Обзор языка программирования C#

(продолжение)

Гутников С.Е.

События

События позволяют классу или объекту уведомлять другие классы или объекты о возникновении каких-либо ситуаций. Класс, отправляющий (или вызывающий) событие, называется **`издателем`**, а классы, принимающие (или обрабатывающие) событие, называются **подписчиками**.

Используйте ключевое слово `event` для объявления события в классе издателя.

Рассмотрим пример:

События

```
using System;
using System.IO;

namespace ExampleEvent
{
    // Делегат для событий:
    public delegate void TEventFirst(
        string msg);
    public delegate bool TEventSecond(
        string msg, int value);
        // возвращает истину если обработано
```

События

```
// простой издатель
public class Sender
{
    // пеменные события для
    // регистрации подписчиков:
    public event TEventFirst evFst;
    public event TEventSecond evSec;
```

События

```
// В этом методе рассылаем события:
public void DoCalls(string msg,
                    int value)
{
    Console.WriteLine(
"*** Send events for ({0},{1}):",
      msg, value);
    if (evFst != null)
    // проверяем есть ли подписчики:
    {
        evFst(msg);
    }
```

События

```
                if (evSec != null)
                // проверяем есть ли подписчики:
                {
                    if (evSec(msg, value)
                            == false)
                    {
                        Console.WriteLine(
 "*** Event ({0},{1}) was not processed!",
                        msg, value);
                    }
                }
}     }
```

События

```
// подписчик 1:
public class ReceiverOne
{// методы обработчики:
    public void OnTEventFirst(
        string msg)
    {
        Console.Write(
        "[1]: got event: ({0})\n", msg);
    }
    public bool OnTEventSecond(
         string msg, int value)
    {
```

События

```
            Console.Write(
            "[1]: got event: ({0},{1})\n",
            msg, value);
            return (value < 0)
                ? false : true;
}} // подписчик 2:
public class ReceiverTwo
{// методы обработчики:
    public void OnTEventFirst(
            string msg) { Console.Write(
            "[2]: got event: ({0})\n", msg);
    }
```

События

```
    public bool OnTEventSecond(
        string msg, int value)
    {
        Console.Write(
          "[2]: got event: ({0},{1})\n",
          msg, value);
        return (value == 0)
            ? false : true;
    }
}

class Program
```

События

```
{
    static void Main(string[] args)
    {
        int an1_called = 0,
            an2_called = 0;

        Console.WriteLine(
            "Event example:\n");

// объявление переменных, создание объектов:
        Sender snd = new Sender();
```

События

```
ReceiverOne rec1 = new ReceiverOne();
ReceiverTwo rec2 = new ReceiverTwo();

// регистрация обработчиков событий:
snd.evFst += rec1.OnTEventFirst;
snd.evFst += rec2.OnTEventFirst;

// регистрация обработчиков событий:
snd.evSec += rec1.OnTEventSecond;
snd.evSec += rec2.OnTEventSecond;
```

События

```
// отправляем события:
snd.DoCalls("event #100", 100);
snd.DoCalls("event #0", 0);
snd.DoCalls("event #-100", -100);

// удаляем подписку:
snd.evFst -= rec1.OnTEventFirst;
snd.evFst -= rec2.OnTEventFirst;
```

События

```
// отправляем события:
snd.DoCalls("event #99", 99);
// регистрация обработчиков событий:
snd.evFst += delegate(string msg)
{
    Console.Write(
        "[0]: got event: ({0})\n", msg);
    // локальные переменные
    // внешнего метода доступны:
    an1_called++;
};
```

События

```
snd.evSec +=
    delegate(string msg, int value)
    {
        Console.Write(
        "[0]: got event: ({0},{1})\n",
        msg, value);
        // локальные переменные
        // внешнего метода доступны:
        an2_called++;
        return true;
    };
```

События

```
    snd.evSec +=
        ((msg, value) => (value != -1));
    // отправляем ещё события
    snd.DoCalls("event #33", 33);
    snd.DoCalls("event #-1", -1);
    // распечатываем локальные переменные:
    Console.WriteLine(
    "an1_called = {0}, an2_called = {1}",
    an1_called, an2_called);
}}}
```

События

Результат работы программы:

```
Event example:
*** Send events for (event #100,100):
[1]: got event: (event #100)
[2]: got event: (event #100)
[1]: got event: (event #100,100)
[2]: got event: (event #100,100)
*** Send events for (event #0,0):
[1]: got event: (event #0)
[2]: got event: (event #0)
[1]: got event: (event #0,0)
[2]: got event: (event #0,0)
```

События

```
*** Event (event #0,0) was not processed!
*** Send events for (event #-100,-100):
[1]: got event: (event #-100)
[2]: got event: (event #-100)
[1]: got event: (event #-100,-100)
[2]: got event: (event #-100,-100)
*** Send events for (event #99,99):
[1]: got event: (event #99,99)
[2]: got event: (event #99,99)
```

События

```
*** Send events for (event #33,33):
[0]: got event: (event #33)
[1]: got event: (event #33,33)
[2]: got event: (event #33,33)
[0]: got event: (event #33,33)
*** Send events for (event #-1,-1):
[0]: got event: (event #-1)
[1]: got event: (event #-1,-1)
[2]: got event: (event #-1,-1)
[0]: got event: (event #-1,-1)
*** Event (event #-1,-1) was not processed!
an1_called = 2, an2_called = 2
```

Интерфейсы и конструкции C#

Рассмотрим основные интерфейсы языка C# и связанные с ними синтаксические конструкции языка.

1) Оператор цикла `foreach` и связанные с ним интерфейсы.

Оператор `foreach` повторяет группу вложенных операторов для каждого элемента массива или коллекции объектов, реализующих интерфейс `System.Collections.IEnumerable`.

Интерфейсы и конструкции C#

Оператор `foreach` используется для итерации коллекции с целью получения необходимой информации, однако его не следует использовать для добавления или удаления элементов исходной коллекции во избежание непредвиденных побочных эффектов. Если нужно добавить или удалить элементы исходной коллекции, следует пользоваться циклом `for`.

После завершения итерации всех элементов коллекции управление переходит к следующему оператору после блока `foreach`.

Интерфейсы и конструкции C#

Так же как в других операторах цикла в любой точке блока `foreach` можно пользоваться операторами `break`, `continue`, `return` или `throw`.

Рассмотрим простой пример использования цикла `foreach` с массивом.

Интерфейсы и конструкции C#

```
class ForEachTest
{    static void Main(string[] args)
     {
          int[] fibarray = new int[]
                    {0, 1, 2, 3, 4, 5};
          foreach (int i in fibarray)
          {
               System.Console.WriteLine(i);
          }
     }
}
```

Интерфейсы и конструкции C#

Результат:

```
0
1
2
3
4
5
```

Интерфейсы и конструкции C#

Рассмотрим интерфейсы связанные с `foreach`.

`System.Collections.IEnumerable`

Для реализации этого интерфейса в вашем классе необходимо реализовать всего один метод:

```
IEnumerator GetEnumerator()
```

В качестве результата возвращается ссылка на интерфейс `IEnumerator`.

Интерфейсы и конструкции C#

`System.Collections.IEnumerator`

Этот интерфейс объявляет два метода и свойство:

```
void Reset();
```

Устанавливает перечислитель в его начальное положение, перед первым элементом коллекции.

```
bool MoveNext();
```

Перемещает перечислитель к следующему элементу коллекции.

Интерфейсы и конструкции C#

```
Object Current { get; }
```

Получает текущий элемент в коллекции.

Рассмотрим пример реализации рассмотренных интерфейсов.

Интерфейсы и конструкции C#

```
using System;
using System.Collections;

public class Person
{
    public Person(string fName, string lName)
    {
        this.firstName = fName;
        this.lastName = lName;
    }

    public string firstName;
    public string lastName;
}
```

Интерфейсы и конструкции C#

```
public class People : IEnumerable
{
    private Person[] _people;
    public People(Person[] pArray){
        _people = new Person[pArray.Length];
        for (int i = 0; i < pArray.Length; i++)
        {
            _people[i] = pArray[i];
        }
    }
    IEnumerator IEnumerable.GetEnumerator(){
        return new PeopleEnum(_people);
    }
}
```

Интерфейсы и конструкции C#

```
public class PeopleEnum : IEnumerator
{
    public Person[] _people;
    int position = -1;
    // positioned before the first element

    public PeopleEnum(Person[] list) {
        _people = list;
    }

    public bool MoveNext() {
        position++;
        return (position < _people.Length);
    }
```

Интерфейсы и конструкции C#

```
    public void Reset() {
        position = -1;
    }
    public object Current {
        get { try {
                return _people[position];
            }
            catch (IndexOutOfRangeException) {
                throw new
                    InvalidOperationException();
            }
        }
    }
}
```

Интерфейсы и конструкции C#

```
class App {
    static void Main() {
        Person[] peopleArray = new Person[3] {
            new Person("John", "Smith"),
            new Person("Jim", "Johnson"),
            new Person("Sue", "Rabon"),
        };
        People peopleList = new People(peopleArray);
        foreach (Person p in peopleList)
            Console.WriteLine(p.firstName +
                " " + p.lastName);
    }
}
```

Интерфейсы и конструкции C#

2) Оператор `using()` и связанный с ним интерфейс `IDisposable`.

Пример фрагмента кода из предыдущих лекций с оператором `using()`:

```
using (StreamReader sr = new
            StreamReader(filename,
              System.Text.Encoding.
                        GetEncoding(1251) )
       ){
      string str = sr.ReadToEnd();
      Console.Write(str);
}
```

Интерфейсы и конструкции C#

Объекты, объявляемые в круглых скобках оператора `using()` должны реализовывать интерфейс `System.IDisposable`. Интерфейс `System.IDisposable` объявляет единственный метод:

```
void Dispose();
```

Интерфейсы и конструкции C#

3) Сравнение и сортировка объектов возможны, если объекты реализуют интерфейсы `System.IComparable` и `System.Collections.IComparer`.

`System.IComparable`

Объявляет единственный метод

```
int CompareTo(Object obj);
```

Сравнивает **this** с другим объектом того же типа.

Интерфейсы и конструкции C#

Возвращаемое значение:

меньше нуля - `this` меньше параметра *obj*;

нуль - `this` и параметр *obj* равны;

больше нуля - `this` больше параметра *obj*.

`System.Collections.IComparer`

`Объявляет единственный метод`

```
int Compare(Object x, Object y);
```

Сравнивает объект `x` с объектом `y`.

Интерфейсы и конструкции C#

Возвращаемое значение:

меньше нуля	- x меньше параметра y;
нуль	- x и параметр y равны;
больше нуля	- x больше параметра y.

Рассмотрим пример использования рассмотренных интерфейсов при решении задачи Контакты.

Интерфейсы и конструкции C#

Условие задачи Контакты:

а) разработать класс Контакт, определяющий запись в электронной книге мобильного телефона и содержащий по меньшей мере следующие поля:

- *Наименование (имя человека или организации)
- *Номер телефона мобильного
- Номер телефона рабочего
- Номер телефона (домашнего)
- Адрес электронной почты
- Адрес веб-страницы
- Адрес почтовый

Обязательными является поля помеченные *, остальные поля могут быть пустыми

Интерфейсы и конструкции C#

б) класс Контакт должен реализовать:

-интерфейс IComparer с выбором одного из полей для сравнения

-интерфейс IDisposable

-индексатор по всем полям

-метод для сохранения значений всех полей в строке текста (переопределить ToString())

-конструктор/метод для инициализации из строки текста

в) Для тестирования класса Контакт, создать консольное приложение позволяющее создать небольшой массив контактов и напечатать отсортированными по выбранному полю.

Интерфейсы и конструкции C#

```
using System;
using System.Collections;

namespace Contacts
{
    class Contact :
        IComparable, IComparer, IDisposable
    {
        public static string[] Names = {
          "*Contact", "*Mobile", "Work", "Home",
          "E-mail", "WWW-page", "Address"
        };
```

Интерфейсы и конструкции C#

```
private static int _sortBy = 0;
public static int SortBy {
    get { return _sortBy; }
    set {
        if (value >= Names.Length ||
            value < 0) {
            throw new
                IndexOutOfRangeException();
        }
        _sortBy = value;
    }
}
```

Интерфейсы и конструкции C#

```
public virtual bool ValidName(string str)
    { return true; }
public virtual bool ValidPhone(string str)
    { return true; }
public virtual bool ValidAddress(string str)
    { return true; }
public virtual bool ValidEMail(string str)
    { return true; }
public virtual bool ValidWWWPage(string str)
    { return true; }

private string[] Areas = null;
```

Интерфейсы и конструкции C#

```
public int Length
    { get { return Areas.Length; }}
public string this[int idx]
{   get {
        if (idx >= Length || idx < 0) {
            throw new
                IndexOutOfRangeException();
        }
        return Areas[idx];
    }
    set {
        if (idx >= Length || idx < 0) {
            throw new
                IndexOutOfRangeException();
        }
```

Интерфейсы и конструкции C#

```
            if((idx == 0 &&
                  ValidName(value)== false)||
               (idx > 0 && idx < 4 &&
                  ValidPhone(value)== false)||
               (idx == 4 &&
                  ValidEMail(value)== false)||
               (idx == 5 &&
                  ValidWWWPage(value)== false)){
                throw new ArgumentException();
            }
            Areas[idx] = value;
        }
    }
```

Интерфейсы и конструкции C#

```
//IDisposable
public void Dispose()
{
    if (Areas != null)
    {
        for (int i = 0; i < Length; i++)
        {
            Areas[i] = null;
        }
        Areas = null;
    }
    GC.SuppressFinalize(this);
}
```

Интерфейсы и конструкции C#

```
//IComparable
public int CompareTo(Object y)
{
    Contact cy = (Contact)y;
    return this.Areas[Contact.SortBy].
           CompareTo(
               cy.Areas[Contact.SortBy]);
}
//IComparer
public int Compare(Object x, Object y)
{
    return ((Contact)x).CompareTo(y);
}
```

Интерфейсы и конструкции C#

```
public override string ToString()
{
    if (Areas == null)
    {
        return "||||||";
    }
    string res = Areas[0];
    for (int i = 1; i < Areas.Length; i++)
    {
        res += "|" + Areas[i];
    }
    return res;
}
```

Интерфейсы и конструкции C#

```
// constructors:
private Contact() { }
private void Setup(string[] args)
{
    if (args.Length < 2 ||
        args.Length > Names.Length){
        throw new ArgumentException();
    }
    Areas = new string[Names.Length];
    int i = 0;
    for (; i < args.Length; i++) {
        this[i] = args[i];
    }
```

Интерфейсы и конструкции C#

```
        while (i < Names.Length)
        {
            Areas[i++] = "";
        }
    }
    public Contact(string str)
    {
        string[] args = str.Split(
            /*new char[] {*/ '|' /*}*/ );
        Setup(args);
    }
    public Contact(params string[] args)
    {
        Setup(args);
    }
```

Интерфейсы и конструкции C#

```
    //format strings for area printout
     public static string Fmt(int i)
     {
         return (i == 4 || i == 5)
             ? "{0,-17}" : "{0,-9}";
     }
}

class Program
{
     static void SortAndPrint(ArrayList pl)
     // simple printout
     {
```

Интерфейсы и конструкции C#

```
            Console.WriteLine(
                "----- Sorted by: {0}",
                Contact.Names[Contact.SortBy]);
            pl.Sort((Contact)pl[0]);
            foreach (Contact c in pl) {
                for (int i = 0; i < c.Length; i++) {
                    string str = c[i];
                    if (str.Length > 0) {
                        Console.Write("{0}: {1} ",
                            Contact.Names[i], str);
                    }
                }
                Console.WriteLine();
            }
        }
```

Интерфейсы и конструкции C#

```
static void SortAndPrint(
    ArrayList pl, IComparer pc)
{
    Console.WriteLine(
        "----- Sorted by: {0}",
         Contact.Names[Contact.SortBy]);
    pl.Sort(pc);
    int n = 0;
    foreach (string str in Contact.Names) {
        Console.Write(
           Contact.Fmt(n++), str);
    }
    Console.WriteLine();
```

Интерфейсы и конструкции C#

```
        foreach (Contact c in pl) {
            for (int i = 0; i < c.Length; i++)
            {
                string str = c[i];
                Console.Write(
                  Contact.Fmt(i),(str.Length > 0
                     ? str : " "));
            }
            Console.WriteLine();
        }
    }
    static void Main(string[] args)
    {
```

Интерфейсы и конструкции C#

```
            using (Contact pc = new Contact(
"Joahn|1234567|9876543||joahn@gmail.com||")
            )
            {
                ArrayList pl = new ArrayList(3);
                pl.Add(pc);
                pl.Add(new Contact(
"Ann|2345678|8765432||nann@gmail.com||"));
                pl.Add(new Contact(
"Mary","3456789","7654321","","mary@gmail.com"));
```

Интерфейсы и конструкции C#

```
                Contact.SortBy = 0;
                SortAndPrint(pl, pc);
                Contact.SortBy = 1;
                SortAndPrint(pl, pc);
                Contact.SortBy = 2;
                SortAndPrint(pl, pc);
                Contact.SortBy = 4;
                SortAndPrint(pl, pc);
            }
        }
    }
}
```

Интерфейсы и конструкции C#

Результат:

```
----- Sorted by: *Contact
*Contact *Mobile  Work     Home     E-mail           WWW-page         Address
Ann      2345678  8765432           nann@gmail.com
Joahn    1234567  9876543           joahn@gmail.com
Mary     3456789  7654321           mary@gmail.com
----- Sorted by: *Mobile
*Contact *Mobile  Work     Home     E-mail           WWW-page         Address
Joahn    1234567  9876543           joahn@gmail.com
Ann      2345678  8765432           nann@gmail.com
Mary     3456789  7654321           mary@gmail.com
----- Sorted by: Work
*Contact *Mobile  Work     Home     E-mail           WWW-page         Address
Mary     3456789  7654321           mary@gmail.com
Ann      2345678  8765432           nann@gmail.com
Joahn    1234567  9876543           joahn@gmail.com
----- Sorted by: E-mail
*Contact *Mobile  Work     Home     E-mail           WWW-page         Address
Joahn    1234567  9876543           joahn@gmail.com
Mary     3456789  7654321           mary@gmail.com
Ann      2345678  8765432           nann@gmail.com
```

Библиотека Windows Forms

Предназначена для разработки приложений с графическим интерфейсом пользователя.

Поддерживается всеми языками платформы .NET, в том числе и языком C#/

Рассмотрим шаблон приложения на простом примере.

Библиотека Windows Forms

```
using System;
using System.IO;
using System.Windows.Forms;
using System.Drawing;
namespace ExampleWinForms1
{    // Application object:
     class Program
     {    [STAThread]//<-Required for WinForms

          static void Main(string[] args)
          {
               Application.Run(new AppWindow(
                    "Hello world!", 300, 200));
```

Библиотека Windows Forms

```
}}
// Главное окно приложения:
class AppWindow : Form
{    // constructor:
     public AppWindow(string title,
          int width, int height)
     {// заголовок и размеры окна:
          Text = title;
          if (width > 0)
          {
               Width = width;
          }
```

Библиотека Windows Forms

```
            if (height > 0)
            {
                Height = height;
            }
            // фиксируем размеры окна:
            MinimumSize = new Size(
                Width, Height);
            MaximumSize = new Size(
                Width, Height);
            // Центрируем окно:
            CenterToScreen();
}}}
```

Библиотека Windows Forms

Результат запуска программы:

Библиотека Windows Forms

Рассмотрим шаблон более сложного приложения Windows.

Это приложения содержит главное меню, строку состояния для вывода подсказок и клиентскую область окна в которую мы помещаем элементы управления.

Этот шаблон соответствует рекомендациям стандарта CUI (Common User Interface).

Библиотека Windows Forms

```
/* Example application Windows Forms #2
 * Demonstrates: Menu, StatusLine, Client area,
 *               Resize event, Mouse events
 * Written by Sergey Gutnikov
 */
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
namespace ExampleWinForms2 {
    static class Program  {
        [STAThread]//<-Required for WinForms
        static void Main() {
            Application.EnableVisualStyles();
```

Библиотека Windows Forms

```
            Application.Run(
                new AppWindow(
                    "Hello world!", 300, 300));
        }
    }
    class AppWindow : Form
    {   // event handler for FormClosing
        private void AppWindow_FormClosing(
            Object sender, FormClosingEventArgs e)
        {
            if (MessageBox.Show(
                "Are you sure to exit?",
```

Библиотека Windows Forms

```
                Text, // message tittle
                MessageBoxButtons.YesNo,
                // message buttons
                MessageBoxIcon.Question)
                // message icon
            != DialogResult.Yes)
                // result
        {
            e.Cancel = true;
        }
}

// event handler for Resize:
```

Библиотека Windows Forms

```
private void AppWindow_Resize(
    object sender, System.EventArgs e)
{
    ArrangeClientArea();
}

// constructor:
public AppWindow(string title,
    int width, int height )
{
    // setup event handler:
    FormClosing += AppWindow_FormClosing;
    Resize += AppWindow_Resize;
    // set application title, width, height:
```

Библиотека Windows Forms

```
        Text = title;
        if (width > 0) Width = width;
        if (height > 0) Height = height;
        // Center window:
        CenterToScreen();
        // Add controls
        BuildControls();
    }
    private void BuildControls()
    {
        // Add tooltip
        BuildToolTip();
```

Библиотека Windows Forms

```
        // Add menu
        BuildMenu();
        // Add StatusLine
        BuildStatusLine();
        // Add button
        BuildButton();
        // Position client area items:
        ArrangeClientArea();
        // Setup minimum of size changing:
        MinimumSize = AppMinSize(
            NcSize.Width =
              (Width - ClientSize.Width + 1),
            NcSize.Height =
              (Height - ClientSize.Height + 1));
}
```

Библиотека Windows Forms

```
private Size NcSize = new Size();

private Rectangle GetClientRectangle()
{
    return new Rectangle(
        NcSize.Width / 2,
        mnuMain.Bounds.Bottom,
        Width - NcSize.Width,
        statusStrip.Bounds.Top -
           mnuMain.Bounds.Bottom );
}
private Size AppMinSize(
    int ncWidth,  // non client width
    int ncHeight) // height
{
```

Библиотека Windows Forms

```
//- a button in client area with size (100,50):
            ncWidth += btnClickMe.Width;
            ncHeight += btnClickMe.Height;
            // plus menu and status height:
            ncHeight += mnuMain.Height +
                statusStrip.Height;
            return new Size(ncWidth, ncHeight);
        }

        private void ArrangeClientArea()
        {
            // center button to client area:
            if (btnClickMe != null)
            {
```

Библиотека Windows Forms

```
            btnClickMe.SetBounds(
                (ClientSize.Width-
                    btnClickMe.Width)/2 + 1,
                (ClientSize.Height-
                    btnClickMe.Height)/2 + 1,
                btnClickMe.Width,
                btnClickMe.Height );
        }
    }
    // 1) declare GUI-control variables:
    //members for status line:
    private StatusStrip statusStrip;
    private
    ToolStripStatusLabel toolStripStatusLabel;
```

Библиотека Windows Forms

```
private void BuildStatusLine()
{   // 2) set-up GUI-controls:
    statusStrip = new StatusStrip();
    toolStripStatusLabel = new
        ToolStripStatusLabel();
    // freeze layout:
    statusStrip.SuspendLayout();
    SuspendLayout();

    statusStrip.Items.AddRange(
        new ToolStripItem[] {
            toolStripStatusLabel });
    statusStrip.Location =
        new Point(0, 248);
```

Библиотека Windows Forms

```
statusStrip.Name = "statusStrip";
statusStrip.Size = new Size(292, 30);
statusStrip.TabIndex = 0;
statusStrip.Text = "statusStrip";

toolStripStatusLabel.BorderSides = 0;
toolStripStatusLabel.BorderStyle =
    Border3DStyle.Flat;
toolStripStatusLabel.IsLink = false;
toolStripStatusLabel.Name =
    "toolStripStatusLabel";
toolStripStatusLabel.Size =
    new Size(246, 28);
```

Библиотека Windows Forms

```
toolStripStatusLabel.Spring = true;
    // fill space
toolStripStatusLabel.Text = "";
toolStripStatusLabel.Alignment =
       ToolStripItemAlignment.Left;
toolStripStatusLabel.TextAlign =
    ContentAlignment.TopLeft;

// 3) Insert GUI-control to
//       Form's ControlCollection
Controls.Add(statusStrip);
// defrost layout:
statusStrip.ResumeLayout(false);
statusStrip.PerformLayout();
```

Библиотека Windows Forms

```
        ResumeLayout(false);
        PerformLayout();
    }
    // 1) declare GUI-control variables:
    //members for simple menu:
    private MenuStrip mnuMain =
        new MenuStrip(); // all menu-system
    private ToolStripMenuItem mnuFile = new
        ToolStripMenuItem(); // pop-down menu
    private ToolStripMenuItem mnuFileChFolder =
         new ToolStripMenuItem();
         // pop-down menu item
    private ToolStripMenuItem mnuFileOpenFile =
         new ToolStripMenuItem();
         // pop-down menu item
```

Библиотека Windows Forms

```
private ToolStripMenuItem
    mnuFileSaveAsFile =
     new ToolStripMenuItem();
     // pop-down menu item
private ToolStripMenuItem mnuFileExit =
     new ToolStripMenuItem();
     // pop-down menu item
private ToolStripMenuItem mnuHelp =
     new ToolStripMenuItem();
     // pop-down menu
private ToolStripMenuItem mnuHelpAbout =
     new ToolStripMenuItem();
     // pop-down menu item
```

Библиотека Windows Forms

```
private void BuildMenu()
{
    // 2) set-up GUI-controls:
    //insert pop-down menu to main menu
    mnuFile.Text = "&File";
        //set text with Alt-command: Alt-F
    mnuMain.Items.Add(mnuFile);
    mnuHelp.Text = "&Help";
        //set text with Alt-command: Alt-H
    mnuMain.Items.Add(mnuHelp);
    //insert Choose Folder command to
    //   pop-down menu
    mnuFileChFolder.Text =
        "&Choose Folder...";
        //set text with command: C
```

Библиотека Windows Forms

```
mnuFile.DropDownItems.Add(
    mnuFileChFolder);

// event for command
//      (System.EventHandler)
mnuFileChFolder.Click +=
    (o, s) => OnFileChooseFolder();
mnuFileChFolder.MouseEnter +=
    (o, s) =>
      toolStripStatusLabel.Text =
         "Choose an existing folder";
mnuFileChFolder.MouseLeave +=
    (o, s) =>
      toolStripStatusLabel.Text = "";
```

Библиотека Windows Forms

```
//insert Open command to pop-down menu
mnuFileOpenFile.Text = "&Open";
    //set text with command: O
mnuFile.DropDownItems.Add(
    mnuFileOpenFile);
// set-up event handler for Open command
mnuFileOpenFile.Click +=
    (o, s) => OnFileOpen();
mnuFileOpenFile.MouseEnter += (o, s) =>
      toolStripStatusLabel.Text =
         "Open an existing file";
mnuFileOpenFile.MouseLeave +=
    (o, s) =>
      toolStripStatusLabel.Text = "";
```

Библиотека Windows Forms

```
//insert SaveAs command to pop-down menu
mnuFileSaveAsFile.Text = "&Save As";
    //set text with command: S
mnuFile.DropDownItems.Add(
    mnuFileSaveAsFile);
// event handler for command
mnuFileSaveAsFile.Click +=
    (o, s) => OnFileSaveAs();
mnuFileSaveAsFile.MouseEnter += (o,s) =>
      toolStripStatusLabel.Text =
        "Save a file with a new name";
mnuFileSaveAsFile.MouseLeave +=
     (o, s) =>
      toolStripStatusLabel.Text = "";
```

Библиотека Windows Forms

```
//insert Separator to pop-down menu
mnuFile.DropDownItems.Add(
    new ToolStripSeparator());
//insert Exit command to pop-down menu
mnuFileExit.Text = "E&xit";
    //set text with command: x
mnuFile.DropDownItems.Add( mnuFileExit);
// handler for command
mnuFileExit.Click +=
    (o, s) => Application.Exit();
mnuFileExit.MouseEnter += (o, s) =>
      toolStripStatusLabel.Text =
         "Exit application...";
mnuFileExit.MouseLeave += (o, s) =>
      toolStripStatusLabel.Text = "";
```

Библиотека Windows Forms

```
//insert About command to pop-down menu
mnuHelpAbout.Text = "&About...";
    //set text with command: A
mnuHelp.DropDownItems.Add(
    mnuHelpAbout);
// set-up event handler
mnuHelpAbout.Click += (o, s) =>
    OnHelpAbout();
mnuHelpAbout.MouseEnter += (o, s) =>
      toolStripStatusLabel.Text =
         "Show information about " +
         " the programm...";
mnuHelpAbout.MouseLeave += (o, s) =>
      toolStripStatusLabel.Text = "";
```

Библиотека Windows Forms

```
        // add menu to form:
        Controls.Add(mnuMain);
        MainMenuStrip = mnuMain;
            // special form member to seting-up
}
// 1) declare GUI-control variables:
private Button btnClickMe = new Button();
private void BuildButton()
{
        // 2) set-up GUI-controls:
        btnClickMe.Text = "Click Me...";
            //set text
        btnClickMe.SetBounds(50, 50, 100, 50);
            //set bounds (x,y,width,height)
```

Библиотека Windows Forms

```
        btnClickMe.Click += (o, s) => {
            MessageBox.Show("You've done that!",
                            btnClickMe.Text);
        };
        btnClickMe.MouseEnter += (o, s) =>
              toolStripStatusLabel.Text =
                "Click button to action...";
        btnClickMe.MouseLeave += (o, s) =>
              toolStripStatusLabel.Text = "";
        // Set up the ToolTip text
        toolTip.SetToolTip(
            btnClickMe as Control, "Click");
        // 3) Insert to ControlCollection
        Controls.Add(btnClickMe);
}
```

Библиотека Windows Forms

```
ToolTip toolTip = new ToolTip();

private void BuildToolTip()
{
    // Set up the delays for the ToolTip.
    toolTip.AutoPopDelay = 5000;
    toolTip.InitialDelay = 1000;
    toolTip.ReshowDelay = 500;
    // Force the ToolTip text to be
    //   displayed whether or not
    //   the form is active.
    toolTip.ShowAlways = true;
}
```

Библиотека Windows Forms

```
private void OnFileOpen()
{
    using (OpenFileDialog openFileDialog =
                new OpenFileDialog())
    {
        openFileDialog.InitialDirectory =
            Environment.GetFolderPath(
                Environment.SpecialFolder.
                    Personal);
            // My Documents folder.
        openFileDialog.Filter =
"Database (*.mdb) |*.mdb|All files (*.*) |*.*";
        openFileDialog.FilterIndex = 0;
        openFileDialog.DefaultExt = "mdb";
        openFileDialog.FileName = "*.mdb";
```

Библиотека Windows Forms

```
            openFileDialog.RestoreDirectory =
                true;
            openFileDialog.Multiselect =
                false;
                // if true - use FileNames[]
                //    to get selected
            if (openFileDialog.ShowDialog()==
                    DialogResult.OK)
            {
                MessageBox.Show(
                  "File selected: " +
                    openFileDialog.FileName);
            }
        }
    }
```

Библиотека Windows Forms

```
private void OnFileSaveAs()
{
    using (SaveFileDialog saveFileDialog=
                new SaveFileDialog())
    {
        saveFileDialog.InitialDirectory =
            Environment.GetFolderPath(
              Environment.SpecialFolder.
                Personal);
                // My Documents folder.
        saveFileDialog.Filter =
"Database (*.mdb) |*.mdb|All files (*.*) |*.*";
        saveFileDialog.FilterIndex = 0;
        saveFileDialog.DefaultExt = "mdb";
        saveFileDialog.FileName = "*.mdb";
```

Библиотека Windows Forms

```
            saveFileDialog.RestoreDirectory =
                true;
            if (saveFileDialog.ShowDialog()==
                    DialogResult.OK )
            {
                MessageBox.Show(
                    "File selected: " +
                    saveFileDialog.FileName);
            }
        }
    }

    private void OnFileChooseFolder()
    {
```

Библиотека Windows Forms

```
        using (FolderBrowserDialog
                folderBrowserDialog =
                    new FolderBrowserDialog())
        {
            // Set the help text
            folderBrowserDialog.Description =
"Select the directory that you want to use";
            // Don't allow to create new files
            folderBrowserDialog.
                ShowNewFolderButton = false;
            // My Documents folder:
            folderBrowserDialog.SelectedPath =
                Environment.GetFolderPath(
                    Environment.SpecialFolder.
                        Personal );
```

Библиотека Windows Forms

```
            folderBrowserDialog.RootFolder =
                    Environment.SpecialFolder.
                        Personal;
            // Show the FolderBrowserDialog.
            if (folderBrowserDialog.
                    ShowDialog(this) ==
                        DialogResult.OK)
            {//SelectedPath=="" for MyComputer
                MessageBox.Show(
                    "Folder choosen: " +
                     folderBrowserDialog.
                     SelectedPath);
            }
    }}
```

Библиотека Windows Forms

```
        private void OnHelpAbout()
        {
                MessageBox.Show(
                    "MinForms #2 sample application\n”+
                    ”Written by Sergey Gutnikov",
                    "About...",
                    // message tittle
                    MessageBoxButtons.OK,
                    // message buttons
                    MessageBoxIcon.Information);
                    // message icon
        }
    }
}
```

Библиотека Windows Forms

Результат: