VISUAL C# İLE
GÖRSEL PROGRAMLAMA
DERS NOTLARI

Öğretim Görevlisi Erkan Kaynak

İçindekiler Tablosu

BİRİNCİ BÖLÜM: VISUAL C# İLE TEMEL İŞLEMLER

1. Visual Studio Programı

Visual Studio, C# ile Windows uygulamaları geliştirmek için kullanılan güçlü bir Entegre Geliştirme Ortamıdır (IDE).

Visual Studio, Microsoft'un resmi web sitesi üzerinden ücretsiz olarak indirilebilir. Öğrenciler için **Visual Studio Community** sürümü yeterlidir.

İndirme adresi:
https://visualstudio.microsoft.com

İndirme sırasında işletim sistemine uygun sürüm seçilmeli ve kurulum aşamasında **.NET Desktop Development** iş yükü mutlaka işaretlenmelidir.

Visual Studio'nun sunduğu temel özellikler şunlardır:

- Kod yazma ve derleme
- Hata ayıklama (Debug)
- Görsel arayüz tasarımı (Form Designer)

Arayüzde sık kullanılan bölümler:

- **Solution Explorer:** Projedeki dosyaları ve sınıfları gösterir
- **Properties Window:** Seçilen nesnenin özelliklerini düzenlemek için kullanılır (F4)
- **ToolBox:** Sürükle-bırak ile kullanılan kontrolleri içerir

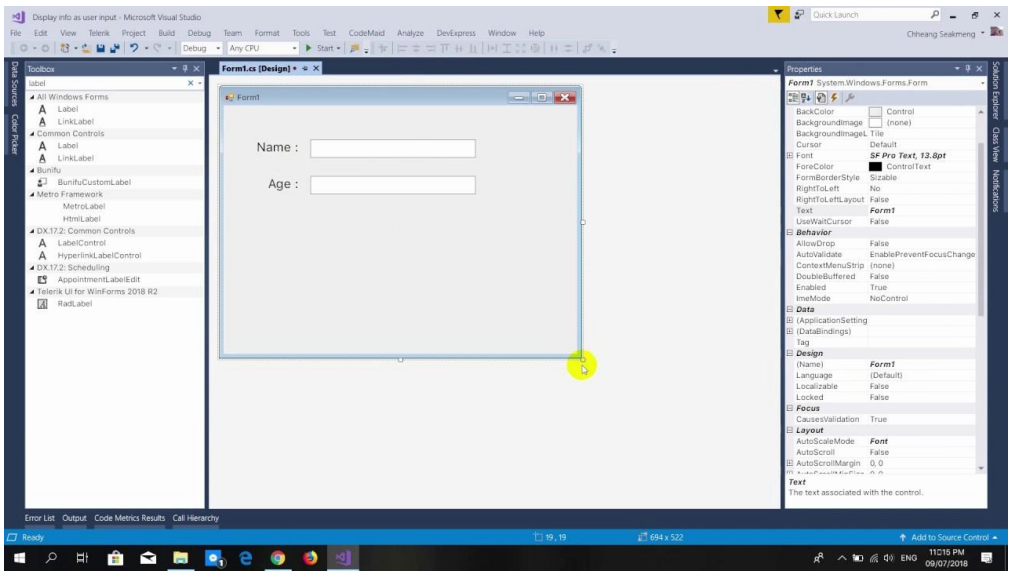
---

2. Görsel Programlama ile İlgili Temel Bilgiler

Görsel programlama, kullanıcı arayüzünün sürükle-bırak yöntemiyle oluşturulduğu ve olay (event) tabanlı çalışılan programlama yaklaşımıdır.

Temel kavramlar:

- **Property (Özellik):** Nesnenin görünümünü veya davranışını belirler
- **Event (Olay):** Kullanıcının yaptığı işlemler sonucunda tetiklenir
- **Method (Metot):** Belirli bir işi yapan kod bloklarıdır



3. Form ve Özellikleri

Form, Windows Forms uygulamasının ana penceresidir.

Sık kullanılan form özellikleri:

- Text
- BackColor
- Size
- StartPosition
- FormBorderStyle
- MaximizeBox

Sık kullanılan olaylar:

- Load
- Click
- FormClosing

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Öğrenci Kayıt Uygulaması";
    this.StartPosition = FormStartPosition.CenterScreen;
}
```

Yeni Bir Form Açmak

Bir Windows Forms uygulamasında birden fazla form kullanılabilir. Yeni bir form açmak için önce projeye yeni bir Form eklenir (Add → Windows Form). Daha sonra bir form üzerinden diğer form nesnesi oluşturularak gösterilir.

Yeni formu açmak için en sık kullanılan yöntemler şunlardır:

- **Show():** Formu modeless (arka plandaki form açık kalır) olarak açar.
- **ShowDialog():** Formu modal (diğer formlar kilitlenir) olarak açar.

```
private void button1_Click(object sender, EventArgs e)
{
    Form2 frm = new Form2();
    frm.Show();
}
```

Modal Form Açma:

```
private void button2_Click(object sender, EventArgs e)
{
    Form2 frm = new Form2();
    frm.ShowDialog();
}
```

Formun başlangıç pozisyonunu değiştirme:

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Öğrenci Kayıt Uygulaması";
    this.StartPosition = FormStartPosition.CenterScreen;
}
```

4. En Çok Kullanılan Bileşenler

4.1 Label

Kullanıcıya bilgi göstermek için kullanılır.

Sık kullanılan özellikler:

Text: Label üzerinde gösterilecek yazıyı belirtir.
ForeColor: Yazının rengini belirler.
BackColor: Label'ın arka plan rengini ayarlar.
Font: Yazı tipi, boyutu ve stilini belirler.
AutoSize: Yazının uzunluğuna göre Label'ın otomatik boyutlanmasını sağlar.
Visible: Label'ın görünür veya gizli olmasını belirler.
TextAlign: Yazının Label içindeki hizalamasını ayarlar.

```
label1.Text = "Ad Soyad:";
label1.ForeColor = Color.Blue;
label1.BackColor = Color.LightYellow;
label1.Font = new Font("Arial", 12, FontStyle.Bold);
label1.TextAlign = ContentAlignment.MiddleLeft;
```

4.2 TextBox

Kullanıcıdan veri almak için kullanılır.

Sık kullanılan özellikler:

Text: TextBox içinde yer alan metni tutar. Kullanıcının girdiği veya program tarafından atanan değerdir.

MaxLength: Kullanıcının girebileceği maksimum karakter sayısını belirler.

ReadOnly: TextBox'ın sadece okunabilir olmasını sağlar. Kullanıcı metni değiştiremez ancak seçip kopyalayabilir.

PasswordChar: Girilen karakterlerin yerine gösterilecek sembolü belirler. Şifre girişleri için kullanılır.

Multiline: TextBox'ın birden fazla satırdan oluşmasını sağlar. Açıklama veya uzun metinler için kullanılır.

Enabled: TextBox'ın aktif veya pasif olmasını belirler. Pasif olduğunda kullanıcı etkileşimi olmaz.

TextAlign: Metnin TextBox içindeki hizalamasını ayarlar (Left, Center, Right).

```
textBox1.Text = "Merhaba Dünya!";
textBox1.MaxLength = 20;
textBox1.ReadOnly = false;
textBox1.PasswordChar = '*';
```

Metin kutusuna girilen değeri kontrol etme:

```
if (string.IsNullOrWhiteSpace(textBox1.Text))
{
    MessageBox.Show("Bu alan boş bırakılamaz");
    textBox1.Focus();
    return;
}
```

4.3 Button

Kullanıcının bir işlemi başlatmasını sağlar.

Sık kullanılan özellikler:

Text: Buton üzerinde görünen yazıyı belirler.

Enabled: Butonun aktif veya pasif olmasını sağlar.

BackColor: Butonun arka plan rengini belirler.

ForeColor: Buton üzerindeki yazının rengini belirler.

FlatStyle: Butonun düz veya klasik görünümde olmasını sağlar.

Width / Height: Butonun genişlik ve yükseklik değerlerini belirler.

Visible: Butonun görünür veya gizli olmasını sağlar.

```
button1.Text = "Kaydet";
button1.BackColor = Color.LightGreen;
button1.FlatStyle = FlatStyle.Flat;
```

Butona basılması:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Butona tıklandı");
}
```

4.4 CheckBox

Birden fazla seçeneğin seçilmesine olanak tanır.

Sık kullanılan özellikler:

Text: CheckBox yanında gösterilecek açıklama yazısını belirler.

Checked: CheckBox'ın seçili olup olmadığını belirtir.

Enabled: CheckBox'ın aktif veya pasif olmasını belirler.

Visible: CheckBox'ın görünür olup olmadığını belirler.

CheckAlign: Onay kutusunun metne göre hizalanmasını ayarlar.

Olaylar:

- CheckedChanged

```
checkBox1.Text = "Bültene abone ol";
checkBox1.Checked = true;
```

4.5 ListBox

Listeleme ve seçim işlemleri için kullanılır.

Sık kullanılan özellikler:

Items: ListBox içerisinde yer alan öğeleri tutar.

SelectedItem: Kullanıcı tarafından seçilen öğeyi verir.

SelectedIndex: Seçili öğenin listedeki sıra numarasını belirtir.

DataSource: ListBox'a bir veri kaynağı bağlamak için kullanılır.

DisplayMember: Nesne listelerinde ekranda gösterilecek alanı belirtir.

ValueMember: Seçilen öğeye ait arka planda kullanılacak değeri belirler.

SelectionMode: Tekli veya çoklu seçim yapılmasını sağlar.

Olaylar:

- SelectedIndexChanged

Eleman Ekleme

```
listBox1.Items.Add("Ankara");
listBox1.Items.Add("İstanbul");
```

Elemanları Silme

```
ListBox1.Items.Clear();
```

Nesne ile ListBox Kullanımı

```
class Ogrenci
{
    public string AdiSoyadi { get; set; }
    public string Telefon { get; set; }
}

List<Ogrenci> ogrenciler = new List<Ogrenci>();
ogrenciler.Add(new Ogrenci { AdiSoyadi = "Ali Veli", Telefon = "555" });

listBox1.DataSource = ogrenciler;
listBox1.DisplayMember = "AdiSoyadi";
```

4.6 PictureBox (Image Kontrolü)

PictureBox, forma resim eklemek ve göstermek için kullanılır.

Sık kullanılan özellikler:

Image: Gösterilecek resim dosyasını belirler.

SizeMode: PictureBox kontrolünde resmin kontrol alanı içinde nasıl görüntüleneceğini belirler. Resmin boyutu ile PictureBox boyutu arasındaki ilişkiyi düzenler.

Normal: Resim orijinal boyutunda gösterilir, PictureBox küçükse resmin bir kısmı görünür.

StretchImage: Resim PictureBox boyutlarına göre enine ve boyuna esnetilir, oran bozulabilir.

AutoSize: PictureBox, resmin boyutuna göre kendini otomatik olarak yeniden boyutlandırır.

CenterImage: Resim, PictureBox içinde ortalanarak gösterilir, boyutu değiştirilmez.

Zoom: Resim, PictureBox alanına oranı korunarak sığdırılır, kenarlarda boşluk kalabilir.

BackColor: PictureBox'ın arka plan rengini belirler.

BorderStyle: PictureBox etrafındaki kenarlık stilini belirler.

Visible: PictureBox'ın görünür veya gizli olmasını sağlar.

```
pictureBox1.Image = Image.FromFile("logo.png");
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
```

4.7 MessageBox

MessageBox, kullanıcıya bilgi vermek, uyarı göstermek veya bir işlem için onay almak amacıyla kullanılan hazır bir mesaj penceresidir.

En basit kullanım şekliyle sadece bir metin gösterilir. Bu kullanımda başlık yoktur ve varsayılan olarak yalnızca OK butonu bulunur.

```
MessageBox.Show("İşlem başarılı");
```

Daha gelişmiş kullanımda MessageBox'a başlık, buton türü ve ikon eklenebilir. Bu sayede kullanıcıya verilen mesaj daha anlaşılır hale gelir.

```
MessageBox.Show(
    "Kayıt başarıyla eklendi",
    "Bilgi",
    MessageBoxButtons.OK,
    MessageBoxIcon.Information
);
```

Bu kullanımda mesaj metni, pencere başlığı, gösterilecek butonlar ve ikon türü belirlenmiş olur.

MessageBox, kullanıcıdan cevap almak için de kullanılabilir. Bu durumda MessageBox'tan dönen değer bir DialogResult değişkenine atanır.

```
DialogResult sonuc = MessageBox.Show(
    "Kaydı silmek istiyor musunuz?",
    "Onay",
    MessageBoxButtons.YesNo,
    MessageBoxIcon.Question
);
```

Kullanıcının verdiği cevap bu değişken üzerinden kontrol edilir. Eğer kullanıcı Yes butonuna basarsa işlem yapılır, aksi durumda işlem iptal edilir.

```
if (sonuc == DialogResult.Yes)
{
    MessageBox.Show("Kayıt silindi");
}
else
{
    MessageBox.Show("Silme işlemi iptal edildi");
}
```

MessageBox, özellikle silme işlemleri, önemli güncellemeler ve programdan çıkış gibi kritik durumlarda kullanıcıdan onay almak için sıkça kullanılır.

Örneğin program kapatılırken kullanıcıya çıkış onayı sorulabilir:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult sonuc = MessageBox.Show(
        "Programdan çıkmak istiyor musunuz?",
        "Çıkış",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Warning
    );

    if (sonuc == DialogResult.No)
    {
        e.Cancel = true;
    }
}
```

Bu örnekte kullanıcı Hayır cevabını verirse program kapanmaz, Evet cevabı verirse program kapanır.

İKİNCİ BÖLÜM: VERİTABANI İŞLEMLERİ

1. Veritabanı Nedir?

Veritabanı, bilgilerin düzenli, güvenli ve kalıcı olarak saklandığı dijital yapılardır. Uygulamalarda kullanıcı, öğrenci, ürün gibi bilgilerin saklanmasını sağlar.

Veritabanı kullanmanın avantajları:

- Veri bütünlüğü
- Güvenli erişim
- Hızlı sorgulama
- Yedekleme imkanı

2. Veritabanı Türleri

- İlişkisel Veritabanları
- NoSQL Veritabanları

3. İlişkisel Veritabanı Kavramları

İlişkisel veritabanı, verilerin tablolar (tables) halinde saklandığı ve bu tabloların birbirleriyle ilişkiler (relations) kurarak çalıştığı veritabanı modelidir. Her tablo belirli bir varlığı temsil eder (örneğin Öğrenciler, Dersler, Siparişler) ve tablolar arasındaki ilişkiler anahtar alanlar aracılığıyla kurulur.

İlişkisel veritabanlarında her tabloda bir Primary Key (Birincil Anahtar) bulunur. Bu anahtar, her kaydı benzersiz şekilde tanımlar. Tablolar arasında bağlantı kurmak için ise Foreign Key (Yabancı Anahtar) kullanılır. Bu yapı sayesinde veriler tutarlı, düzenli ve tekrar etmeyecek şekilde saklanır.

İlişkisel veritabanları; öğrenci bilgi sistemleri, banka uygulamaları, muhasebe ve finans yazılımları, e-ticaret siteleri, hastane otomasyonları, personel ve stok takip sistemleri gibi verinin düzenli ve güvenli tutulmasının önemli olduğu birçok alanda kullanılır. Özellikle birbiriyle ilişkili çok sayıda verinin bulunduğu kurumsal uygulamalarda tercih edilir.

İlişkisel veritabanlarının en önemli avantajlarından biri veri bütünlüğü sağlamasıdır. Anahtar kısıtlamaları ve ilişkiler sayesinde hatalı veya tutarsız veri girişi büyük ölçüde engellenir. Ayrıca SQL dili sayesinde veriler kolayca sorgulanabilir, filtrelenebilir, sıralanabilir ve birleştirilebilir.

Bunun yanında ilişkisel veritabanları güvenlik açısından güçlüdür; kullanıcı yetkilendirme ve erişim kontrolü yapılabilir. Yedekleme ve geri yükleme işlemleri kolaydır. Aynı anda birden fazla kullanıcının veriye erişmesine olanak tanır ve bu erişim sırasında veri tutarlılığı korunur.

Özetle, ilişkisel veritabanları düzenli yapı, güçlü sorgulama yetenekleri, veri güvenliği ve bütünlüğü sağladığı için günümüzde en yaygın kullanılan veritabanı modellerinden biridir.

- Tablo
- Alan (Field)
- Kayıt (Record)
- Primary Key
- Foreign Key

4. SQL

SQL (Structured Query Language), **ilişkisel veritabanları** üzerinde işlem yapmak için kullanılan standart bir sorgulama dilidir. SQL sayesinde veritabanında bulunan veriler eklenebilir, silinebilir, güncellenebilir ve listelenebilir. Aynı zamanda tablolar oluşturmak, tablolar arasındaki ilişkileri yönetmek ve veri yapısını kontrol etmek için de kullanılır.

SQL; Microsoft SQL Server, MySQL, PostgreSQL, Oracle, SQLite ve MS Access gibi birçok ilişkisel veritabanı yönetim sistemi tarafından desteklenir. Bu sayede öğrenilen SQL bilgisi farklı sistemlerde de büyük ölçüde aynıdır.

SQL'in temel kullanım amaçları şunlardır:

veri listeleme (SELECT), veri ekleme (INSERT), veri güncelleme (UPDATE), veri silme (DELETE), tablo ve veritabanı oluşturma (CREATE),veri yapısını değiştirme (ALTER).

SQL, özellikle veriye hızlı ve doğru şekilde erişmenin önemli olduğu öğrenci otomasyonları, e-ticaret sistemleri, banka uygulamaları, stok ve personel takip programları gibi birçok yazılımda temel bir rol oynar.

Kısaca SQL, ilişkisel veritabanlarındaki verilerle etkili, güvenli ve düzenli bir şekilde çalışmayı sağlayan vazgeçilmez bir sorgulama dilidir.

Veri Listeleme:

```
SELECT * FROM Ogrenciler
```

Filtreleme:

```
SELECT * FROM Ogrenciler WHERE AdiSoyadi='Ali Veli'
```

Sıralama:

```
SELECT * FROM Ogrenciler ORDER BY AdiSoyadi ASC
```

Tablo Birleştirme:

```
SELECT * FROM Ogrenciler o
INNER JOIN Bolumler b ON o.BolumID = b.ID
```

Kayıt Ekleme:

```
INSERT INTO Ogrenciler (AdiSoyadi, Telefon) VALUES ('Ali','555')
```

Güncelleme:

```
UPDATE Ogrenciler SET Telefon='444' WHERE ID=1
```

Silme:

```
DELETE FROM Ogrenciler WHERE ID=1
```

ÜÇÜNCÜ BÖLÜM: VISUAL C# İLE VERİTABANI UYGULAMALARI

1. Bağlantının Oluşturulması

```
OleDbConnection baglanti = new OleDbConnection(
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=okul.accdb");
```

2. Veri Çekip Listeleme

```
baglanti.Open();
OleDbCommand cmd = new OleDbCommand("SELECT * FROM Ogrenciler", baglanti);
OleDbDataReader dr = cmd.ExecuteReader();

listBox1.Items.Clear();
while (dr.Read())
{
    listBox1.Items.Add(dr["AdiSoyadi"].ToString());
}
baglanti.Close();
```

3. Veri Ekleme

```
baglanti.Open();
OleDbCommand cmd = new OleDbCommand(
"INSERT INTO Ogrenciler (AdiSoyadi, Telefon) VALUES (@ad,@tel)", baglanti);
cmd.Parameters.AddWithValue("@ad", textBox1.Text);
cmd.Parameters.AddWithValue("@tel", textBox2.Text);
cmd.ExecuteNonQuery();
baglanti.Close();
```

4. Veri Güncelleme

```
baglanti.Open();
OleDbCommand cmd = new OleDbCommand(
"UPDATE Ogrenciler SET AdiSoyadi=@ad, Telefon=@tel WHERE ID=@id", baglanti);
cmd.Parameters.AddWithValue("@ad", textBox1.Text);
cmd.Parameters.AddWithValue("@tel", textBox2.Text);
cmd.Parameters.AddWithValue("@id", textBox3.Text);
cmd.ExecuteNonQuery();
baglanti.Close();
```

5. Veri Silme

```
DialogResult sonuc = MessageBox.Show("Silmek istiyor musunuz?","Onay",
MessageBoxButtons.YesNo,MessageBoxIcon.Warning);

if (sonuc == DialogResult.Yes)
{
    baglanti.Open();
    OleDbCommand cmd = new OleDbCommand(
    "DELETE FROM Ogrenciler WHERE ID=@id", baglanti);
    cmd.Parameters.AddWithValue("@id", textBox3.Text);
    cmd.ExecuteNonQuery();
    baglanti.Close();
}
```