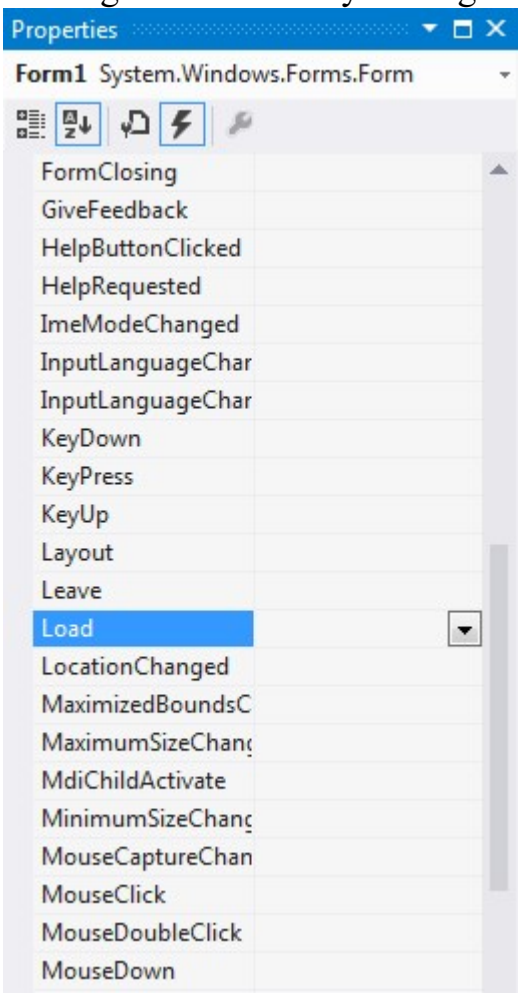
14-ma’ruza. Hodisalar

- ❖ Windows Forms da hodisalar. Forma hodisalari
- ❖ Forma hodisalarining qisqacha tavsifi

Kalit so‘zlar: *Click, CollectionChanged, CollectionChanging, DoubleClick, KeyDown, KeyPress, KeyUp, Load, MouseCaptureChanged, MouseClick, MouseDoubleClick, Paint, PreviewKeyDown, qayta ishlovchi (обработчик), Xodisa.*

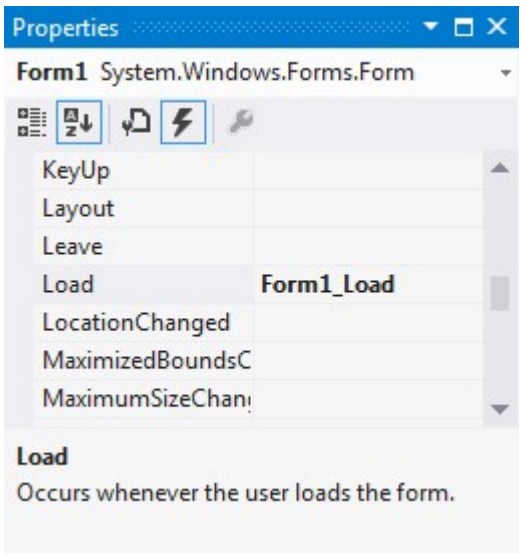
Windows Forms da hodisalar. Forma hodisalari

Windows Forms da foydalanuvchi bilan muloqot uchun hodisalar mexanizmi qo‘llaniladi. Windows Formsda hodisalar faqat C# ning visual komponentalariga qo‘llaniladigan standart hodisalarini ifodalaydi va C# ning hodisalari bo‘ysunadigan qoidalarga amal qiladi. Lekin Windows Formsda hodisalarni qayta ishlovchisini yaratish ham o‘ziga xos xususiyatlarga ega.



WinFormsda bir qator standart xodisalar to‘plami mavjud, ularning kattagina qismi barcha visual komponentalarda keltirilgan. Alohida elementlar o‘z xodisalarini qo‘shadi, lekin ular bilan ishlash prinsipi juda o‘xshash. Elementning barcha xodisalarini ko‘rish uchun grafik dizayner maydonidan shu elementni tanlash va forma panelida xodisalar bo‘limiga o‘tish zarur. Masalan, forma xodisalari:

Hodisani qayta ishlovchini qo‘shish uchun hodisa nomining yonidagi bo‘sh maydonda sichqonchaning o‘ng tugmasini ikki marta bosish yetarli, shundan so‘ng Visual Studio hodisani qayta ishlashni avtomatik generatsiyalaydi. Masalan, Load hodisasini qayta ishlovchini yaratish uchun:



Bu maydonda Load hodisasini qayta ishlovchi metod nomi aks etgsn. Sukut bo‘yicha u Form1_Load deb nomlanadi.

Agar Form1.cs kod fayliga o‘tsak, unda avtomatik generatsiyalangan Form1_Load metodini ko‘rishimiz mumkin:

```
public partial class Form1 : Form
{
   public Form1()
   {
      InitializeComponent();
   }
   private void Form1_Load(object sender, EventArgs e)
   {
   }
}
```

Formaning har bir yuklanishida Form1_Load hodisasi uchun yozilgan kod ishga tushadi.

Odatda, ko‘pchilik turli visual komponentalar hodisa ishlovchilari ikkita parametrga ega bo‘ladi: sender – hodisa initsializatsiyalangan ob’yekt va hodisa haqidagi ma’lumotni saqlovchi argument (mazkur holatda EventArgs e).

Lekin bu faqat qayta ishlovchi. Bunday usulda yaratilgan qayta ishlovchini qo‘shish Form1.Designer.cs faylida amalga oshiriladi:

```
namespace HelloApp
{
```

```
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
        private void InitializeComponent()
        {
          this.SuspendLayout();
          this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
          this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
          this.ClientSize = new System.Drawing.Size(284, 261);
          this.Name = "Form1";
         // hodisa ishlovchini qo‘shish
          this.Load += new System.EventHandler(this.Form1_Load);
          this.ResumeLayout(false);
        }
    }
}
```

Hodisa ishlovchisini qo‘shish uchun standart C# sintaksisi qo‘llaniladi:

```
this.Load += new System.EventHandler(this.Form1_Load)
```

Shu sababli bunday usulda yaratilgan hodisa ishlovchisini o‘chirish uchun nafaqat Form1.cs forma kodidan metodni, balki hodisa ishlovchisini qo‘shishni ham o‘chirish talab qilinadi.

Shu bilan birga, hodisa ishlovchisini dasturiy ravishda, masalan, forma konstruktorida qo‘shish ham mumkin:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HelloApp
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         this.Load += LoadEvent;
      }
      private void Form1_Load(object sender, EventArgs e)
      {
      }
      private void LoadEvent(object sender, EventArgs e)
      {
         this.BackColor = Color.Yellow;
      }
   }
}
```

Oldin yaratilgan Form1_Load hodisa ishlovchisidan tashqari bu yerda fon sifatida sariq rangni o‘rnatuvchi boshqa formani yuklash ishlovchisi ham qo‘shilgan: this.Load += LoadEvent

Forma hodisalarining qisqacha tavsifi

Tashqi ko‘rinish bo‘yicha:

Paint – boshqaruv elementini qayta chizish kerak bo‘lganida sodir bo‘ladi.

Quyidagi kod misoli formada PictureBox boshqaruvini yaratadi va unga chizish uchun Paint hodisasidan foydalanadi:

```
// This example creates a PictureBox control on the form and draws to it.
// This example assumes that the Form_Load event handler method is
// connected to the Load event of the form.
private PictureBox pictureBox1 = new PictureBox();
// Cache font instead of recreating font objects each time we paint.
private Font fnt = new Font("Arial",10);
private void Form1_Load (object sender, System.EventArgs e)
{
   // Dock the PictureBox to the form and set its background to white.
   pictureBox1.Dock = DockStyle.Fill;
   pictureBox1.BackColor = Color.White;
   // Connect the Paint event of the PictureBox to the event handler method.
pictureBox1.Paint += new System.Windows.Forms.PaintEventHandler
(this.pictureBox1_Paint);
   // Add the PictureBox control to the Form.
   this.Controls.Add(pictureBox1);
}
private void pictureBox1_Paint (object sender,
System.Windows.Forms.PaintEventArgs e)
{
   // Create a local version of the graphics object for the PictureBox.
   Graphics g = e.Graphics;
   // Draw a string on the PictureBox.
   g.DrawString("This is a diagonal line drawn on the control",
      fnt, System.Drawing.Brushes.Blue, new Point(30,30));
   // Draw a line in the PictureBox.
 g.DrawLine (System.Drawing.Pens.Red, pictureBox1.Left,
pictureBox1.Top, pictureBox1.Right, pictureBox1.Bottom);
}
```

Berilganlar bilan ishlashda:

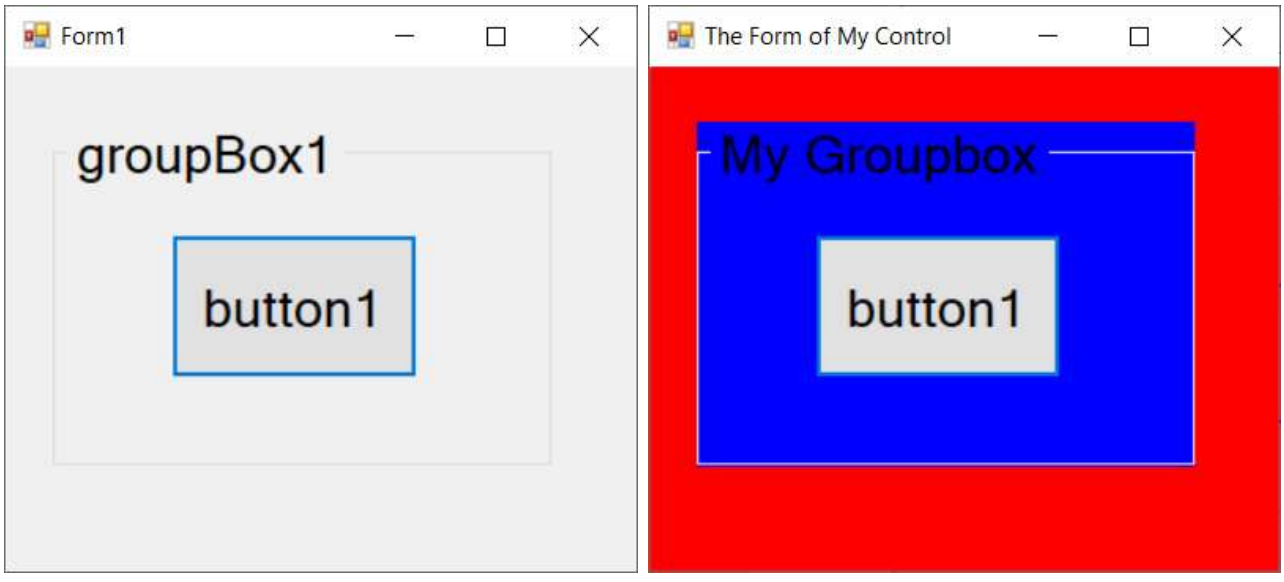
CollectionChanged – joriy kolleksiyaga tegishli har bir o‘zgarishlar kiritilganida sodir bo‘ladi.

CollectionChanging – kolleksiya tarkibi o‘zgarganida sodir bo‘ladi.

Click – boshqaruv elementi bosilganida yuz beradi.

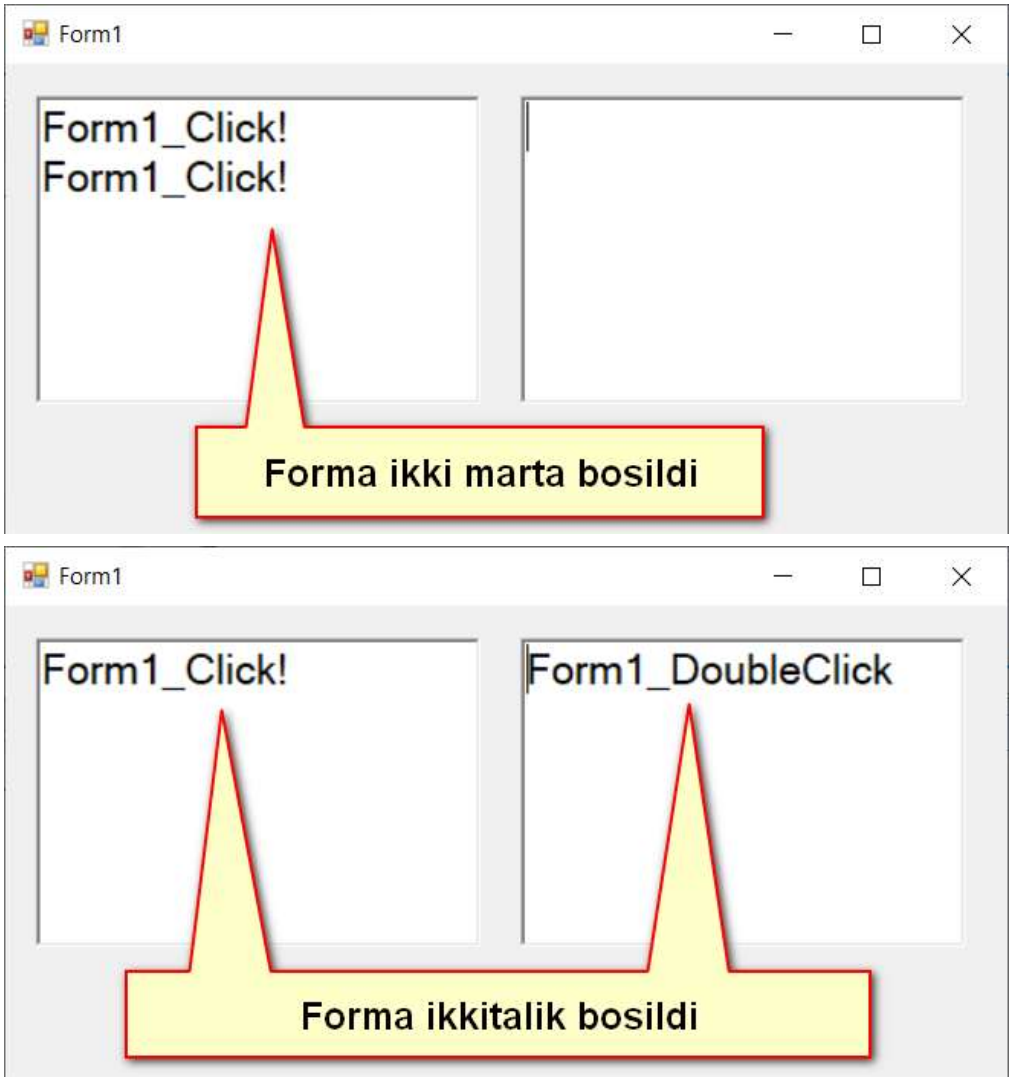
Quyidagi kod misolida tugmachaning Click hodisasi ko‘rsatilgan:

```
// This example uses the Parent property and the Find method of Control to set
// properties on the parent control of a Button and its Form. The example assumes
// that a Button control named button1 is located within a GroupBox control. The
// example also assumes that the Click event of the Button control is connected to
// the event handler method defined in the example.
private void button1_Click (object sender, System.EventArgs e)
{
  // Get the control the Button control is located in. In this case a GroupBox.
  Control control = button1.Parent;
  // Set the text and backcolor of the parent control.
  control.Text = "My Groupbox";
  control.BackColor = Color.Blue;
  // Get the form that the Button control is contained within.
  Form myForm = button1.FindForm();
  // Set the text and color of the form containing the Button.
  myForm.Text = "The Form of My Control";
  myForm.BackColor = Color.Red;
}
```



DoubleClick – boshqaruv elementi ikki marta bosilganida yuz beradi.

Izoh. Click hodisasi uning hodisa ishlovchisida EventArgs parametriga o‘tadi, bu faqat bir marta bosish yuz berganligini ko‘rsatadi. Agar Sizga sichqoncha haqida (tugmasi, bosishlar soni, g‘ildakchasini aylantirish yoki joylashuv o‘rni bo‘yicha) to‘liqroq ma’lumot kerak bo‘lsa, MouseClick hodisasidan foydalaning. Agar sichqoncha harakatidan farqli bosish hodisasida, masalan, klaviaturadan Enter tugmasi bosilganida MouseClick hodisasi chaqirilmaydi. Ikkitalik bosish foydalanuvchi operatsion tizimidagi sichqoncha sozlamalari bilan aniqlanadi. Foydalanuvchi bosishlar o‘rtasidagi vaqt oralig‘ini ham belgilab olishi mumkin, ya’ni ikkitalik bosish va ikki marta bosish farqi. Click hodisasi boshqaruv elementi ikki marta bosilishining har birida chaqiriladi. Masalan, formada Click va DoubleClick hodisalari ishlovchilari mavjud bo‘lsa, ikki marta bosilganida faqat Click hodisasi ikki marta chaqiriladi, ikkitalik bosilganida esa Click hodisasi ham DoubleClick hodisasi ham chaqiriladi.

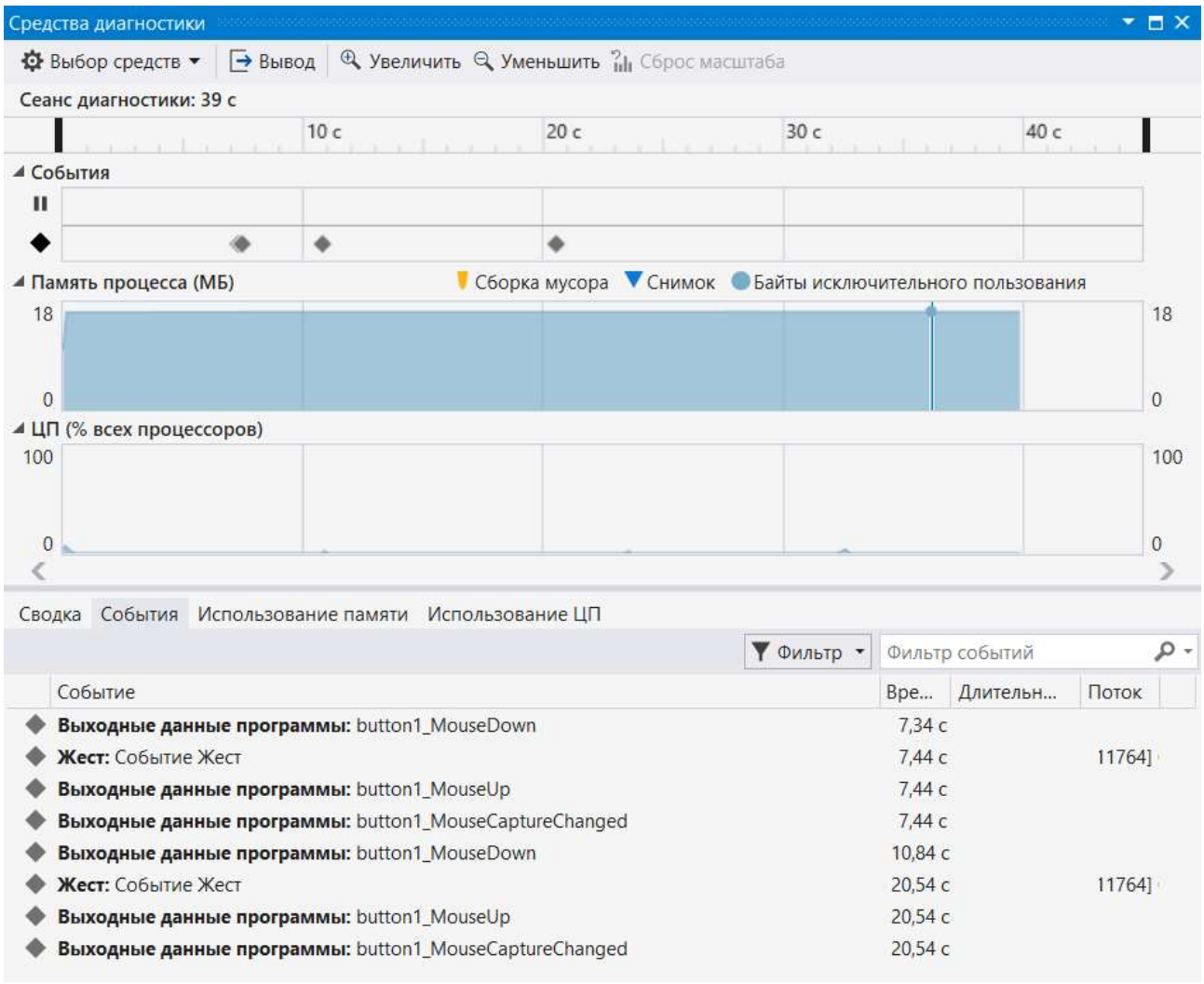


```
private void Form1_Click (object sender, EventArgs e)
        {
          richTextBox1.Text = richTextBox1.Text + "Form1_Click!\n";
        }
private void Form1_DoubleClick (object sender, EventArgs e)
        {
          richTextBox2.Text = richTextBox2.Text + "Form1_DoubleClick\n";
        }
```

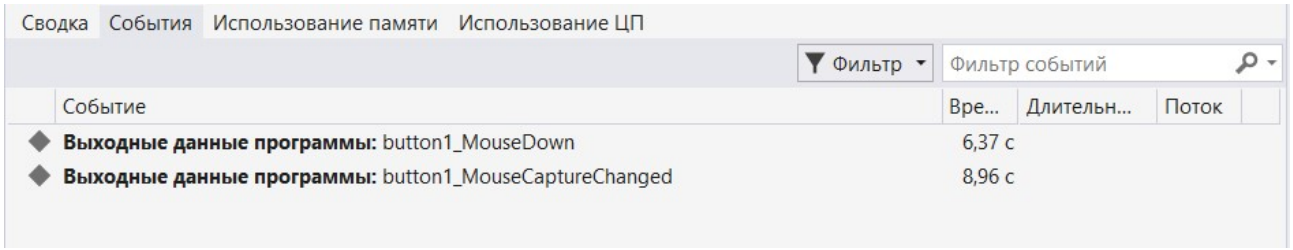
MouseCaptureChanged – sichqonchadan kelgan xabarni “tutish” o‘zgarganidan so‘ng sodir bo‘ladi.

Quyidagi kod misolida Button boshqaruv elementi uchun MouseCaptureChanged hodisasi namoyish etilgan:

```
private void button1_MouseDown (object sender, MouseEventArgs e)
{
   System.Diagnostics.Debug.WriteLine ("button1_MouseDown");
}
private void button1_MouseUp (object sender, MouseEventArgs e)
{
   System.Diagnostics.Debug.WriteLine ("button1_MouseUp");
}
private void button1_MouseCaptureChanged (object sender, EventArgs e)
{
   System.Diagnostics.Debug.WriteLine ("button1_MouseCaptureChanged");
}
```



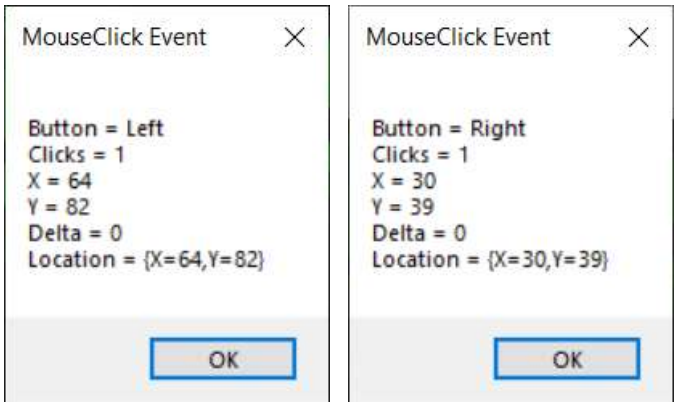
Sichqoncha tugmasi bosib turilgan holatda boshqa dasturga o‘tilganida (masalan, Alt+Tab orqali) MouseUp hodisasi chaqrilmaydi:



MouseClick – boshqaruv elementida sichqoncha bosilganida sodir bo‘ladi.

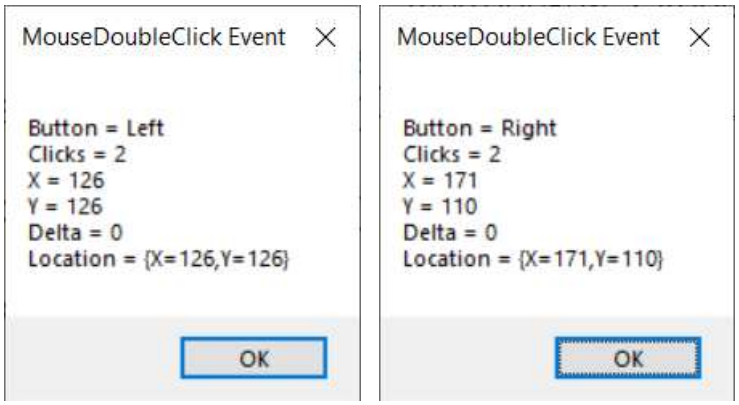
```
private void Form1_MouseClick(object sender, MouseEventArgs e)
    {
      System.Text.StringBuilder messageBoxCS = new System.Text.StringBuilder();
      messageBoxCS.AppendFormat("{0} = {1}", "Button", e.Button);
      messageBoxCS.AppendLine();
      messageBoxCS.AppendFormat("{0} = {1}", "Clicks", e.Clicks);
      messageBoxCS.AppendLine();
      messageBoxCS.AppendFormat("{0} = {1}", "X", e.X);
      messageBoxCS.AppendLine();
      messageBoxCS.AppendFormat("{0} = {1}", "Y", e.Y);
      messageBoxCS.AppendLine();
```

```
        messageBoxCS.AppendFormat("{0} = {1}", "Delta", e.Delta);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "Location", e.Location);
        messageBoxCS.AppendLine();
        MessageBox.Show(messageBoxCS.ToString(), "MouseClick Event");
    }
```



MouseDoubleClick – sichqoncha tugmasi ikkitalik bosilganida sodir bo‘ladi.

```
private void Form1_MouseDoubleClick(object sender, MouseEventArgs e)
    {
        System.Text.StringBuilder messageBoxCS = new System.Text.StringBuilder();
        messageBoxCS.AppendFormat("{0} = {1}", "Button", e.Button);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "Clicks", e.Clicks);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "X", e.X);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "Y", e.Y);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "Delta", e.Delta);
        messageBoxCS.AppendLine();
        messageBoxCS.AppendFormat("{0} = {1}", "Location", e.Location);
        messageBoxCS.AppendLine();
        MessageBox.Show(messageBoxCS.ToString(), "MouseDoubleClick Event");
    }
```



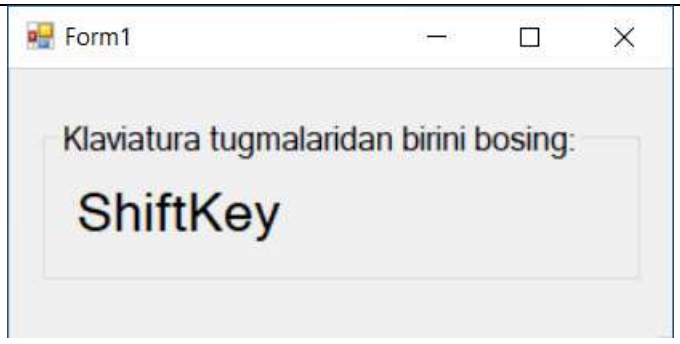
KeyDown – Klaviatura tugmasi birinchi bosilganida sodir bo‘ladi.

```
private bool nonNumberEntered = false;
    private void textBox1_KeyDown(object sender, KeyEventArgs e)
    {
        nonNumberEntered = false;
        if (e.KeyCode < Keys.D0 || e.KeyCode > Keys.D9)
        {
            if (e.KeyCode < Keys.NumPad0 || e.KeyCode > Keys.NumPad9)
            {
                if (e.KeyCode != Keys.Back)
                {
                    nonNumberEntered = true;
                }
            }
        }
        if (Control.ModifierKeys == Keys.Shift)
        {
            nonNumberEntered = true;
        }
    }
    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (nonNumberEntered == true)
        {
            e.Handled = true;
        }
    }
```

KeyPress – boshqaruv elementi fokusda bo‘lganida foydalanuvchi klaviatura tugmasi bosib-qo‘yib yuborganida sodir bo‘ladi.

KeyUp – bosilgan klaviatura tugmasi qo‘yib yuborilganida sodir bo‘ladi.

```
private void Form1_KeyUp(object sender, KeyEventArgs e)
    {
        label1.Text = e.KeyCode.ToString();
    }
```



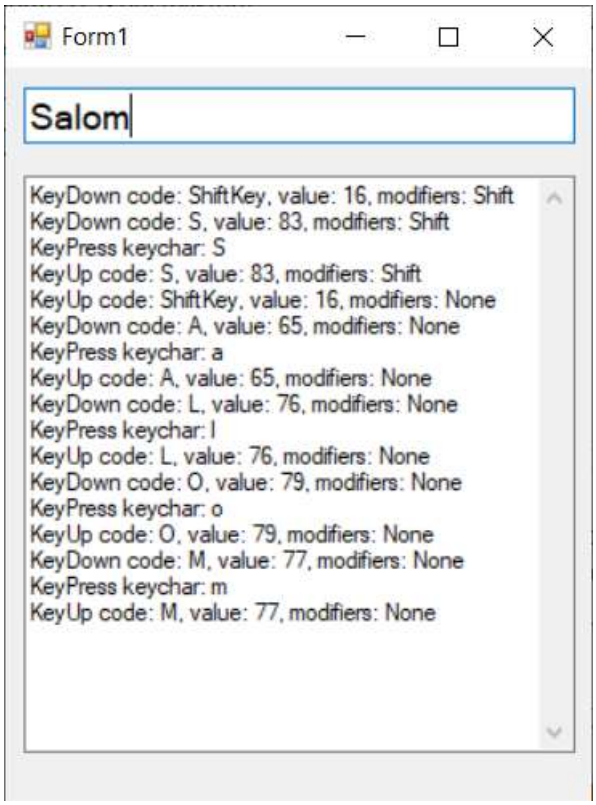

```
public Form1()
    {
```

```
    InitializeComponent();
    textBox2.Multiline = true;
    textBox2.ScrollBars = ScrollBars.Both;

    textBox1.KeyDown += TextBox1_KeyDown;
    textBox1.KeyPress += TextBox1_KeyPress;
    textBox1.KeyUp += TextBox1_KeyUp;
}
private void TextBox1_KeyUp(object sender, KeyEventArgs e)
{
    textBox2.AppendText($"KeyUp code: {e.KeyCode}, value: {e.KeyValue},
    modifiers: {e.Modifiers}" + "\r\n");
}

private void TextBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    textBox2.AppendText($"KeyPress keychar: {e.KeyChar}" + "\r\n");
}

private void TextBox1_KeyDown(object sender, KeyEventArgs e)
{
    textBox2.AppendText($"KeyDown code: {e.KeyCode}, value: {e.KeyValue},
    modifiers: {e.Modifiers}" + "\r\n");
}
```



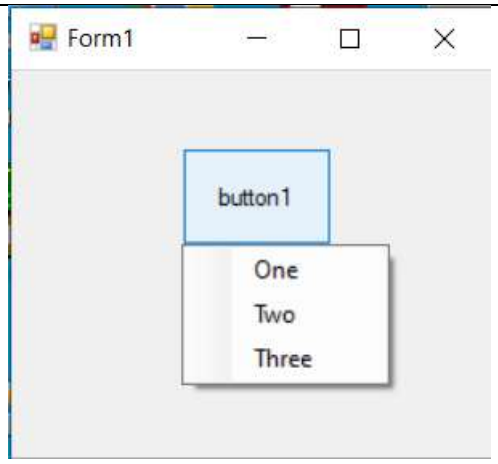
PreviewKeyDown – agar focus joriy boshqaruv elementida bo‘lsa, klaviatura tugmasi bosilganida KeyDown hodisasidan oldin sodir bo‘ladi.

```
public Form1()
```

```
    {
        InitializeComponent();
        button1.PreviewKeyDown += new PreviewKeyDownEventHandler
        (button1_PreviewKeyDown);
        button1.KeyDown += new KeyEventHandler(button1_KeyDown);
        button1.ContextMenuStrip = new ContextMenuStrip();
        button1.ContextMenuStrip.Items.Add("One");
        button1.ContextMenuStrip.Items.Add("Two");
        button1.ContextMenuStrip.Items.Add("Three");
    }

private void button1_KeyDown(object sender, KeyEventArgs e)
    {
        switch (e.KeyCode)
        {
            case Keys.Down:
            case Keys.Up:
                if (button1.ContextMenuStrip != null)
                {
                    button1.ContextMenuStrip.Show(button1, new Point(0,
                    button1.Height), ToolStripDropDownDirection.BelowRight);
                }
                break;
        }
    }

private void button1_PreviewKeyDown(object sender, PreviewKeyDownEventArgs e)
    {
        switch (e.KeyCode)
        {
            case Keys.Down:
            case Keys.Up:
                e.IsInputKey = true;
                break;
        }
    }
```



Nazorat savollari

1. Hodisa nima?
2. Windows Forms da hodisalar qanday ishlanadi?
3. Windows Forms standart hodisalariga misollar keltiring.
4. Sichqoncha hodisalarini tavsiflang.
5. Klaviatura hodisalariga misollar keltiring

Foydalanish uchun tavsiya etiladigan adabiyotlar

1. Троелсен Эндрю, Джепикс Филипп. Язык программирования C# 7 и платформы .NET и .NET Core. Вильямс. 2018
2. Албахари Бен, Албахари Джозеф. C# 7.0. Справочник. Полное описание языка. Пер. с англ.-СПб: “Альфа-книга”, 2018, -1024 с.
3. Ю.С. Магда C#. Язык программирования Си Шарп. – Изд. ДМК Пресс, 2013, 190 с.
4. Лабор В.В. C#: Создание приложение для Windows. – Мн.: Харвест, 2003, 384 с.
5. https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.click?view=windowsdesktop-5.0