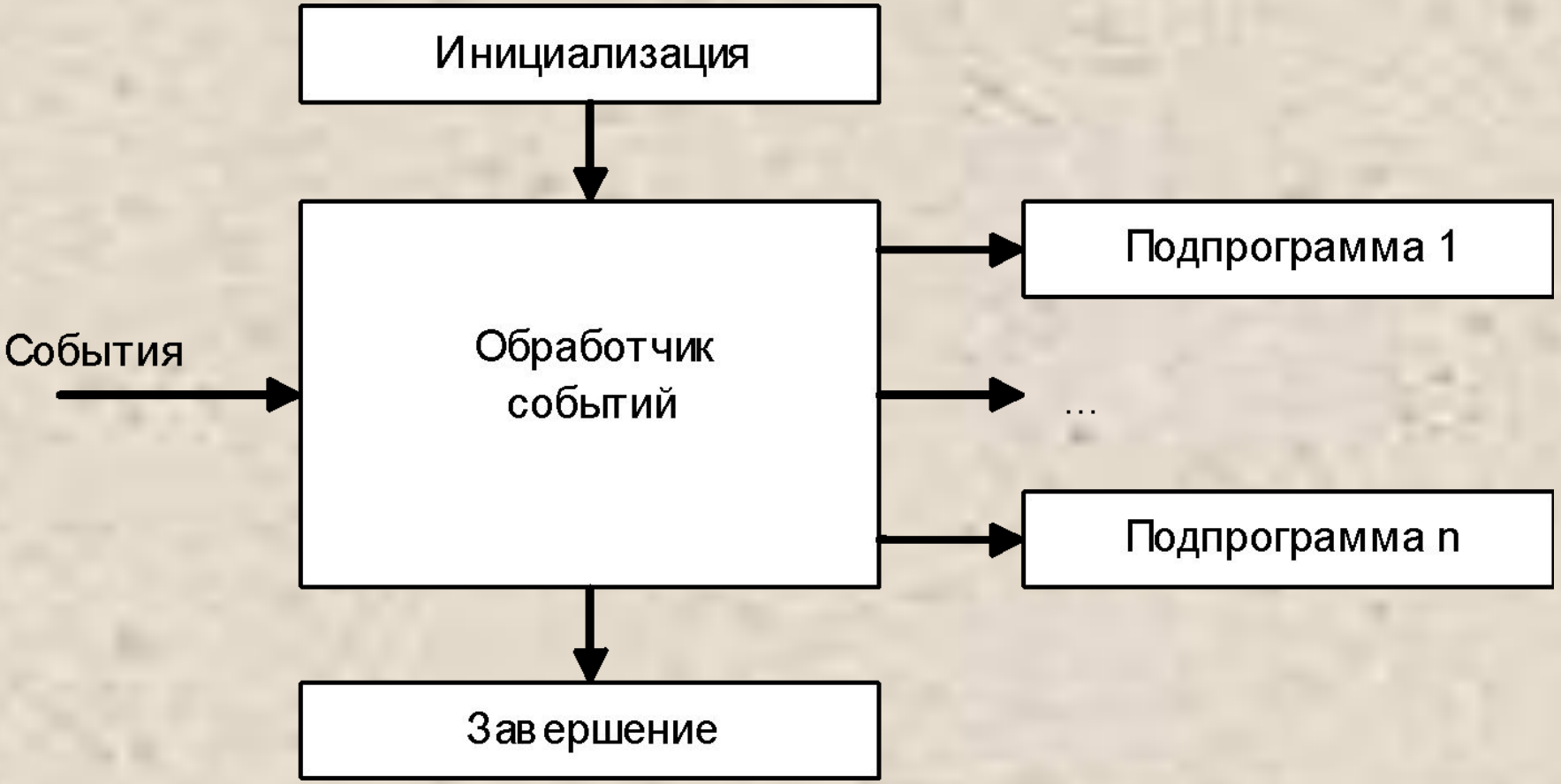
Основы программирования для многозадачных операционных систем

Основные особенности Windows

- Многозадачность
- Независимость программ от аппаратуры
- Стандартный графический интерфейс с пользователем
- Поддержка виртуального адресного пространства для каждого приложения.
- Возможность обмена данными между приложениями
- Возможность запуска старых программ

Событийно-управляемое программирование

- Структура программы, управляемой событиями:



Структура Windows-приложения

Приложение: консольное или Windows?

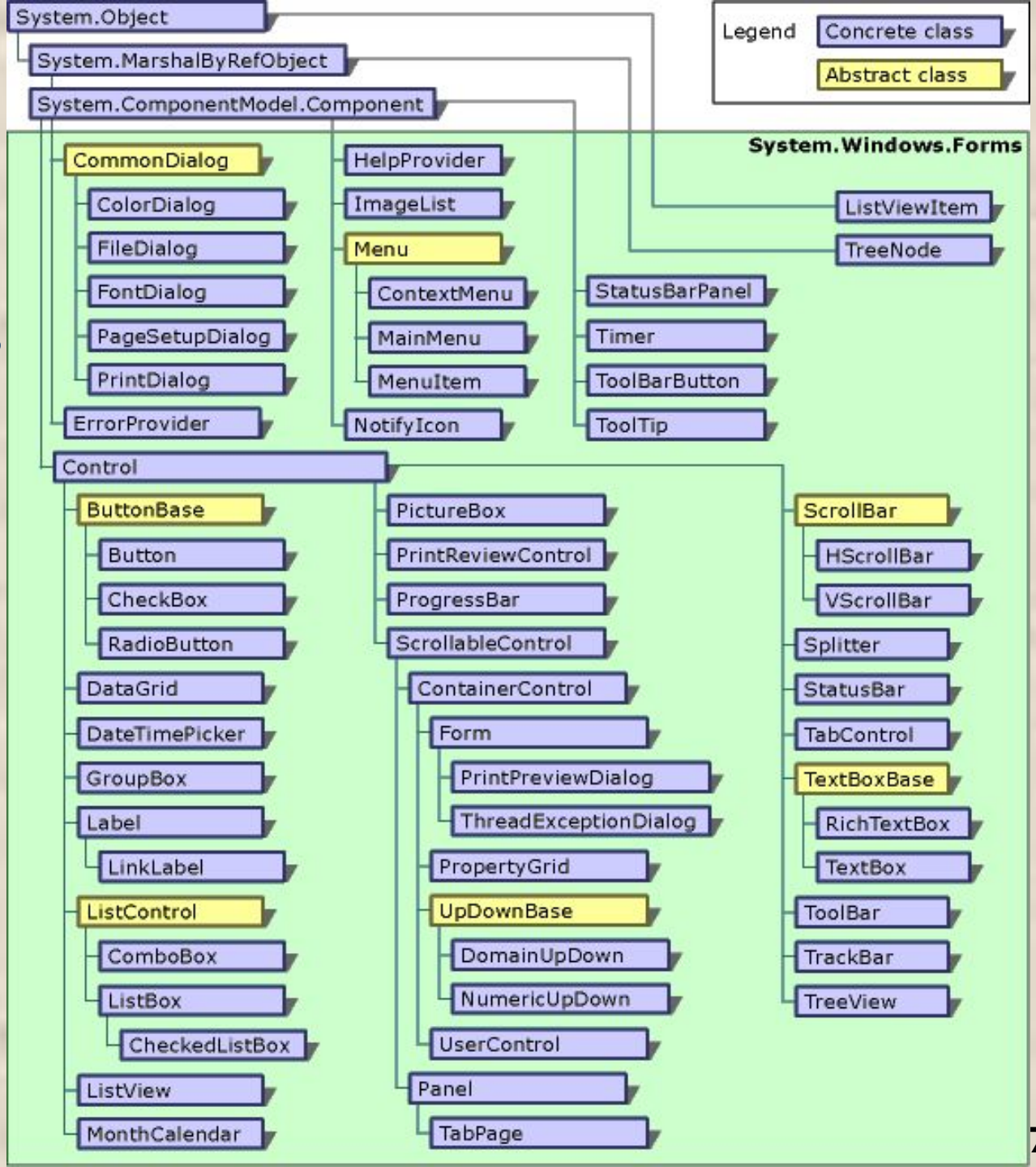
```
class A
{
    public static void Main()
    {
        System.Windows.Forms.MessageBox.Show("Hello!");
    }
}
```

```
// Необходимо добавить ссылку на System.Windows.Forms
// (Add Reference…)
```

Помощь .NET при создании приложений

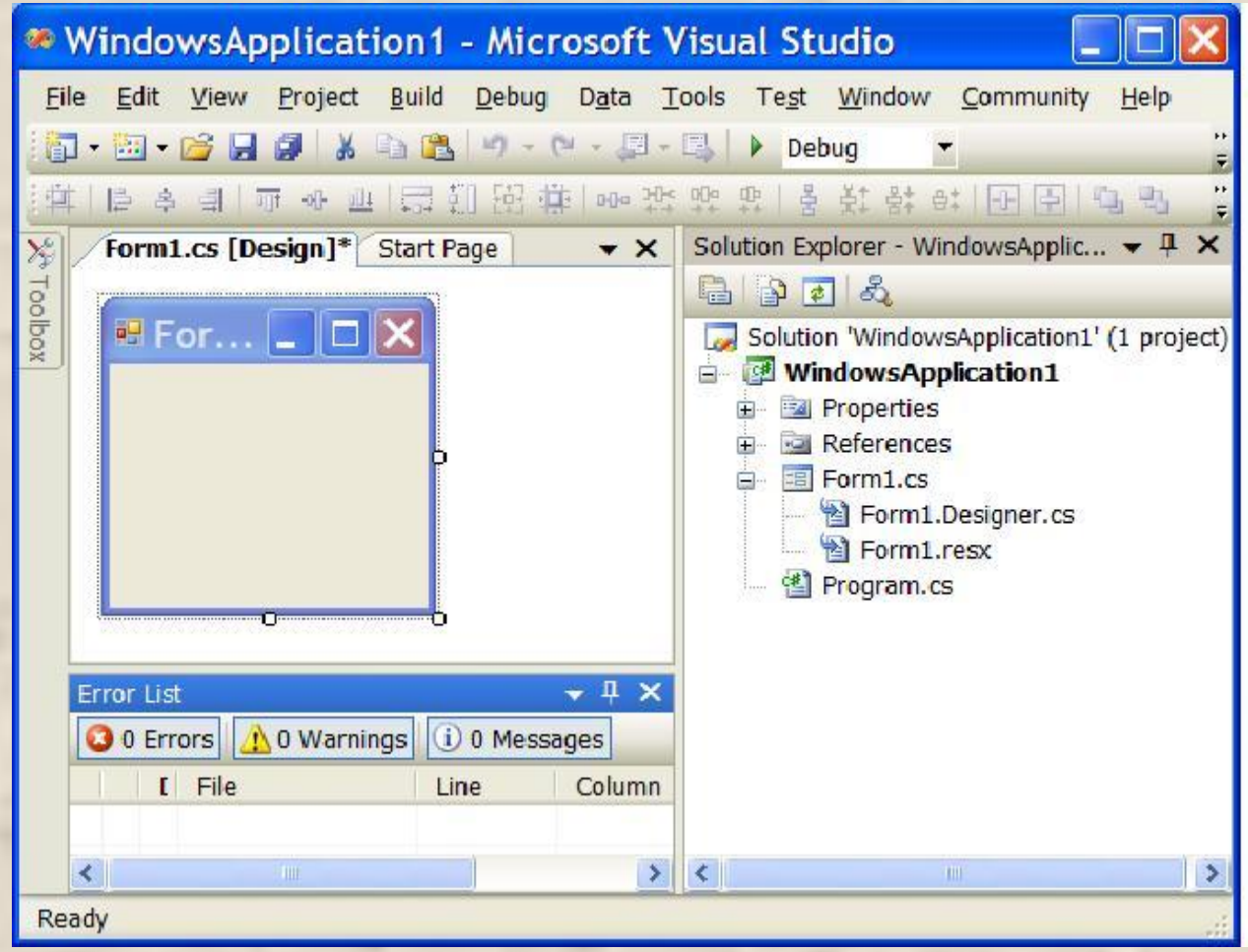
- **Среда** Visual Studio.NET содержит удобные средства разработки Windows-приложений, например:
  - создание шаблонов приложения и форм;
  - создание заготовок обработчиков событий.
- **Библиотека** классов .NET включает пространство имен System.Windows.Forms, содержащее огромное количество типов - строительных блоков Windows-приложений, например:
  - Application
  - Button, CheckBox, DataGrid, GroupBox, ListBox, PictureBox
  - Form
  - ColorDialog, FileDialog, FontDialog
  - Menu, MainMenu, MenuItem
  - Clipboard, Help, Timer, Screen, Cursors
  - StatusBar, ToolBar, ScrollBar

Элементы управления Windows.Forms



Создание Windows-приложения

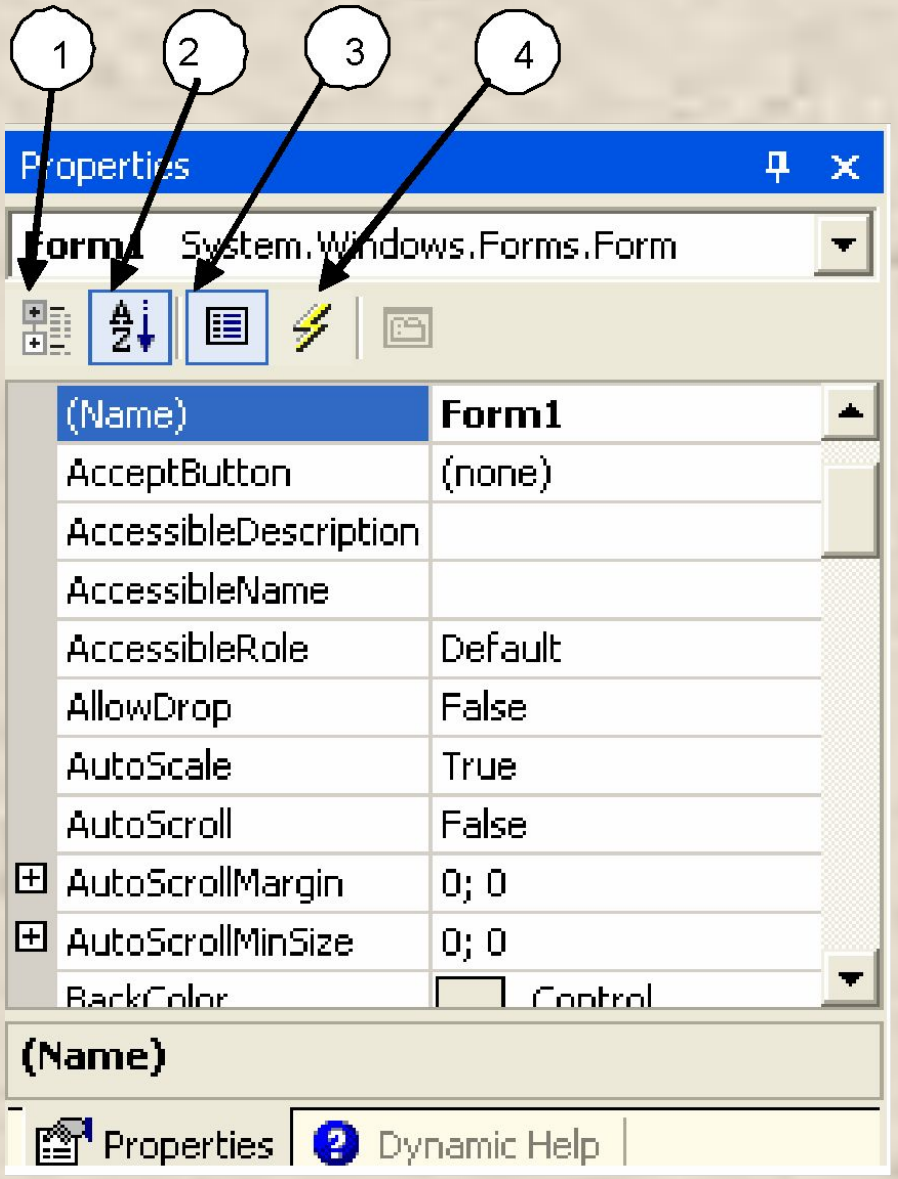
- новый проект (File□New□Project)
- шаблон Windows Application



Создание Windows-приложения

- Процесс создания Windows-приложения состоит из двух основных этапов:
  - *визуальное проектирование*, то есть задание внешнего облика приложения
  - *определение поведения* приложения путем написания процедур обработки событий.
- *Визуальное проектирование* заключается в помещении на форму компонентов (элементов управления) и задании их свойств и свойств самой формы.

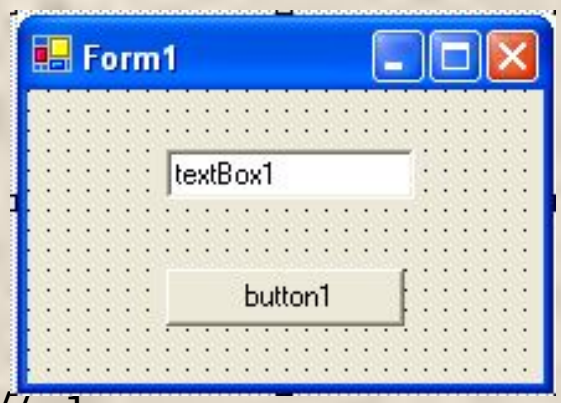
Окно свойств



- View ☐Properties Window

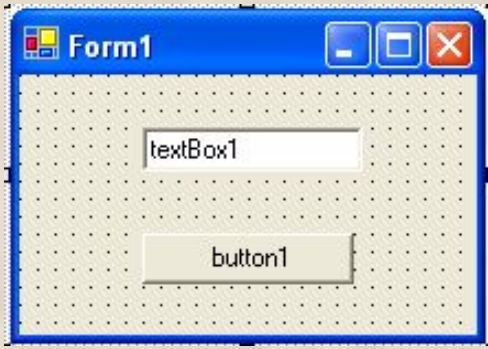
Задание свойств выполняется либо выбором имеющихся в списке вариантов, либо вводом требуемого значения с клавиатуры.
Если около имени свойства стоит значок +, это означает, что свойство содержит другие свойства. Они становятся доступными после щелчка на значке.

Шаблон Windows-приложения




```
using System;
using System.Windows.Forms;
namespace WindowsApplication1
{    public class Form1 : Form {
    private System.Windows.Forms.TextBox textBox1;  //  1
    private System.Windows.Forms.Button   button1;   //  2
    private System.ComponentModel.Container components = null;
    public Form1()  {        InitializeComponent();        }
    protected override void Dispose( bool disposing )        { … }
#region Windows Form Designer generated code
    private void InitializeComponent()
      {  this.components = new System.ComponentModel.Container();
         this.Size = new System.Drawing.Size(300,300);
         this.Text = "Form1"; …
      }
 #endregion
      static void Main()   {    Application.Run(new Form1());    }   }}
```

Пример вставки в InitializeComponent




```
this.textBox1 = new System.Windows.Forms.TextBox();   //  3
this.button1  = new System.Windows.Forms.Button();    //  4
this.SuspendLayout();                                 //  5
// textBox1
   this.textBox1.Location = new System.Drawing.Point(24, 16);
   this.textBox1.Name = "textBox1";
   this.textBox1.Size = new System.Drawing.Size(240, 20);
   this.textBox1.TabIndex = 0;
   this.textBox1.Text = "textBox1";
   this.textBox1.KeyPress += new                      //  6
   System.Windows.Forms.KeyPressEventHandler(this.textBox1_KeyPress);
// button1
    this.button1.Location = new System.Drawing.Point(192, 80);
    this.button1.Name = "button1";
    this.button1.TabIndex = 1;
    this.button1.Text = "button1";
    this.button1.Click += new                         //  7
    System.EventHandler(this.button1_Click);
```

```
// Form1
      this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
       this.ClientSize = new System.Drawing.Size(292, 126);
       this.Controls.Add(this.button1);                    //  8
       this.Controls.Add(this.textBox1);                   //  9
       this.Name = "Form1";
       this.Text = "Form1";
       this.ResumeLayout(false);                          // 10
  }
  #endregion
…
private void button1_Click(object sender, System.EventArgs e)   {      // 11
       }
private void textBox1_KeyPress(object sender,
                          System.Windows.Forms.KeyPressEventArgs e) {   // 12
       }
```

Размещение компонента на форме

- *Создать экземпляр* соответствующего класса
  - (код создается автоматически при размещении компонента на заготовке формы)
- *Настроить свойства экземпляра*, в том числе зарегистрировать обработчик событий
  - (через окно свойств)
- *Поместить экземпляр в коллекцию* компонентов формы
  - (автоматически)

Определение поведения программы

- Определение поведения программы начинается с принятия решений, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе пунктов меню и т. д. (по каким событиям будут выполняться действия, реализующие функциональность программы).
- Заготовка шаблона обработчика события формируется двойным щелчком на поле, расположенном справа от имени соответствующего события на вкладке Events окна свойств, при этом появляется вкладка окна редактора кода с заготовкой соответствующего обработчика.
- Для каждого класса определен свой набор событий, на которые он может реагировать.

Часто используемые события

- Activated — получение формой фокуса ввода;
- Click, DoubleClick — одинарный и двойной щелчки мышью;
- Closed — закрытие формы;
- Load — загрузка формы;
- KeyDown, KeyUp — нажатие и отпускание любой клавиши и их сочетаний;
- KeyPress — нажатие клавиши, имеющей ASCII-код;
- MouseDown, MouseUp — нажатие и отпускание кнопки мыши;
- MouseMove — перемещение мыши;
- Paint — возникает при необходимости прорисовки формы.

Примеры обработчиков событий

```
private void Form1_Load(object sender, EventArgs e)
{
        rnd = new Random();
        i = rnd.Next(max);
}
private void Exit_Click(object sender, EventArgs e)
{                                          // имя пункта меню - Exit
   Close();
   // или:
   // Application.Exit();
}
```

Класс Control

- Класс Control является базовым для всех отображаемых элементов и реализует их базовую функциональность.
- Он содержит методы обработки ввода пользователя с помощью мыши и клавиатуры, определяет размер, положение, цвет фона и другие характеристики элемента.
- Основные элементы управления (потомки Control):
  - метка Label
  - Кнопка Button
  - Поле ввода TextBox
  - Меню MainMenu и ContextMenu
  - Флажок CheckBox
  - Переключатель RadioButton
  - Панель GroupBox
  - Список ListBox

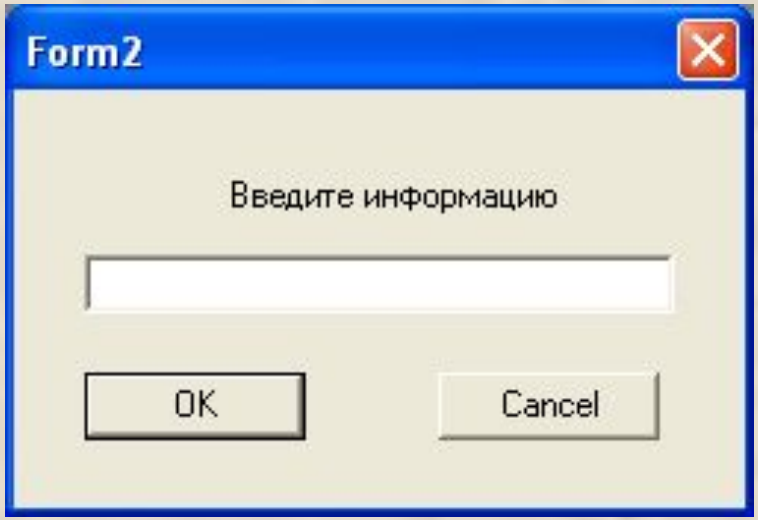
Виды окон

- ***Модальное окно*** не позволяет пользователю переключаться на другие окна того же приложения, пока не будет завершена работа с текущим окном.
- В виде модальных обычно оформляют ***диалоговые окна***, требующие от пользователя ввода какой-либо информации.
- ***Немодальное окно*** позволяет переключаться на другие окна того же приложения. Немодальные окна являются, как правило, информационными. Они используются в тех случаях, когда пользователю желательно предоставить свободу выбора — оставлять на экране какую-либо информацию или нет.
- Каждое приложение содержит одно ***главное окно***. Класс главного окна приложения содержит точку входа в приложение (статический метод Main). При закрытии главного окна приложение завершается.
- Вид окна определяет его функциональность, например, окно с одинарной рамкой не может изменять свои размеры.

Диалоговые окна

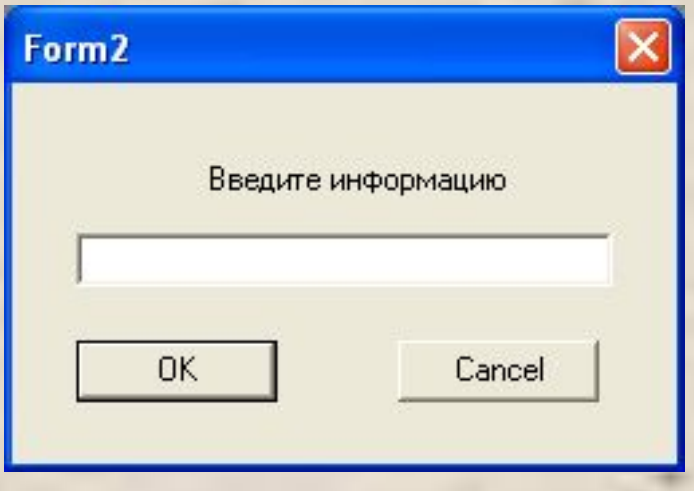
- Диалоговое окно характеризуется:
  - неизменяемыми размерами (FormBorderStyle = FixedDialog);
  - отсутствием кнопок восстановления и свертывания в правом верхнем углу заголовка формы (MaximizeBox = False, MinimizedBox = False);
  - наличием кнопок наподобие OK, подтверждающей введенную информацию, и Cancel, отменяющей ввод пользователя, при нажатии которых окно закрывается (AcceptButton = имя_кнопки_OK, CancelButton = имя_кнопки_Cancel);
  - установленным значением свойства DialogResult для кнопок, при нажатии которых окно закрывается.
- Для отображения диалогового окна следует создать экземпляр объекта соответствующей формы, а затем вызвать для этого объекта метод ShowDialog.

Пример отображения диалогового окна




```
private void menuItem1_Click( object sender, EventArgs e )
{  Form2 f = new Form2();          // создание экземпляра класса окна
   if ( f.ShowDialog() == DialogResult.OK )        // отображение окна
       label1.Text = f.Info;
}
private void menuItem2_Click( object sender, EventArgs e )
{    Close();                                  // закрытие главного окна
}
```

Пример диалогового окна



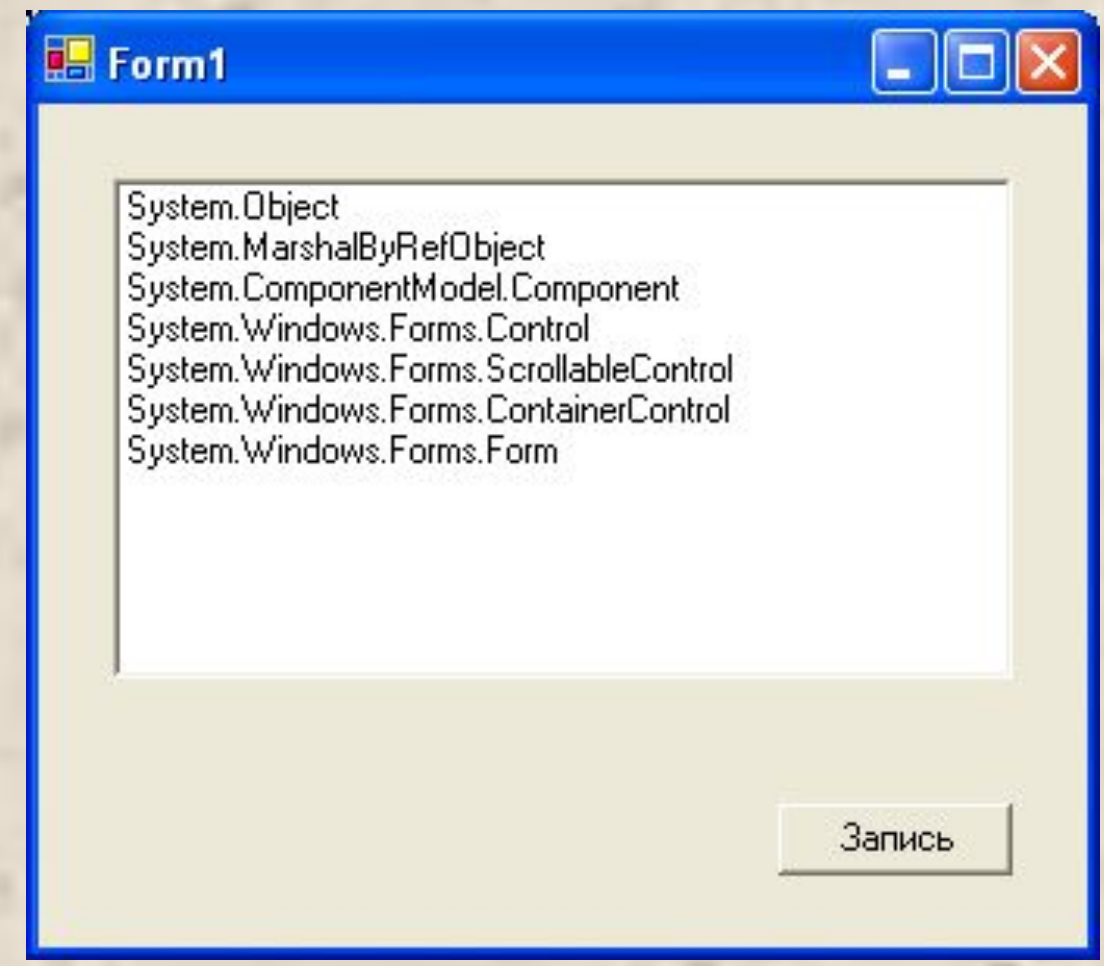

```
public class Form2 : Form
{
    private Label     label1;
    private TextBox   textBox1;
    private Button    btnOK;
    private Button    btnCancel;
    private Container components = null;
    public string Info   // свойство для передачи информации из окна
    {
        get
        {
            return textBox1.Text;
        }
    }
    ...
}
```

Список ListBox

- Список служит для представления перечней элементов, в которых пользователь может выбрать одно или несколько значений
- Чаще всего используются списки строк, но можно выводить и произвольные изображения. Список может состоять из нескольких столбцов и быть отсортированным в алфавитном порядке
- Элементы списка нумеруются с нуля. Они хранятся в свойстве Items, представляющем собой коллекцию.
  - В Items можно добавлять элементы с помощью методов Add, AddRange и Insert.
  - Для удаления элементов служат методы Remove и RemoveAt

Пример использования списка

Приложение отображает в списке типа ListBox строки, считанные из входного файла, а затем по щелчку на кнопке Запись выводит выделенные пользователем строки в выходной файл.



Фрагменты приложения

```
public class Form1 : Form
   {  private ListBox   listBox1;
      private Button    button1;
...
private void Form1_Load(object sender, EventArgs e)
      {  try
         {  StreamReader f = new StreamReader( "input.txt" );
            string buf;
            while ( ( buf = f.ReadLine() ) != null )  // чтение из файла
               listBox1.Items.Add(buf);          // занесение в список
         }
         catch ( FileNotFoundException exc )
               {  MessageBox.Show( exc.Message );  return; }
    catch {  MessageBox.Show( "Неопознанное искл-е" );  return; }
      }
```

```
private void button1_Click(object sender, EventArgs e)
        {
            StreamWriter f = new StreamWriter( "output.txt" );

            foreach ( string item in listBox1.SelectedItems )
                f.WriteLine(item);                          // запись в файл
            f.Close();
        }
```

Введение в графику

- Для вывода линий, геометрических фигур, текста и изображений необходимо создать экземпляр класса Graphics, описанного в пространстве имен System.Drawing. Существуют различные способы создания этого объекта.
- *Первый способ* состоит в том, что ссылку на объект Graphics получают из параметра PaintEventArgs, передаваемого в обработчик события Paint, возникающего при необходимости прорисовки формы или элемента управления:

```
private void Form1_Paint( object sender, PaintEventArgs e )
{   Graphics g = e.Graphics;
    // использование объекта
}
```

- *Второй способ* — использование метода CreateGraphics, описанного в классах формы и элемента управления:

```
Graphics g;
g = this.CreateGraphics();
```

- *Третий способ* — создание объекта с помощью объекта-потомка Image. Этот способ используется для изменения существующего изображения:

```
Bitmap bm = new Bitmap( "d:\\picture.bmp" );
Graphics g = Graphics.FromImage( bm );
```

- После создания объекта типа Graphics его можно применять для вывода линий, геометрических фигур, текста и изображений. Основными объектами, которые при этом используются, являются объекты классов Pen (рисование линий и контуров геометрических фигур), Brush (заполнение областей), Font (вывод текста) и Color (цвет).

Пример

```
using System;
using System.Drawing;
using System.Windows.Forms;
namespace WindowsApplication1 {
   public partial class Form1 : Form    {
      public Form1() { InitializeComponent(); }
      private void Form1_Paint( object sender, PaintEventArgs e )
         using ( Graphics g = e.Graphics) {                    //
               using ( Pen pen = new Pen( Color.Red ) ) {      // 2
                g.DrawLine( pen, 0, 0, 200, 100 );
                g.DrawEllipse( pen, new Rectangle(50, 50, 100, 150) );  }
                string s = "Sample Text";
                Font font = new Font( "Arial", 18 );                 // 3
                SolidBrush brush = new SolidBrush( Color.Black );   // 4
                float x = 100.0F;
                float y = 20.0F;
                g.DrawString( s, font, brush, x, y );
                font.Dispose();                                     // 5
                brush.Dispose();                                    // 6
}}}}
```