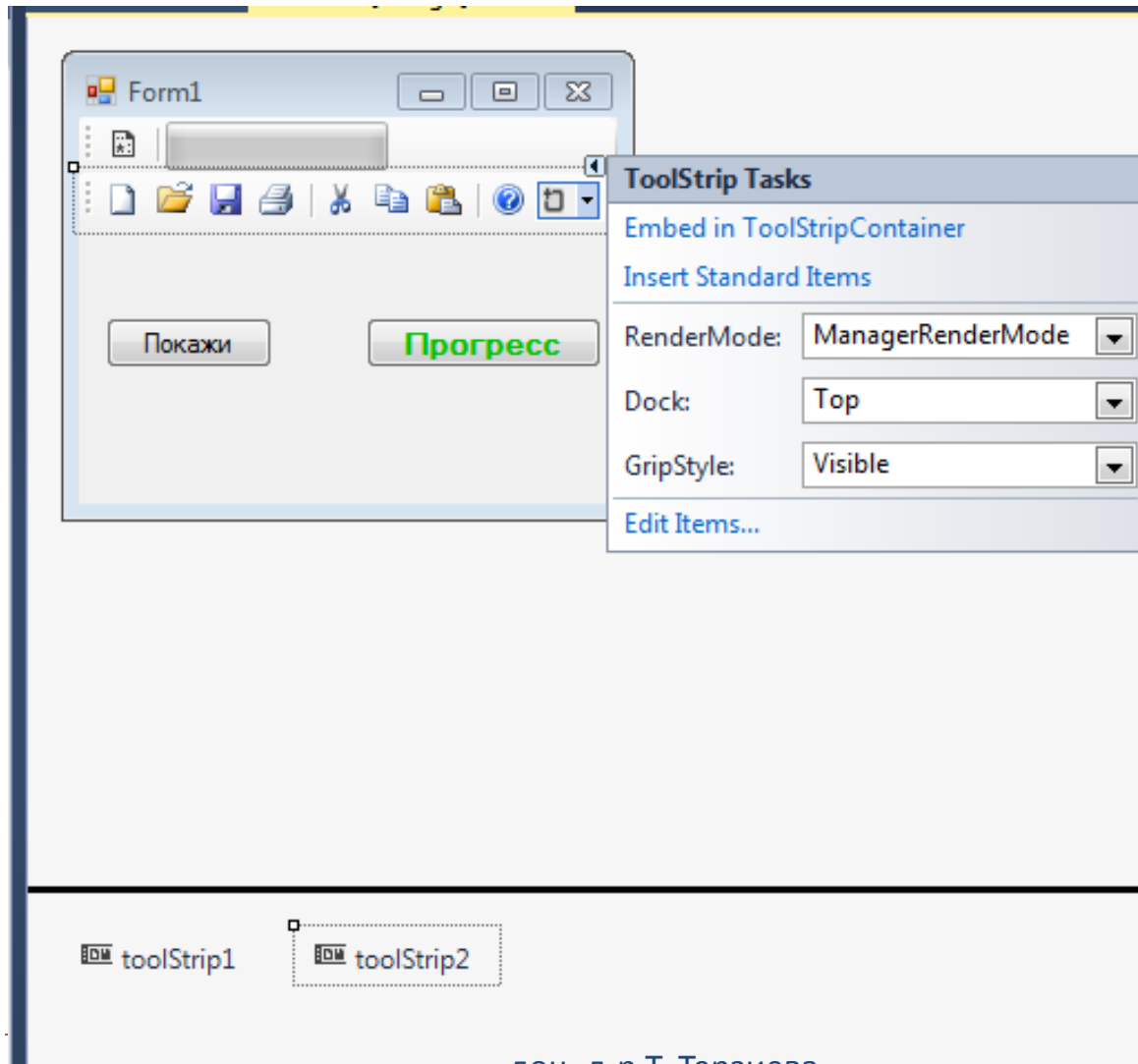
## ToolStrip Control

- ▶ Лентите с **инструменти** са често използвани при приложенията с ГПИ и могат да съдържат **различни по тип елементи**: бутони, комбинирани кутии, текстови полета, етикети и др.
- ▶ Те се използват за **визуализация на информация**, свързана със състоянието на приложението
- ▶ За реализиране на ленти с инструменти в средата Visual C# се използва компонента **ToolStrip**.

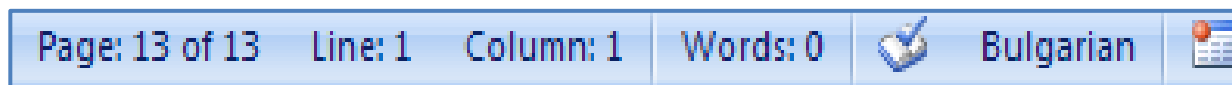




# СЪЗДАВАНЕ НА ЛЕНТА С ИНСТРУМЕНТИ

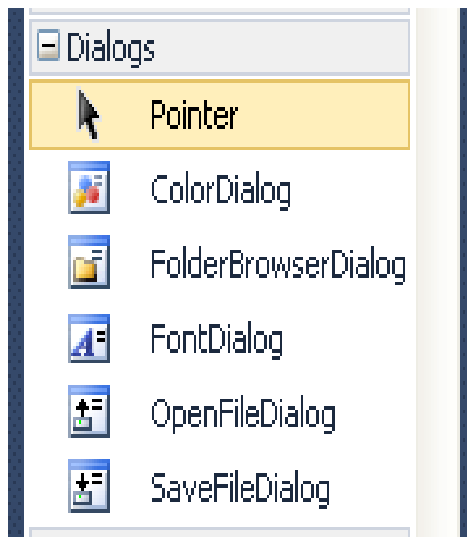


- ▶ **Лентите за състояние** са още една от типичните компоненти в приложения с ГПИ
- ▶ Обикновено те се състоят от **отделни секции** (панели), които могат да съдържат **текст** или **икони**
- ▶ За реализиране на **лента за състоянието** в средата Visual C# се използва компонента **StatusStrip**
- ▶ Те се използват за **визуализация на информация**, свързана със **състоянието на приложението**
- ▶ Например в текстовите редактори много често в **статус лентата** се показва **номерът на текущия ред и на колона**

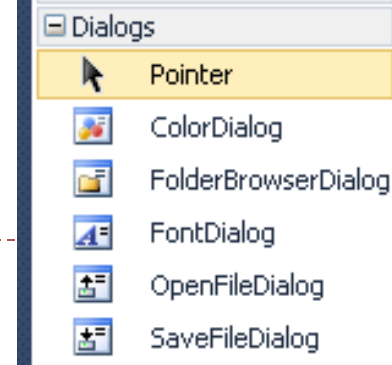


# ДИАЛОГ ЗА ИЗБОР НА ФАЙЛ

- ▶ При създаване на приложения с ГПИ често се налага да се осъществява **достъп до локалната файлова система** с цел да се реализира възможност за:
  - ▶ **отваряне, модификация и съхраняване** на даден тип файлове



# ДИАЛОГ ЗА ИЗБОР НА ФАЙЛ



- ▶ За тази цел .NET (Winforms) предоставя **стандартни диалози**, които осигуряват достъп до стандартно настроените **диалогови прозорци**, извършващи **операции по отваряне и съхранение на файлове** (прозорци Open и Save As), **избор на цвят**, или **отпечатване на документ**
- ▶ .NET предоставя **диалогови контроли**, които ви **позволяват да извикате диалог** за изпълнение на конкретна задача
- ▶ Прозорците могат да бъдат **персонализирани** с помощта на **свойствата** на тези контроли

# OpenFileDialog

- ▶ **OpenFileDialog** представлява **диалог за избор на файл при отваряне**. Този клас ни позволява да проверим **дали даден файл съществува** и да **го отворим**
- ▶ **По-важни свойства** на диалога са:
  - ▶ **Title** – задава **заглавие на диалога**
  - ▶ **InitialDirectory** – задава **началната директория**, от която започва **изборът на файл**. Ако не бъде зададена изрично, се използва последната директория, от която потребителят е избирал файл по време на работа с текущото приложение
  - ▶ **Filter** – задава възможните **файлови разширения**, между които потребителят може да избира (например \*.txt, \*.doc, ...)
  - ▶ **FilterIndex** – задава **активния филтър**
  - ▶ **MultiSelect** – указва **дали** в диалога могат да бъдат **избирани много файлове едновременно или само един**
  - ▶ **FileName, FileNames** – съдържа **избраните файлове**

# OpenFileDialog

---

```
openFileDialog1.FileName = string.Empty;
// показване на диалог за Отваряне на файл
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    fn = openFileDialog1.FileName;
    // показване на името на файла в заглавието на прозореца
    this.Text = fn;
    try
    {
        // прочитане на данни от файл
        StreamReader sr = new StreamReader(fn);
        textBox1.Text = sr.ReadToEnd();
        sr.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show("Грешка при четене на файл. \n" +
                        exc.ToString(), "Четене на файл",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

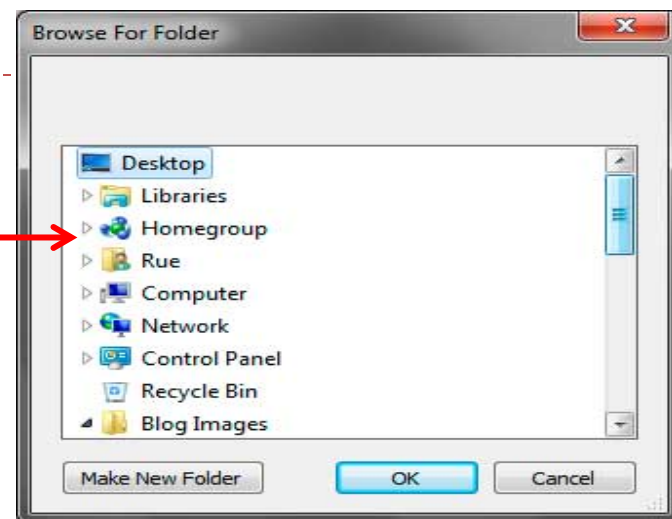
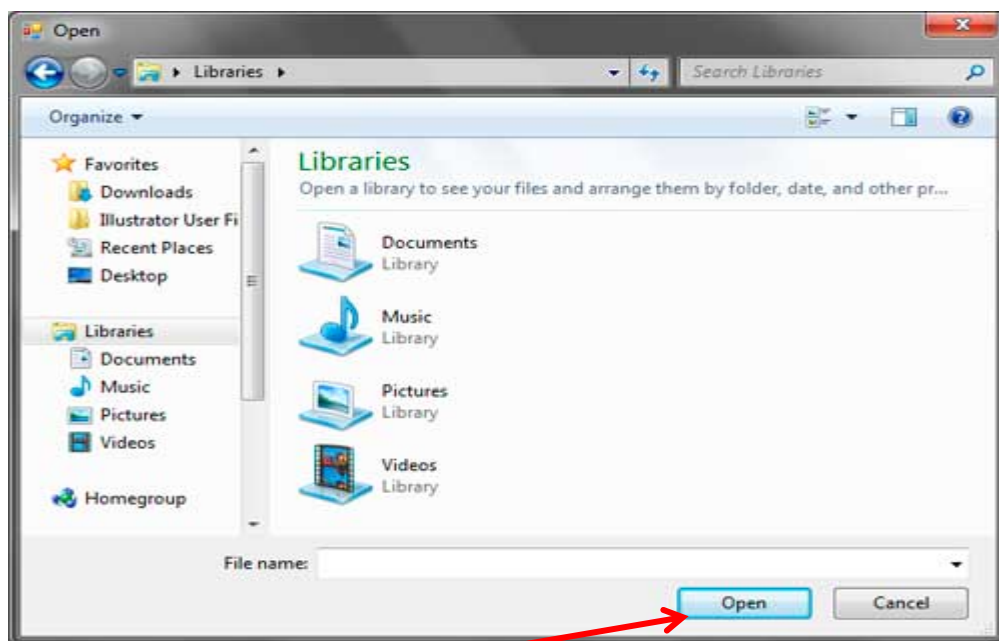
# OpenFileDialog

---

```
private void mnuOpen_Click(object sender, EventArgs e)
{
    string textfile = "";
    openFileDialog1.InitialDirectory = "C";
    openFileDialog1.Title = "Отваряне на текстов файл!";
    openFileDialog1.FileName = "";
    openFileDialog1.Filter = "Text files|*.txt|Word Documents|*.doc|RTF files|*.rtf";
    if(openFileDialog1.ShowDialog() == DialogResult.OK &&
        openFileDialog1.FileName.Length>0)
    {
        textfile = openFileDialog1.FileName;
        if (openFileDialog1.FilterIndex == 1 || openFileDialog1.FilterIndex==2)
            richTextBox1.LoadFile(textfile, RichTextBoxStreamType.PlainText);
        else if (openFileDialog1.FilterIndex == 3)
            richTextBox1.LoadFile(textfile, RichTextBoxStreamType.RichText);
    }
}
```

# РАЗЛИЧНИ ДИАЛОЗИ

- ▶ Диалог за избор на **директория**  
**Browsing Folders**

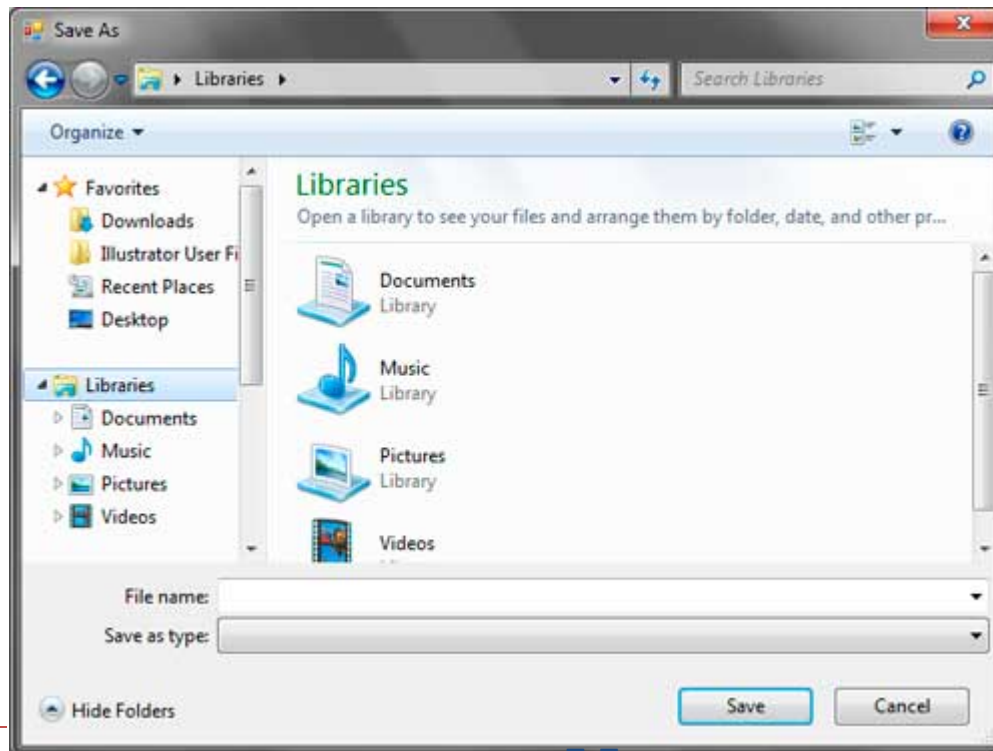


- ▶ Диалог за **отваряне на файл**



# SaveFileDialog

- ▶ **SaveFileDialog** представлява **диалог за избор на файл при записване**
- ▶ Този клас ни позволява да **презапишем съществуващ** или да **създадем нов файл**



# SaveFileDialog

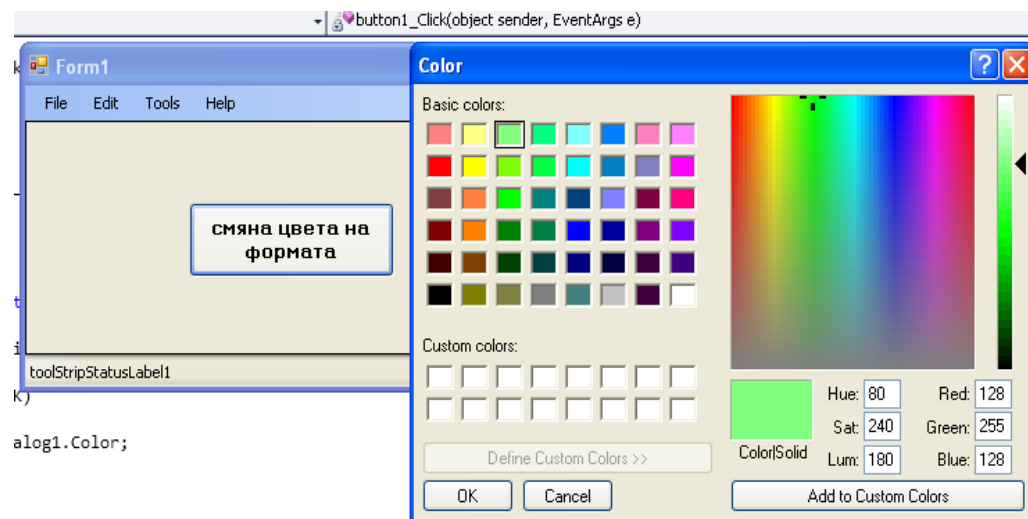
---

```
private void SaveButton_Click(object sender, EventArgs e)
{
    // SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    saveFileDialog1.InitialDirectory = @"C:\";
    saveFileDialog1.Title = "Save text Files";
    saveFileDialog1.DefaultExt = "txt";
    saveFileDialog1.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
    saveFileDialog1.FilterIndex = 2;
    saveFileDialog1.RestoreDirectory = true;

    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = saveFileDialog1.FileName;
    }
}
```

# Диалози за избор на шрифт и цвят

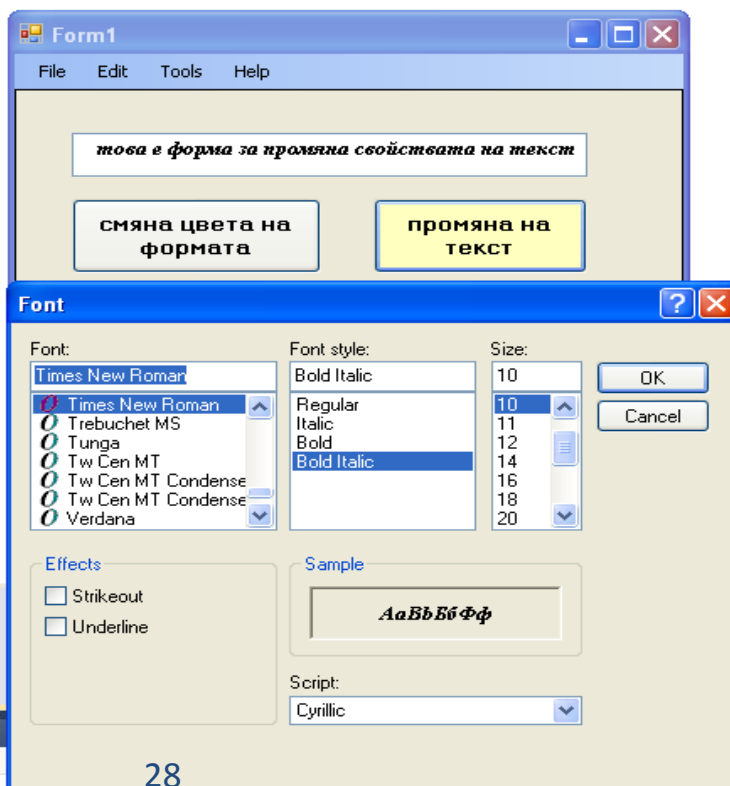
- ▶ Компонентът **ColorDialog** **представлява диалог за избор на цвят** чрез визуализиране на стандартния диалогов прозорец **Color** за избор на цвят
- ▶ Поставяме контрола **button** във формата и добавяме събитие **Click**



```
private void button1_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
    {
        this.BackColor = colorDialog1.Color;
    }
}
```

# Диалози за избор на шрифт и цвят

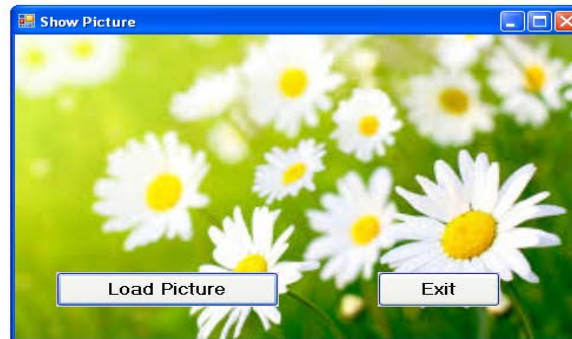
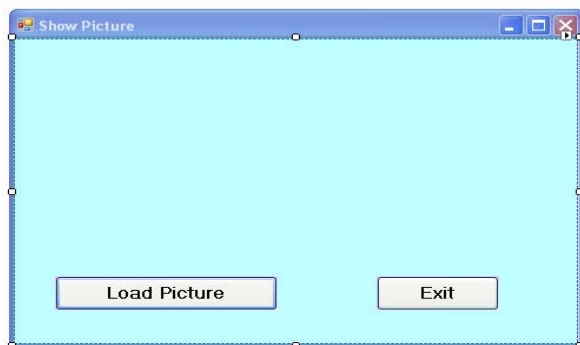
- ▶ Контрола **FontDialog** се използва за **избор на различни видове шрифт и свойства** на шрифтовете
- ▶ Чрез **FontDialog**, може да промените **вида на шрифта, стила, размера, цвета, да добавяте ефекти** и да **определи набора от символи за шрифта**



```
private void button1_Click(object sender, EventArgs e)
{
    DialogResult result = fontDialog1.ShowDialog();
    if (result == DialogResult.OK)
    {
        textBox1.Font = fontDialog1.Font;
    }
}
```

# Пример за използване на файлова диалогова кутия

- Създайте приложение с ГПИ, което съдържа два бутона, поле за зареждане на картина (PictureBox) и компонент за файлова диалогова кутия за отваряне на файл, както е показано на фигурата.



Контрола	Име на елемент	Свойства
Form	Form1	Text="Зареждане на картинка"
PictureBox	pictureBox1	BorderStyle=FixedSingle, SizeMode=StretchImage
Button	button1	Font=Times New Roman, Bold, 12, BackColor – по избор, Text="Load Picture"
Button	buttonExit	Font=Times New Roman, Bold, 12, BackColor – по избор, Text="Exit"
OpenFileDialog	openFileDialog1	Name= openFileDialog1, FileName= openFileDialog1

# Добавяне на програмен код - примерен първи вариант

---

```
private void button1_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
        pictureBox1.Image = new Bitmap(openFileDialog1.FileName);
}
private void buttonExit_Click(object sender, EventArgs e)
{
    Form1.ActiveForm.Close();
}
```

# Добавяне на програмен код

## втори вариант

```
private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "jpg files (*.jpg)|*.jpg|all files|*.*";
    dlg.Title = "Open Picture...";
    if (dlg.ShowDialog() == DialogResult.OK)
        pictureBox1.Image = new Bitmap(dlg.FileName);
    dlg.Dispose();
}

private void buttonExit_Click(object sender, EventArgs e)
{
    Form1.ActiveForm.Close();
}
```

# Пример с файлов диалог

## OpenFileDialog Control

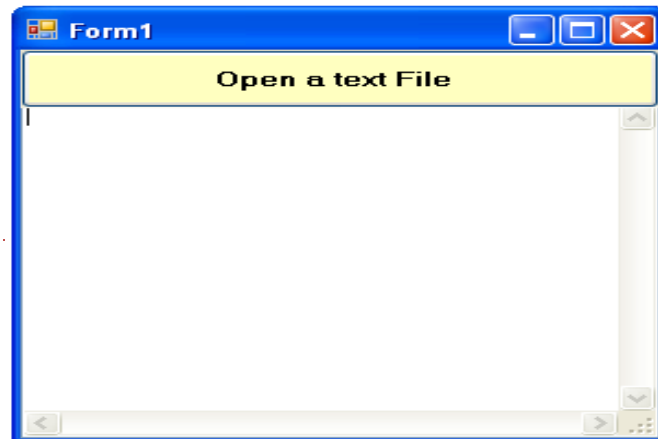
---

- ▶ Ще създадем примерно приложение, което използва основните възможности на контрола **OpenFileDialog**
- ▶ Приложението ще позволи на потребителя да **открие** (browse) **текстов файл**, както и да **преглежда** съдържанието му с помощта на **многоредово текстово поле**
- ▶ Трябва да **включим пространството** от имена **System.IO** в нашия код  
`using System.IO;`



# Пример с файлов диалог

- ▶ Създаване на **формуляр**, подобен на показаният на фигурата:
- ▶ Поставете **текстово поле (textBox)** с **Name = textBox1**. Използвайте многоредово текстово поле чрез създаване на свойство **Multiline = true**
- ▶ Настройте плъзгачите **ScrollBars = Both** на текстовото поле, както и **WordWrap = false**, **Dock = Bottom**
- ▶ Добавете бутон и задайте свойство **Name = buttonOpenFile**, **Text = “Open a Text File”**, **Dock = Top**
- ▶ Плъзнете контрол **OpenFileDialog** от кутията с инструменти към формуляра
- ▶ Кликнете два пъти върху бутона, за да създадете **събитие Click** и добавете **програмен код** към него



# Добавяне на програмен код

```
private void buttonOpenFile_Click(object sender, EventArgs e)
{
    // Задаване на филтър само за текстови файлове text files
    openFileDialog1.Filter = "Text Files|*.txt" + "Log files (*.log)|*.log|" + " Doc Files (*.docx)|*.docx";
    // No initial file selected
    openFileDialog1.FileName = String.Empty;
    // Отваряне на file dialog и съхраняване на върнатата стойност
    DialogResult result = openFileDialog1.ShowDialog();
    // Ако се избере Button OK за отваряне на файл
    if (result == DialogResult.OK)
    {
        // Създаване на поток за отваряне на файл
        Stream fs = openFileDialog1.OpenFile();
        // Създаване на инстанция reader на поток за четене от файл
        StreamReader reader = new StreamReader(fs);
        // прочитане на цялото съдържание
        textBox1.Text = reader.ReadToEnd();
        // Close the reader and the stream
        reader.Close();
    }
}
```

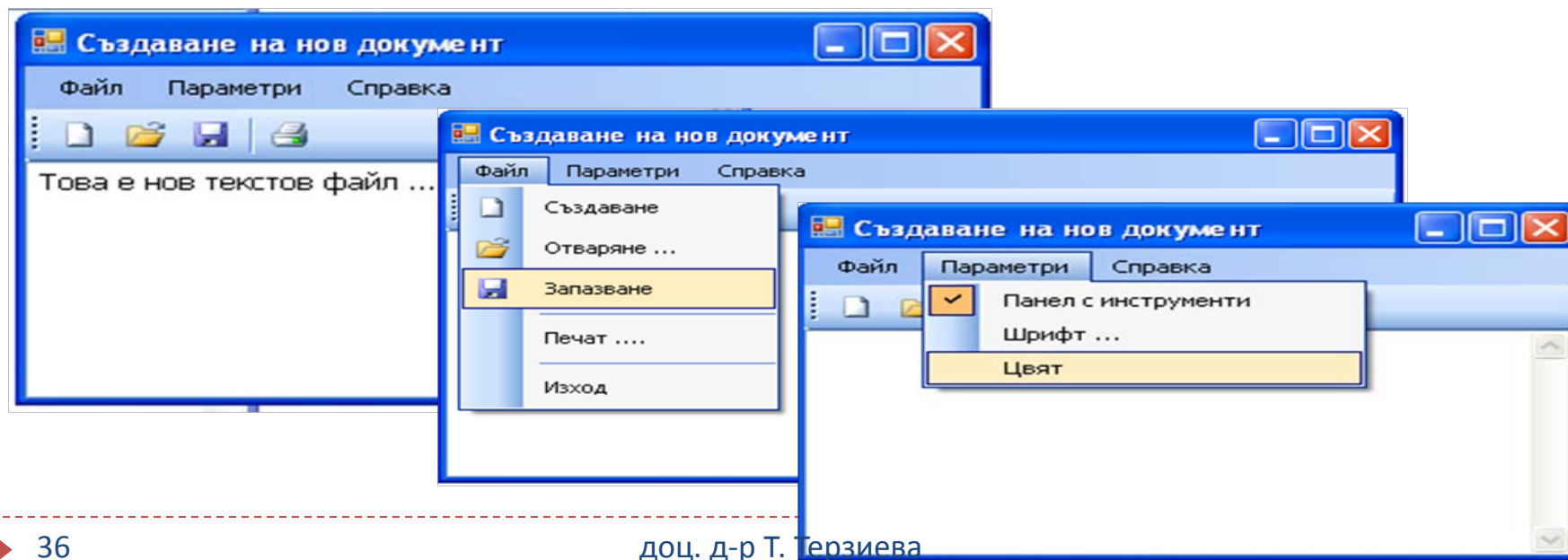
## Втори вариант с обработка на изключения

```
private void button1_Click(object sender, EventArgs e)
{
    Stream streamOpen = null;
    OpenFileDialog fileOpen = new OpenFileDialog();
    fileOpen.Title = "Отваряне на текстов файл";
    fileOpen.InitialDirectory = "C:/";
    fileOpen.Filter = "TXT Files|*.txt";
    if (fileOpen.ShowDialog() == DialogResult.OK)
    {
        label1.Text = fileOpen.FileName.ToString();
        try
        {
            if ((streamOpen = fileOpen.OpenFile()) != null)
            {
                using (StreamReader reader = new StreamReader(streamOpen))
                {
                    // Insert code to read the stream here.
                    textBox1.Text = reader.ReadToEnd();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Грешка: Не можете да отворите файла за четене:" + ex.Message);
        }
    }
}
```

# Пример: Създаване на прост текстов редактор **mEdit**



- ▶ Създайте приложение „Текстов редактор“, което съдържа следните елементи на ГПИ: **MenuStrip**, **ContextMenuStrip**, **ToolStrip**, **StatusStrip**, **RichTextBox**, **OpenFileDialog**, **SaveFileDialog**, **FontDialog**, **ColorDialog**, както е показано на фигурата:



# Създаване на прост текстов редактор

1. Да се създаде **система от менюта** за избор и изпълнение на основните действия, свързани със **създаване, преглеждане, редактиране, форматиране, съхраняване и отпечатване** на текстови файлове.
2. Да се създаде **лента с инструменти за бърз достъп до често използвани действия**, свързани със създаване, преглеждане, редактиране, форматиране и съхраняване на текстови файлове. За всеки от бутоните да се зададе съответен подсказващ текст.
3. Да се създаде **контекстно меню**, включващо командите **Cut, Copy и Paste** и улесняващо работата по **редактиране на текст**. Това меню да се визуализира при щракване с десен бутон върху елемента **RichTextBox**.
4. Да се създаде **лента за състояние на приложението**, която съдържа етикет, показващ името на текущия файл, с който работи приложението.

**Благодаря за вниманието!**

