



Kırklareli Üniversitesi  
Lüleburgaz Meslek Yüksekokulu

## Görsel Programlama-II

### Hafta-5 : Formlar, Özellikler ve Olaylar

Öğr. Gör. Özcan EKEN  
ozcan.eken@klu.edu.tr

## Formlar

Form nesneleri ve Form üzerinde kullandığımız diğer kontroller, System.Windows.Forms sınıfından türetilirler.

Windows Form uygulamalarımızda kontrollerimizi form üzerine yerleştirerek kullanırız. Form dosyaları disk üzerinde cs uzantısı ile kaydedilirler. Bir ya da birden fazla form kullanmamız mümkündür.

Projemize yeni bir form eklemek için **Solution Explorer** panelinde proje adının üzerinde farenin sağ tuşuna bastığımız açılan menüden **Add> Windows Form** ya da **Project** menüsünden **Add> Windows Form** seçeneği ile yeni bir form ekleyebiliriz.

Açılan formlar arası geçişlerde **Show** (formu açmak için), **ShowDialog** (formu Dialog Penceresi olarak açmak için), **Hide** (formu gizlemek için) ve **Close** (formu kapatmak için) metotları kullanılabilir.

Formlar arası geçişlerde başlangıç formu Close metodu ile kapatılamaz, Başlangıç formunda Close metodu programı sonlandırma görevini gerçekleştirir.

## Formlar

Birden fazla form olan projelerde program çalıştığında ilk sırada yer alan başlangıç formu varsayılan olarak Form1'dir.

Diğer formlardan birini başlangıç formu olarak seçmek için Solution Explorer panelinde bulunan **program.cs** dosyasını açarak;

```
Application.Run(new Form1());
```

satırındaki Form1 yerine başlangıç formu istenilen form ismi yazılarak başlangıç formu seçilebilir.

Bir formun içerisinde başka bir formda bulunan nesneye erişim sağlayabilmek için Solution Explorer panelinde nesnenin bulunduğu formun **Form2Designer.cs** dosyasını açılarak;

```
private System.Windows.Forms.Label label1;
```

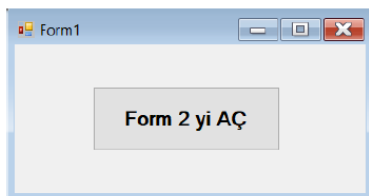
Erişim tipi **private** olarak tanımlanmış nesnenin erişim tipi **public** olarak değiştirilir.

```
public System.Windows.Forms.Label label1;
```

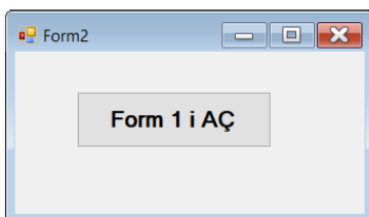
## Örnek (Show-Hide-Close)

Form1'e bir button nesnesi ekleyiniz, Project Menüünden **Add Form (Windows Form)** komutunu seçerek Form2'yi projenize ekleyiniz ve Form2'ye bir button nesnesi ekleyiniz. İlgili buttonların içerisine aşağıdaki kodları yazarak Formlar arası geçiş yapınız.

Programdaki Hide ve Close metodlarını değiştirerek programı tekrar çalıştırınız ve gözlemleyiniz.



```
private void btn_form1deki_Click(object sender, EventArgs e)
{
    Form2 frm2 = new Form2();
    frm2.Show();
    this.Hide();
}
```



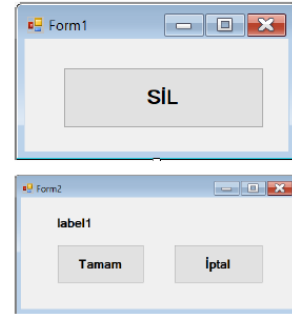
```
private void btn_form2deki_Click(object sender, EventArgs e)
{
    Form1 frm1 = new Form1();
    frm1.Show();
    this.Close();
}
```

## Örnek(ShowDialog)

Form1'deki Sil butonundan Form2'de yer alan label1 nesnesine erişim sağlayabilmek için **Form2.Designer.cs** dosyasından label1'in erişim tipini **public** olarak değiştirmeyi unutmayınız.

```
public System.Windows.Forms.Label label1;
```

```
private void sil_Click(object sender, EventArgs e)
{
    Form2 frm = new Form2();
    frm.label1.Text = "Silmek İstediginize Emin Misiniz?";
    frm.ShowDialog();
    if (frm.DialogResult == DialogResult.OK)
    {
        MessageBox.Show("Dosyalar Silindi!");
    }
}
```



```
private void iptal_Click(object sender, EventArgs e)
{
    this.DialogResult = DialogResult.Cancel;
    this.Close();
}
```

```
private void ok_Click(object sender, EventArgs e)
{
    this.DialogResult = DialogResult.OK;
    this.Close();
}
```

## SDI Form ve MDI Form

Windows Form Uygulamalarımızda bir ya da birden fazla form kullanmamız mümkündür. Form uygulamalarımızı SDI ve MDI olmak üzere iki ana biçimde gerçekleştirebiliriz.

SDI Formlar, şu ana kadar kullandığımız içerisinde bir adet form penceresi olan form yapısıdır. Birden fazla formda kullanılabilir.

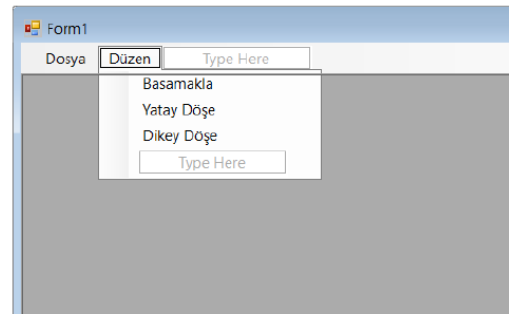
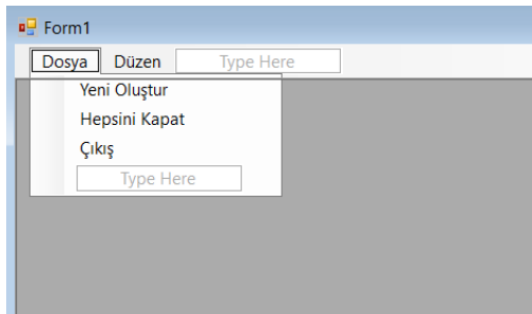
MDI Formlar, ise içerisinde birden fazla formu tek pencerede kullanılabilen form yapısıdır. MDI formuna parent(ebeveyn) form, içerisindeki formlara ise child(çocuk) form adı verilir.

Bir formu MDI Form yapmak için Properties panelinden **IsMdiContainer** özelliğini **True** olarak ayarlanmalıdır.

## Örnek (MDI Form)

Windows Form Application projesi açarak Form'u MdiForm olarak ayarlamak için IsMdiContainer özelliğini True olarak ayarlayınız. Form1 MdiForm olduğunda koyu gri ye dönecektir.

Form'a MenuStrip nesnesi ekleyerek Type Here yazan alanlara resimdeki bilgileri yazarak Dosya ve Düzen menülerini tasarlayınız.



## Örnek (MDI Form)

```
byte sayi;  
1 reference  
private void yeniOluşturToolStripMenuItem_Click(object sender, EventArgs e)  
{  
    Form yeniiform = new Form();  
    yeniiform.MdiParent = this;  
    sayi += 1;  
    yeniiform.Text = "Form " + sayi;  
    yeniiform.Show();  
}  
  
private void hepsiniKapatToolStripMenuItem_Click(object sender, EventArgs e)  
{  
    foreach (Form cocukForm in this.MdiChildren)  
    {  
        cocukForm.Close();  
    }  
}  
  
1 reference  
private void çıkışToolStripMenuItem_Click(object sender, EventArgs e)  
{  
    this.Close(); //Application.Exit();  
}
```

## Örnek (MDI Form)

```
private void basamaklaToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.Cascade);
}
```

1 reference

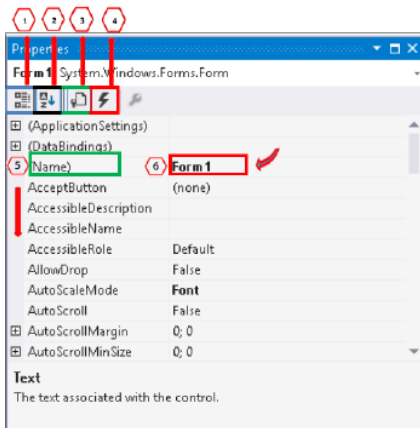
```
private void yatayDöşeToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileHorizontal);
}
```

1 reference

```
private void dikeyDöşeToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.LayoutMdi(MdiLayout.TileVertical);
}
```

## Özellikler (Properties)

Özellikler, form ya da kontrollerin görünümü, yerleşimi veya davranışlarına özel niteliklerdir. Örneğin bir formun Text özelliği, Başlığında yazan metine erişmemizi sağlar. Formların özelliklerine, tasarım anında Properties panelinden ulaşılabileceği gibi ayrıca kod tarafında da okunup değiştirilebilir.



1. **Kategori Mod:** Seçili form ya da kontrollerin özelliklerini gruplar hâlinde gösterilmesini sağlar.
2. **Alfabetik Sırala:** Nesneye ait özelliklerin alfabetik sıralanmasını sağlar.
3. **Özellikler modu:** Panelde seçili nesnenin özelliklerini gösterir.
4. **Olaylar modu:** Panelde seçili nesneye ait olaylar gösterilir.
5. **Özellik adı:** Seçili nesnelerin özellik adını gösterir.
6. **Seçenekler:** Özellikte bulunan seçenekleri ya da metin alanlarını gösterir.

## Özellikler (Properties)

**Name:** Bu özellik Form'un ismini değiştirmeye yarar. Kullanıcı sonradan verilen Form isminin değişmesi durumunda yeni ismin bütün kodlarda olacak şekilde yeniden değiştirilmesi gerekir. Name ve Text sıklıkla karıştırılır. Text, nesnenin adres çubuğunda görünecek metni ayarlarken Name ise nesneye isim verir.

Program kodları içinde Name ile verilen isim kullanıldığı için genellikle default değerlerin olmasına dikkat edilir.

**AcceptButton:** Bu özellik form ekranında bulunan "Enter" Tuşuna basıldığında hangi buton ayarlanmışsa o buton tıklanmış olur. Alabileceği değerler: Button sınıfından oluşturulur.

**AutoScaleMode:** Bu özellik Form ekranını kullanıcı elle genişlettiğinde hangi değer de aynı oranda genişlemesi gerektiğini ayarlayan bir özelliktir. Özellikte Default olarak Font seçilidir. Yani siz ekranı genişlettiğinizde yazı fontları da aynı oranda genişleyecektir.

## Özellikler (Properties)

**AutoScroll:** Kontroller formun üzerine sığmayınca kaydırma çubuğunun görünürlüğünü kontrol eder. Alabileceği değer true veya false olur.

**AutoSize:** Bu özellik Formun içindeki nesnelere göre kendi boyutunu otomatik olarak ayarlamasını sağlar. Alabileceği değer true veya false olur.

**BackColor:** Formun arka plan rengini değiştirir. Alabileceği değerler SystemColor, WebColors veya Color Sınıfından oluşturulur.

**BackgroundImage:** Bu özellik Form'a bir arka plan resmi yüklenmesini sağlar. Alabileceği değerler Image sınıfından oluşturulur.

**BackgroundImageLayout:** Bu özellik Form ekranına eklenen arka plan resminin boyutlandırmasının nasıl yapılacağını ayarlar.

- None (Hiçbiri)
- Stretch (Tam Boyut Yap)
- Tile (Döşe)
- Zoom (Yakınlaştır)
- Center (Ortala)



## Özellikler (Properties)

**CancelButton:** Bu özellik sayesinde "Esc" Tuşuna bastığınızda hangi buton ayarlanmışsa O buton tıklanmış olur. Alabileceği değerler: Button sınıfından oluşturulur.

**ContextMenuStrip:** Form ekranına birden fazla ContextMenuStrip(Sağ Tık) Menüsü eklenebilir. Bu özellik o Formda sağ tıklanınca hangi ContextMenuStrip'in aktif edileceğini seçmenizi sağlar. Alabileceği değerler true ya da false olur.

**ControlBox:** Bu özellik Formun sağ üst köşesindeki Kapat-Küçült-Büyüt butonlarının kapatılıp açılabilmesini sağlar. Alabileceği değerler true ya da false olur.

**Cursor:** Bu özellik Formun üzerine geldiğinde Mouse nin simgesinin değişmesini sağlar. Alabileceği değerler Cursor sınıfından oluşturulan nesnelerdir.

**Enabled:** Bu özellik Form Ekranını ve içindeki bileşenlerin tıklanma değiştirilme gibi özelliklerini kapatır. Alabileceği değerler true ya da false olur. False olması durumunda kullanıcı Form Ekranını kullanamaz ve müdahale edemez.

## Özellikler (Properties)

**Font:** Bu özellik Form alanına eklenen her nesnenin Yazı Tipini otomatik olarak seçilen fonta dönüştürür. Alabileceği değerler Font sınıfından oluşturulur.

**ForeColor:** Bu özellik Form ekranına eklenen her nesnenin Yazı Rengini otomatik olarak burada seçilen renge dönüştürür. Alabileceği değerler Color sınıfından oluşturulur.

**FormBorderStyle:** Bu özellik sayesinde Formun kenarlıkları ve davranışları ayarlayabilir. Alabileceği değerler:

- None olunca formun kenarlık ve başlığı oluşmaz.
- Fixed'li bir değer olunca form çalışma anında büyütülemez veya küçültülemez.
- Sizable'li değerler çalışma anında ebatları değiştirilebilir.

**HelpButton:** Bu özellik sayesinde başlığın bulunduğu çubukta Yardım butonunun gösterilip gösterilmeyeceğini ayarlayabilirsiniz. Alabileceği değerler true ya da false olur. Bu butonun form üzerine eklenebilmesi MaximizeBox ve MinimizeBox butonlarının false olmasına bağlıdır.

## Özellikler (Properties)

**Icon:** Bu özellik sayesinde Form ekranına bir icon ekleyebilmeyi sağlar. Bu icon Formun başlığının sol tarafında gözükecektir. Alabileceği değerler Icon sınıfından oluşturulur.

**Is MDI Container:** Bu özellik sayesinde mevcut Form ekranınızın içine Form eklenebilmesine izin verilir. Alabileceği değerler true ya da false olur.

**Location:** Bu özellik Formun ekranın sol üst köşesinden itibaren ekranın neresinde açılacağını belirler. Alabileceği değer Point sınıfından oluşturulur. Point sınıfı X ve Y olmak üzere iki bileşene sahiptir.

**Locked:** Bu özellik Form ekranını ve içindeki elemanları kitleyerek kullanıcının müdahale etmesini engeller. Alabileceği değerler true ya da false olur.

**MainMenuStrip:** Bu özellik Form ekranına eklenen menustriplerden hangisinin ana menustrip olarak ayarlanacağını seçmesini sağlar. Alabileceği değerler eklenen menüstriplerden biri seçilir.

## Özellikler (Properties)

**MaximizeBox:** Formun Sağ üst köşesinde görülen büyültme düğmesinin görünüp görünmeyeceğini ayarlar. Alabileceği değerler true ya da false olur.

**MaximumSize:** Bu özellik Formun alabileceği maksimum büyüklüğün ayarlanmasını sağlar. Kullanıcı Form ekranını ne kadar büyültmek isterse istesin bu bölümden verilen değerden büyük bir değer alamaz. Alabileceği değerler Size sınıfından oluşturulur.

**MinimizeBox:** Bu özellik Formun sağ üst köşesinde yer alan Form ekranını alta almaya yarayan düğmenin gösterilip gösterilmeyeceğini ayarlar. Alabileceği değerler true ya da false olur.

**MinimumSize:** Formun alabileceği en küçük boyutu gösterir. Kullanıcı ne kadar Form ekranını küçültmeye çalışırsa çalışsın Form ekranı bu bölümden verilen boyuttan küçük olamaz. Alabileceği değerler Size sınıfından oluşturulur.

**Opacity:** Bu özellik Formun saydamlığını ayarlar. Alabileceği değerler %100 tam görünür, %0 görünmez anlamına gelir.



## Özellikler (Properties)

**ShowIcon:** Form ekranına eklenen icon(simge) nin gösterilip gösterilmeyeceğini ayarlar. Alabileceği değerler true ya da false olur.

**ShowInTaskBar:** Yaptığımız uygulamanın görev çubuğunda gözükp gözükmeyeceğini ayarlamamızı sağlar. Alabileceği değerler true ya da false olur.

**Size:** Bu özellik Formun boyutlarını ayarlamamızı sağlar. Alabileceği değer Size sınıfının Width(genişlik) ve Height(yükseklik) şeklinde iki değerden oluşur.

**StartPosition:** Formun ekranın neresinde açılacağını ayarlamayı sağlar. Alabileceği değerler:

- CenterParent: Form kendisini başlatan ana formun merkezinde gözüktür.
- CenterScreen: Form ekranın ortasında gözüktür.
- Manual: Form Location ile belirtilen noktada gözüktür.
- WindowsDefaultBounds ve WindowsDefaultLocation değerlerinde işlemler sisteme bırakılır.

## Özellikler (Properties)

**Text:** Bu özellik Formun başlığını ayarlamayı sağlar. Alabileceği değer String sınıfından oluşturulur.

**TopMost:** Bu özellik Form ekranı açıldığında başka Form ekranı açık ise diğer tüm Form ekranlarının üzerinde görünüp görünmeyeceğini ayarlar. Son açılan Form ekranı kapatılmadan diğer Form ekranı görünmez. Alabileceği değerler true ya da false olur.

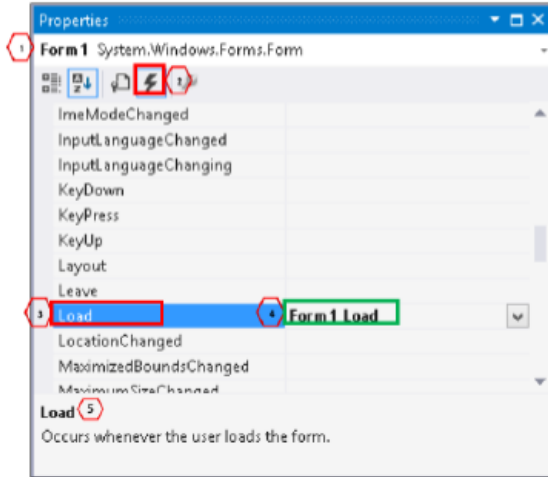
**WindowState:** Bu özellik Form ekranımızın açılışta Tam ekran, Normal Ekran, Küçük Ekran gibi görünümlerden hangileri ile açılacağını belirlemeyi sağlar.

Alabileceği değerler:


- Maximized durumunda form ekranı kaplamış bir şekilde başlar.
- Minimized durumunda form simge durumunda başlar.
- Normal durumunda varsayılan büyüklük ile başlar.

## Olaylar (Events)

Olay(Event), görsel programlama dillerinde bir olay oluşturulduktan sonra çalıştırılacak metotları tetikleyen bir yapıdır. Olaylar gerçekleştiği zaman yapılması gereken işlemler, ilgili olayın yordamına yazılır. Form1 penceresi yüklenirken gerçekleştirmek istenen eylemler genelde Form1\_Load yordamına yazılır.



1. **Kontrol Adı:** Seçili Form ya da kontrol adını gösterir.
2. **Olaylar modu:** Panelde seçili nesneye ait olayların gösterilmesini sağlar.
3. **Olay Türü:** Form ya da kontrole ait olay türünün seçimi yapılır.
4. **Olayın yordam Adı:** Olaylar gerçekleştiğinde çalıştırılacak metot ismidir.
5. **Aktif seçilmiş metod açıklaması:** Seçili olay hakkında bilgi verir.

Olaylara geçmek için properties penceresinden (  ) iconu tıklanır. Oluşturulmak istenen e Olayın adının üzerine çift tıklanır. Çift tıklandıktan sonra kod bölümünde aşağıdaki gibi otomatik olarak çalışacak Olay metodu oluşturulur.

```
private void Form1_Load(object sender, EventArgs e)
{
    // Form1 yüklenirken çalıştırılması istenen
    // kodları buraya yazınız
}
```

## Olaylar (Events)

Windows Formları ve Kontrollerin çok sayıda olayı vardır. Bu olaylardan bazıları uygulamada çok sık kullanılır. Bunlar aşağıda liste halinde gösterilmektedir.

**BackColorChanged:** Formun BackColor özelliği değiştiği zaman devreye girer.

```
private void Form1_BackColorChanged(object sender, EventArgs e)
{
}
```

**BackgroundImageChanged:** Formun BackgroundImage özelliği değiştiğinde devreye girer.

```
private void Form1_BackgroundImageChanged(object sender, EventArgs e)
{
}
```

**Click:** Forma tıklanınca devreye girer.

```
private void Form1_Click(object sender, EventArgs e)
{
}
```

## Olaylar (Events)

*ControlAdded:* Formun üzerine yeni bir kontrol eklenince devreye girer.

```
private void Form1_ControlAdded(object sender, ControlEventArgs e)
{
}
}
```

*ControlRemoved:* Formun üzerindeki kontrollerden biri silinince devreye girer.

```
private void Form1_ControlRemoved(object sender, ControlEventArgs e)
{
}
}
```

*CursorChanged:* Formun Cursor özelliği değişince devreye girer.

```
private void Form1_CursorChanged(object sender, EventArgs e)
{
}
}
```

## Olaylar (Events)

*DoubleClick:* Form çift tıklanıldığında devreye girer.

```
private void Form1_DoubleClick(object sender, EventArgs e)
{
}
}
```

*DragDrop:* Formun üzerine bir şey sürüklenip bırakılınca devreye girer.

```
private void Form1_DragDrop(object sender, DragEventArgs e)
{
}
}
```

*DragEnter:* Formun üzerine bir şey sürüklerken Formun sınırı geçilip içeriye girerken devreye girer.

```
private void Form1_DragEnter(object sender, DragEventArgs e)
{
}
}
```

## Olaylar (Events)

*DragLeave:* Formdan dışarıya bir şey sürüklerken Formun sınırı geçilip dışarıya çıkarken devreye girer.

```
private void Form1_DragLeave(object sender, EventArgs e)
{
}
```

*DragOver:* Formun üzerine bir şey sürüklenip henüz bırakılmamışken devrededir.

```
private void Form1_DragOver(object sender, DragEventArgs e)
{
}
```

*FormClosed:* Form kapandıktan sonra gerçekleşir.

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
}
```

## Olaylar (Events)

*FormClosing:* Form kapanmadan hemen önce gerçekleşir.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
}
```

*KeyDown:* Form aktifken bir tuşa basılınca devreye girer.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
}
```

*KeyPress:* Form aktifken bir tuşa basılıp bırakılınca devreye girer.

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
}
```

## Olaylar (Events)

*KeyUp:* Form aktifken bir tuşa basılıp bırakılınca devreye girer.

```
private void Form1_KeyUp(object sender, KeyEventArgs e)
{
}
}
```

*Layout:* Form üzerine nesneleri yerleştirirken devreye girer.

```
private void Form1_Layout(object sender, LayoutEventArgs e)
{
}
}
```

*Load:* Form ekrana yüklenirken devreye girer.

```
private void Form1_Load(object sender, EventArgs e)
{
}
}
```

## Olaylar (Events)

*LocationChanged:* Formun yeri değişince devreye girer.

```
private void Form1_LocationChanged(object sender, EventArgs e)
{
}
}
```

*MouseClick:* Kontrolle Mouse ile tıklanınca devreye girer.

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
}
}
```

*MouseDoubleClick:* Kontrolle Mouse ile çift tıklanınca devreye girer.

```
private void Form1_MouseDoubleClick(object sender, MouseEventArgs e)
{
}
}
```



## Olaylar (Events)

*MouseDown:* Kontrolle Mouse ile tıklanınca Mouse butonu henüz inerken devreye girer.

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
}
```

*MouseEnter:* Formun üzerine Mouse ile girince sınırını geçip içeri girince devreye girer.

```
private void Form1_MouseEnter(object sender, EventArgs e)
{
}
```

*MouseLeave:* Formun üzerine Mouse ile girilmişken sınırını geçip dışarı çıkınca devreye girer.

```
private void Form1_MouseLeave(object sender, EventArgs e)
{
}
```

## Olaylar (Events)

*MouseMove:* Formun üzerinde her Mouse hareketinde devreye girer.

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
}
```

*MouseUp:* Forma Mouse ile tıklandıktan sonra butonu geri bırakınca devreye girer.

```
private void Form1_MouseUp(object sender, MouseEventArgs e)
{
}
```

*Move:* Formun yerini değiştirince (hareket ederken) devreye girer.

```
private void Form1_Move(object sender, EventArgs e)
{
}
```

## Olaylar (Events)

*Paint:* Form ve üzerindeki controllerler ekrana çizilip yerleştiği zaman devreye girer.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
}
}
```

*Resize:* Formun büyüklüğü değiştirilince devreye girer.

```
private void Form1_Resize(object sender, EventArgs e)
{
}
}
```

*SizeChanged:* Formun Size özelliği değişince devreye girer.

```
private void Form1_SizeChanged(object sender, EventArgs e)
{
}
}
```

## Olaylar (Events)

*TextChanged:* Formun Text özelliği değişince devreye girer.

```
private void Form1_TextChanged(object sender, EventArgs e)
{
}
}
```

*VisibleChanged:* Formun Visible özelliği (Ekranda görünürlük) değişince devreye girer.

```
private void Form1_VisibleChanged(object sender, EventArgs e)
{
}
}
```

## Örnek (Events)

Windows Form Application projesi açarak Form'un FormClosing, KeyDown, KeyPress olaylarına aşağıdaki ilgili kodları yazınız.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (MessageBox.Show("Çıkmak istediğinize emin misiniz?", "ÇIKIŞ", MessageBoxButtons.YesNo) == DialogResult.No)
    {
        e.Cancel = true;
    }
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Control && e.KeyCode == Keys.Q)
    {
        MessageBox.Show("Ctrl+Q kısayolu ile Programı Kapatıyorsunuz!!!");
        this.Close();
    }
}

byte k;
1 reference
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    k++;
    if (e.KeyChar==13) // enter e basıldığında
    {
        MessageBox.Show(k +" defa tuşa basıldı.");
    }
}
```