

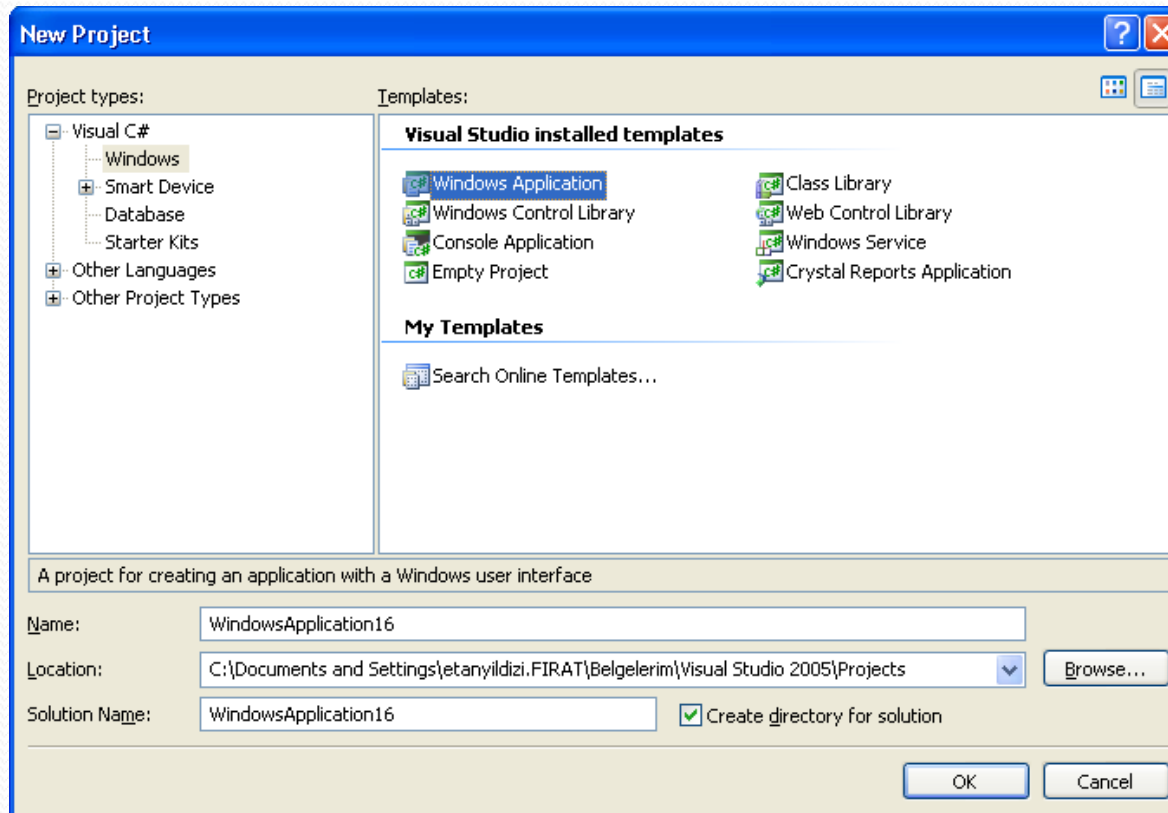
Visual Studio.Net -C#

9. HAFTA

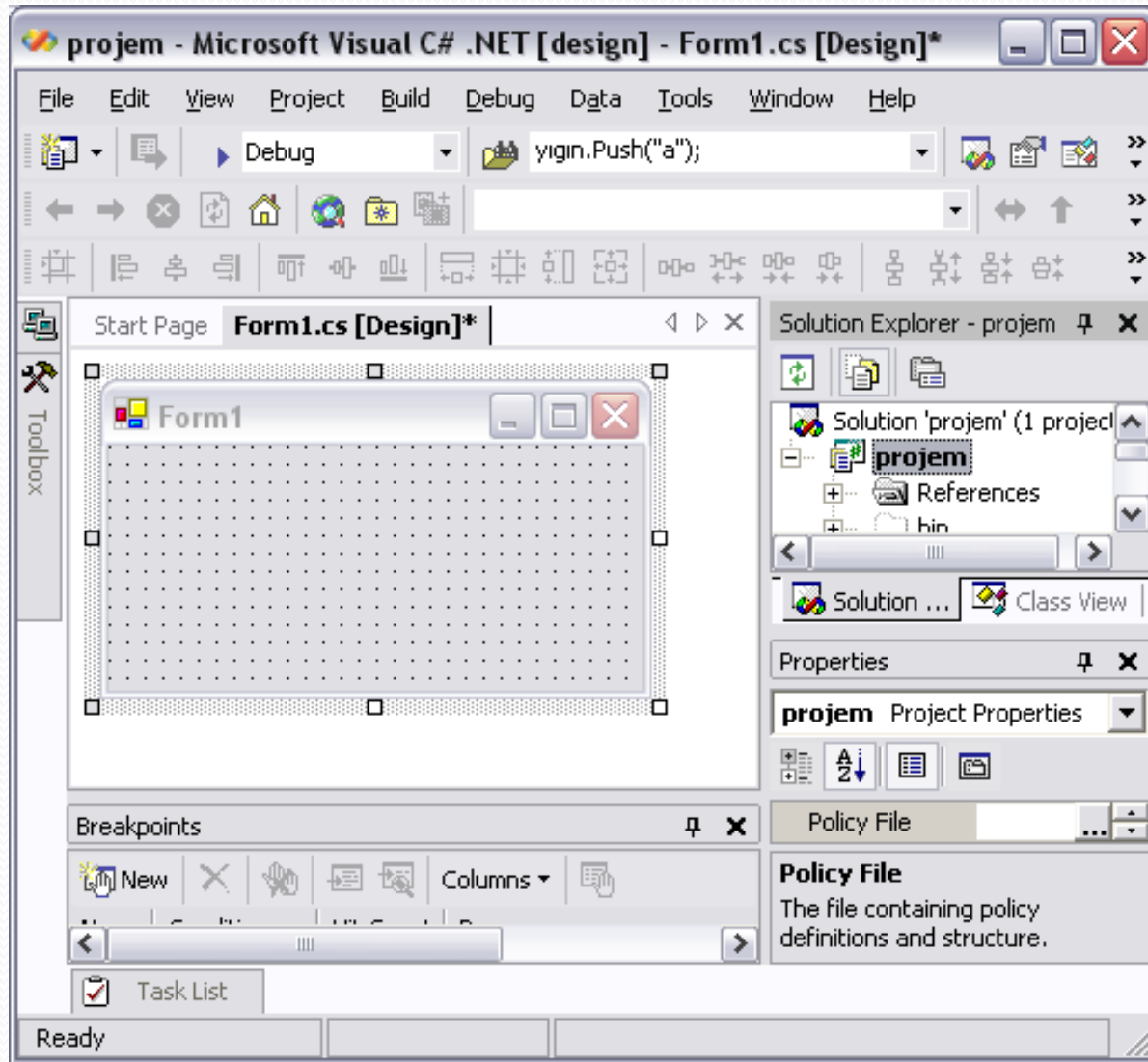
FORM UYGULAMALARI

Proje Başlatma

- **New Project: Yeni Proje Hazırlamak**
- **Open Project: Mevcut Projeleri Açmak**

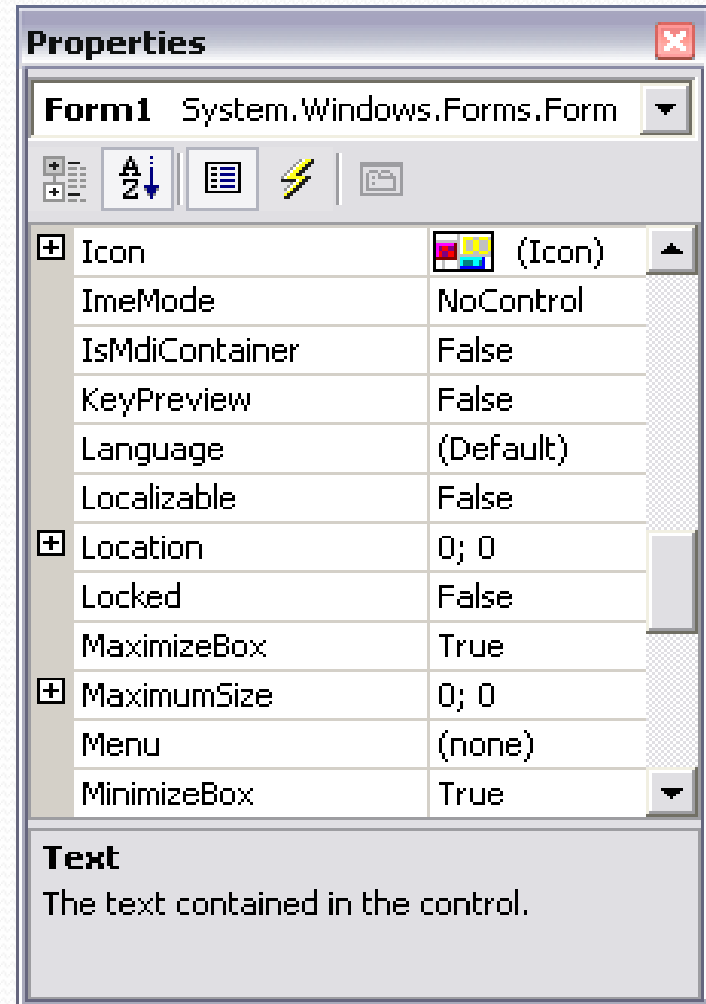


Proje Çalışma Ekranı



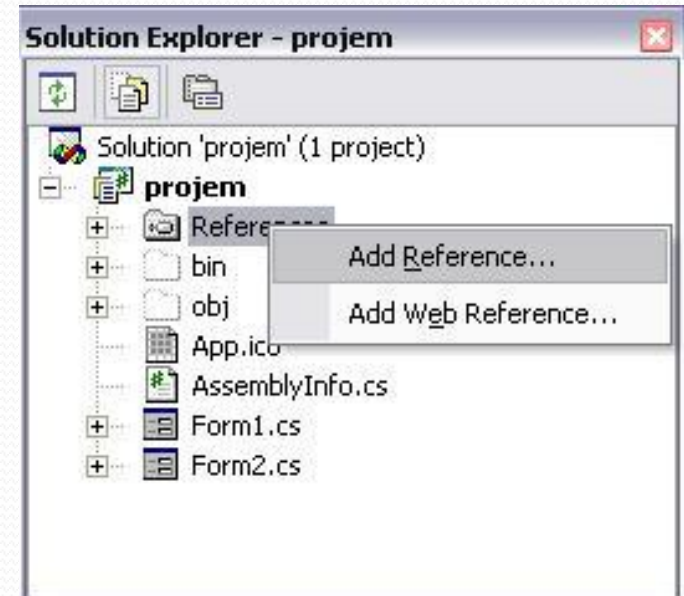
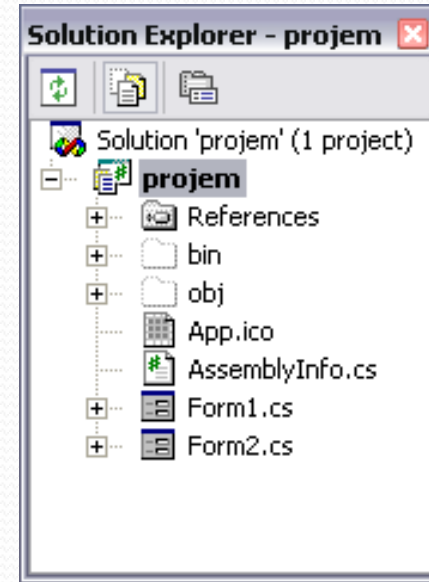
Proje Çalışma Ekranı

- **Properties Penceresi**
- Form ve diğer elamanlara ait özelliklerin belirlediği bölümdür.
- Properties penceresi eğer ekranda yoksa F4 fonksiyon tuşu ya da **View/ Properties Window** tıklanarak ekrana yerleştirilir.



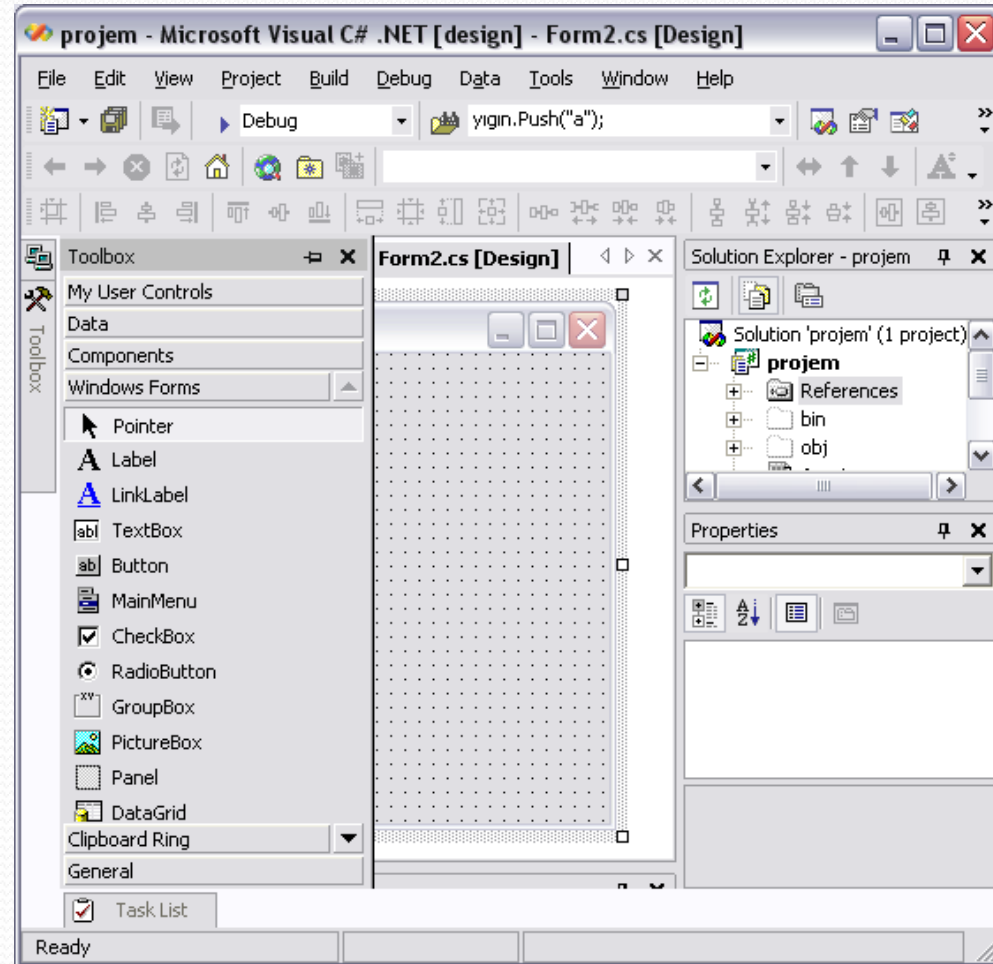
Proje Çalışma Ekranı

- **Solution Explorer Penceresi**
- Bu pencere proje içerisinde bulunan dosyalara erişim için kullanılır.
- Çalıştığımız projeye **ToolBox'** ta bulunmayan bir referans eklemek için Solution Explorer penceresinde **References** seçeneğine sağ tıklayarak **Add Reference** komutunu seçilmelidir.
- Solution Explorer penceresini görünür hale getirmek istersek Ctrl+Alt+L kısayol tuşunu ve ya View menüsünden Solution Explorer tıklanarak seçilir.



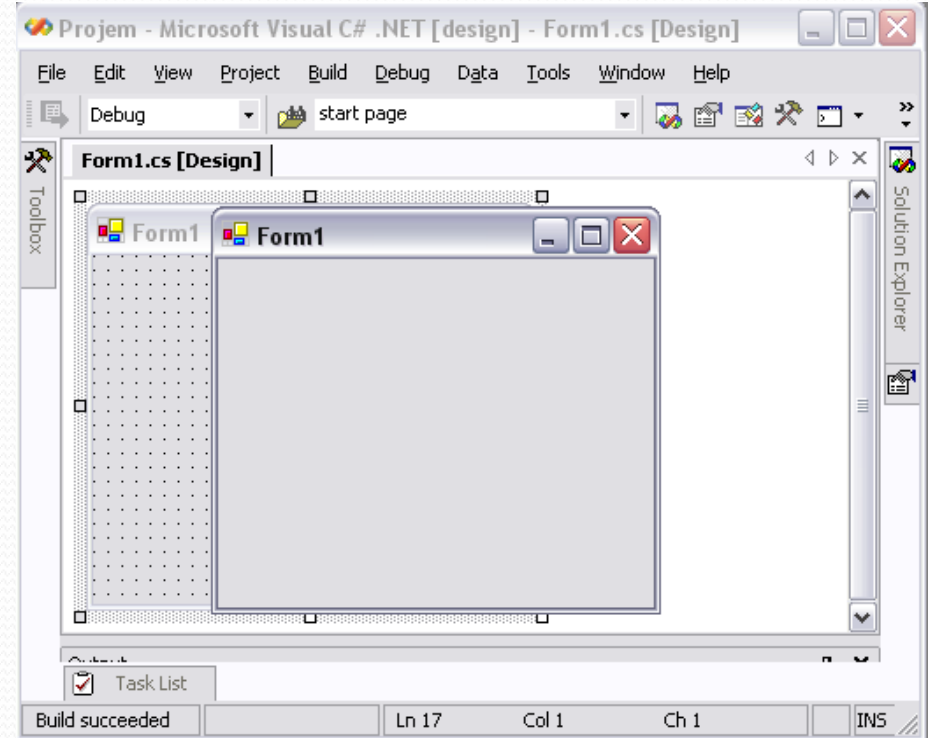
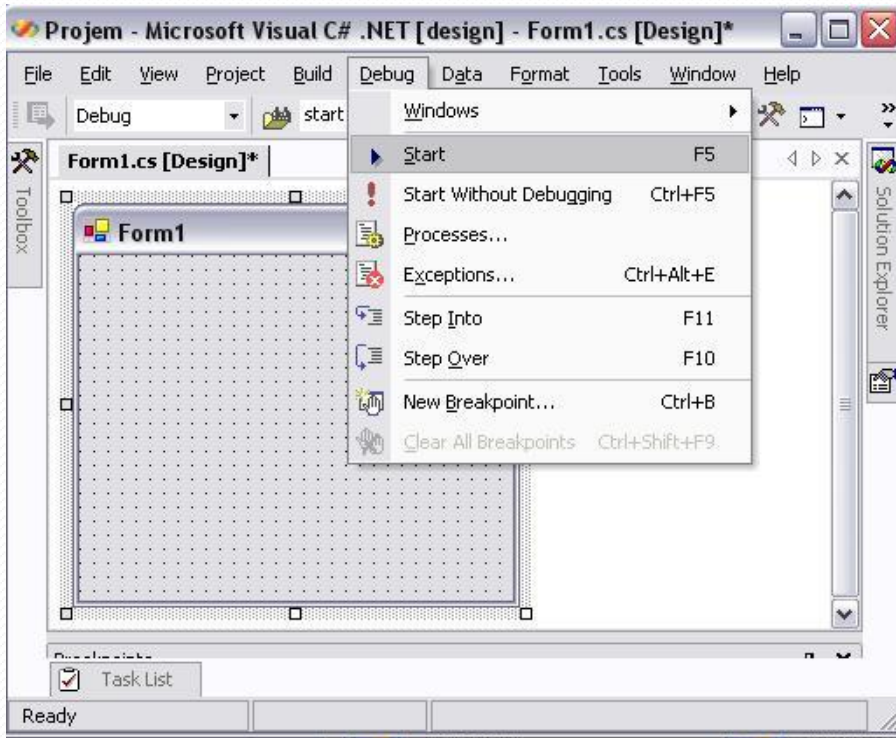
Proje Çalışma Ekranı

- **ToolBox Penceresi**
- Windows tabanlı uygulamalar geliştirirken sıkça kullanacağımız bir grup kontrol vardır.
- Form kontrolü hariç diğer bütün kontroller **Toolbox** panelinden seçilir.
- Bu kontroller sürüklenip Form üzerine istenilen pozisyona bırakılır.



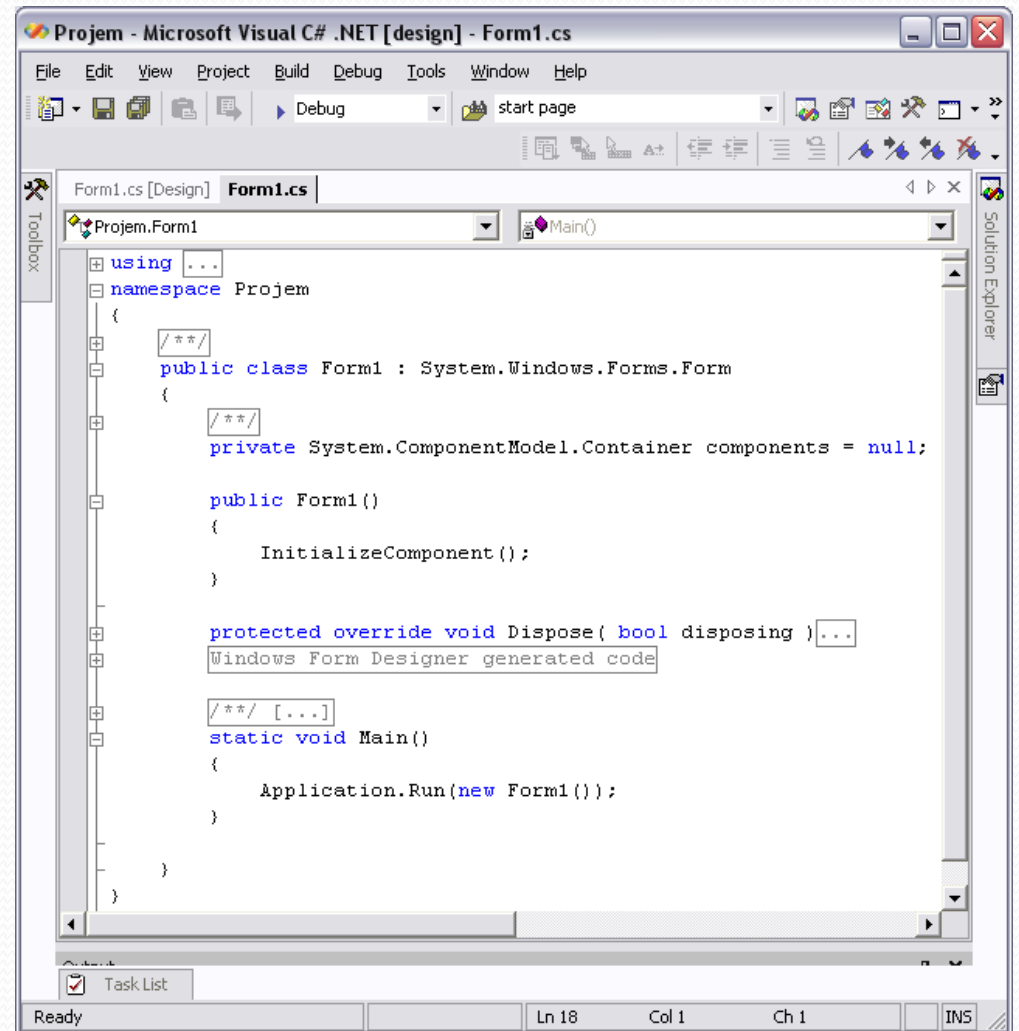
Proje Çalışma Ekranı

- Projeleri Çalıştırmak



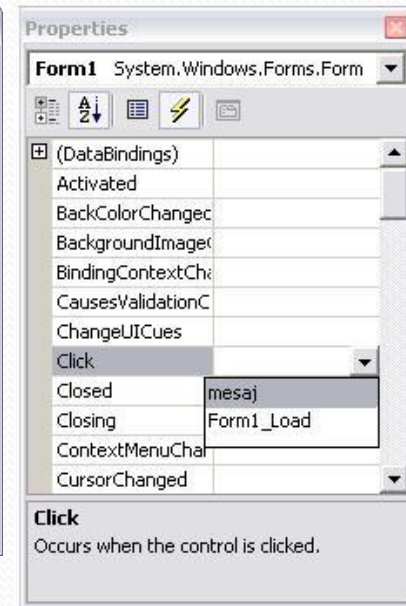
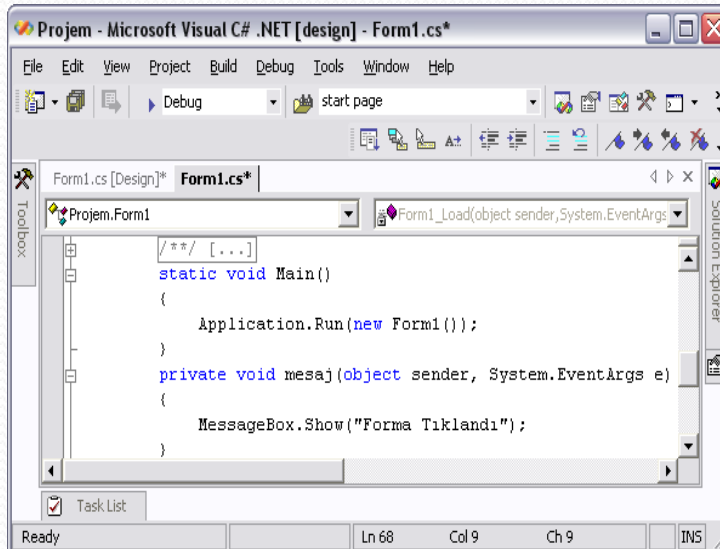
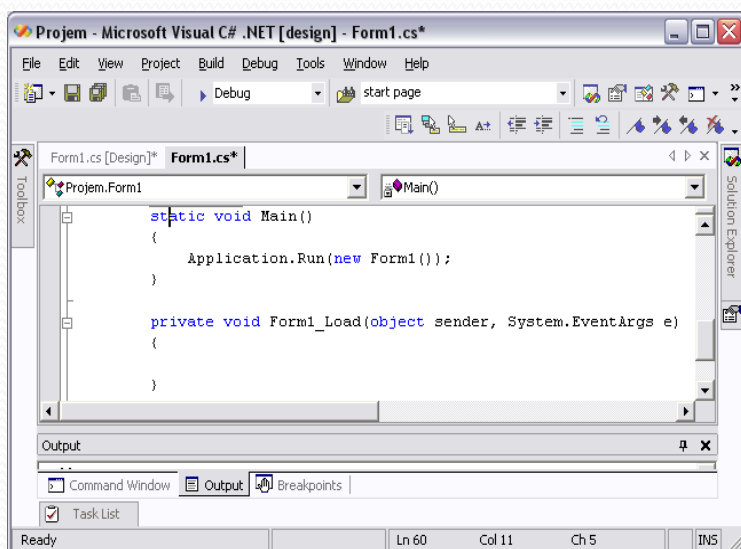
FORM UYGULAMALARI

- Program Kodu Yazmak



FORM UYGULAMALARI

- Olay ve Yordam Hazırlamak



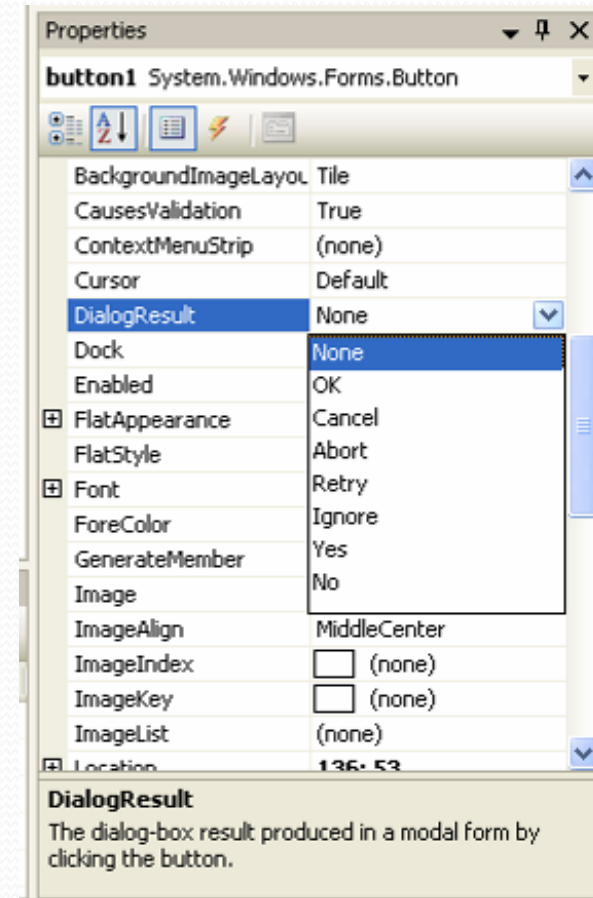
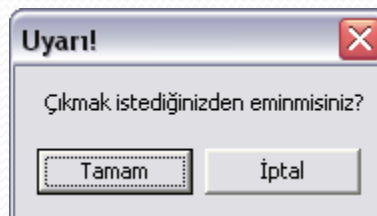
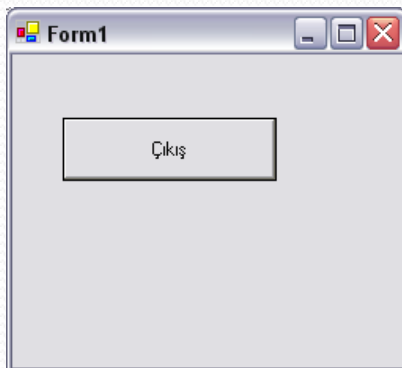
FORM UYGULAMALARI

- **MessageBox Sınıfı**
- Proje içerisinde kullanıcıya bilgi vermek veya onay almak için kullanılan sınıftır
- **MessageBox (Açıklama, Başlık, Onay Butonu, Mesaj Sembolü)**
- `MessageBox("Form1 yüklendi", "Yükleme Bilgisi", MessageBoxButtons.OK, MessageBoxIcon.Information);`



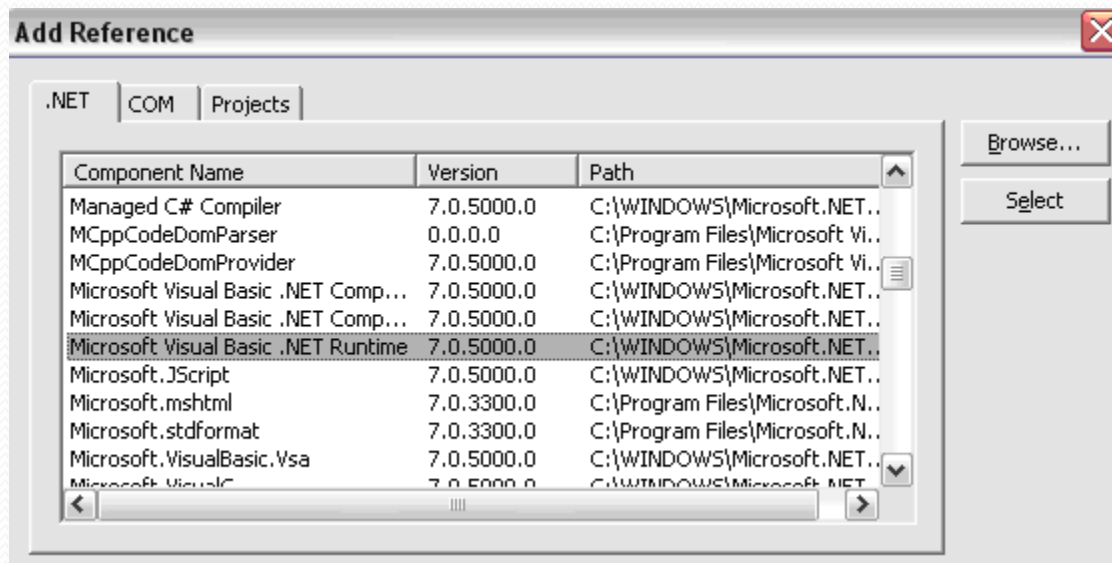
FORM UYGULAMALARI

- **MessageBox** sınıfındaki onay butonları
- **Dialog Result:** Basılan tuşa göre işlem yapma
- `private void button1_Click(object sender, System.EventArgs e) {`
- **DialogResult** sonuc;
- `sonuc=MessageBox.Show ("Çıkmak istediğinizden emin misiniz?", "Uyarı!",`
`MessageBoxButtons.OKCancel);`
- `if (sonuc==DialogResult.OK)`
- `{ Form1.ActiveForm.Close(); }`
- `}`



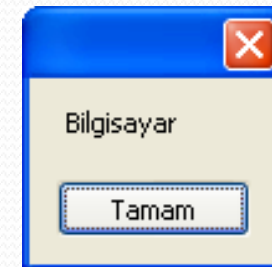
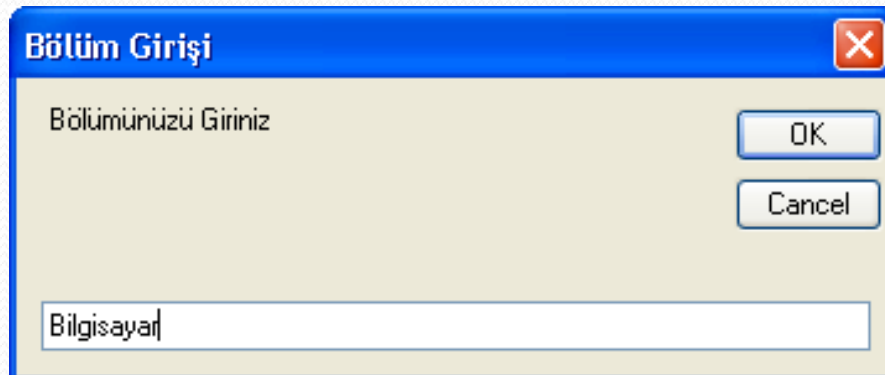
FORM UYGULAMALARI

- **InputBox Sınıfı**
- Kullanıcıdan bilgi almak için kullanılan sınıftır. C sharp içinde dirak olarak bulunmaz. Proje içerisine Add Reference ile dahil edilmesi gerekir.



FORM UYGULAMALARI

- **InputBox Sınıfı**
- `private void Form1_Load(object sender, System.EventArgs e)`
- `{`
- `string bolum= Microsoft.VisualBasic.Interaction.InputBox ("Bölümünüzü Giriniz" ,"Bölüm Girişi","",20,20);`
- `MessageBox.Show(bolum);`
- `}`



FORM UYGULAMALARI

- **Kontrol elamanlarını nesne ile tanımlama**
- Toolbox içerisinden kontrol eklemesi yapmadan sadece o kontrol sınıfına ait nesne ile kontrol elamanları kullanılabilir. Örnek: Proje içerisinde yeni bir form oluşturmak.

```
private void button1_Click (object sender, System.EventArgs e)
```

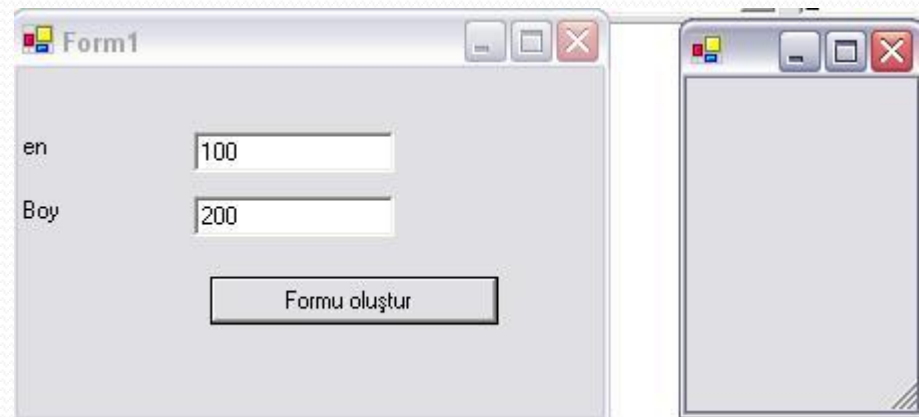
```
{ Form f=new Form();
```

```
    f.Width=Convert.ToInt32(textBox1.Text);
```

```
    f.Height=Convert.ToInt32(textBox2.Text);
```

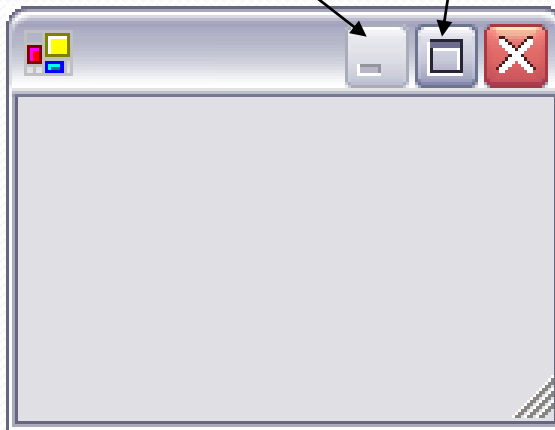
```
    f.ShowDialog();
```

```
}
```



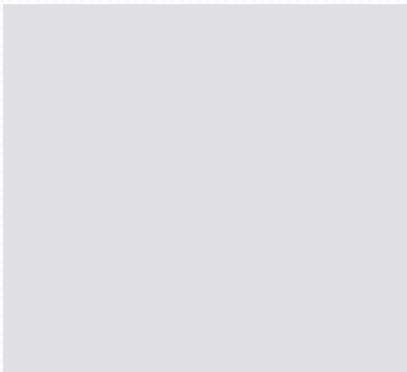
FORM UYGULAMALARI

- **Form Özellikleri**
- **MinimizeBox Özelliği:** Simge Durumuna Küçültme
- **MaximizeBox Özelliği:** Ekranı Kaplama
- **MinimizeBox =false, MaximizeBox=true**

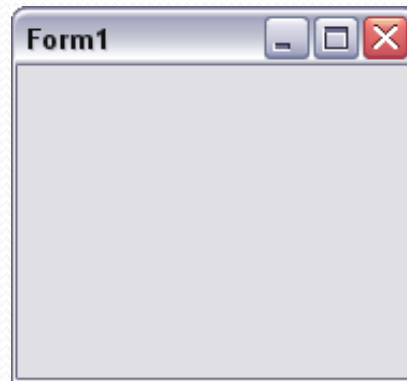


FORM UYGULAMALARI

- **Form BorderStyle Özelliği:** Formun sınır özelliklerini belirler
- Sizable seçenekleri dışındakilerde ekran boyutu fare ile değiştirilemez.
- Örneğin
- **None**



FixedDialog

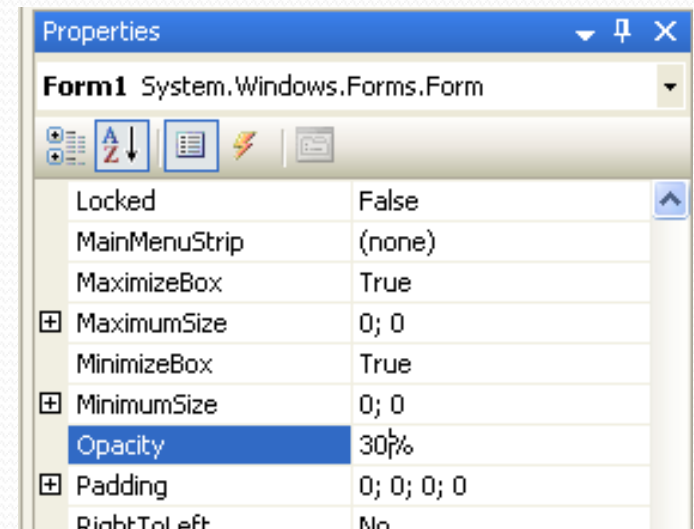


Fixed ToolWindow



FORM UYGULAMALARI

- **Formun boyutları:** Width (x boyutu) , Height (y boyutu)
- **Formun Ekrandaki Konumu (Loaction):**
 - `private void Form1_Load(object sender, EventArgs e)`
 - `{ this.Location=new Point (150, 250);}`
- **BackgroundImage:** Form arka planına resim ekleme
- **Oppacity:** Formun şeffaflığını ayarlama

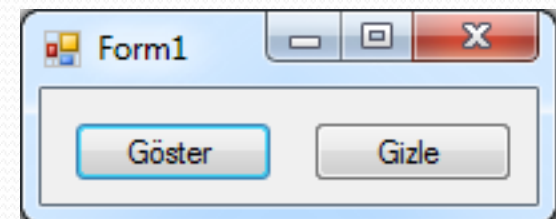


FORM UYGULAMALARI

- **Show ve ShowDialog Metotları**
- **Show** metodu birden fazla formu aynı anda ekrana getirmek için, **ShowDialog** ise sadece çalışmak istediğimiz formu ekrana getirmek için kullanılır. Bu form dışındakilere erişilemez.
- Birinci Form içerisinde;
- `private void button1_Click(object sender, EventArgs e)`
- `{ // Project –Add Windows Form ile form eklendikten sonra`
- `Form2 yeni = new Form2();`
- `yeni.Show();`
- `}`

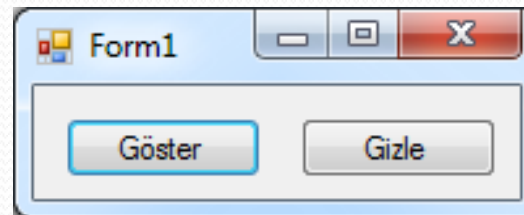
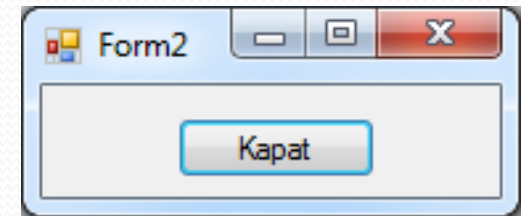
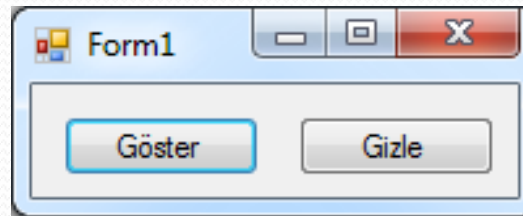
FORM UYGULAMALARI

- **Örnek:** Form1 üzerinden Form2 yi çağırma. Eğer aşağıdaki kod satırları olmazsa her show metodunda form2 yeni pencerede açılır.
- `public partial class Form1 : Form {`
- `Form2 s = new Form2();`
- `private void button1_Click(object sender, EventArgs e) {`
- `s.Visible = true; // Hide ile gizlemenin etkisini kaldırıyor`
- `if (s.Created) // Form oluşturulmuş ise sadece aktif ediyor`
- `s.Activate();`
- `else`
- `s.Show(); // Oluşturulmamış ise gösteriyor`
- `}`
- `}`



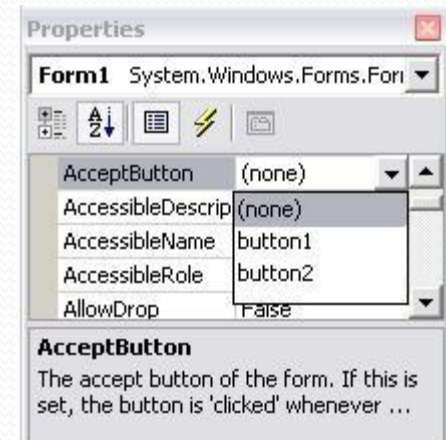
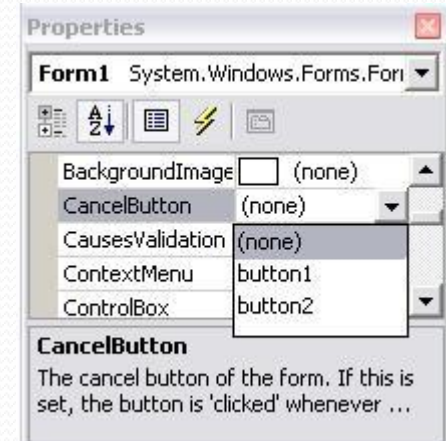
FORM UYGULAMALARI

- `private void button2_Click(object sender, EventArgs e)`
- `{`
- `if (s.Created)`
- `s.Hide();`
- `}`
- `//Form2`
- `public partial class Form2 : Form {`
- `private void button1_Click(object sender, EventArgs e)`
- `{`
- `this.Close();`
- `}`
- `}`



FORM UYGULAMALARI

- **İptal (CancelButton) düğmesi**
- **Esc** ile aynı işleve sahiptir. Forma yerleştirmiş olduğunuz düğmeye benzer işlevi vermek istiyorsanız formun özelliğine söz konusu düğmenin adını aktarmalısınız.
- **Tamam (AcceptButton) düğmesi**
- **Enter** ile tuşu aynı işleve sahiptir. Herhangi bir anda Enter tuşuna basıldığında formdaki düğmelerden birisi tıklanmış gibi işlem yapılmasını istiyorsanız o düğmenin adını formun AcceptButton özelliğine aktarmalısınız.



FORM UYGULAMALARI

- Form Özellikleri:

Özellik	Açıklama
AcceptButton	Form üzerinde Enter tuşuna basıldığı zaman “tıklanacak” Button kontrolü
CancelButton	Form üzerinde Esc tuşuna basıldığı zaman “tıklanacak” Button kontrolü
Opacity	Formun şeffaflık oranı (0 -1 arası)
MaximizeBox	Ekranı Kapla düğmesinin görünürlüğü
MinimizeBox	Simge Durumunda Küçült düğmesinin görünürlüğü
ControlBox	Close, Maximize ve Minimize düğmelerinin tümünün görünürlüğü
StartPosition	Form açıldığı zaman, ekran üzerindeki konumu
TopMost	Formun tüm pencerelerin üzerinde gözükmesi
FormBorderStyle	Formun kenar stili
MaximumSize	Formun alabileceği maksimum büyüklük
MinimumSize	Formun alabileceği maksimum büyüklük

FORM UYGULAMALARI

- Form Olayları:

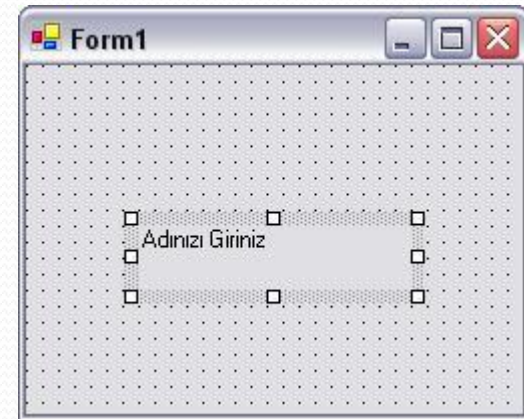
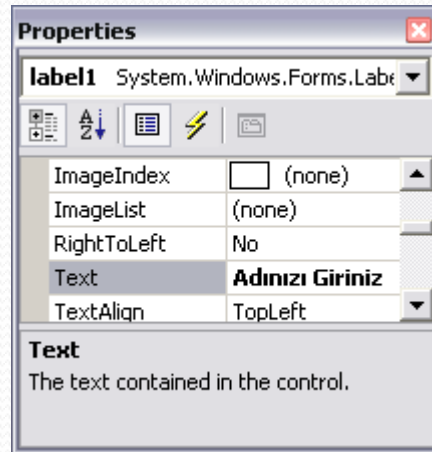
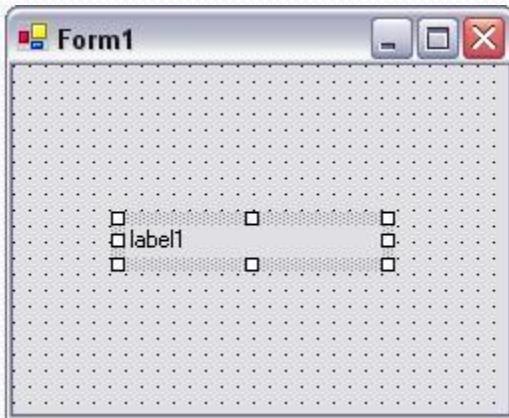
Olay	Açıklama
Click	Form üzerine tıklandığı zaman gerçekleşir
Closing	Form kapanmadan hemen önce gerçekleşir
Closed	Form kapandıktan sonra gerçekleşir
Load	Form yüklenirken gerçekleşir
KeyDown	Form üzerindeyken bir tuşun basılması ile gerçekleşir
KeyUp	Basılan tuşun kaldırılması ile gerçekleşir

- Form Metotları:

Metot	Açıklama
Hide	Formu Visible özelliğini False yaparak, gizler
Close	Formu kapatır. Eğer form başlangıç formuysa uygulama sonlanır
Show	Formu gösterir. Hide ile gizlenmişse, Visible özelliği True yapılır.
ShowDialog	Formu diyalog kutusu olarak gösterir.

FORM KONTROLLERİ- LABEL KONTROLÜ

- **Label** kontrolü Form üzerinde kullanıcıya bilgi vermek amaçlı kullanılan etikettir.



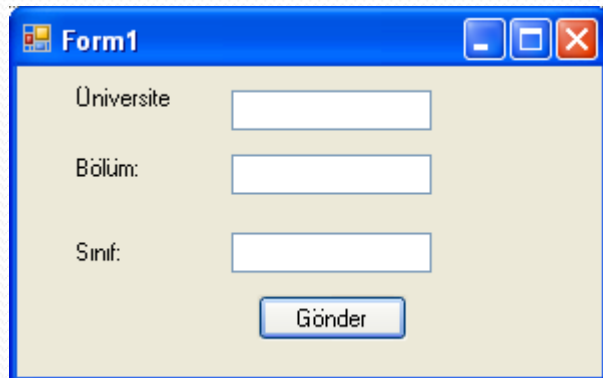
LABEL KONTROLÜ

- Label Özellikleri :

Özellik	Açıklama
TextAlign	Yazının, etiket üzerindeki pozisyonu belirler.
BorderStyle	Kontrolün kenar stilidir. FixedSingle değeri, kontrolün kenar çizgilerini gösterir. Fixed3D değeri, kenarların üç boyutlu olmasını sağlar
Image	Etiket üzerinde görüntülenmek istenen resmi tutar
ImageAlign	Etiket üzerindeki resmin nerede duracağını belirler
RightToLeft	Etiket üzerindeki yazının yönünü belirler. Eğer Yes değerini alırsa, yazılar sağdan sola gösterilir

TextBox KONTROLÜ

- Metin kutuları, kullanıcıdan bilgi almak için kullanılır.
- **private void** button1_Click(**object** sender, **EventArgs** e)
- {
- **MessageBox.Show** (**textBox1.Text** + "\n" + **textBox2.Text** +
" \n" + **textBox3.Text**);
- }



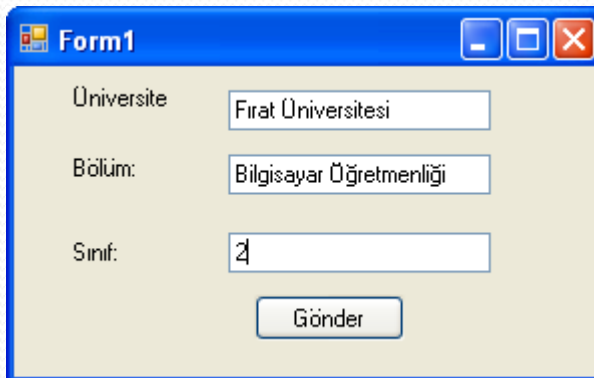
Form1

Üniversite

Bölüm:

Sınıf:

Gönder



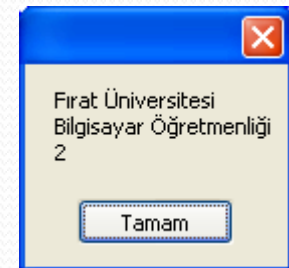
Form1

Üniversite

Bölüm:

Sınıf:

Gönder



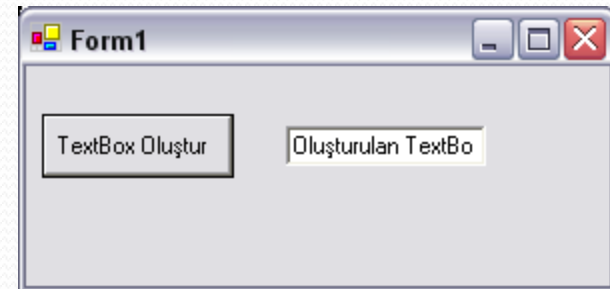
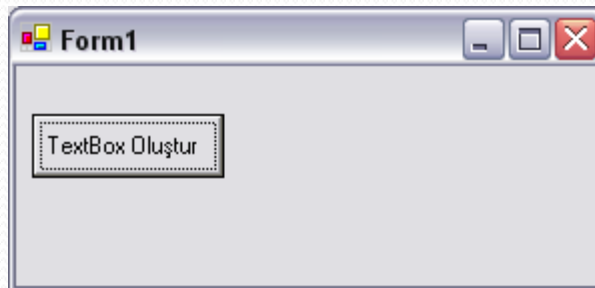
Firat Üniversitesi
Bilgisayar Öğretmenliği
2

Tamam

TextBox KONTROLÜ

Çalışma Anında Forma Kontrol Yerleştirmek

- `private void button1_Click(object sender, System.EventArgs e)`
- `{`
- `TextBox metin=new TextBox(); // metin kutusu nesnesi`
- `this.Controls.Add(metin);`
- `metin.Text="Oluşturulan TextBox";`
- `metin.Top=30;`
- `metin.Left=130;`
- `}`

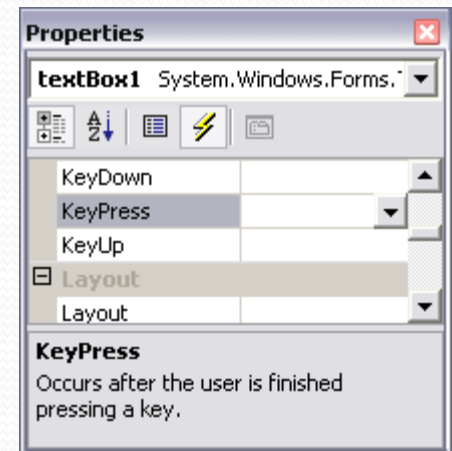


TextBox KONTROLÜ

- **KeyDown:** Bir tuşa basıldığı durumda meydana gelen olay
- **KeyUp:** Tuştan elin çekildiği durumda meydana gelen olay
- **KeyPress:** Bir tuşa basılma ve çekme anı arasında meydana gelen olaydır.
- Formun üzerine yerleştirilen nesnelerin **KeyDown** metodu yerine formun **KeyDown** metodunun işletilmesini istiyorsanız formun **KeyPreview** özelliğini **true** yapmalısınız. Bu özellik true iken öncelik formun KeyDown olayını temsil eden metoda verilir ve aktif nesnenin KeyDown metodu, formun KeyDown metodunun çalışması sona erdikten sonra işletilir. Bu durum KeyUp ve KeyPress içinde geçerlidir.

TextBox KONTROLÜ

- Örnek:
- `private void textBox1_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)`
- `{`
- `if(e.KeyChar==13) // Enter Tuşu`
- `MessageBox.Show(textBox1.Text);`
- `}`
- `// enter tuşuna basıldığında ortaya çıkan sonuç`

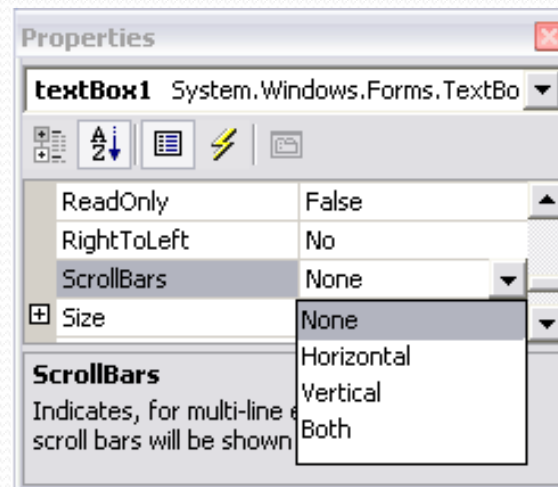


TextBox KONTROLÜ

- **Örnek:** Alt tuşu ile formun kapatılması
- `private void Form1_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)`
- `{`
- `if(e.Modifiers==Keys.Alt)`
- `//(e.KeyCode==Keys.Control) olabilir.`
- `this.Close();`
- `}`

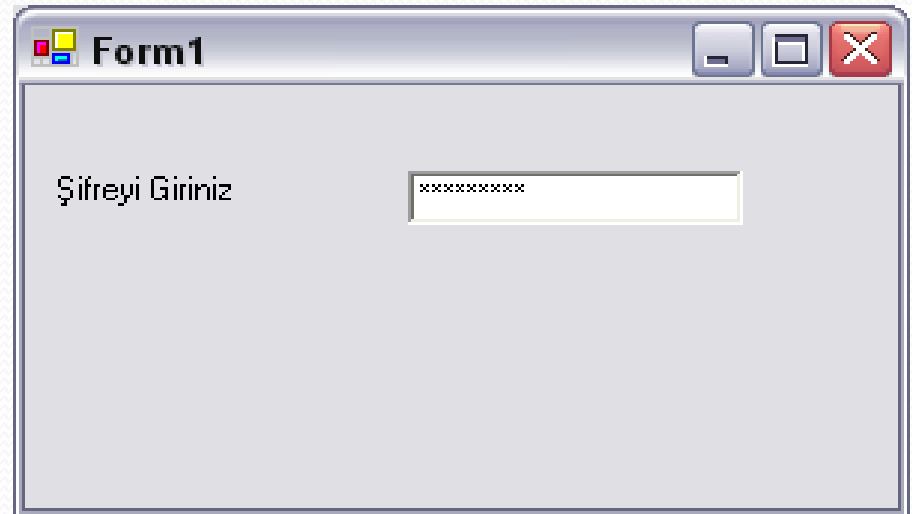
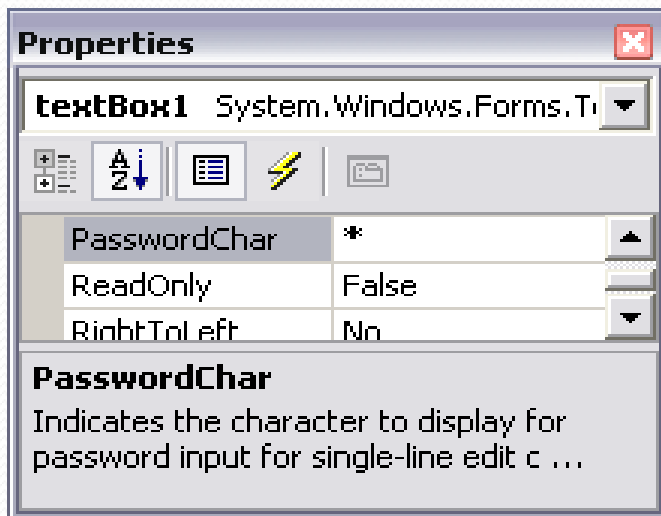
TextBox KONTROLÜ

- **MultiLine Özelliği:** Metin kutusuna bilginin çoklu satır olarak girilmesini sağlar.
- **ScrollBars Özelliği:** Eğer satır sayısı metin kutusunun boyutundan fazla ise kaydırma çubuklarına gerek duyulabilir. Kaydırma çubukları eklemek için Scroll Bars özelliği kullanılır, bu özellik dört değişik değer alır.



TextBox KONTROLÜ

- **PasswordChar Özelliği:** Metin kutusuna girilen bilginin belirlenen karakter ile gizlenmesini sağlar.



TextBox KONTROLÜ

- **CharacterCasing Özelliği:** Metin kutusuna girilen karakterlerin büyük ya da küçük harfe çevrilmesini sağlar. Upper ve Lower değerleri alır.
- `private void Form1_Load(object sender, System.EventArgs e)`
- `{`
- `textBox1.CharacterCasing=CharacterCasing.Upper;`
- `}`

TextBox KONTROLÜ

- TextBox Özellikleri

Özellik	Açıklama
MultiLine	Metin kutusuna birden fazla satırda değer girilebilmesini sağlar. False durumunda ise, metin kutusunun yüksekliği değiştirilemez
ScrollBars	Metin kutusunda kaydırma çubuklarının görünmesi. Varsayılan olarak kaydırma çubuğu görüntülenmez, ancak Horizontal, Vertical kaydırma çubukları ya da ikisi birden gösterilebilir.
PasswordChar	Metin kutusuna parola girilecekse, girilen karakterlerin hangi karakter olarak görüneceğini belirler
WordWrap	Metin kutusuna girilen değerlerin, satır sonlandığında bir alt satıra geçilmesini belirtir. Eğer MultiLine özelliği False ise, alt satırlar tanımlı olmayacağı için bu özelliğin bir etkisi görülmez.
MaxLength	Metin kutusunun alabileceği maksimum karakter sayısını belirtir.
ReadOnly	Metin kutusunun yazmaya karşı korumalı olduğunu belirtir.
CharacterCasing	Metin kutusuna karakterler girilirken büyük veya küçük harfe çevrilmesini sağlar. Upper değeri büyük, Lower değeri küçük harfe çevrimi sağlar.

TextBox KONTROLÜ

- **TextBox Olayları**

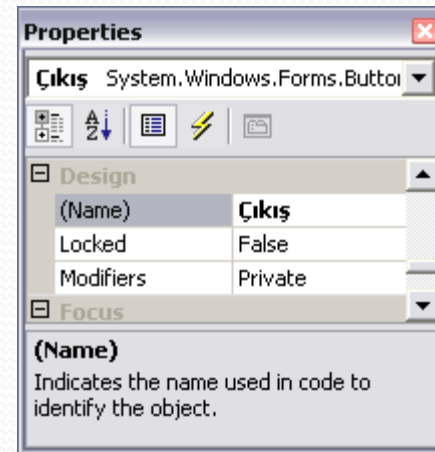
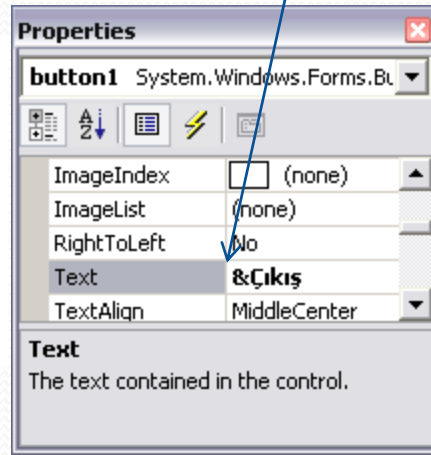
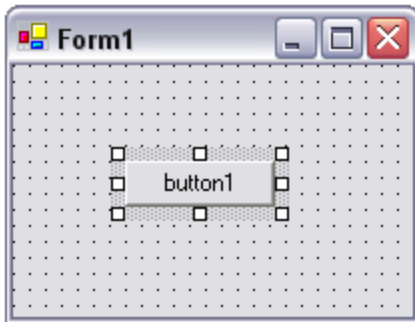
Olay	Açıklama
TextChanged	Metin kutusundaki yazı değiştiği zaman gerçekleşir.
KeyPress, KeyPreview, KeyDown	Bir tuşa basılma durumunda gerçekleşir

- **TextBox Metotları**

Metot	Açıklama
Cut	Seçilen karakterleri siler ancak hafızada tutar.
Copy	Seçilen karakterleri kopyalar
Paste	Hafızaya alınan karakterleri metin kutusuna yapıştırır
Clear	Metin kutusundaki yazıları temizler
SelectAll	Metin kutusundaki tüm yazıyı seçer

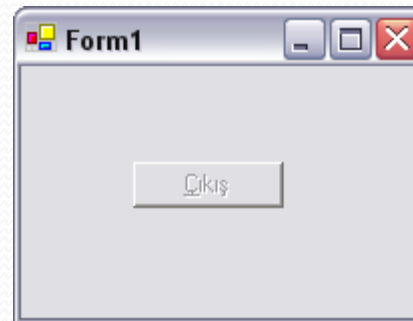
Buton KONTROLÜ

- Windows uygulamalarında, form üzerinde komut düğmeleri olarak kullanılır. Kısayol tuşları ile buton kontrolüne erişmek için **&** işareti kullanılır. (Alt+ Ç tuşuna basarak butona tıklanma sağlanır)



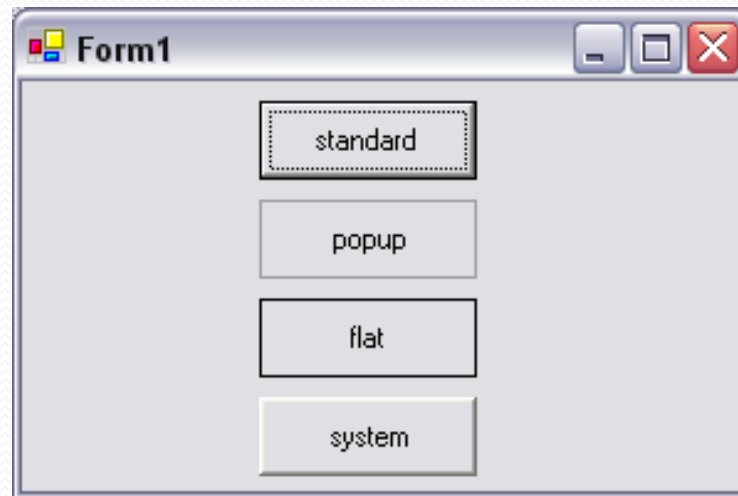
Buton KONTROLÜ

- **Enabled ve Visible Özellikleri:** Butonun aktif veya görünür olmasını kontrol eder.
- `private void Çıkış_Click(object sender, System.EventArgs e)`
- `{ this.Close(); }`



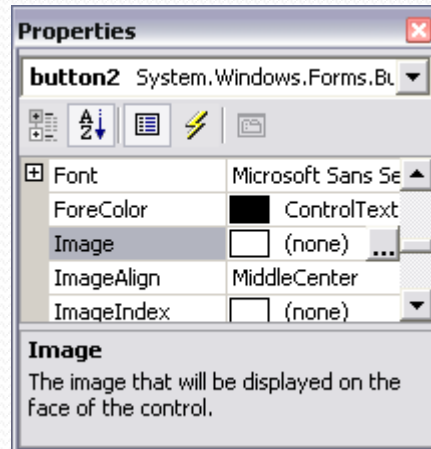
Buton KONTROLÜ

- **TabIndex ve TabStop Özellikleri**
- TabStop false olursa o nesne atlanır
- **FlatStyle Özelliği**
- Dört farklı değer alabilir. Bunlar **Standart**, **popup**, **flat**, **system**' dir. Görünüm şekilleri aşağıdaki gibidir.



Buton KONTROLÜ

- **PerformClick() metodu:**
- Butonların çalışması için üzerine tıklamak gerekir. Ancak PerformClick() metodu ile düğme tıklanmış gibi bir etki sağlanır.
 - `private void Form1_Load(object sender, EventArgs e)`
 - `{ button1.PerformClick(); }`
- **Image Özelliği:** Düğmelerin üzerine resim yerleştirmek için kullanılır.



Buton KONTROLÜ

- Button Özellikleri

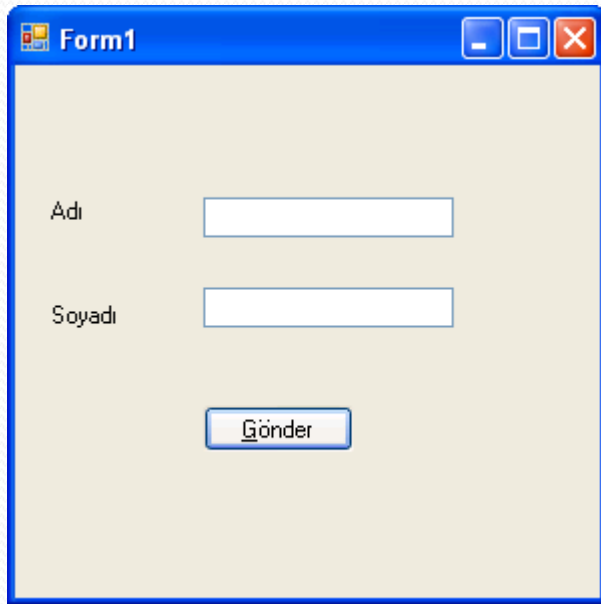
Özellik	Açıklama
DialogResult	Ait olduğu form ShowDialog metodu ile çağrıldığı zaman, dönüş değerini belirler
FlatStyle	Düğmeye basıldığında ve düğmenin üzerine gelindiğinde görünen formatı belirler

- Button Olayları

Olay	Açıklama
Click	Düğme üzerine tıklandığı zaman gerçekleşir

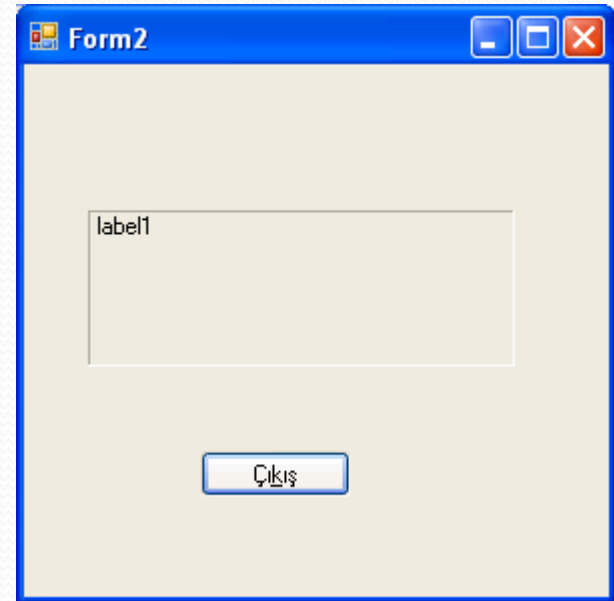
Örnek: Oluşturulan formlar arası bilgi gönderme

- İlk Form



A screenshot of a Windows form titled "Form1". The form has a light beige background and a blue title bar. It contains two text input fields: the first is labeled "Adı" and the second is labeled "Soyadı". Below these fields is a button labeled "Gönder".

Bilgi Gönderilecek Form



A screenshot of a Windows form titled "Form2". The form has a light beige background and a blue title bar. It contains a large text area labeled "label1" and a button labeled "Çıkış" at the bottom.

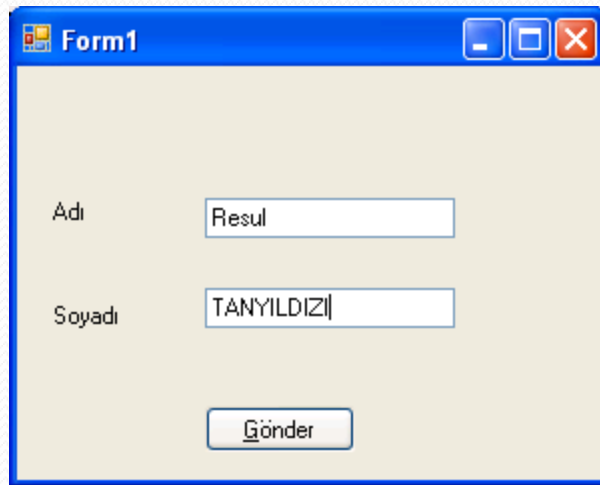
Örnek: Oluşturulan formlar arası bilgi gönderme

- İlk Formun kod kısmında Form2 show metodu ile gösterileceği zaman onun yapılandırıcısına Form1 de girilen değerleri parametre olarak yazıyoruz.
- ```
private void button1_Click(object sender, EventArgs e)
```
- ```
{
```
- ```
 string a, b;
```
- ```
    a = textBox1.Text;
```
- ```
 b = textBox2.Text;
```
- ```
    Form2 f = new Form2(a,b);
```
- ```
 f.Show();
```
- ```
}
```

Örnek: Oluşturulan formlar arası bilgi gönderme

- **Form2 nin yapılandırıcısında alabacağımız değerler için parametreleri ekliyoruz. Normal durumda burası boştur.**
- `string aa,bb;`
- `//object yazmak bazen daha uygun olur.`
- `public Form2(string a, string b)`
- `{ aa = a;`
- `bb = b;`
- `InitializeComponent();`
- `}`
- `private void Form2_Load(object sender, EventArgs e)`
- `{`
- `label1.Text = " Adı: "+aa + "\n \n" + " Soyadı: "+ bb;`
- `}`
- `private void button1_Click(object sender, EventArgs e)`
- `{ this.Close(); }`

Örnek: Oluşturulan formlar arası bilgi gönderme



Form1

Adı:

Soyadı:

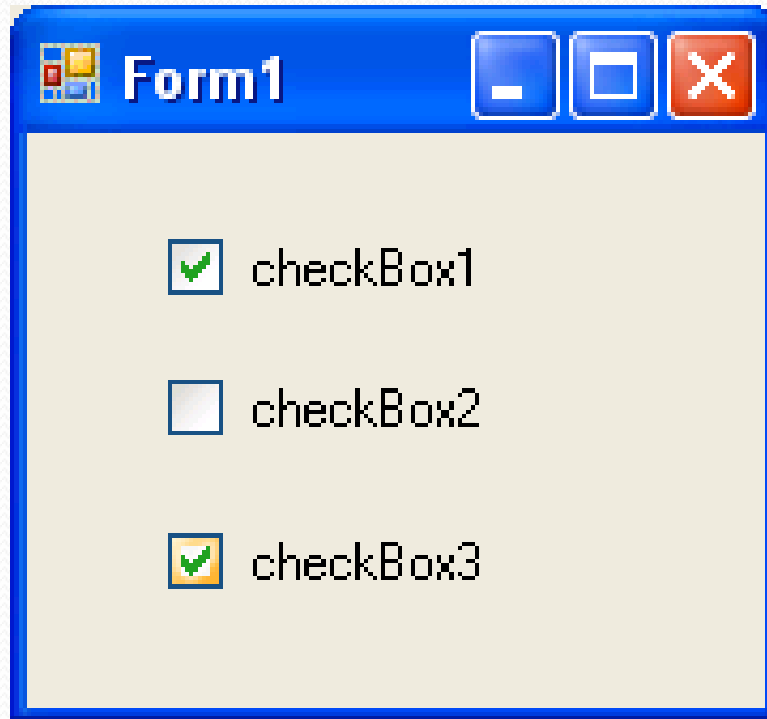


Form2

Adı: Resul
Soyadı: TANYILDIZI

CheckBox KONTROLÜ

- Kontrol Kutusu, kullanıcıya birden çok seçeneği seçme imkanı sağlar. Checked özelliği kontrol kutusunun seçilip seçilmediğini kontrol eder. Seçili ise , true değilse false değerini alır.

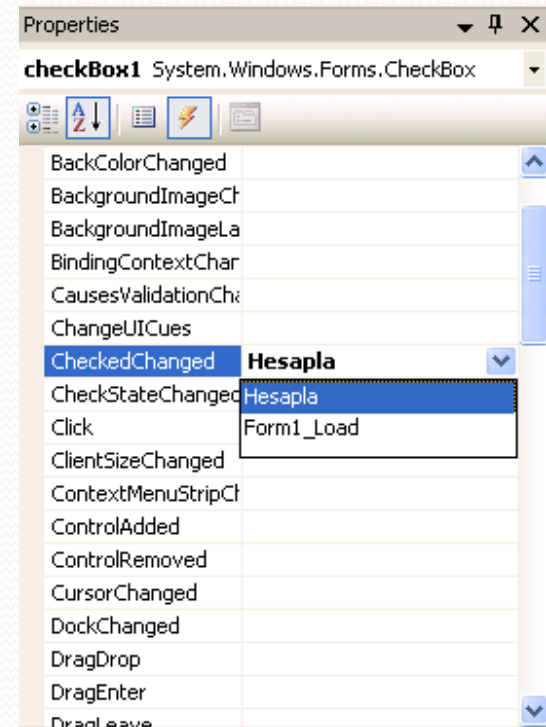
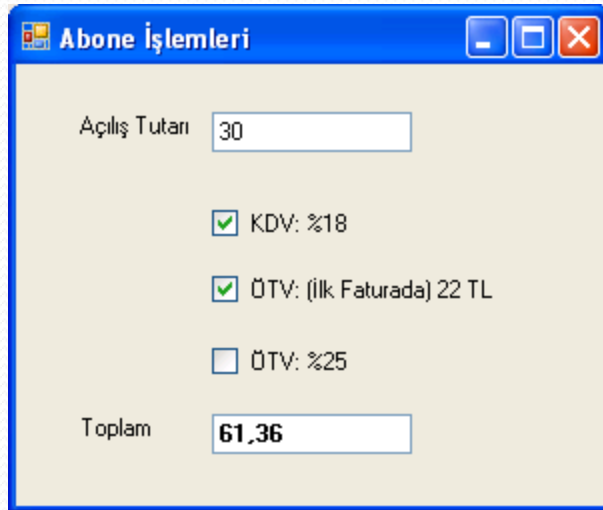


The image shows a screenshot of a Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains three checkboxes, each with a label to its right:

- ☒ checkBox1
- ☐ checkBox2
- ☒ checkBox3

Örnek:

- Bir GSM şebekesinden faturalı hat açılışında toplam tutar hesaplanması.
- Öncelikle CheckBox özelliğinden kontrol ifadelerinin her birinin CheckedChange olayına Hesapla diye bir metot tanımlayınız.



Örnek :

- using System;
-
- namespace uygulama {
- public partial class Form1 : Form {
- public Form1() { InitializeComponent(); }
- private void Hesapla(object sender, EventArgs e)
- { // Form üzerindeki tüm seçme kutularının durumu değiştiği zaman, toplam fiyat tekrar hesaplanır
- double toplam = Convert.ToDouble(textBox1.Text);
- // İlk faturada 22 YTL açılış bedeli eklenir
- if (checkBox2.Checked) { toplam += 22; }
- // KDV eklenir
- if (checkBox1.Checked) { toplam *= 1.18; }
- // Özel İletişim vergisi eklenir
- if (checkBox3.Checked) { toplam *= 1.25; }
- textBox2.Text = Convert.ToString(toplam); }}}

CheckBox KONTROLÜ

- CheckBox Özellikleri

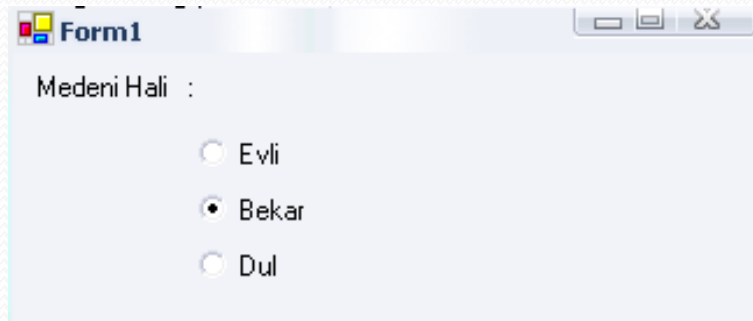
Özellik	Açıklama
Checked	Kontrolün seçili olup olmadığını belirler
CheckAlign	Seçme kutusunun ve üzerinde yazan metnin birbirlerine göre konumlarını belirler
Appearance	Kontrolün seçme kutusu ya da düğme şeklinde olmasını belirler
ThreeState	Seçili olup olmaması dışında, Intermediate durum da eklenir. Eğer kontrol Intermediate durumundaysa Checked özelliği True olur.
AutoChecked	Kontrole basıldığı zaman seçili duruma geçileceğini belirtir. Eğer bu özellik False ise, kontrolün durumunu değiştirmek için, Click olayında, Checked özelliğini güncellemek gerekir

- CheckBox Olayları

Olay	Açıklama
CheckChanged	Seçme kutusunun durumu değiştiği zaman gerçekleşir.

Radiobutton KONTROLÜ

- RadioButton kontrolü CheckBox'tan farklı olarak birkaç seçenekten sadece birini seçme imkanı veren bir onay kontrolüdür.
- Bu kontrolün tek başına kullanılması anlamsızdır. Bir kaç seçenekten birini seçme imkanı veren bir kontrol olduğu için en az iki tane birlikte verilmelidir.
- Formda düğmelerden biri seçildiğinde seçilmiş olan kendiliğinden kalkacaktır.
- GroupBox kontrolleri sayesinde farklı seçim butonları yapabilirsiniz.



Form1

Medeni Hali :

☐ Evli

☒ Bekar

☐ Dul



Form1

medeni hal :

☒ Evli

☐ Bekar

☐ Dul

Öğrenim Durumu :

☐ İlk

☐ Orta

☐ Lise

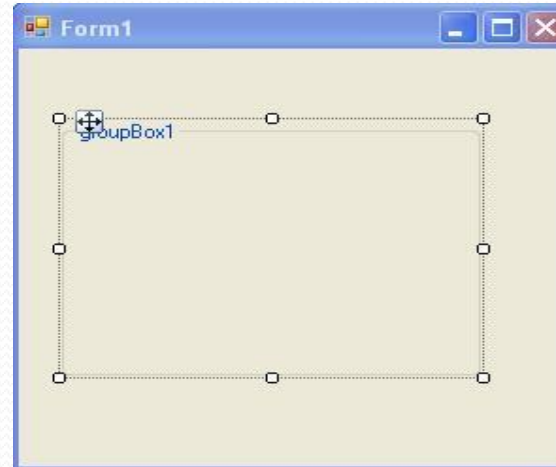
☒ Üniversite

Radiobutton KONTROLÜ

- Bu kontrolün özellikleri CheckBox'a çok benzemektedir. CheckBox'ta olduğu gibi;
-
- **Appearance** özelliği ile komut düğmesi görünümüne sahip RadioButton düğmeleri oluşturulabilir,
- **Text özelliği** ile içindeki metni yazabilir,
- **TextAlign** özelliği ile içindeki metnin yerleşmesini belirleyebilir,
- **Image** özellikleri ile içinde resim gösterebilir,
- **CheckAlign** özelliği ile seçenek düğmesini sola, sağa alınabilir.

GroupBox KONTROLÜ

- Bu kontrol tek başına değil, diğer kontrolleri gruplamak için kullanılır. Kontrolleri bu GroupBox ile gruplamanın birçok avantajı vardır.
- GroupBox içine yerleştirilen kontroller, GroupBox'a bağımlıdırlar ve konumları bu çerçeve dışına taşamaz. Özellikle birkaç kontrolü birden görünür ya da görünmez yapmak için hepsinin Visible özelliğini tek tek değiştirmek yerine çerçevenin Visible özelliğini değiştirilerek çerçeve içindeki tüm kontroller aynı anda değiştirilebilir. Her bir kontrol tek tek taşınmak yerine çerçeve taşınır. Çerçevelerin buna benzer pek çok faydaları vardır. Ayrıca radioButton' ların gruplanmasında çerçeve kullanmak kaçınılmaz olabilir.



Örnek

- Aşağıdaki form dizaynında aile bilgileri bölümünün medeni hali bekar olanlar için gösterilmemesi, mezun olduğu fakülte bölümünün ise sadece öğrenim durumu üniversite olması durumunda gösterilmesi gerekir. Aile bilgileri ve mezun olduğu fakülte içindeki bütün kontrolleri tek seferde gizleyip, göstermek için sadece onun içinde bulunduğu GroupBox kontrolünü gizleyip göstermek yeterlidir.

The screenshot shows a Windows form titled "Form1" with a light beige background and a blue title bar. The form is organized into four distinct sections, each with a blue header and a light beige border:

- personelin**: Contains three text input fields labeled "Adı ve Soyadı", "Telefonu", and "Adresi".
- Medeni Hali**: Contains three radio button options: "bekar", "evli", and "dul".
- Aile Bilgileri**: Contains two text input fields labeled "Çocuk Sayısı" and "Evlenme Tarihi".
- Mezun Olduğu Fakülte**: Contains four radio button options: "mühendislik", "teknik eğitim", "fen edebiyat", and "tıp", followed by a text input field labeled "Diğer".

Örnek

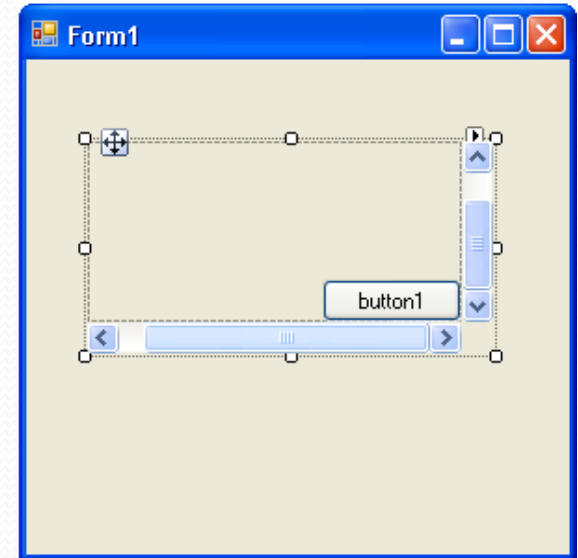
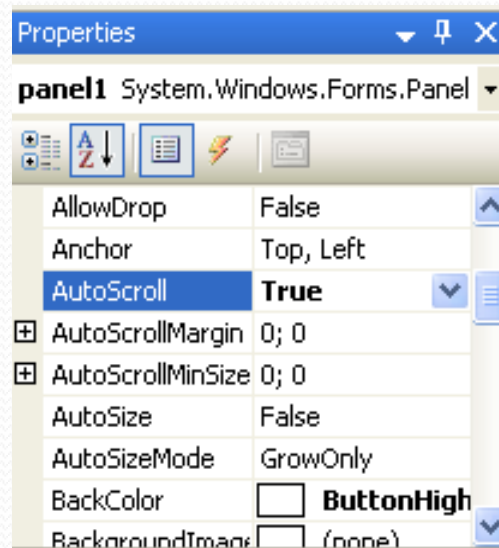
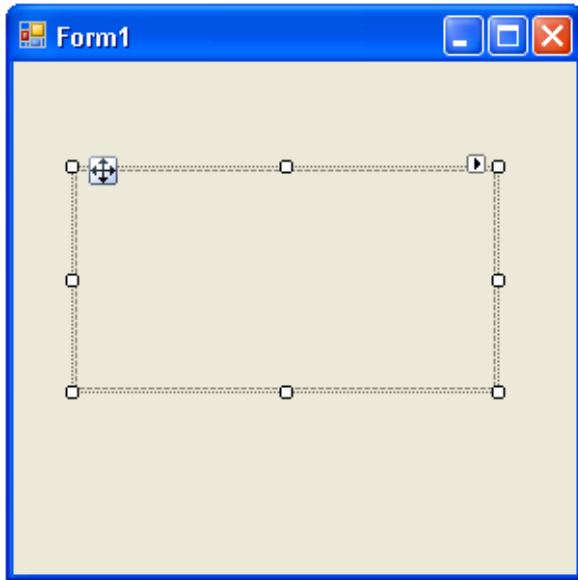
- Bekar seçeneği işaretli ise (radioButton1) Aile Bilgilerini (groupBox4) gizlememiz diğer durumlarda göstermemiz gerekir. Fakülte seçeneği işaretli ise (radioButton7) mezun olduğu fakülteyi (groupBox5) göstermemiz, diğer durumlarda gizlememiz gerekir. Buna göre;

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
```

- { if (radioButton1.Checked == true) groupBox4.Visible = false;
- else groupBox4.Visible = true; }
- private void radioButton6_CheckedChanged(object sender, EventArgs e)
- { if (radioButton6.Checked == true) groupBox5.Visible = true; }
- private void radioButton4_CheckedChanged(object sender, EventArgs e)
- { if (radioButton4.Checked == true) groupBox5.Visible = false; }
- private void radioButton5_CheckedChanged(object sender, EventArgs e)
- { if (radioButton5.Checked == true) groupBox5.Visible = false; }
- private void radioButton7_CheckedChanged(object sender, EventArgs e)
- { if (radioButton7.Checked == true) groupBox5.Visible = false; }

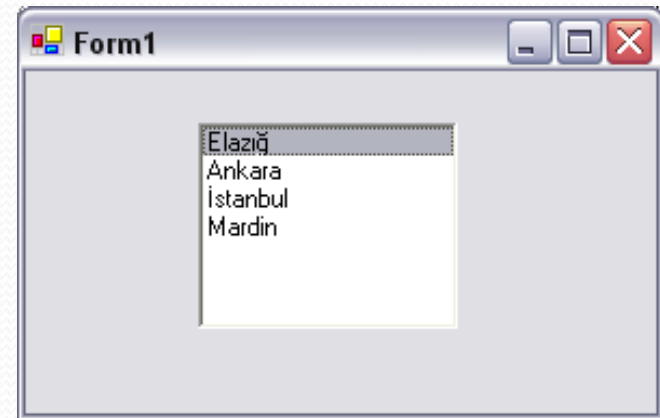
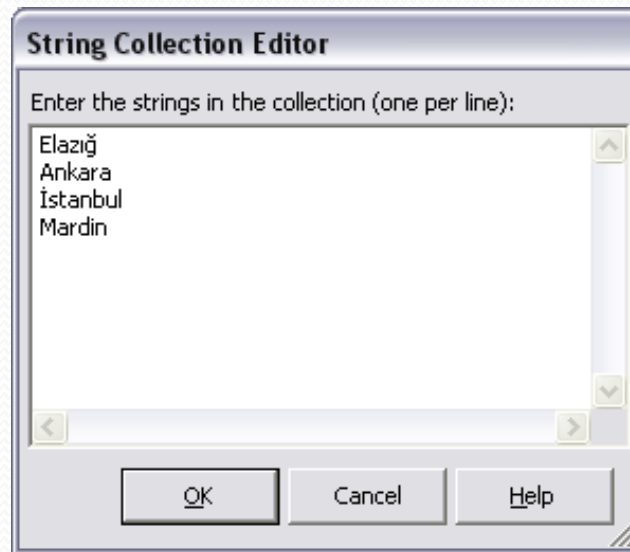
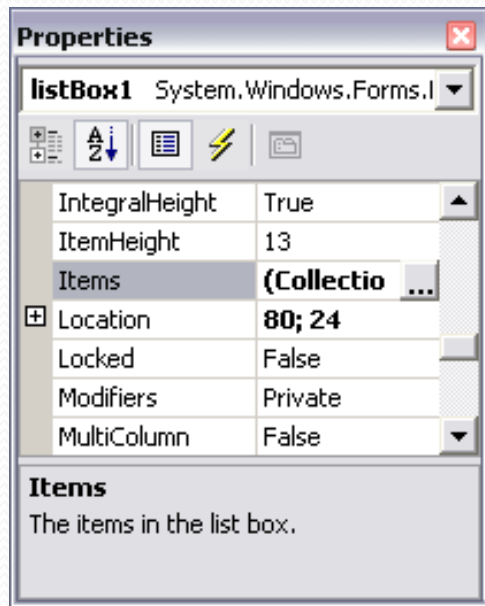
Panel KONTROLÜ

- Kullanıma sokulan grupları kontrol etmek, üzerinde daha rahat işlem yapabilmek için bir arada tutan bir toolbox komutudur. Panel üzerine istediğimiz toolbox komutunu rahatça bırakabilir üzerlerinde işlem yapabiliriz. GroupBox gibi ayarlama yapan bir kontroldür. GroupBox'a göre en büyük artısı kaydırma çubuklarını (AutoScroll) desteklemesidir.



ListBox KONTROLÜ

- Kullanıcıya sunulan seçeneklerin bir liste halinde görünmesini sağlar. Liste kutusundan istenen sayıda öge seçilebilir.



- Kod ortamında ListBox'a elaman ekleme
 - `private void Form1_Load(object sender, EventArgs e)`
 - `{`
 - `listBox1.Items.Add("Elazığ");`
 - `listBox1.Items.Add("Bursa");`
 - `listBox1.Items.Add("Manisa"); }`

ListBox KONTROLÜ

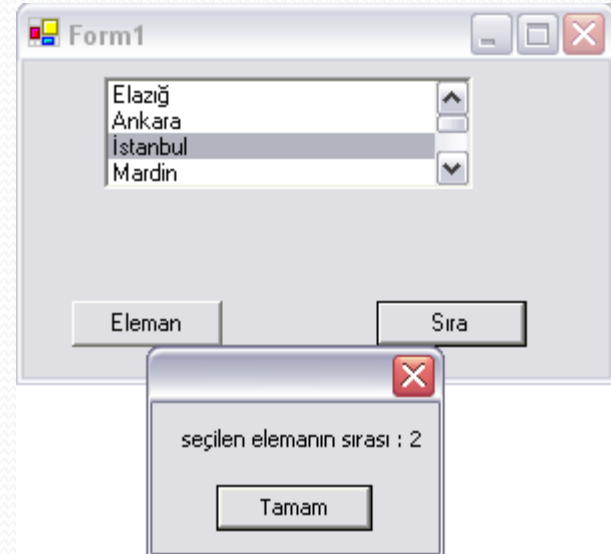
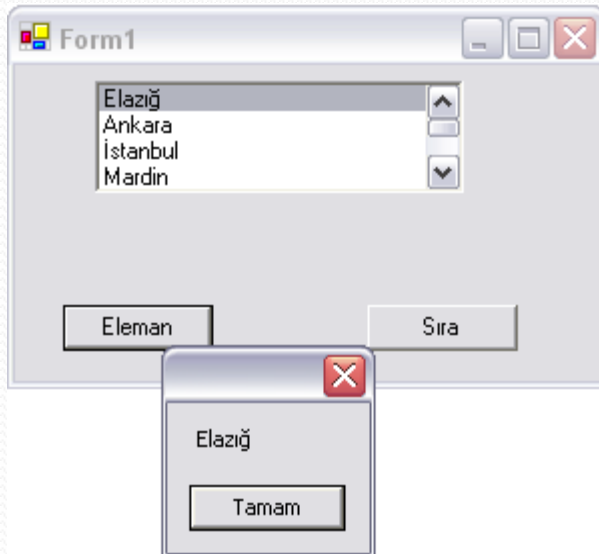
- **AddRange() metodu:** Birden fazla eleman bir seferde listBox nesnesine ekler. String bir dizinin elemanlarını da ekleyebilirsiniz.
 - listBox2.Items.AddRange(listBox1.Items);
 - listBox1.Items.AddRange(string dizi adı);
- **CopyTo() metodu:** listBox ın içeriğini bir dizi değişkene aktarır
 - string[] sehir; sehir=new string[3];
 - listBox1.Items.CopyTo(sehir,0);
- **Insert(sıra_no, eleman) metodu:** İstedığımız bir sıraya eleman yerleştirmek
 - listBox1.Items.Insert(1, "eleman");
- **Remove() metodu:**Listboxtan bir eleman silmek
 - listBox1.Items.Remove("eleman");

ListBox KONTROLÜ

- **Clear()metodu** : Liste kutusunu tamamen boşaltmak
- **Contains (string) metodu**: Liste içerisinde bir elamanın olup olmadığını kontrol eder varsa **true** yoksa **false** değeri geri döner
 - listBox1.Items.Contains("a");
- **FindStringExact (string) metodu**: Contains ile aynı işleve sahip fakat elamanın ilk geçtiği index numarasını geri döndürür.
 - listBox1. FindStringExact("a");
- **Count Özelliği**: Liste kutusundaki elaman sayısını verir.
 - private void button1_Click(object sender, EventArgs e)
 - { for (int i = 0; i < listBox1.Items.Count; i++)
 - {MessageBox.Show(""+listBox1.Items[i]); }
 - }

ListBox KONTROLÜ

- **Text Özelliği:** Liste kutusundaki seçilen elemanın değerini döndürür.
- **SelectedIndex Özelliği:** Liste kutusundaki elemanın indis numarasını verir.
 - `private void button1_Click(object sender, System.EventArgs e)`
 - `{ MessageBox.Show(listBox1.Text); }`
 - `private void button2_Click(object sender, System.EventArgs e)`
 - `{MessageBox.Show("seçilen elemanın sırası : "+ listBox1.SelectedIndex.ToString());}`



ListBox KONTROLÜ

- **SelectedIndexChanged()** metodu: Liste kutusundan herhangi bir eleman seçildiğinde çalışan olaydır.
- `private void listBox1_SelectedIndexChanged(object sender, EventArgs e)`
- `{ MessageBox.Show(listBox1.Text); }`



ListBox KONTROLÜ

- ListBox Özellikleri

Özellik	Açıklama
Items	Liste kutusuna eklenen öğelerin tutulduğu koleksiyon nesnesidir.
SelectedItem	Liste kutusundan seçilen öğeyi alır.
SelectedItems	Liste kutusundan seçilen öğeleri alır. Seçilen öğeler dinamik bir dizide tutulur.
SelectedIndex	Liste kutusundan seçilen öğenin indisini alır.
SelectedIndices	Liste kutusundan seçilen öğelerin indislerini bir koleksiyon nesnesinde tutar.
DataSource	Listenin öğelerinin tutulduğu veri kaynağıdır. Veri kaynağı boş geçilirse Items koleksiyonuna eklenen öğeler görüntülenir.
DisplayMember	Veri kaynağından gelen öğelerin, kullanıcıya gösterilecek özelliğidir.
ValueMember	Veri kaynağından gelen öğelerin, dönüş değerini belirleyen özelliğidir.
SelectedValue	Seçilen öğenin, liste kutusunun ValueMember ile belirtilen özelliğidir.
SelectionMode	Liste kutusundan kaç tane öğe seçilebileceğini belirtir. None değeri 0, One değeri 1, MultiSimple ve MultiExtended değerleri birden fazla öğenin seçilebileceğini belirtir.
MultiColumn	Liste kutusundaki öğelerin birden fazla kolonda görüntülenmesini belirler.

ListBox KONTROLÜ

- ListBox Olayları

Olay	Açıklama
SelectedIndexChanged	Liste kutusunda bir öge seçildiği zaman gerçekleşir.

- ListBox Metotları

Metot	Açıklama
GetItemText	Parametre olarak verilen nesnenin liste kutusunda gösterilen yazısını döndürür.
GetSelected	Parametre olarak verilen indisteki öğenin seçili olup olmadığını döndürür.
FindString	Parametredeki String ifadesini liste kutusunda arayarak, bulduğu ilk öğenin indisini döndürür

Örnek:

- Tedarikçiden alınacak ve stokta bulunan ürünleri listelemek ve alım satım işlemi yapmak için ListBox kontrolleri kullanma.

The screenshot shows a Windows application window titled "Form1" with a blue title bar and standard Windows window controls (minimize, maximize, close). The main area has a light beige background and contains two vertical lists of products, three buttons in the center, and two text boxes at the bottom.

Tedarikçideki Ürünler (Products from Supplier):

- Defter - 4,99 TL
- Cetvel - 1,99 TL
- Pergel - 2,99 TL

Stoktaki Ürünler (Products in Stock):

- Kalem - 1,49 TL (highlighted)
- Silgi - 0,39 TL
- Not Defteri - 3,79 TL

Buttons:

- Ekle >>> (Add)
- <<< Çıkar (Remove)
- Hesapla (Calculate)

Bottom Section:

- Seçilen Ürün (Selected Product): 1,49 TL
- Toplam (Total): 5,67

Örnek

- **Uygulama Adımları**
- **1-** Ürünlerin tutulması için bir Struct oluşturulur. Bu ürün yapısının ToString metodu tekrar yazılmıştır. Bunun nedeni, ListBox kontrolünde listelenen nesnelerin görüntülediği değer ToString metodu çağırılarak belirlenir. Dolayısıyla liste kutularında istenen formatta değerın gözükmesini sağlamak için ToString metodunun tekrar yazılması gerekir.
-
- **public struct Urun {**
- **public string Ismi; public double Fiyat;**
- **public Urun(string UrunIsmi, double UrunFiyat)**
- **{ Ismi = UrunIsmi; Fiyat = UrunFiyat;}**
- **public override string ToString()**
- **{ return string.Format("{0} - {1:C}", Ismi, Fiyat); }**
- **}**

Örnek

- 2- Liste kutularının özellikleri ayarlanır ve içine eleman doldurulur.
- **private void Form1_Load(object sender, EventArgs e)**
 - { **listBox1.SelectionMode = SelectionMode.MultiExtended;**
 - **listBox2.SelectionMode = SelectionMode.MultiExtended;**
 - **UrunEkle(); }**
- **public void UrunEkle() {**
 - **Urun u = new Urun();**
 - **u = new Urun("Kalem", 1.49);** **listBox1.Items.Add(u);**
 - **u = new Urun("Silgi", 0.39);** **listBox1.Items.Add(u);**
 - **u = new Urun("Defter", 4.99);** **listBox1.Items.Add(u);**
 - **u = new Urun("Cetvel", 1.99);** **listBox1.Items.Add(u);**
 - **u = new Urun("Pergel", 2.99);** **listBox1.Items.Add(u);**
 - **u = new Urun("Not Defteri", 3.79);** **listBox1.Items.Add(u);**
 - **}**

Örnek

- **3-Tedarikçi** liste kutusundan, **stok** liste kutusuna öğe aktarılması için, seçilen değerler önce liste kutusuna eklenir. Daha sonra bu seçilen değerler, diğer listede olmayacağı için tek tek çıkartılır.
- **private void button2_Click(object sender, EventArgs e)**
- **{ // Tedarikçiden alınan ürünler stok listesine eklenir**
- **foreach (object item in listBox1.SelectedItems)**
- **{ listBox2.Items.Add(item); }**
- **// Stok listesine eklenen tüm ürünler tedarikçi listesinden çıkartılır**
- **foreach (object item in listBox2.Items)**
- **{ listBox1.Items.Remove(item); }**
- **button3.Enabled = true;**
- **button4.Enabled = true;**
- **}**

Örnek:

- 4-Stok listesinden öge çıkarmak için, ekleme işlemine benzer kodlar çalıştırılır.
- **private void button3_Click(object sender, EventArgs e)**
- { // Tedarikçiden alınan ürünler stok listesine eklenir
- **foreach (object item in listBox2.SelectedItems)**
- { **listBox1.Items.Add(item);**
- // Stok listesine eklenen tüm ürünler tedarikçi listesinden çıkartılır
- }
- **foreach (object item in listBox1.Items)**
- { **listBox2.Items.Remove(item);** }
- **if (listBox2.Items.Count == 0)**
- { **button3.Enabled = false; button4.Enabled = false;** }
- }

Örnek

- 5-Stoktaki toplam fiyatın hesaplanması işlemi, ürünlerin fiyatlarının alınıp toplanması ile gerçekleşir.
- **private void button4_Click(object sender, EventArgs e)**
- **{ double toplam = 0;**
- **for (int i=0; i<=listBox2.Items.Count - 1; i++)**
- **{ Urun urun = (Urun)(listBox2.Items[i]) ;**
- **toplam += urun.Fiyat;**
- **}**
- **textBox2.Text = System.Convert.ToString(toplam);**
- **}**

Örnek

- 6-Stok listesindeki bir öğenin seçildiği durumda, bu öğenin fiyatı görüntülenir.
- **private void listBox2_SelectedIndexChanged(object sender, EventArgs e)**
- **{**
- **Urun secilen = new Urun();**
- **secilen = (Urun)(listBox2.SelectedItem);**
- **textBox1.Text = string.Format("{0:C}", secilen.Fiyat);**
- **}**
- **}**
- **}**

CheckedListBox KONTROLÜ

- Liste kutusunun tüm özellik, metot ve olaylarını alır ve listedeki öğelerin işaret kutusu ile gösterilmesini sağlar.

- `private void button1_Click(object sender, System.EventArgs e)`
- `{`
- `int eleman=checkedListBox1.Items.Count;`
- `for(int i=0;i<eleman;i++)`
- `{ if(checkedListBox1.GetItemChecked(i)==true)`
- `listBox1.Items.Add(checkedListBox1.Items[i]);`
- `}`
- `}`



Örnek:

- Kategori başına stoktaki toplam ürünlerin gösterildiği bir uygulamada listelenen kategorileri seçmek için CheckedListBox kullanımı.

Form2

Kategori Adı:

Ekle

☐ Film
☐ Müzik
☐ Bilgisayar
☐ Kitap

label2

Form2

Kategori Adı:

Ekle

☒ Film
☒ Müzik
☐ Bilgisayar
☐ Kitap

Seilen kategorilerdeki toplam ürün: 1110

Örnek:

- `private void chlistKategoriler_SelectedIndexChanged(object sender, EventArgs e)`
- `{ int toplam=0;`
- `// Listedeki seçilen öğelerin ürün adeti toplanır.`
- `for (int i = 0; i <= chlistKategoriler.Items.Count - 1; i++)`
- `{ if (chlistKategoriler.GetItemChecked(i))`
- `{ object secilen = null;`
- `secilen = chlistKategoriler.Items[i];`
- `// Stok durumunu gsteren fonksiyon arlr`
- `toplam += StokDurumu(secilen.ToString());`
- `}`
- `}`
- `lblToplamUrun.Text = "Seilen kategorilerdeki toplam ürün: " + toplam;`
- `}`

Örnek:

- // Kategoriye göre, stoktaki ürünlerin belirlenmesi

```
public int StokDurumu(string kategori)
{
    switch (kategori)
    {
        case "Film": return 1100;
        case "Müzik": return 982;
        case "Bilgisayar": return 302;
        case "Kitap": return 1222;
        default: return 10;
    }
}
```

- //Ekleme işlemi

```
private void btnKategoriEkle_Click(object sender, EventArgs e)
{
    chlistKategoriler.Items.Add(txtKategoriAdi.Text);
}
```


CheckedListBox KONTROLÜ

- **CheckedListBox Özellikleri**

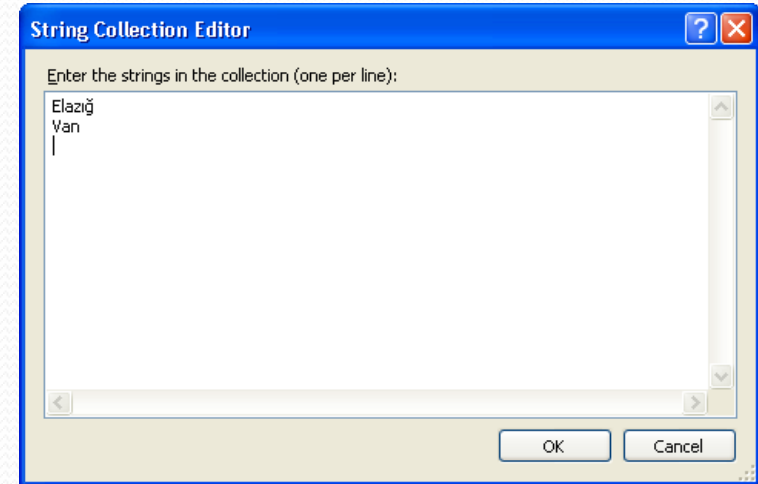
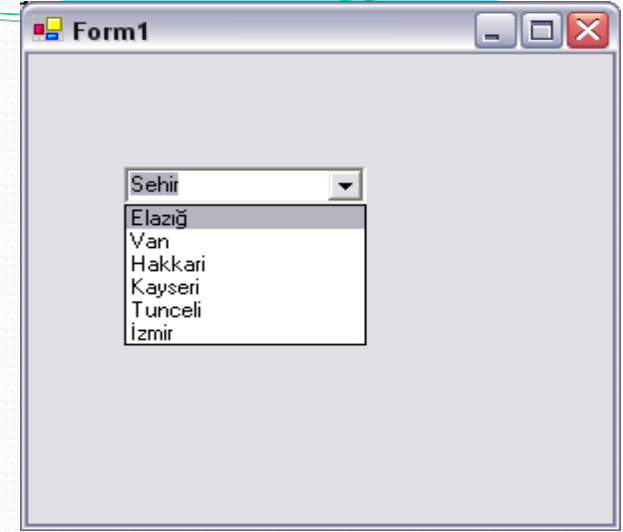
Özellik	Açıklama
CheckedItems	Liste kutusunda işaretlenmiş öğeleri tutar
CheckedIndices	Liste kutusunda işaretlenmiş öğelerin indislerini tutar
CheckOnClick	Liste kutusunda öğeye tıklandığı zaman işaretlenmesini belirler. False ise ilk tıklamada öğe seçilir, ikinci tıklamada seçme kutusu işaretlenir.

- **CheckedListBox Metotları**

Metot	Açıklama
GetItemSelected	Parametre olarak verilen indisteki öğenin seçili olup olmadığını döndürür
SetItemSelected	İlk parametrede verilen indisteki elemanın seçili olup olmadığını, ikinci parametrede verilen Boolean değeri ile belirler

ComboBox KONTROLÜ

- Liste kutusu ile aynı özelliklere sahiptir. Ancak listelenen öğeler açılan bir kutuda görüntülenir ve listeden en fazla bir tane öğe seçilebilir.
- Liste kutusuna göre bir başka farklılığı ise, isteğe bağlı olarak, kullanıcının açılan kutu üzerinde değer girebilmesidir. Dolayısıyla bir TextBox kontrolü gibi de davranabilir.
- Listede bulunmasını istediğimiz elemanlar Items-> String Collection Editor diyalog kutusuna yazılır.

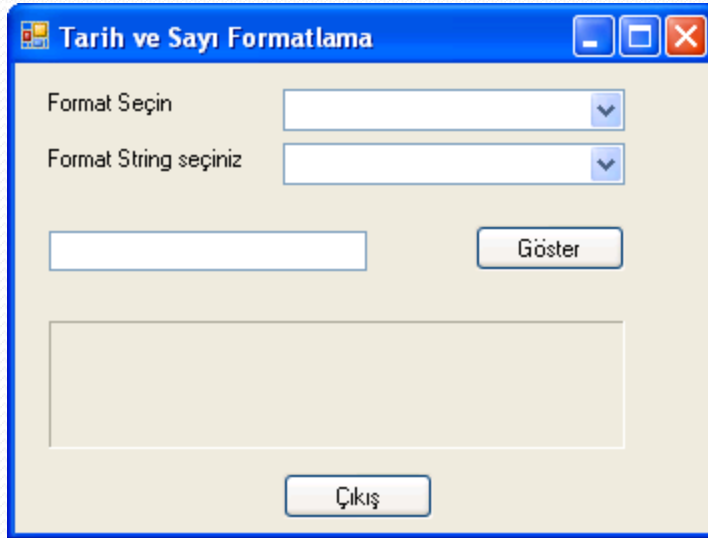


ComboBox KONTROLÜ

- Sayfa içerisinde kodlar ile ComboBox a eleman eklenebilir.
- `private void Form1_Load(object sender, System.EventArgs e)`
- `{`
- `string [] meslekler = { "İşçi","Memur","Mühendis","Eğitimci",
"Programcı", "Tekniker", "Veteriner" };`
- `comboBox1.Items.AddRange(meslekler);`
- `}`

Örnek

- Tarih ve sayı formatlarını, kullanıcının seçimine bırakarak bir sayı veya tarih yazdırma işlemi ComboBox kontrolleri ile yapılabilir.



The screenshot shows a Windows application window titled "Tarih ve Sayı Formatlama". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main content area is light beige. It contains two dropdown menus: "Format Seçin" and "Format String seçiniz". Below these is a text input field and a "Göster" button. At the bottom of the window is a "Çıkış" button.

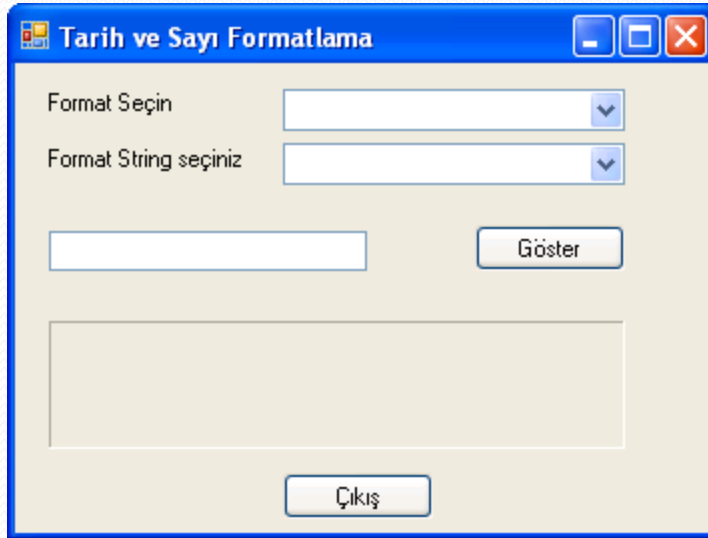
ComboBox KONTROLÜ

- ComboBox Özellikleri

Özellik	Açıklama
DropDownStyle	Kontrolün listeleme stilini belirler. Simple stil, listedeki sadece bir öğeyi görüntüler. DropDown stili, listenin tüm elemanlarını görüntüleyerek seçilmelerini ve kullanıcının değer girmesini sağlar. DropDownList kullanıcının değer girmesini engeller.
DropDownWidth	ComboBox kontrolünün açılan listesinin genişliğini belirler.
MaxDropDownItems	Kontrole eklenebilecek maksimum öğe sayısını belirler.
MaxLength	Kullanıcının girebileceği maksimum karakter sayısını belirler.
SelectedText	Seçilen öğenin görüntülenen yazısını belirler.

Örnek

- Tarih ve sayı formatlarını, kullanıcının seçimine bırakarak bir sayı veya tarih yazdırma işlemi ComboBox kontrolleri ile yapılabilir.



The screenshot shows a Windows application window titled "Tarih ve Sayı Formatlama". The window has a blue title bar with standard Windows controls (minimize, maximize, close). The main area is light beige. It contains two dropdown menus: "Format Seçin" and "Format String seçiniz". Below these is a text input field and a "Göster" button. At the bottom is a "Çıkış" button.

Örnek

- **1-ComboBox kontrollerinin özelliklerinin ayarlanması ve format tiplerine öğe eklenmesi**
- `private void Form1_Load (System.Object sender, System.EventArgs e)`
- `{`
- `cmbFormat.DropDownStyle = ComboBoxStyle.DropDownList;`
- `cmbFormatString.DropDownStyle = ComboBoxStyle.DropDownList;`
- `cmbFormat.Items.Add("Tarih Formatı");`
- `cmbFormat.Items.Add("Sayı Formatı");`
- `}`

Örnek

- **2-** Tarih ya da sayı formatlarından biri seçildiği zaman, ikinci ComboBox kontrolüne değişik format seçenekleri eklenir.
- `private void cmbFormat_SelectedIndexChanged (object sender, EventArgs e)`
- `{ cmbFormatString.Items.Clear();`
- `switch (cmbFormat.SelectedIndex)`
- `{ case 0:`
- `cmbFormatString.Items.Add("dd - MM - yyyy");`
- `cmbFormatString.Items.Add("yyyy*MM*dd hh:mm");`
- `cmbFormatString.Items.Add("dddd dd.MM.yy hh:mm:ss"); break;`
- `case 1:`
- `cmbFormatString.Items.Add("C");`
- `cmbFormatString.Items.Add("P");`
- `cmbFormatString.Items.Add("N"); break;`
- `}`
- `}`

Örnek

- **3-** Format seçildikten sonra metin kutusuna girilen değer alınır ve ilgili formatta gösterilir.
- ```
private void btnGoster_Click(object sender, EventArgs e) {
```
- ```
    switch (cmbFormat.SelectedIndex)
```
- ```
 {
```
- ```
        case 0:
```
- ```
 DateTime d = txtYazi.Text;
```
- ```
            lblSonuc.Text = d.ToString(cmbFormatString.Text);
```
- ```
 break;
```
- ```
        case 1:
```
- ```
 int i = txtYazi.Text;
```
- ```
            lblSonuc.Text = i.ToString(cmbFormatString.Text);
```
- ```
 break; }
```
- ```
}
```

VISUAL STUDIO.NET ve FORM UYGULAMALARI

The image displays six screenshots of a Windows application titled "Tarih ve Sayı Formatlama" (Date and Number Formatting). The application is organized into two rows of three windows each. Each window contains a "Format Seçin" (Select Format) dropdown menu, a "Format String seçiniz" (Select Format String) dropdown menu, an input field for a value, a "Göster" (Show) button, a text area for the formatted result, and a "Çıkış" (Exit) button.

Top Row (Date Formatting):

- Window 1:** "Format Seçin" is set to "Tarih Formatı" (Date Format). "Format String seçiniz" is set to "dddd dd.MM.yy hh:mm:ss". The input field contains "29.04.2009 13:07:05". The "Göster" button is visible. The text area displays "Çarşamba 29.04.09 01:07:05".
- Window 2:** "Format Seçin" is set to "Tarih Formatı". "Format String seçiniz" is set to "dd - MM - yyyy". The input field contains "29.04.2009". The "Göster" button is visible. The text area displays "29 - 04 - 2009".
- Window 3:** "Format Seçin" is set to "Tarih Formatı". "Format String seçiniz" is set to "yyyy*MM*dd hh:mm". The input field contains "29.04.2009". The "Göster" button is visible. The text area displays "2009*04*29 12:00".

Bottom Row (Number Formatting):

- Window 4:** "Format Seçin" is set to "Sayı Formatı" (Number Format). "Format String seçiniz" is set to "C". The input field contains "50". The "Göster" button is visible. The text area displays "50,00 TL".
- Window 5:** "Format Seçin" is set to "Sayı Formatı". "Format String seçiniz" is set to "P". The input field contains "50". The "Göster" button is visible. The text area displays "%5.000,00".
- Window 6:** "Format Seçin" is set to "Sayı Formatı". "Format String seçiniz" is set to "N". The input field contains "50". The "Göster" button is visible. The text area displays "50,00".