

FORM EKRANI

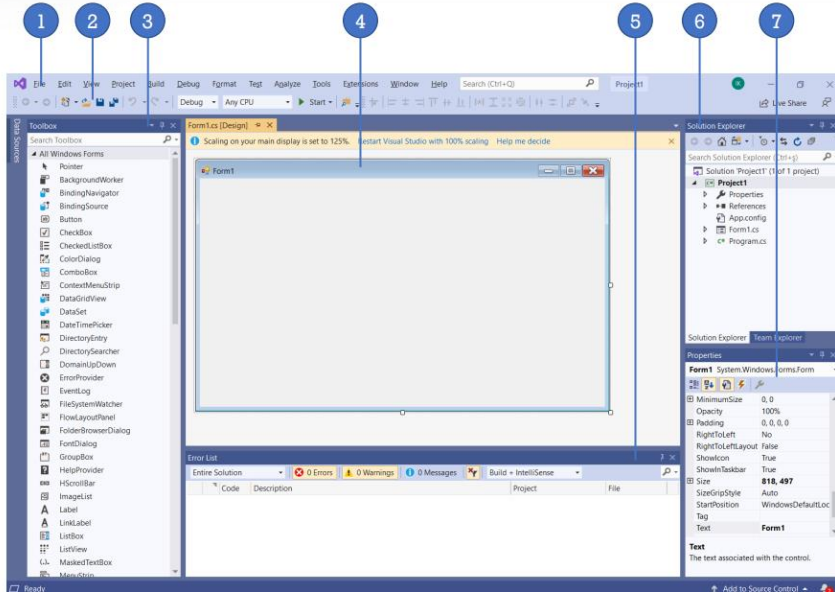
FORM EKRANI

- ✓ *Form Ekranına Giriş*

KAZANIMLAR

- ✓ *Visual Studio menülerini sıralar.*
- ✓ *Visual Studio menülerinin görevlerini açıklar.*
- ✓ *Windows form uygulamalarını kavrar.*
- ✓ *Program üzerinde Visual Studio menülerini kullanır.*

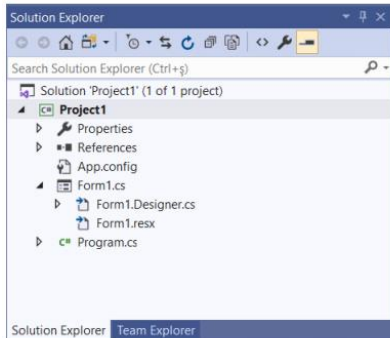
FORM EKRANI



1. Menü Çubuğu (Menu Bar)
2. Standart Araç Çubuğu (Standard Toolbar)
3. Araç Kutusu (ToolBox)
4. Form Tasarımcısı (Forms Designer)
5. Hata Listesi (Output Window-Error List),
6. Çözüm Gezgini (Solution Explorer)
7. Özellikler ve Olaylar (Properties / Events)

Çözüm Gezgini (Solution Explorer)

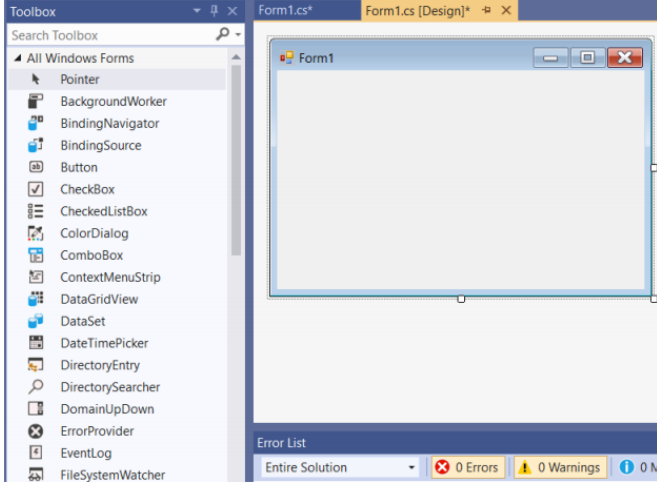
- ✓ Çözüm Gezgini (Solution Explorer), çözüm ve çözüme bağlı projelerdeki kontrol, sınıf, metod vb. bileşenleri görüntülemek amacıyla kullanılan Visual Studio aracıdır.
- ✓ Solution, projeleri gruplandırarak bir çatı altında toplamak amacıyla kullanılır. Bir Solution içerisinde, bir veya daha fazla proje olabilir. Solution ile ilgili kodlar *.sln uzantılı dosyalarda tutulmaktadır.



Dosya Adı	Açıklama
App.config	Uygulama ile ilgili genel tanımlama ve bilgileri içerir.
Form1.cs	Windows Form Uygulaması Kod sayfasıdır. (Her form için ayrı bir kod sayfası mevcut olabilir.)
Form1.Designer.resx	Form tasarımında kullanılan nesnelerin görsel özellikleri ile ilgili kodların yer aldığı dosyadır. C# tarafından otomatik olarak oluşturulur.
Form1.resx	Form tasarımında kullanılan yazı, resim vb. medya öğelerini içeren "resource-kaynak" dosyadır.
Program.cs	Proje ortam ayarlarını içeren dosyadır. (Örneğin birden çok forma sahip Projelerde, başlangıç formu bu dosya içerisinde belirlenir.)

Toolbox

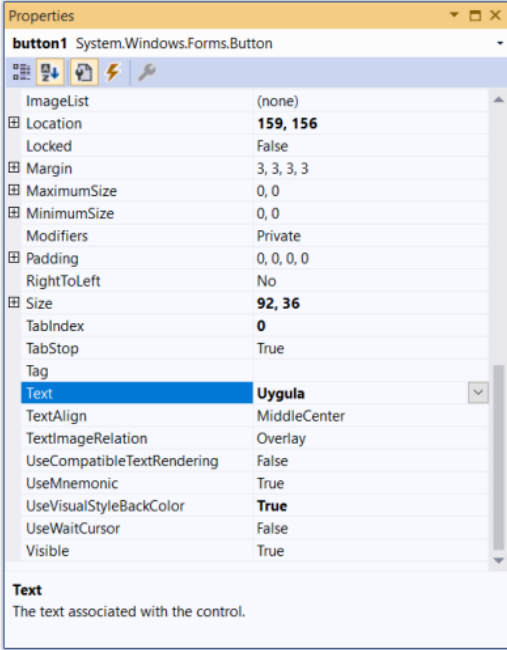
- ✓ Windows tabanlı uygulamalar geliştirirken sıkça kullanacağımız bir grup kontrol vardır.
- ✓ Form kontrolü hariç diğer bütün kontroller Toolbox panelinden seçilir.
- ✓ Bu kontroller sürüklenip Form üzerine istenilen pozisyona bırakılır.



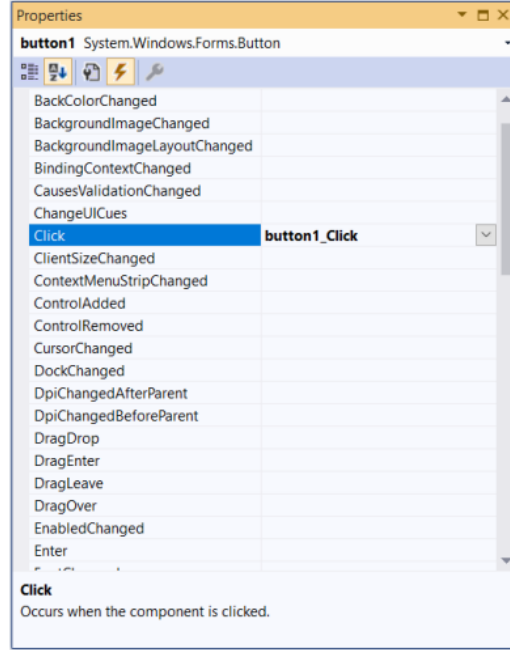
Toolbox kapalı ise **Ctrl + Alt + X** tuşlarına birlikte basarak aktif hale getirilebilir.

Özellikler Penceresi (Properties & Events)

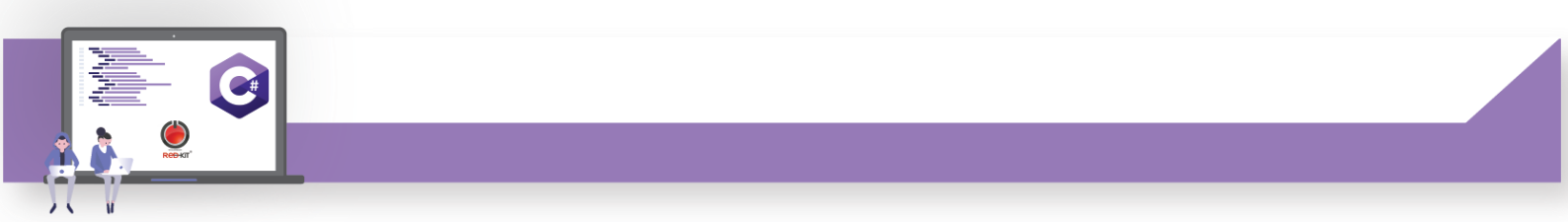
Seçili olan nesnenin özelliklerini ve olaylarını görüntüleyip düzenlemek amacıyla kullanılan Visual Studio aracıdır.



Properties



Events



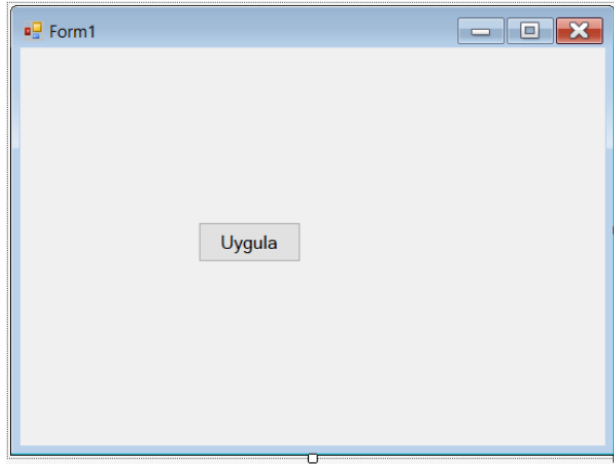
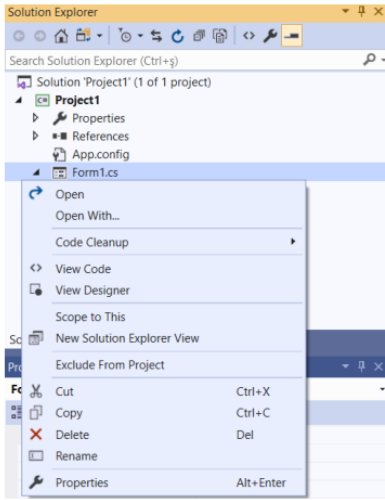
Özellikler Penceresi (Properties & Events)

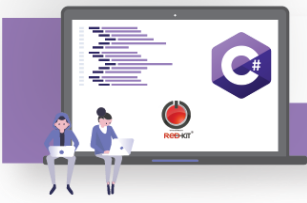
Simge	Simge Adı	Açıklama
	Özellikler (Properties)	Seçili nesneye ait özellikleri listeler.
	Olaylar (Events)	Seçili nesneye ait olayları listeler.
	Kategorik Sıralama	Özellik ve olayların kategorilere ayrılarak listelenmesini sağlar
	Alfabetik Sıralama	Özellik ve olayların alfabetik olarak sıralı listelenmesini sağlar

Kodu Görüntüleme

Çözüm Gezgini penceresinde Form1 üzerinde sağ klik yapılarak elde edilen menüden proje formu ve kaynak kodları görüntülenebilmektedir.

Diğer bir yol ise, Designer alanında Form1'in üzerine çift tıklamaktır.







Kodu Görüntüleme

```
Form1.cs* x Form1.cs [Design]*
Project1
Project1.Form1
Form1_Load(object sender, EventArgs e)

7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace Project1
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void button1_Click(object sender, EventArgs e)
21         {
22         }
23
24         private void Form1_Load(object sender, EventArgs e)
25         {
26         }
27     }
28 }
29
30
31
```

Start (Başlat)

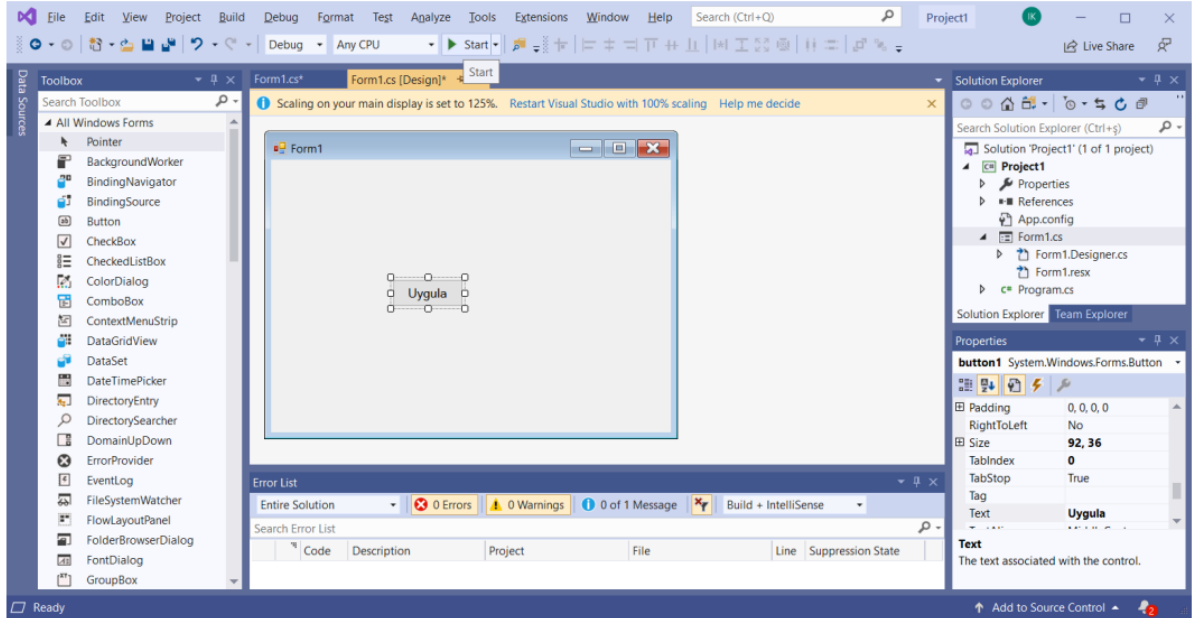
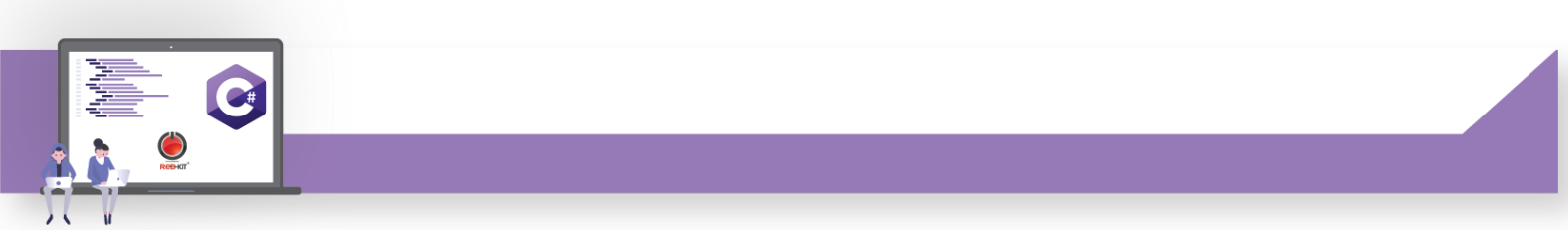
C# program geliştirme işlemlerinde yapılan değişiklikler için Kaydetme işlemi yapılmalıdır

Program Kaydetme Seçenekleri		
Simge	Kısayol Adı	Açıklama
	Ctrl + S	Aktif Form dosyaları kaydedilir.
	Ctrl + Shift + S	Tüm Dosyalar Kaydedilir.

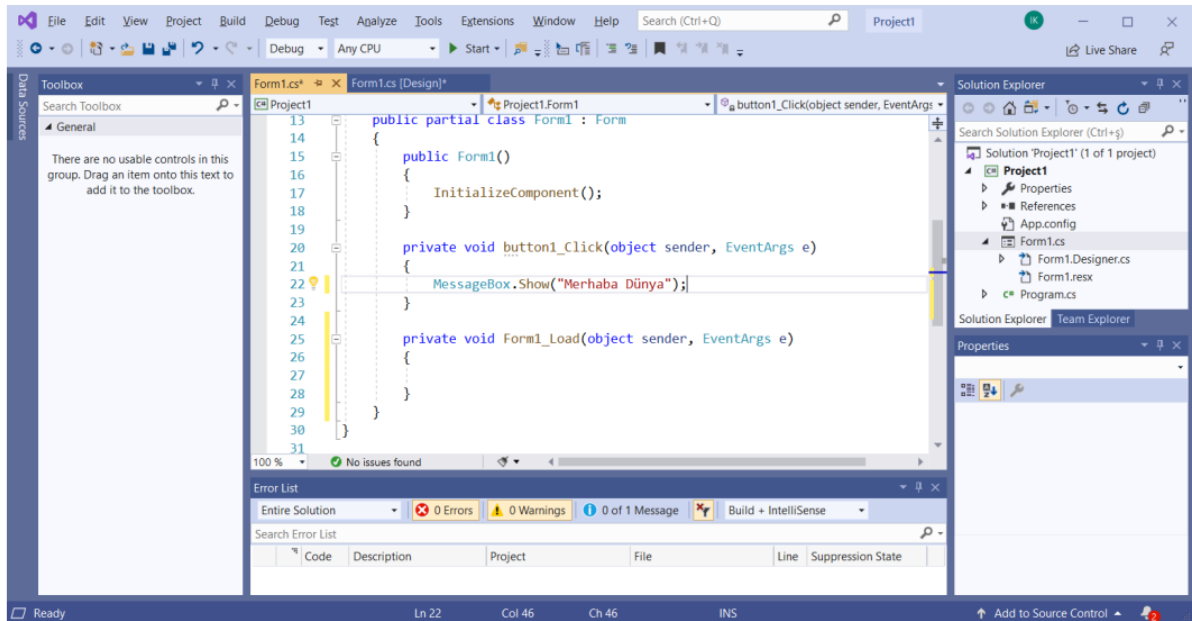
PROGRAMIN DERLENMESİ VE BAŞLATILMASI

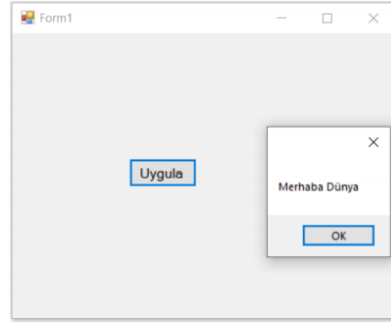
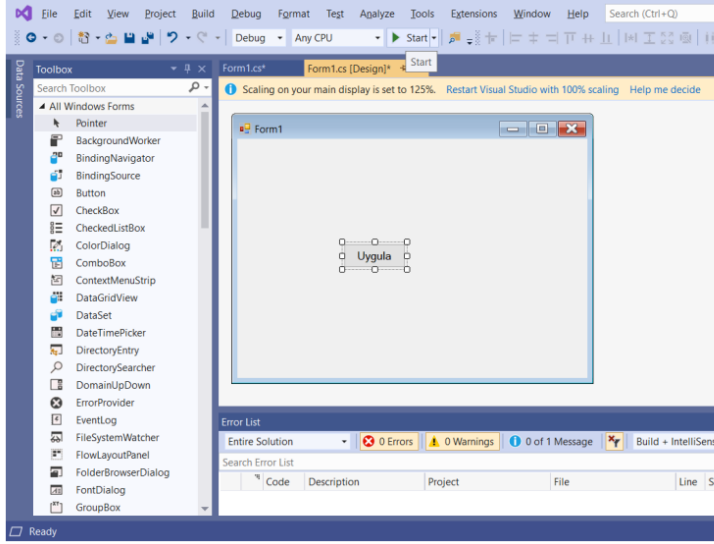
C# dilinde yazılmış bir Windows Forms Uygulamasını çalıştırmak için aşağıdaki yollardan birisi izlenebilir:

- ✓ Araç çubuğundan “Başlat (START)” butonuna tıklamak
- ✓ Menüden “Hata Ayıkla (DEBUG) ->Hata Ayıklamayı Başlat (START DEBUGGING)” a tıklamak
- ✓ F5 fonksiyon tuşunu kullanmak



Designer ekranından butonun üzerine çift tıklanarak Click olayına komut yazılabilir. Oluşturulan butonun Click olayına yazılan komut aşağıdaki gibidir.

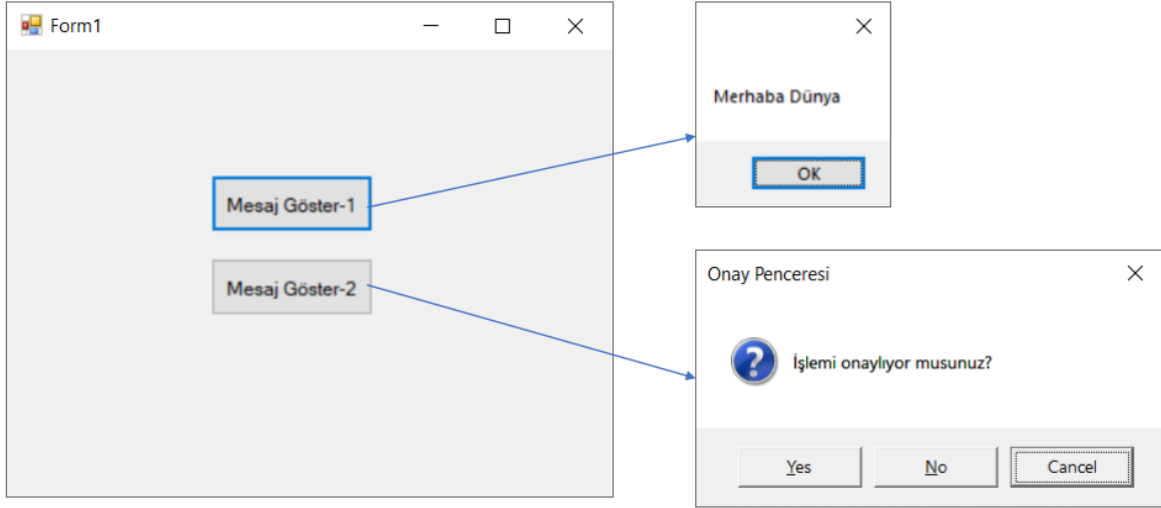




MessageBox

Kullanıcıya diyalog penceresi içerisinde mesaj verme işlemi için MessageBox sınıfı Show metodu kullanılır

```
Form1.cs*  Form1.cs [Design]*
Project1  Project1.Form1  button2_Click(o
17
20 private void button1_Click(object sender, EventArgs e)
21 {
22     MessageBox.Show("Merhaba Dünya");
23 }
24
25 private void button2_Click(object sender, EventArgs e)
26 {
27     MessageBox.Show("İşlemi onaylıyor musunuz?", //Mesaj içeriği
28         "Onay Penceresi", //Mesaj Başlığı
29         MessageBoxButtons.YesNoCancel, //Mesaj Butonları
30         MessageBoxIcon.Question, //Mesaj Iconu
31         MessageBoxDefaultButton.Button3, //Default Seçili Buton
32         MessageBoxOptions.DefaultDesktopOnly //PencereVerleşimSeçeneği
33     );
34 }
35 }
36 }
```



1. MessageBox.Show metodunun alabileceği parametreler şunlardır:

- ✓ Mesaj İçeriği
- ✓ Mesaj Başlığı
- ✓ Butonlar
- ✓ Icon
- ✓ Default Seçili Buton
- ✓ Mesaj Penceresi Yerleşim Seçenekleri

2. MessageBox.Show metodunun alabileceği bu parametrelerin tanımlama sıralamalarına dikkat edilmelidir.

3. Ayrıca sonraki parametrenin kullanılabilmesi için önceki parametrelerin kullanılması gereği de unutulmamalıdır.

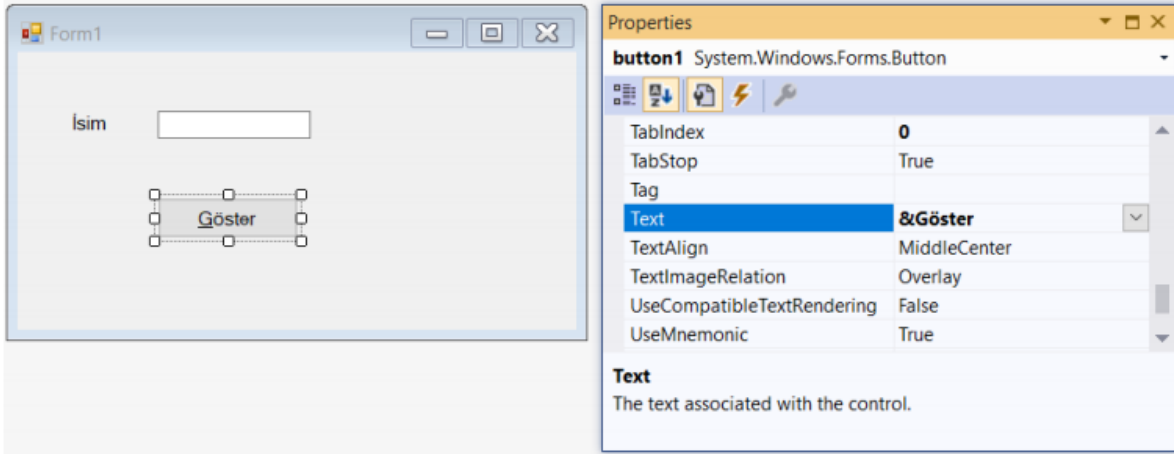
Icon Adı	Icon Görünümü
MessageBoxIcon.Information MessageBoxIcon.Asteriks	
MessageBoxButtons.Question	
MessageBoxButtons.Error MessageBoxButtons.Hand MessageBoxButtons.Stop	
MessageBoxButtons.Exclamation MessageBoxButtons.Warning	
MessageBoxButtons.None veya Bu parametre hiç kullanılmazsa	Icon Görünmez



Button

Windows uygulamalarında, form üzerinde komut düğmeleri olarak kullanılır.

Kısayol tuşları ile buton kontrolüne erişmek için & işareti kullanılır. Örneğimizde Alt+ G tuşuna basarak butona tıklanma sağlanır.



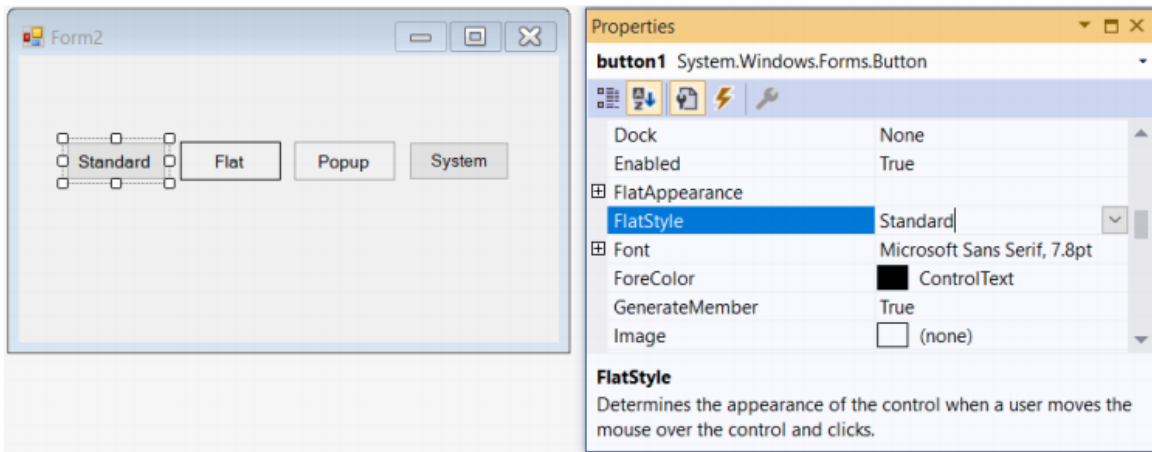
Enabled ve Visible Özellikleri: Butonun aktif veya görünür olmasını kontrol eder.

- ✓ 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.



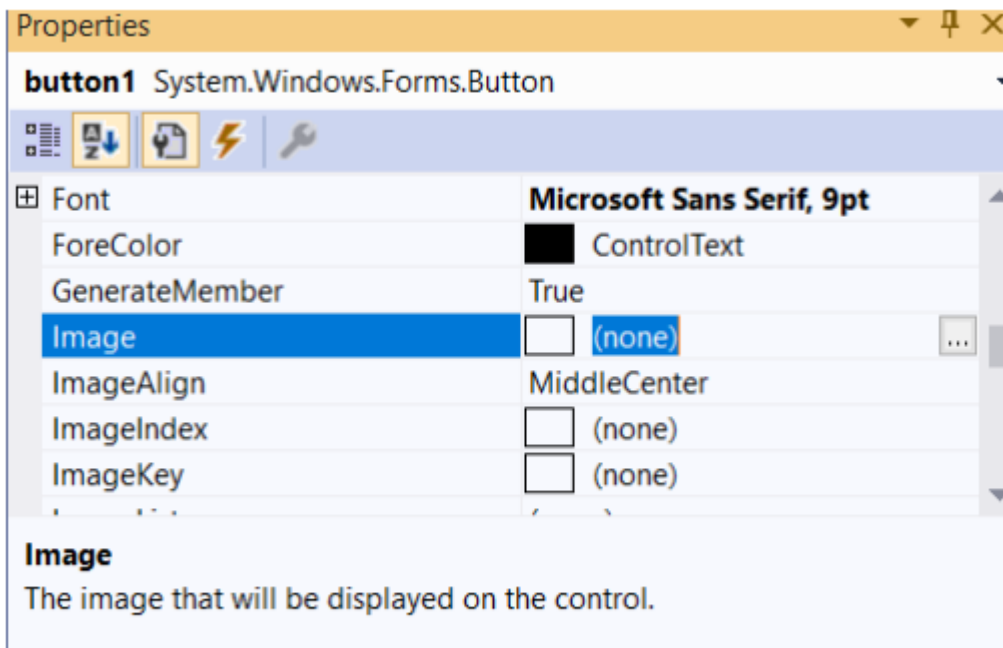


PerformClick() metodu:

Butonların çalışması için üzerine tıklamak gerekir. Ancak PerformClick() metodu ile buton tıklanmış gibi bir etki sağlanır.

```
private void Form1_Load(object sender, EventArgs e)
{
    button1.PerformClick();
}
```

Image Özelliği: Butonların üzerine resim yerleştirmek için kullanılır.



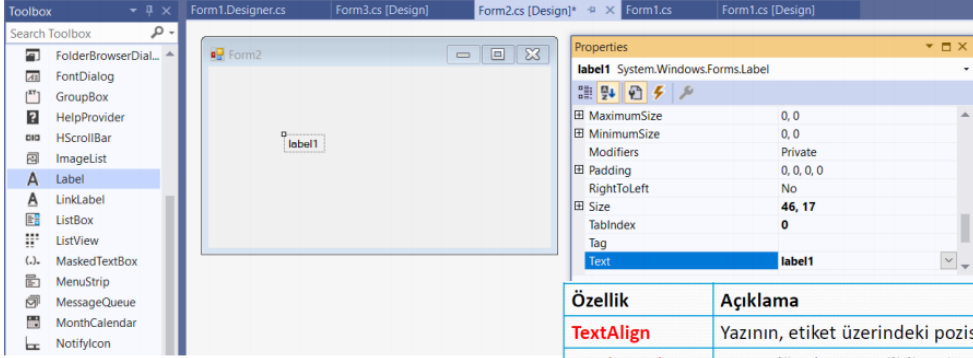
Örnek : Form ekranındaki butona bastığımızda form ekranımızın arka plan rengi değişsin.

```
private void button1_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Red;
}
```



Label

Label kontrolü Form üzerinde kullanıcıya bilgi vermek amaçlı kullanılan etikettir.



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

Metin kutuları, kullanıcıdan bilgi almak için kullanılır.

```
private void button1_Click(object sender, EventArgs e)
{
    int sayi1 = Convert.ToInt32(textBox1.Text);
    int sayi2 = Convert.ToInt32(textBox2.Text);
    textBox3.Text = Convert.ToString(sayi1 + sayi2);
}
private void button2_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

TextBox

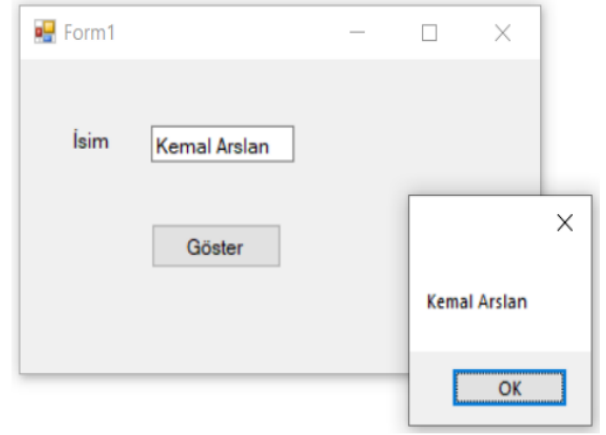
- ✓ **KeyDown:** Bir tuşa basıldığı durumda meydana gelen olay
- ✓ **KeyUp:** Tuştan elin çekildiği durumda meydana gelen olay
- ✓ **KeyPress:** Bir tuşa basılma ve çekme anı arasında meydana gelen olaydır.



- ✓ 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 için de geçerlidir.

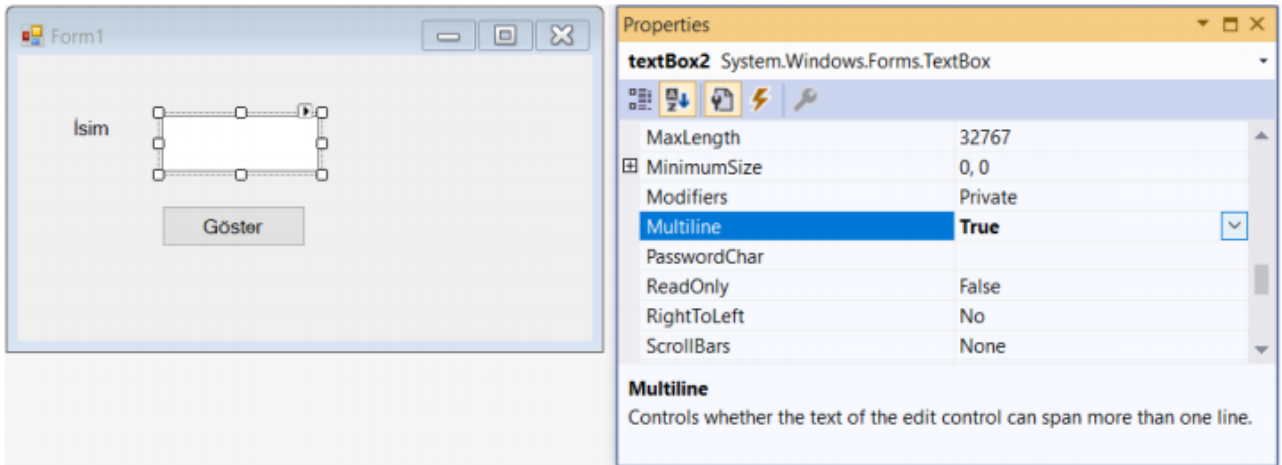
```
private void textBox2_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
        MessageBox.Show(textBox2.Text);
}
```

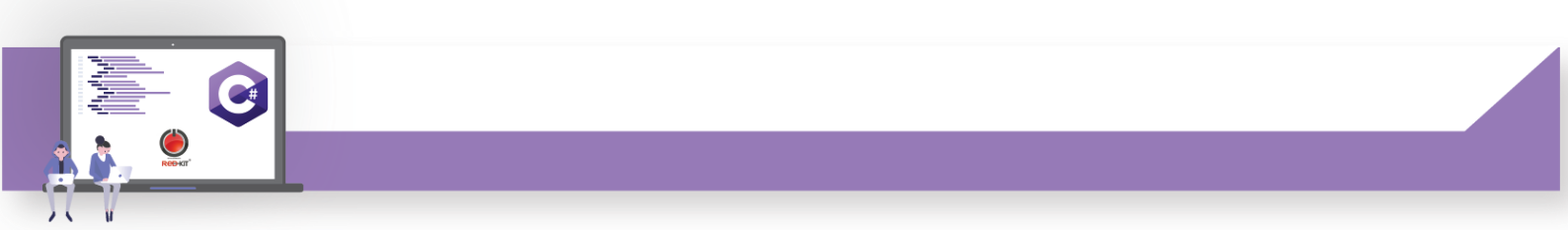
Yukarıdaki komut TextBox'ın KeyDown olayına yazılmıştır. ENTER tuşuna basıldığında, TextBox'a yazılmış olan metin MessageBox vasıtasıyla gösterilmektedir



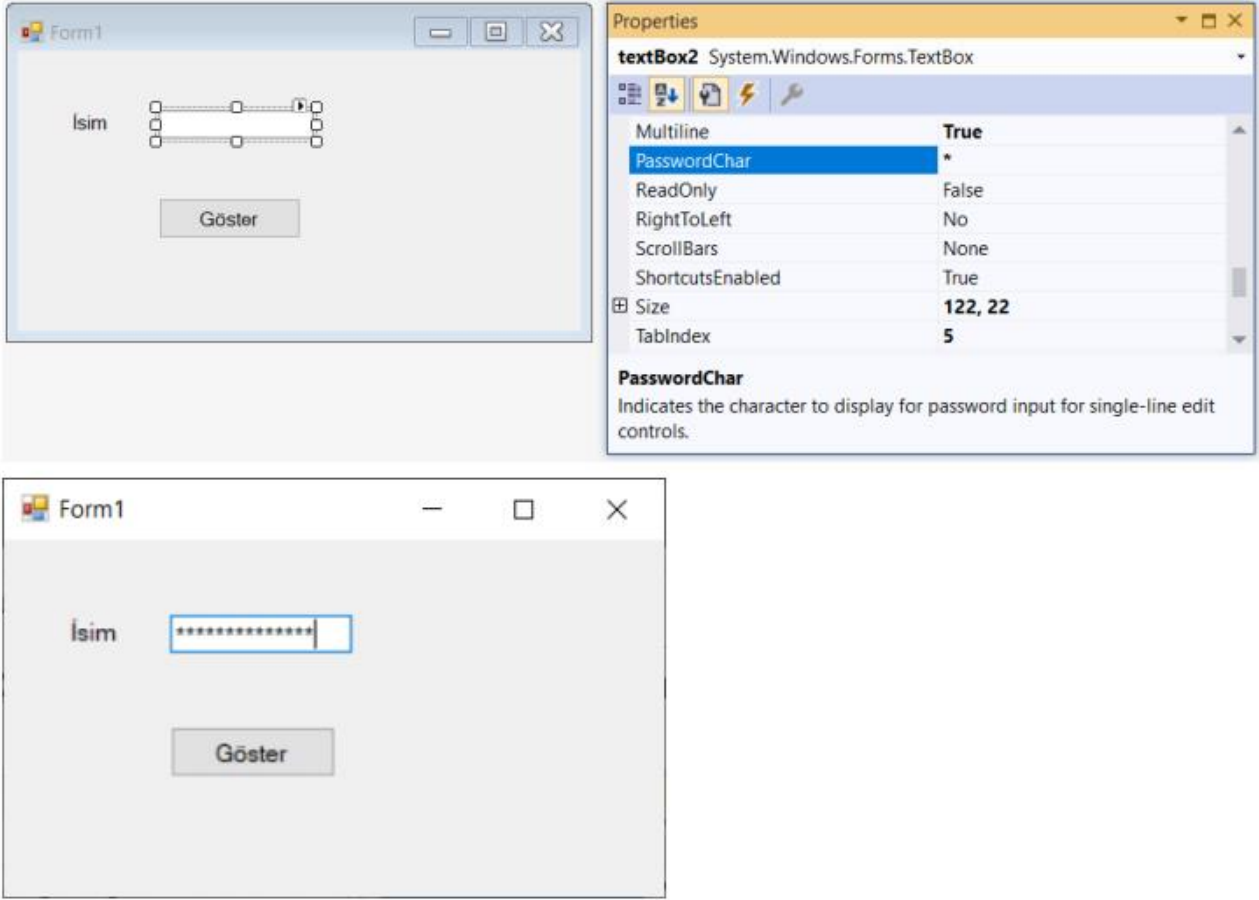
MultiLine Özelliği: Metin kutusuna bilginin çoklu satır olarak girilmesini sağlar.

ScrollBars Özelliği: Eğer satır sayısı metin kutusunun boyutundan fazla ise kaydırma çubuklarına gerek duyulabilir. Kaydırma çubukları eklemek için ScrollBars özelliği kullanılır, bu özellik dört değişik değer alır.





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



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.



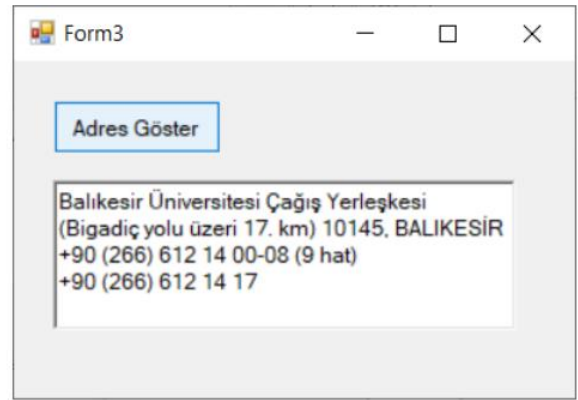
Textbox olayları

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

RichTextBox

Zengin metin kutusu anlamına gelmektedir. Normal metin kutusundan farklı olarak alt satıra da yazmaya imkan vermektedir.

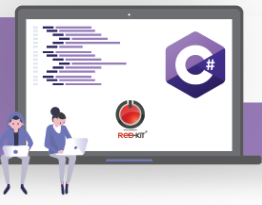
Kod yapısı olarak Lable veya TextBox ile aynı yapıda kullanılmaktadır



CheckBox

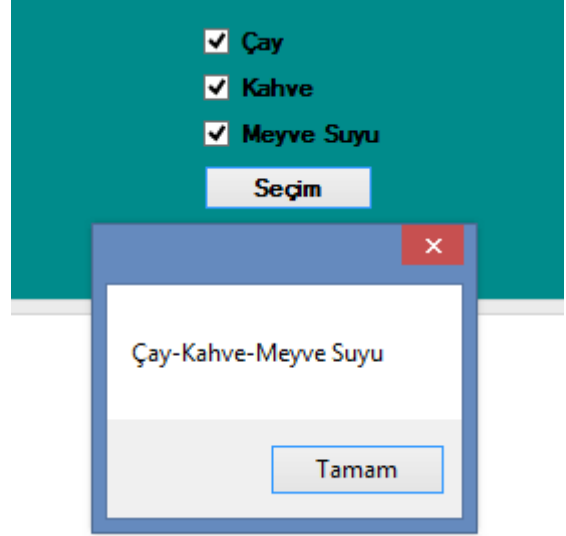
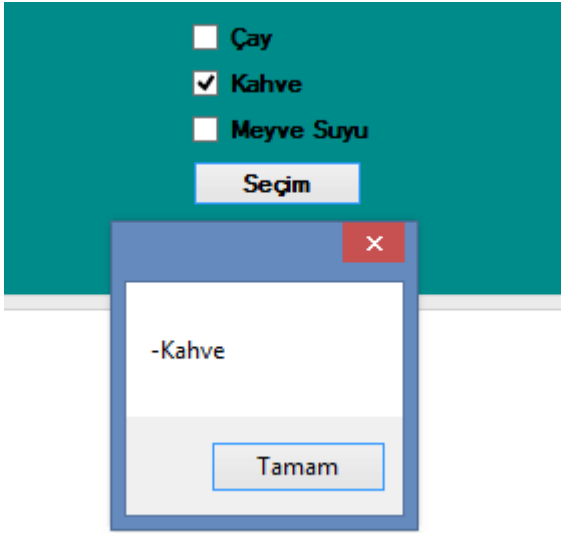
Kontrol Kutusu, kullanıcıya birden çok seçeneği seçme imkanı sağlar. Checked özelliği kontrol kutusunun seçilip seçilmediğini kontrol eder. Seçili ise , true değilse false değerini alır

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



Olay	Açıklama
CheckChanged	Seçme kutusunun durumu değiştiği zaman gerçekleşir.

Örnek: Checkbox dan seçilen içeceğin butona basıldığında mesajla bildirim vermesi. Kullanıcı Çoklu seçimde yapabilir.



```
private void button1_Click(object sender, EventArgs e)
{
    string secim="";
    if (checkBox1.Checked == true)
    { secim += "Çay"; }
    if (checkBox2.Checked == true)
    { secim += "-Kahve"; }
    if (checkBox3.Checked == true)
    { secim += "-Meyve Suyu"; }
    MessageBox.Show(secim+" içeceklerini sipariş ettiniz.");
}
```



Örnek: Beni Yakala isimli bir buton olsun. Buton rastgele hareket etsin ve butona tıkladığımızda karşımıza mesaj kutusunda “beni yakaladın” yazısı çıksın çıksın.

Form Ekranında Kullanılan Bileşenler

Timer → interval değeri:300

Enabled:True

Buton → name:btn

Text:beni yakala

Kod Blokları

```
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 WindowsFormsApp7
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {
                MessageBox.Show("Beni yakaladın");
                Application.Exit();
            }

            private void timer1_Tick(object sender, EventArgs e)
            {
                Random rastgele = new Random();
                btn.Location = new Point(rastgele.Next(500), rastgele.Next(500));
            }
        }
    }
}
```