

Label Kontrolü

İçeriği sabit kalıp değiştirilmeyecek bilgilerin görüntülenmesinde kullanılır. Text özelliği ile görüntülenecek metin belirlenir.

Örnek :

```
private void Form1_Load(object sender, EventArgs e)
{
    Label label1 = new Label();
    this.Controls.Add(label1);
    label1.Text = "Lüleburgaz Meslek Yüksekokulu\nBilgisayar Programcılığı Programı";
    label1.Font = new Font("Comic Sans MS", 15, FontStyle.Italic);
    label1.ForeColor = Color.DarkBlue;
    label1.AutoSize = true;
}
```



LinkLabel Kontrolü

Tıklandığında herhangi bir web adresi yada mail adresini açan label nesnesidir. **LinkLabel** nesnesinin **Links** özelliğine **Add** metodu ile web adresi girilerek hedef site yada mail adresi belirlenir.

```
private void Form1_Load(object sender, EventArgs e)
{
    linkLabel1.Links.Add(0, 24, "https://luleburgazmyo.klu.edu.tr");
    linkLabel2.Links.Add(0, 15, "lmyo@klu.edu.tr");
}
1 başvuru
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start(e.Link.LinkData.ToString());
    linkLabel1.LinkVisited = true;
}
1 başvuru
private void linkLabel2_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    System.Diagnostics.Process.Start("mailto:lmyo@klu.edu.tr?subject= Ders-1 Hakkında");
    linkLabel2.LinkVisited = true;
}
```



textBox Kontrolü

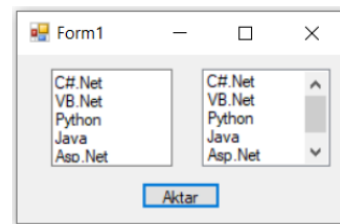
İçeriği değiştirilebilecek bilgilerin görüntülenmesi amacı ile kullanılır. Bilgi girişi amacıyla da kullanılır. Text özelliği ile görüntülenecek yada girişi yapılacak metin belirlenir.

- **MultiLine** özelliği True yapılarak bir satırdan daha uzun olan metinlerin daha fazla satırda görüntülenmesi sağlanabilir.
- **ScrollBars** özelliği ile textbox nesnemiz multiline modundayken kaydırma çubukları eklenebilir. Vertical -> Dikey , Horizontal -> Yatay , Both ise hem Dikey hem Yatay kaydırma çubuğu eklemek içindir.
- **PasswordChar** özelliği ile textbox içerisine girilen bilginin gizlenip başkaları tarafından görülmemesi amacıyla kullanılır. Örneğin buraya * karakteri girilirse yazılanlar * karakteri şeklinde görülür.
- **ReadOnly** özelliği False iken yazı girişi yapılabilir. Ancak True konumunda ise yazı girişi yapılamaz, textbox içerisindeki bilgiler yalnız okunabilir.
- **Lines** özelliği Textbox içerisine girilen satırların String() Dizi olarak saklanması amacıyla kullanılır. Textbox içerisine girilen satır sayısına Lines özelliğinin Length metodu ile ulaşılabilir.

Örnekler

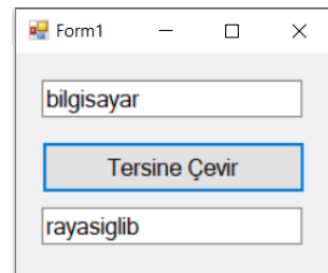
Button1'e basıldığında textBox1'e girilen her bir satırı, ListBox1'e eleman olarak aktaran program.

```
private void button1_Click(object sender, EventArgs e)
{
    int i;
    for(i=0;i<textBox1.Lines.Length;i++)
    {
        listBox1.Items.Add(textBox1.Lines[i]);
    }
}
```



textBox1'e girilen yazıyı tersine çeviren program.

```
private void button1_Click(object sender, EventArgs e)
{
    char[] dizi = textBox1.Text.ToCharArray();
    Array.Reverse(dizi);
    string tersi = new string(dizi);
    textBox2.Text = tersi;
}
```



textBox Kontrolünün Metodları

Cut : Seçili olan metni keserek taşınmak üzere hafızaya alır.

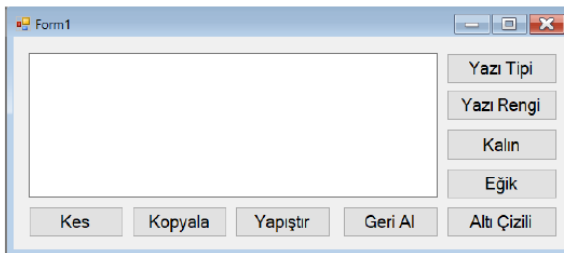
Copy: Seçili olan metni seçerek kopyalamak üzere hafızaya alır.

Paste: Hafızada olan metni yapıştırır.

Undo : Son yapılan işlemi geri alır.

Clear : Textbox içerisindeki yazıyı temizleyerek siler.

Dispose : Textbox nesnesini siler.



Örnek olarak bir textbox içerisinde temel işlemleri gerçekleştirelim. Projemize yazı rengini değiştirmek için bir adet ColorDialog nesnesi, yazı tipini değiştirmek içinde bir adet FontDialog nesnesi ekleyelim.

```
private void btn_kes_Click(object sender, EventArgs e)
{
    if (textBox1.SelectionLength > 0)
        textBox1.Cut();
    else
        MessageBox.Show("Seçili Alan Yok");
}

1 başvuru
private void btn_kopyala_Click(object sender, EventArgs e)
{
    if (textBox1.SelectionLength > 0)
        textBox1.Copy();
    else
        MessageBox.Show("Seçili Alan Yok");
}

1 başvuru
private void btn_yapistir_Click(object sender, EventArgs e)
{
    textBox1.Paste();
}

1 başvuru
private void btn_gerial_Click(object sender, EventArgs e)
{
    textBox1.Undo();
}
```

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

1 başvuru
private void btn_yazirengi_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
        textBox1.ForeColor = colorDialog1.Color;
}

1 başvuru
private void btn_kalin_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font(Font, FontStyle.Bold);
}

1 başvuru
private void btn_egik_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font(Font, FontStyle.Italic);
}

1 başvuru
private void btn_alticizili_Click(object sender, EventArgs e)
{
    textBox1.Font = new Font(Font, FontStyle.Underline);
}
```

button Kontrolü

Bir işlemi, olayı gerçekleştirmek amacı ile kullanılan nesnedir.

- **Image** buton nesnesinin üzerinde görüntülenecek olan resmi belirler.
- **TabStop** Buton nesnesinin TAB tuşu ile aktifleştirilebilmesini sağlar.
- **TabIndex** ise TAB tuşuna basıldığında nesnelerin hangi sıra ile aktifleşeceğini belirler. Bu özelliği nesnenin sıra numarası yazılır. TabIndex değerleri 0'dan başlamak üzere yazılır.

- **Click** bir defa tıklama olayıdır.
- **DoubleClick** çift tıklama olayıdır.
- **MouseMove** Farenin imleci buton nesnesinin üzerine geldiğinde gerçekleşen olaydır.
- **MouseLeave** Farenin imleci buton nesnesinin üzerinden ayrıldığında gerçekleşen olaydır.

```
Color renk;  
1 başvuru  
private void Form1_Load(object sender, EventArgs e)  
{  
    renk = button1.BackColor;  
}  
1 başvuru  
private void button1_MouseMove(object sender, MouseEventArgs e)  
{  
    button1.BackColor = Color.DarkGray;  
}  
1 başvuru  
private void button1_MouseLeave(object sender, EventArgs e)  
{  
    button1.BackColor = renk;  
}
```

comboBox Kontrolü

İçerisindeki değerleri açılabilen liste şeklinde görüntüleyen nesnedir. Comboboxiçerisindeki elemanlara index numaraları yardımı ile ulaşılabilir. İlk eleman 0 index numarası ile belirtilir.

ComboBox Kontrolünün Özellikleri

- **Items:** Combobox içerisinde değer listesini eklemek amacı ile kullanılır.
- **Text:** Comboboxta tıklanmadan önce combobox içerisinde görüntülenen metindir.
- **SelectedIndex :** Combobox içerisinde seçili olan elemanın index numarasını verir.
- **Sorted:** true değeri aldığında combobox içerisindeki elemanların sıralı şekilde listelenmesini sağlar.

comboBox Kontrolü

ComboBox Kontrolünün Metodları

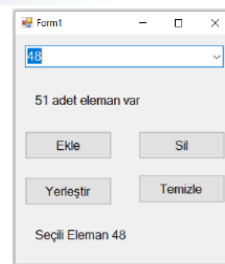
Items özelliği aşağıdaki metodları içerir.

- **Add** : Combobox içerisine belirtilen değeri listesinin sonuna eklemek amacı ile kullanılır.
- **Remove**: Combobox içerisindeki belirtilen değeri silmek amacı ile kullanılır.
- **RemoveAt** : Combobox içerisindeki belirtilen index numaralı değeri silmek amacı ile kullanılır.
- **Count** : Combobox içerisindeki değer sayısını verir.
- **Insert** : Combobox içerisine belirtilen değeri listenin istenilen bir yerine eklemek amacı ile kullanılır.
- **Clear** : Combobox içerisindeki değerlerin hepsini siler.
- **Contains** : Eğer belirtilen değer mevcutsa True sonucunu, mevcut değilse False değerini döndürür.
- **CopyTo** : Belirtilen elemanı bir dizi içerisine kopyalayarak aktarmak amacı ile kullanılır.

Örnek

```
private void btn_ekle_Click(object sender, EventArgs e)
{
    for(byte i=0;i<=100;i++)
    {
        if (i % 2 == 0)
            comboBox1.Items.Add(i);
    }
    label1.Text = comboBox1.Items.Count + " adet eleman var";
    comboBox1.SelectedIndex = 0;
}
```

```
1 bagvuru
private void btn_yerlestir_Click(object sender, EventArgs e)
{
    comboBox1.Items.Insert(0, -1);
    label1.Text = comboBox1.Items.Count + " adet eleman var";
    comboBox1.SelectedIndex = 0;
}
```



```
private void btn_temizle_Click(object sender, EventArgs e)
{
    comboBox1.Items.Clear();
    label1.Text = comboBox1.Items.Count + " adet eleman var";
}
```

```
1 bagvuru
private void btn_sil_Click(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex > 0)
        comboBox1.Items.RemoveAt(comboBox1.SelectedIndex);
    else
        comboBox1.Items.Remove(comboBox1.SelectedItem);
    label1.Text = comboBox1.Items.Count + " adet eleman var";
    label2.Text = "Seçili Eleman " + comboBox1.SelectedItem;
}
```

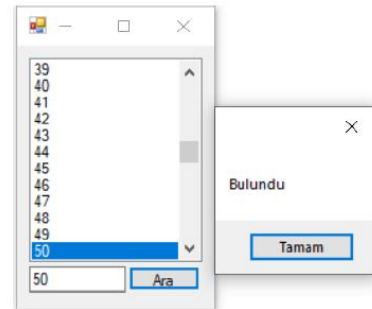
```
1 bagvuru
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label2.Text = "Seçili Eleman " + comboBox1.SelectedItem;
}
```

listBox Kontrolü

İçindeki değerleri liste şeklinde görüntüleyen nesnedir.

```
int i;  
Boolean bulundu = false;  
1 başvuru  
private void Form1_Load(object sender, EventArgs e)  
{  
    for(i=0;i<=100;i++)  
    { listBox1.Items.Add(i); }  
}  
  
1 başvuru  
private void btn_ara_Click(object sender, EventArgs e)  
{  
    for(i=0;i<=listBox1.Items.Count-1;i++)  
    {  
        if(textBox1.Text==listBox1.Items[i].ToString())  
        {  
            bulundu = true;  
            listBox1.SelectedIndex = Convert.ToInt32(listBox1.Items[i]);  
            break;  
        }  
    }  
    if (bulundu == true)  
        MessageBox.Show("Bulundu");  
    else  
        MessageBox.Show("Bulunamadı");  
    bulundu = false;  
}
```

listBox nesnesinin özellik, metod ve olayları comboBox ile hemen hemen aynıdır.



Drag and Drop Olayları

Bir nesnenin içerisindeki içeriği alıp başka bir nesnenin içerisine taşımaya drag and drop yada sürükle ve bırak denilir. Bir Listbox nesnesi içerisinde sürükle ve bırak olayını gerçekleştirebilmek için öncelikle AllowDrop özelliğinin True olarak ayarlanması gerekmektedir. Aksi takdirde sürükle ve bırak işlemi gerçekleşmez.

- **All** : Veri kopyalanır, sürükleme kaynağından silinir ve hedefe bırakılır.
- **Copy** : Sürükleme kaynağındaki veri kopyalanarak hedefe bırakılır.
- **Link** : Sürükleme kaynağındaki veri hedef nesneye bağlanır.
- **Move** : Sürükleme kaynağındaki veri taşınarak hedefe bırakılır.
- **None** : Hedef nesne sürüklenip bırakılan verileri kabul etmez.
- **Scroll** : Hedef nesnede kaydırma yapıldığını bildirir.

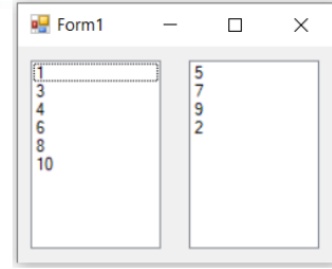
Örnek

```
private void Form1_Load(object sender, EventArgs e)
{
    for (byte i = 1; i <= 10; i++)
        listBox1.Items.Add(i);
}

1 başvuru
private void listBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (listBox1.Items.Count == 0) return;
    string deger = listBox1.Items[listBox1.IndexFromPoint(e.X, e.Y)].ToString();
    if (DoDragDrop(deger, DragDropEffects.All) == DragDropEffects.All)
        listBox1.Items.RemoveAt(listBox1.IndexFromPoint(e.X, e.Y));
}

1 başvuru
private void listBox2_DragDrop(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(DataFormats.StringFormat))
        listBox2.Items.Add(e.Data.GetData(DataFormats.StringFormat));
}

1 başvuru
private void listBox2_DragOver(object sender, DragEventArgs e)
{
    e.Effect = DragDropEffects.All;
}
```



IndexFromPoint metodunu kullanarak belirtilen X ve Y koordinatlarında listbox nesnesinin hangi index numaralı değeri olduğunu kontrolü yapılarak listbox'ın içindeki ilgili elemanı deger adlı değişkene aktarılır.

DoDragDrop metodu taşıma işleminin başlamasını sağlar.

DragDropEffects, taşınacak olan verinin nasıl taşınacağını belirler. Alabileceği değerler;

checkBox Kontrolü

İki ya da üç duruma sahip olabilen, onay ve seçim işlemleri için kullanılan bir nesnedir.

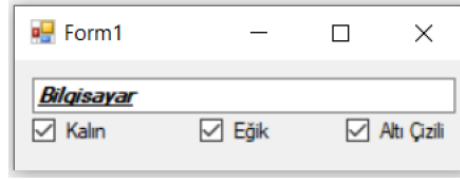
checkBox Kontrolünün Özellikleri

- **Checked** : True değerini aldığı anda checkBox nesnesi işaretlidir. False değerini aldığı anda ise işaretli değildir.
- **CheckState**: Üç adet değer alır. Unchecked(İşaretsiz), Checked(İşaretli), Indeterminate(Belirsiz)
- **ThreeState** : checkBox'ın Checked, Unchecked şeklinde iki değer değilde checkstate te belirtilen üç değeri alabilmesini sağlar.
- **FlatStyle** : checkBox nesnesinin görünüm şeklini değiştirir. Aldığı değerler Flat , Popup, Standart, System şeklindedir.

checkBox Kontrolünün Olayları

- **CheckedChange** : checkBox nesnesinin durumu değiştirildiğinde gerçekleşen olaydır

Örnek



```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    textBox1.Font = new Font(textBox1.Font.Name, textBox1.Font.Size, textBox1.Font.Style ^ FontStyle.Bold);
}

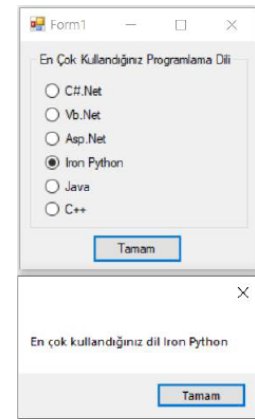
1 başvuru
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{
    textBox1.Font = new Font(textBox1.Font.Name, textBox1.Font.Size, textBox1.Font.Style ^ FontStyle.Italic);
}

1 başvuru
private void checkBox3_CheckedChanged(object sender, EventArgs e)
{
    textBox1.Font = new Font(textBox1.Font.Name, textBox1.Font.Size, textBox1.Font.Style ^ FontStyle.Underline);
}
```

radioButton Kontrolü

Birden fazla seçenektan yalnızca birini seçme durumlarında kullanılan nesnedir. Form üzerine aynı anda eklenen birden fazla radioButton içerisinde yalnızca biri seçili olabilir.

- **Checked Özelliği:** RadioButton nesnesinin seçili olup olmasını belirler.



```
private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked)
        MessageBox.Show("En çok kullandığınız dil C#.Net");
    else if (radioButton2.Checked)
        MessageBox.Show("En çok kullandığınız dil Vb.Net");
    else if (radioButton3.Checked)
        MessageBox.Show("En çok kullandığınız dil Asp.Net");
    else if (radioButton4.Checked)
        MessageBox.Show("En çok kullandığınız dil Iron Python");
    else if (radioButton5.Checked)
        MessageBox.Show("En çok kullandığınız dil Java");
    else if (radioButton6.Checked)
        MessageBox.Show("En çok kullandığınız dil C++");
}

private void button1_Click(object sender, EventArgs e)
{
    foreach (Control radio in groupBox1.Controls)
    {
        RadioButton rdb = (RadioButton)(radio);
        if (rdb.Checked)
            MessageBox.Show("En çok kullandığınız dil " + rdb.Text);
    }
}
```