Visual C# ile Görsel Programlama

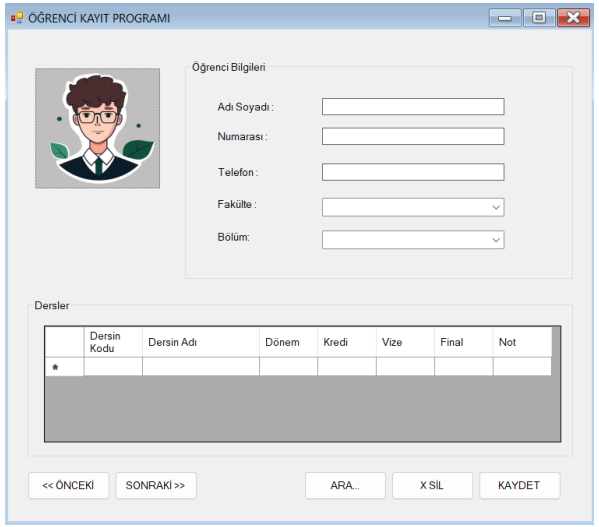
Öğr. Gör. Erkan Kaynak

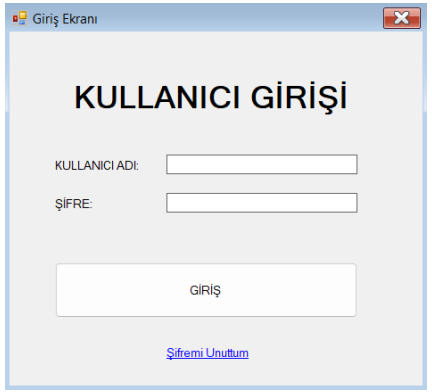
Dersin Amacı

- Öğrencilerin **C# programlama** dilindeki temel bilgilerini pekiştirerek, **Windows Forms** uygulamaları aracılığıyla görsel programlama becerilerini geliştirmektir.
- Öğrenciler, nesne yönelimli programlama (OOP) kavramlarını kullanarak **görsel arayüze** sahip yazılımlar geliştirmeyi öğrenirler.
- Ayrıca, **olay tabanlı programlama**, **form kontrolleri** ile etkileşim ve veri işleme konularında uygulamalı projeler yaparak yazılım geliştirme süreçlerine hakimiyet kazanırlar.

Görsel Programlama

- Görsel programlama, **kullanıcı arayüzlerini** (UI) ve **etkileşimli bileşenleri** kullanarak programlamaya olanak sağlayan bir yazılım geliştirme yaklaşımıdır.
- Bu yöntem, yazılım geliştirme sürecini **kolaylaştırarak**, kullanıcıların programın işlevselliği ile etkileşimini daha **anlaşılır ve sezgisel** hale getirir.
- Özellikle masaüstü ve mobil uygulamalarda yaygın olarak kullanılır.





Temel Özellikleri

1. **Olay Tabanlı Programlama:**
   Görsel programlamada kullanıcı etkileşimleri (tıklama, seçim yapma gibi) olaylar olarak tanımlanır ve bu olaylara verilen tepkiler programlanır.
2. **Kolay Kullanıcı Arayüzü (UI) Tasarımı:**
   Bileşenler sürükle-bırak yöntemiyle yerleştirilebilir ve bu sayede UI hızlı bir şekilde tasarlanabilir.
3. **Form ve Bileşen Tabanlı Yapı:**
   Formlar, UI temelini oluşturur ve üzerinde düğmeler, metin kutuları, etiketler gibi kontroller bulunur.
4. **Kullanıcı Deneyimi (UX) Odaklı:**
   Görsel programlama, uygulamaların kullanıcı dostu ve erişilebilir olmasını hedefler. Kullanıcıların sezgisel olarak uygulama ile etkileşim kurabilmesini sağlar.
5. **Arka Plan Kodu:**
   Görsel programlama sadece UI tasarımından ibaret değildir; her bir bileşenin arka planda işleyen kodlarla birleştirilmesi gerekir. Olaylar tetiklendiğinde belirlenen işlevler çalışır.

Ders İçeriği

1. Visual Studio

- Kurulum ve Ayarlar
- Proje Oluşturma
- Visual Studio Arayüzü
- Temel Özellikleri
- Olay Tabanlı Programlama
- Proje Yönetimi

Ders İçeriği

2. C# Programlama Dili Tekrarı

- Değişkenler, veri tipleri ve operatörler
- Koşullu ifadeler (if-else, switch)
- Döngüler (for, while, foreach)
- Sınıflar ve nesneler

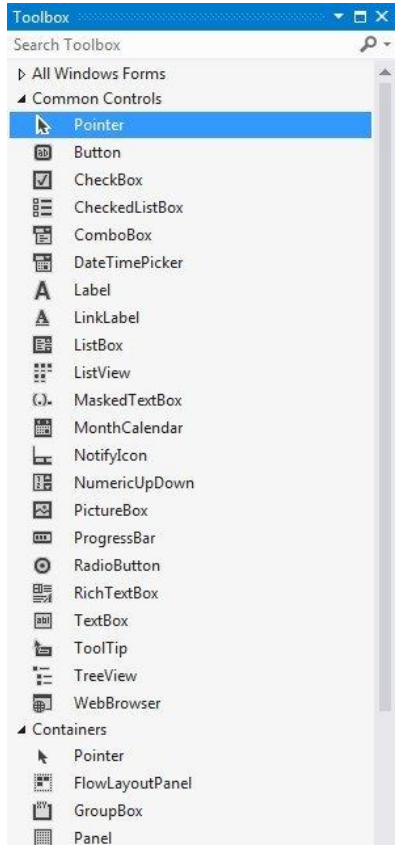
Ders İçeriği

3. Windows Forms Giriş

- Windows Forms nedir?
- Temel Bileşenler
- Olay Tabanlı Programlama
- Formlar Arası Veri Aktarma

4. Temel ve Sık Kullanılan Kontroller

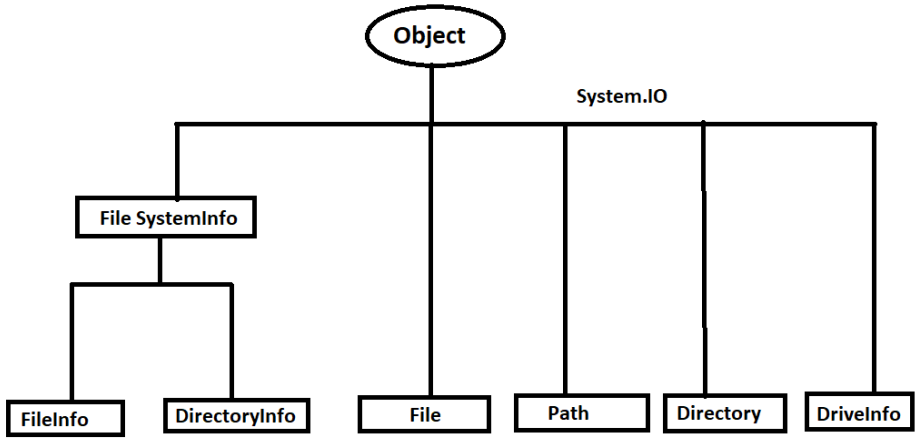
- Button
- TextBox
- Label
- CheckBox ve RadioButton
- ListBox ve Dropdown List



Ders İçeriği

5. Ek Kontroller ve Bileşenler

- Timer
- Layouts
- Dialog Pencereleri
- DataTable ve DataGrid

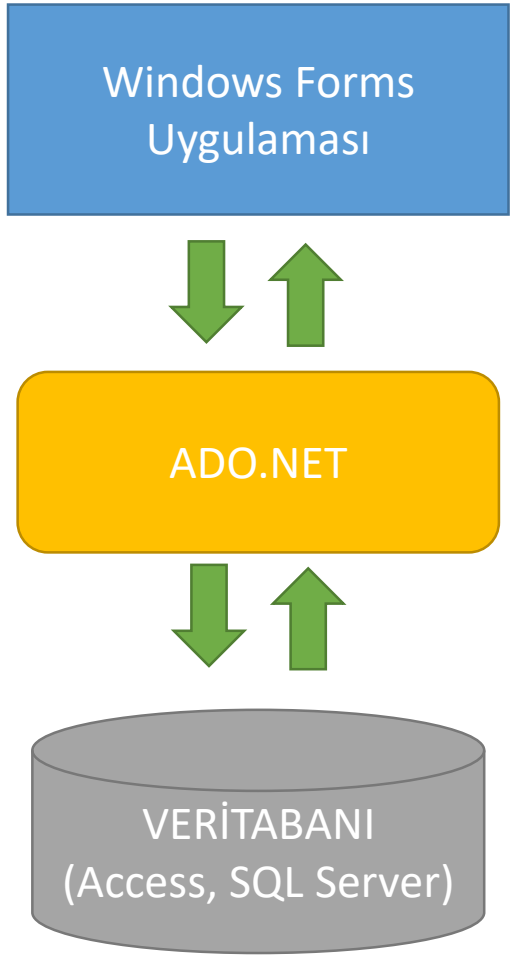


6. Dosya İşlemleri

- Dosya ve Klasör Oluşturma
- Dosyadan Okuma
- Dosyaya Yazma

Ders İçeriği

7. Veritabanı Kavramları ve SQL Kullanımı
   - Veritabanı Nedir?
   - Tablolar ve Alanlar
   - İlişkisel Veritabanları
   - SQL Sorgu Dili
   - Visual C# ile veritabanına bağlanma.
   - DataReader ve DataAdapter
   - Veri Ekleme ve Veri Çekme
   - CRUD İşlemleri

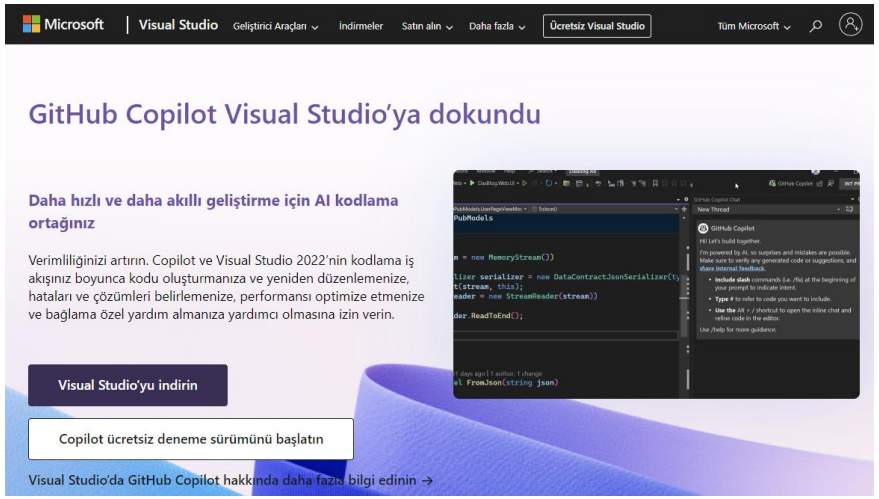


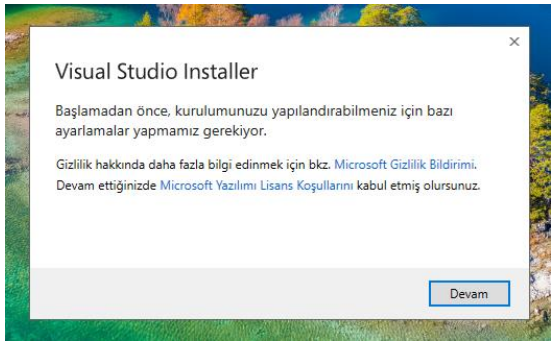
Bölüm 1: Visual Studio

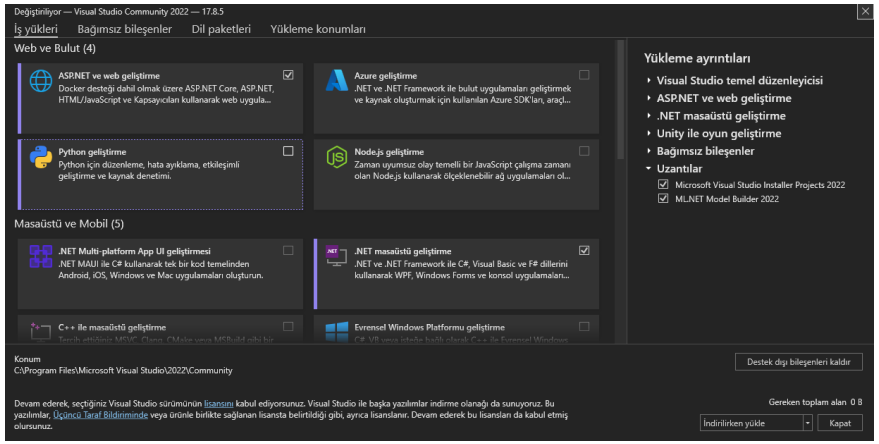
C# ile Görsel Programlama

Kurulum ve Ayarlar

1. Visual Studio resmi internet sitesine girin.
   - https://visualstudio.microsoft.com/tr/
2. [Visual Studio'yu İndirin] bağlantısına tıklayın.
3. Bilgisayarınıza inen «Visual Studio Setup» programını çalıştırın.
4. Kurulacak paketlerden «.NET Masaüstü Geliştirme» seçeneğini işaretleyin.
5. Program gerekli dosyaları bilgisayarınıza indirip kurulumu otomatik olarak yapacaktır.

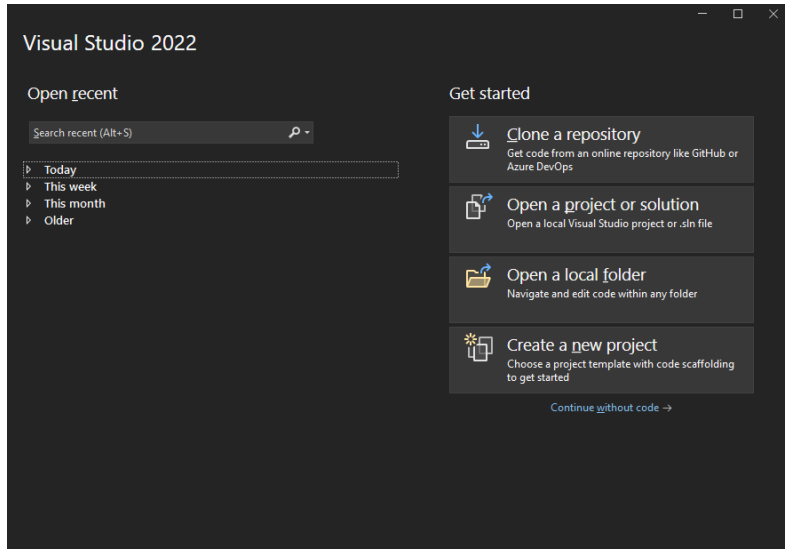

Microsoft | Visual Studio
Geliştirici Araçları
İndirmeler
Satın alın
Daha fazla
Ücretsiz Visual Studio
Tüm Microsoft
GitHub Copilot Visual Studio'ya dokundu
Daha hızlı ve daha akıllı geliştirme için AI kodlama ortağınız
Verimliliğinizi artırın. Copilot ve Visual Studio 2022'nin kodlama iş akışınız boyunca kodu oluşturmanıza ve yeniden düzenlemenize, hataları ve çözümleri belirlemenize, performansı optimize etmenize ve bağlama özel yardım almanıza yardımcı olmasına izin verin.
Visual Studio'yu indirin
Copilot ücretsiz deneme sürümünü başlatın
Visual Studio'da GitHub Copilot hakkında daha fazla bilgi edinin →

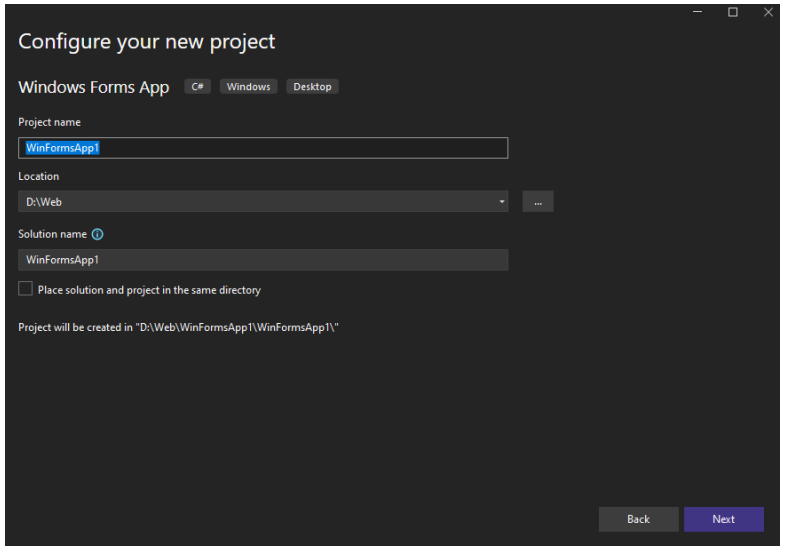

Visual Studio Installer
Başlamadan önce, kurulumunuzu yapılandırabilmeniz için bazı ayarlamalar yapmamız gerekiyor.
Gizlilik hakkında daha fazla bilgi edinmek için bkz. Microsoft Gizlilik Bildirimi.
Devam ettiğinizde Microsoft Yazılımı Lisans Koşullarını kabul etmiş olursunuz.
Devam

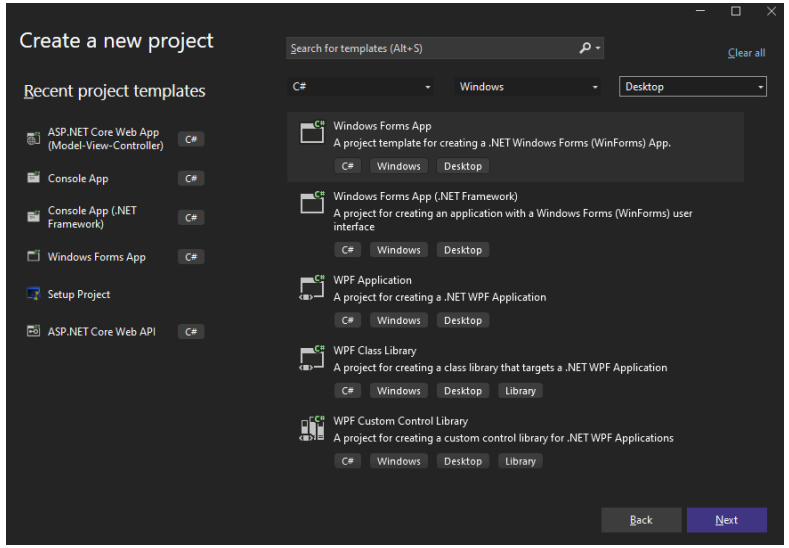

Değiştiriliyor — Visual Studio Community 2022 — 17.8.5
İş yükleri
Bağımsız bileşenler
Dil paketleri
Yükleme konumları
Web ve Bulut (4)
ASP.NET ve web geliştirme
Azure geliştirme
Python geliştirme
Node.js geliştirme
Masaüstü ve Mobil (5)
.NET Multi-platform App UI geliştirmesi
.NET masaüstü geliştirme
C++ ile masaüstü geliştirme
Evrensel Windows Platformu geliştirme
Yükleme ayrıntıları
Visual Studio temel düzenleyicisi
ASP.NET ve web geliştirme
.NET masaüstü geliştirme
Unity ile oyun geliştirme
Bağımsız bileşenler
Uzantılar
Microsoft Visual Studio Installer Projects 2022
ML.NET Model Builder 2022
Konum
C:\Program Files\Microsoft Visual Studio\2022\Community
Destek dışı bileşenleri kaldır
Devam ederek, seçtiğiniz Visual Studio sürümünün lisansını kabul ediyorsunuz. Visual Studio ile başka yazılımlar indirme olanağı da sunuyoruz. Bu yazılımlar, Üçüncü Taraf Bildirimi'nde veya ürünle birlikte sağlanan lisansta belirtildiği gibi, ayrıca lisanslanır. Devam ederek bu lisansları da kabul etmiş olursunuz.
Gereken toplam alan 0 B
İndirilirken yükle
Kapat

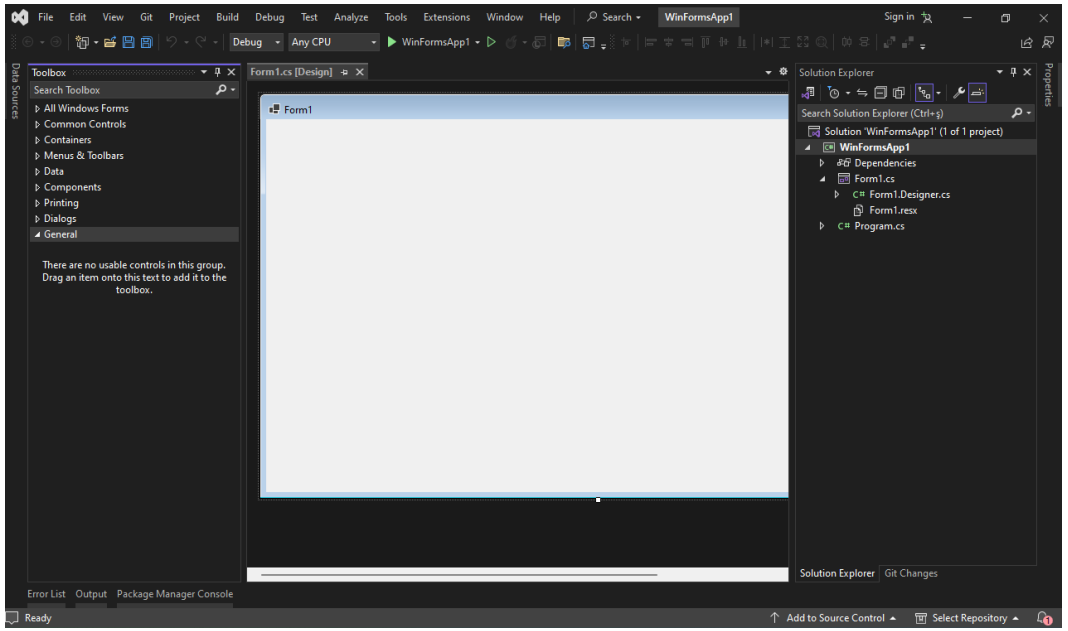
Proje Oluşturma

- Visual Studio’yu başlatın.
- Sol Menüde bulunan «Create a new project» seçeneğini seçin.
- Açılan pencerede proje tipi olarak «Windows Forms App» seçeneğine tıklayın.
- Daha kolay filtrelemek için yukarıdaki seçeneklerden C#, Windows ve Desktop seçeneklerini seçebilirsiniz.
- [Next] Butonu ile devam edin. Projenin adını ve konumunu yazın.


Visual Studio 2022
Open recent
Search recent (Alt+S)
Today
This week
This month
Older
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →


Configure your new project
Windows Forms App
C#
Windows
Desktop
Project name
WinFormsApp1
Location
D:\Web
Solution name
WinFormsApp1
Place solution and project in the same directory
Project will be created in "D:\Web\WinFormsApp1\WinFormsApp1\"
Back
Next


Create a new project
Recent project templates
ASP.NET Core Web App (Model-View-Controller)
Console App
Console App (.NET Framework)
Windows Forms App
Setup Project
ASP.NET Core Web API
Search for templates (Alt+S)
Clear all
C#
Windows
Desktop
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
WPF Application
A project for creating a .NET WPF Application
WPF Class Library
A project for creating a class library that targets a .NET WPF Application
WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
Back
Next

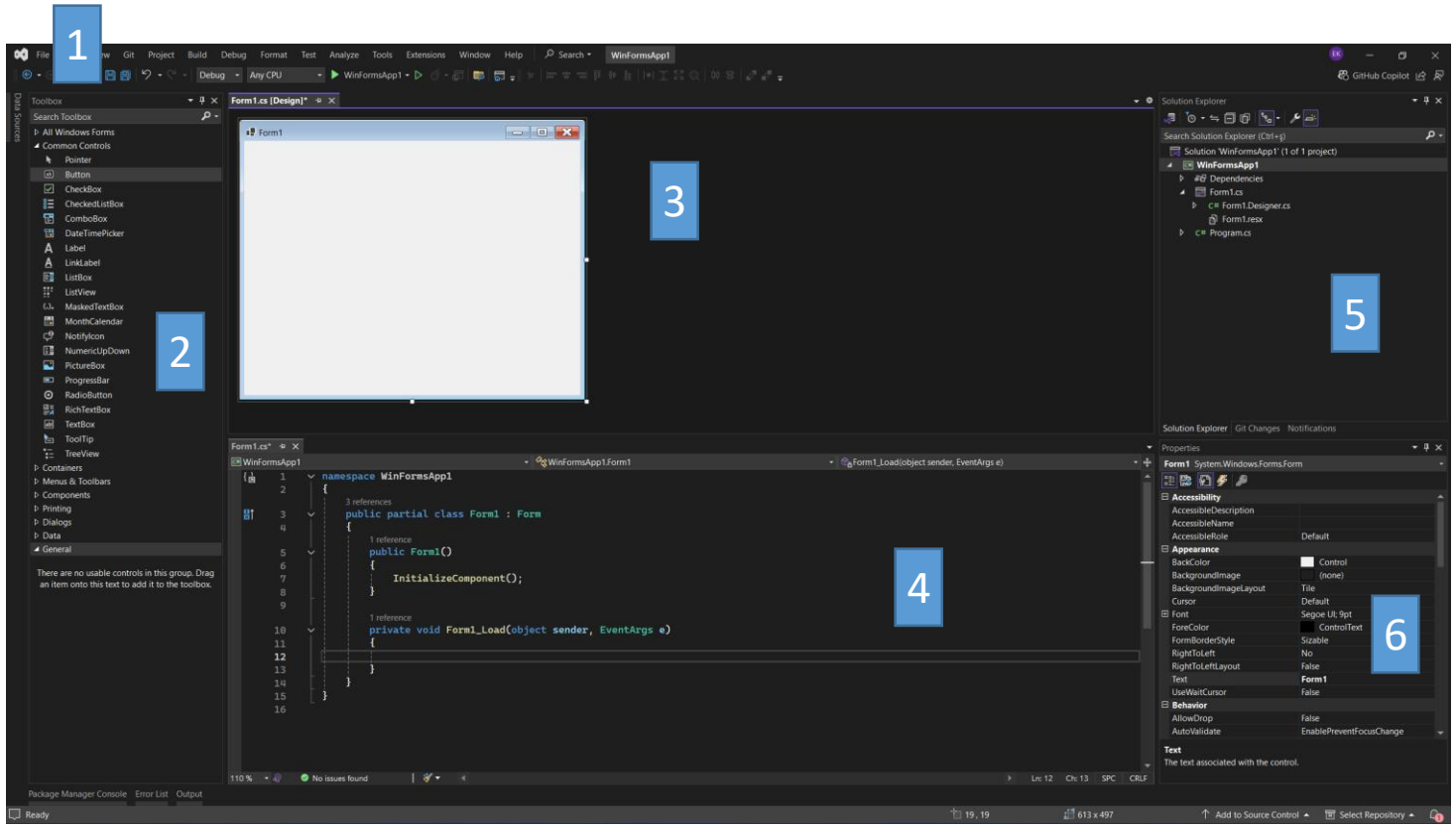

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
WinFormsApp1
Debug
Any CPU
Toolbox
Search Toolbox
All Windows Forms
Common Controls
Containers
Menus & Toolbars
Data
Components
Printing
Dialogs
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Form1.cs [Design]
Form1
Solution Explorer
Solution 'WinFormsApp1' (1 of 1 project)
WinFormsApp1
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Error List
Output
Package Manager Console
Ready
Add to Source Control
Select Repository

Visual Studio Ortamı

C# ile Görsel Programlama

Genel Görünüm

1. Araç Çubukları
2. Toolbox
3. Tasarım Penceresi
4. Kod Yazım Penceresi
5. Solution Explorer
6. Properties Window

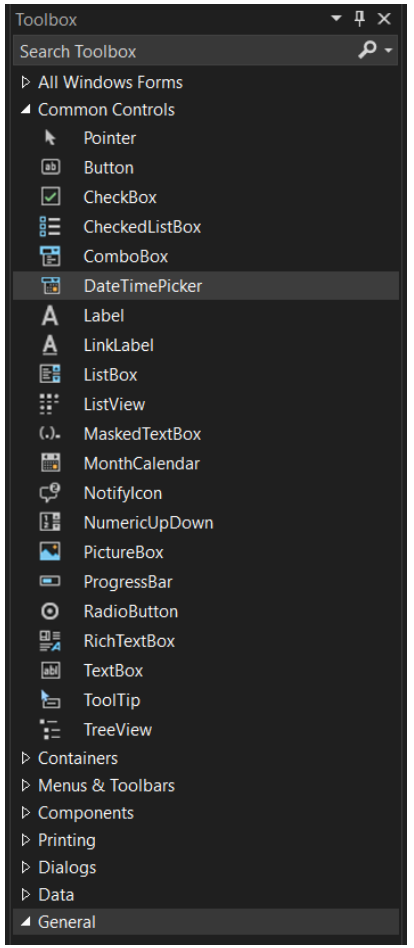


1. Araç Çubukları

- Visual Studio kullanırken ve projeyi geliştirirken ihtiyacımız olan kısayolları içeren yapılardır.
- Başlangıç olarak aşağıdaki butolarla birlikte açılır.
  - Aç, Kapa, Kaydet gibi dosya işlemleri,
  - Projeyi Çalıştır, Hata Ayıkla gibi proje işlemleri,
  - Hizalama, Yerleşim gibi Form tasarım ayarları
- Sağ tuş ile tıklayarak, açılan menüden başka özellik ve araçlar da eklenebilir.

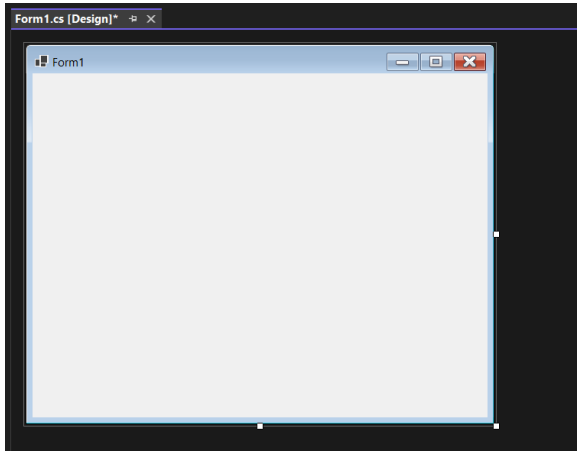
2. Toolbox (Alet Çantası)

- Görsel bileşenlerin bulunduğu penceredir.
- Windows Forms projesinde buton, metin kutusu, label gibi bileşenleri buradan form üzerine sürükleyip bırakabilirsiniz.
- Kapalıysa açmak için: Menüden View > Toolbox yolunu izleyebilirsiniz veya Ctrl + Alt + X kısayolunu kullanabilirsiniz.
- Form tasarımına uygun bileşenleri seçip sürükleyerek hızlıca kullanıcı arayüzü oluşturabilirsiniz.



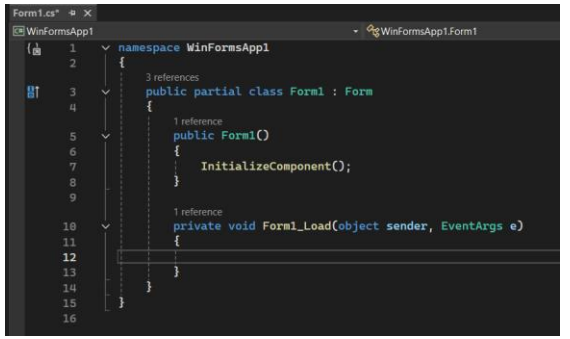
3. Tasarım Penceresi

- Açık olan formun görsel tasarımını yaptığımız kısımdır.
- Solution Explorer da bir forma çift tıkladığımızda bu pencere açılır.
- Fare ile kontrolleri yerleştirip, sürükle-bırak ile form tasarımını kolaylıkla yapabiliriz.



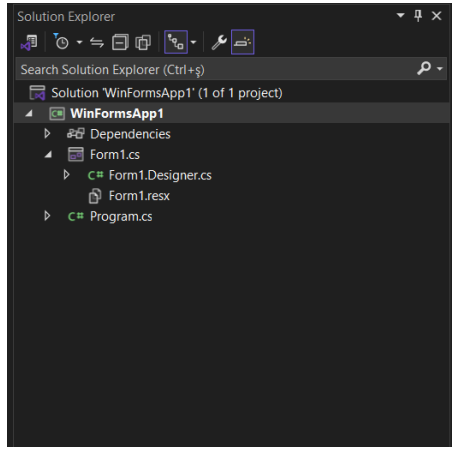
4. Kod Yazım Penceresi

- C# kodlarını yazdığınız ve düzenlediğiniz alandır.
- Formun üzerinde sağ tuşla tıklayarak «View Code» seçeneği ile açılır.
- Visual Studio, kod yazarken çeşitli özelliklerle size yardımcı olur:
  - Kod yazarken otomatik tamamlama özelliği (intellisense)
  - Kodun daha okunabilir olması için renkli bir yapı sunar.
  - Kodda bir hata varsa ilgili satırın altında kırmızı bir çizgiyle gösterilir.



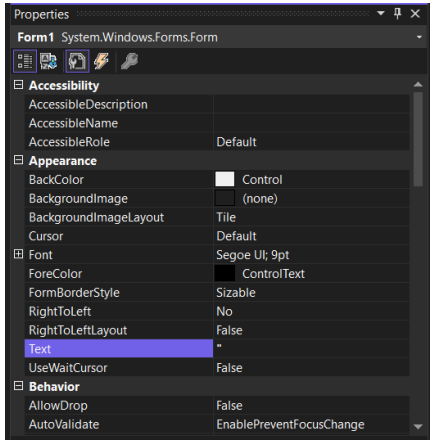
5. Solution Explorer

- Projede yer alan tüm dosya ve klasörleri gösteren penceredir.
- Kapalıysa açmak için: Menüden View > Solution Explorer yolunu izleyebilirsiniz veya Ctrl + Alt + L kısayolunu kullanabilirsiniz.
- Sağ tıklayarak dosya ve klasör ekleme, yeniden adlandırma, kaldırma işlemleri yapabilirsiniz.



6. Properties Window (Özellikler Penceresi)

- Seçili bir nesne veya dosyanın özelliklerini görüntüleyip düzenleyebileceğiniz penceredir.
  - Örneğin, bir buton seçtiğinizde Text özelliğini değiştirerek üzerinde yazan metni değiştirebilirsiniz.
- Menüden View > Properties Window yolunu izleyebilirsiniz veya F4 kısayolunu kullanabilirsiniz.



En Çok Kullanılan Bileleşenler

C# ile Görsel Programlama

1. Form

- Form, bir Windows Forms uygulamasında kullanıcı arayüzünü (UI) oluşturan temel bileşendir.
- Kullanıcının etkileşimde bulunabileceği düğmeler, metin kutuları, listeler ve diğer bileşenler form üzerine eklenir.
- Bir pencere veya kapsayıcı olarak düşünebilirsiniz.
- C# ile Windows Forms uygulamaları geliştirirken, her uygulama en az bir form içerir.

Form Nesnesinin Temel Özellikleri

- **Text:** Formun başlık çubuğunda görünen metindir.
- **Name:** Formun kod tarafında kullanılacak adıdır.
  - Varsayılan olarak Form1, Form2 gibi isimler atanır. Ancak daha anlamlı bir isim kullanmak iyi bir pratiktir
- **Size:** Formun genişlik ve yükseklik değerleridir.
  - Örneğin, Size özelliğine (800, 600) girerseniz, form 800 piksel genişlikte ve 600 piksel yükseklikte görüntülenir.
- **BackColor:** Formun arka plan rengini ayarlamak için kullanılır.
  - Örneğin, BackColor özelliğini LightBlue olarak seçerseniz, formun arka plan rengi açık mavi olur.
- **StartPosition:** Formun ekranda ilk kez nerede görüneceğini belirler.
  - CenterScreen: Form ekranın tam ortasında açılır.
  - Manual: Form, manuel olarak belirlenen koordinatlarda açılır.
- **FormBorderStyle:** Formun kenarlık stilini ayarlar.
  - None: Form kenarlığı olmaz (tam ekran oyun gibi).
  - FixedSingle: Sabit bir kenarlık. Kullanıcı formun boyutunu değiştiremez.
  - Sizable: Kullanıcı formun boyutunu değiştirebilir.

Form Nesnesinin Temel Olayları

- **Load:** Form açıldığında tetiklenir.

```
private void MainForm_Load(object sender, EventArgs e)
{
    MessageBox.Show("Form açıldı!");
}
```

- **Closing:** Form kapanırken tetiklenir.

```
private void MainForm_Closing(object sender, FormClosingEventArgs e)
{
    if (MessageBox.Show("Formu kapatmak istediğinize emin misiniz?", "Onay",
    {
        e.Cancel = true; // Formun kapanmasını iptal eder
    }
}
```

2. Label Nesnesi

- Label, kullanıcı arayüzünde (UI) statik metinleri veya bilgiler göstermek için kullanılan bir kontrol nesnesidir.
- Amacı:
  - Kullanıcıya bilgi vermek (örneğin: bir giriş alanının ne işe yaradığını açıklamak).
  - Dinamik olarak güncellenen bilgiler göstermek (örneğin: “Hoş Geldiniz, [Kullanıcı Adı]”).
- Label, sadece metin göstermek içindir. Kullanıcı tarafından tıklanamaz veya düzenlenemez. Ancak, arka planda kod ile dinamik olarak içeriği değiştirilebilir.

Label Nesnesinin Temel Özellikleri

- **Text:** Label üzerinde görünen metni belirler.
  - Örneğin, Text özelliğine Adınızı Giriniz: yazarsanız, label üzerinde bu ifade görüntülenir.
- **Name:** Label'ın kod tarafında kullanılacak adıdır.
  - Varsayılan olarak label1, label2 gibi isimler atanır. Anlamlı bir isim kullanmak iyi bir pratiktir (örneğin, lblWelcomeMessage).
- **Font:** Label'ın metin yazı tipi, boyutu ve stili (bold, italik vb.) bu özellik ile ayarlanabilir.
- **ForeColor:** Metnin rengini belirler.
- **BackColor:** Label'ın arka plan rengini ayarlar.
- **AutoSize:** Label'ın boyutunun içindeki metne göre otomatik olarak ayarlanıp ayarlanmayacağını belirler.
  - True (Varsayılan): Metin uzunluğuna göre boyut otomatik değişir.
  - False: Label'ı manuel olarak yeniden boyutlandırabilirsiniz.
- **TextAlign:** Label üzerindeki metnin hizalamasını ayarlar.
  - Örneğin: Sol, sağ, ortalı.

3. Textbox (Metin Kutusu)

- TextBox, bir Windows Forms uygulamasında kullanıcıdan metin veya veri girişini almak için kullanılan bir kontrol nesnesidir.
- Kullanıcı adları, şifreler, adresler veya diğer metinsel verilerin girilmesini sağlar.
- Kullanıcıdan bilgi toplamak veya dinamik bir şekilde metin göstermek için kullanılır.

Textbox Nesnesinin Temel Özellikleri

- **Text:** TextBox'a girilen veya kodla belirlenen metni temsil eder.
  - Örneğin, bir TextBox'a girilen değeri almak için textBox1.Text kullanılır.
  - Kodla TextBox'a metin eklemek için: textBox1.Text = "Merhaba Dünya!";
- **Name:** TextBox'ın kod tarafında kullanılacak adıdır.
  - Varsayılan olarak textBox1, textBox2 gibi isimler atanır. Ancak daha anlamlı bir isim vermek iyi bir pratiktir (örneğin, txtUserName, txtPassword).
- **MaxLength:** TextBox'a girilebilecek maksimum karakter sayısını sınırlar.
  - Örneğin, MaxLength = 10 olarak ayarlanırsa, kullanıcı en fazla 10 karakter girebilir.
- **Multiline:** TextBox'ı tek satır mı yoksa birden fazla satır içerecek şekilde mi kullanacağınızı belirler.
  - False (Varsayılan): Tek satır, True: Birden fazla satır.
- **PasswordChar:** TextBox'ın şifre girişleri gibi gizli veri girişi için kullanılmasını sağlar. Girilen karakterler gizlenir.
  - Örneğin, PasswordChar = '*' olarak ayarlanırsa, kullanıcı yazarken metin yerine yıldız (*) görünür.
- **ReadOnly:** TextBox'ın sadece okuma modunda olmasını sağlar.
  - True: Kullanıcı metni değiştiremez, sadece görebilir, False (Varsayılan): Kullanıcı metni düzenleyebilir.
- **ScrollBars:** Eğer Multiline özelliği açıksa, kaydırma çubuklarını eklemek için kullanılır.
  - None (Varsayılan): Kaydırma çubuğu yoktur.
  - Horizontal: Yatay kaydırma çubuğu ekler.
  - Vertical: Dikey kaydırma çubuğu ekler.
  - Both: Hem yatay hem dikey kaydırma çubuğu ekler.

Textbox Nesnesinin Temel Olayları

- **TextChanged:** Kullanıcı TextBox'a her yeni bir karakter eklediğinde veya sildiğinde tetiklenir.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    lblCharacterCount.Text = "Karakter sayısı: " + textBox1.Text.Length;
}
```

- **KeyPress:** Kullanıcı bir tuşa bastığında tetiklenir. Girilen veriyi sınırlandırmak için kullanılabilir

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))
    {
        e.Handled = true; // Rakamlardan başka bir girişe izin verme
    }
}
```

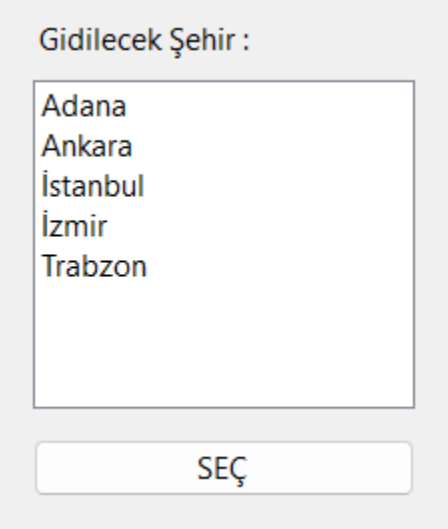
Örnek Kod (1)

- Kullanıcı adı ve şifre girişi yapılan bir program.
- Eğer kullanıcı adı veya şifre girilmemiş ise uyarı veriyor.
- Bilgiler girilmişse, girilen isimi kullanarak «hoş geldiniz» yazan bir pencere açıyor.

```
using System;
using System.Windows.Forms;

namespace TextBoxExample
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            string userName = txtUserName.Text;
            string password = txtPassword.Text;

            if (string.IsNullOrEmpty(userName) || string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Lütfen kullanıcı adı ve şifre giriniz!");
            }
            else
            {
                MessageBox.Show("Hoş geldiniz, " + userName + "!");
            }
        }
    }
}
```

4. Listbox (Liste Kutusu)

- ListBox, kullanıcıya birden fazla öğeden oluşan bir liste sunan ve bu listeden bir veya birden fazla öğe seçmesine olanak tanıyan bir Windows Forms kontrolüdür.
- Amaç:
  - Kullanıcıdan sınırlı bir seçenek arasından seçim yapmasını istemek.
  - Dinamik olarak bir liste oluşturmak ve kullanıcıya bu liste üzerinde işlem yaptırmak.



Listbox Nesnesinin Temel Özellikleri

- **Items:** ListBox'ta görüntülenen öğeleri temsil eder.
  - listBox1.Items.Add("Elma"); // listeye elma ekler
  - listBox1.Items.Remove("Elma"); // listeden elma'yı çıkartır
  - listBox1.Items.Clear(); // tüm öğeleri temizler
- **Name:** ListBox'ın kod tarafında kullanılacak adıdır.
  - Varsayılan olarak listBox1, listBox2 gibi isimler atanır. Daha anlamlı bir isim vermek iyi bir pratiktir (örneğin, lstFruits).
- **SelectedItem:** Kullanıcının seçtiği öğeyi temsil eder.
  - Örneğin, bir ListBox'ta seçilen öğeyi almak için:
    - string secim = listBox1.SelectedItem.ToString();
    - MessageBox.Show("Seçilen: " + secim);
- **SelectedIndex:** Kullanıcının seçtiği öğenin dizinini (sırasını) temsil eder. (Dizilerde olduğu gibi, sıfırdan başlar.)
  - Seçili öğenin dizinini almak:
    - int index = listBox1.SelectedIndex;
    - MessageBox.Show("Seçilen öğenin sırası: " + index);
- **Sorted:** Liste öğelerinin otomatik olarak alfabetik sırada görüntülenmesini sağlar.
  - True: Öğeler otomatik olarak sıralanır.
  - False (Varsayılan): Öğeler eklendiği sırada kalır.

Listbox Nesnesinin Temel Olayları

- SelectedIndexChanged: Kullanıcı listeden bir eleman seçtiğinde tetiklenir.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (listBox1.SelectedItem != null)
    {
        lblSelectedItem.Text = "Seçilen: " + listBox1.SelectedItem.ToString();
    }
}
```

Örnek Kod (2)

- Meyve seçim uygulaması.
- Form yüklenirken liste kutusuna meyveler eklenir.
- Kullanıcı formdaki «Seç» butonuna bastığı zaman seçilen meyve ekrana gelir.
- Eğer bir meyve seçilmemiş ise program uyarı verir.

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
    }

    private void MainForm_Load(object sender, EventArgs e)
    {
        // ListBox'a öğeler ekle
        listBox1.Items.Add("Elma");
        listBox1.Items.Add("Armut");
        listBox1.Items.Add("Karpuz");
        listBox1.Items.Add("Muz");
    }

    private void btnShowSelection_Click(object sender, EventArgs e)
    {
        if (listBox1.SelectedItem != null)
        {
            MessageBox.Show("Seçtiğiniz meyve: " + listBox1.SelectedItem.ToString());
        }
        else
        {
            MessageBox.Show("Lütfen bir meyve seçiniz.");
        }
    }
}
```

5. Checkbox (Onay Kutusu)

- CheckBox, kullanıcıya bir veya daha fazla seçeneği işaretleme (onaylama) veya kaldırma imkanı sunan bir Windows Forms kontrolüdür.
- Kullanıcı bir CheckBox'ı seçtiğinde bir işaret (✓) görüntülenir, seçimini kaldırdığında işaret kaybolur.
- Genellikle seçeneklerin bağımsız olduğu durumlarda kullanılır (örneğin, tercih seçenekleri, özellik aktivasyonları).

Checkbox Nesnesinin Temel Özellikleri

- **Text:** CheckBox'ın yanında görüntülenen metni belirler.
  - Örneğin, Text özelliği "E-posta bildirimi al" şeklinde ayarlanabilir.csharpKodu kopyalacheckBox1.Text = "E-posta bildirimi al";
- **Checked:** CheckBox'ın seçili (işaretli) olup olmadığını kontrol eder veya ayarlar.
  - True: CheckBox işaretlidir.
  - False: CheckBox işaretli değildir.
  - if (checkBox1.Checked) { MessageBox.Show("CheckBox işaretli."); }
- **Name**: CheckBox'ın kod tarafında kullanılacak adıdır.
  - Varsayılan olarak checkBox1, checkBox2 gibi isimler atanır. Daha anlamlı bir isim vermek iyi bir pratiktir (örneğin, chkEmailNotification).
- **Enabled:** CheckBox'ın aktif olup olmadığını belirler.
  - True: Kullanıcı CheckBox'ı kullanabilir (Varsayılan).
  - False: CheckBox devre dışı bırakılır (gri renkte görünür).
- **Visible:** CheckBox'ın görünür olup olmadığını kontrol eder.
  - True: Görünür (Varsayılan).
  - False: Görünmez.

Checkbox Nesnesinin Temel Olayları

- **CheckedChanged**
- Kullanıcı CheckBox'ı işaretlediğinde veya işaretini kaldırdığında tetiklenir

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked)
    {
        lblMessage.Text = "CheckBox işaretli.";
    }
    else
    {
        lblMessage.Text = "CheckBox işaretli değil.";
    }
}
```

Örnek Kod (3)

- Bir kayıt formu örneği.
- Kullanıcı tüm bilgileri doldurduktan sonra kullanım koşullarını kabul etmek zorundadır.
- Eğer kabul kutucuğunu işaretlemezse program hata verir.

```
using System;
using System.Windows.Forms;

namespace CheckBoxRegistrationForm
{
    public partial class RegistrationForm : Form
    {
        public RegistrationForm()
        {
            InitializeComponent();
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            if (chkTerms.Checked)
            {
                MessageBox.Show("Kaydınız başarıyla tamamlandı!");
            }
            else
            {
                MessageBox.Show("Lütfen kullanım koşullarını kabul edin.");
            }
        }
    }
}
```