# DOSYA VE KLASÖR İŞLEMLERİ

İÇİNDEKİLER

- Dosya ve Klasör İşlemleri
  - Klasör İşlemleri
  - Dosya İşlemleri
- Dosya Okuma ve Yazma İşlemleri

HEDEFLER

- Bu üniteyi çalıştıktan sonra;
  - Görsel programlama dillerinde klasör ve dosyalar üzerinde işlemler yapabilecek,
  - Klasör ve dosya işlemlerinde yaygın kullanılan metot ve özellikler hakkında bilgi sahibi olacak,
  - Dosyalar üzerinde okuma ve yazma işlmelerini etkin bir şekilde yapabileceksiniz.

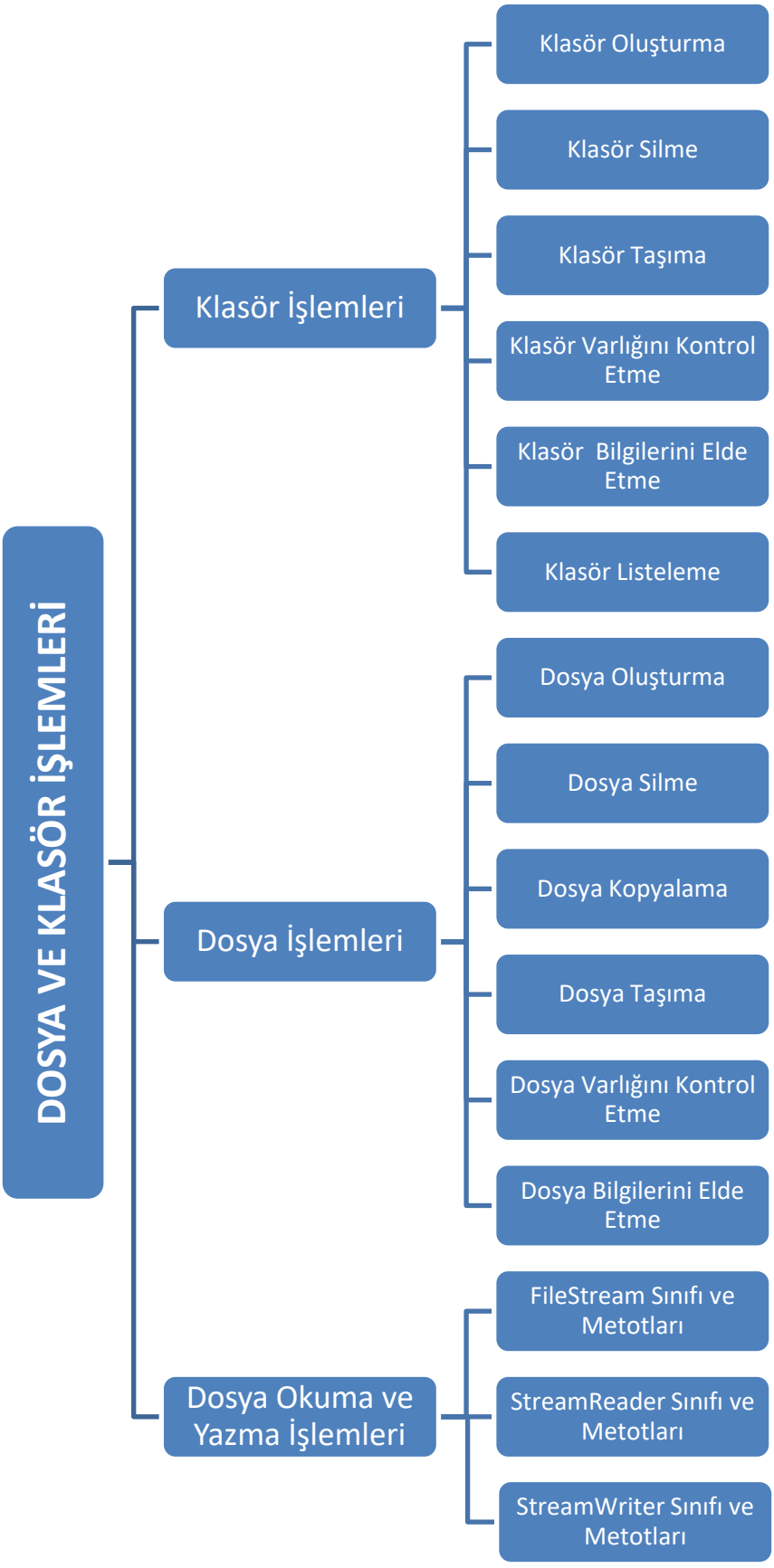
Atatürk Üniversitesi
Açıköğretim Fakültesi

GÖRSEL PROGRAMLAMA I

İshak Metehan
SİS

ÜNİTE

12


DOSYA VE KLASÖR İŞLEMLERİ
Klasör İşlemleri
Klasör Oluşturma
Klasör Silme
Klasör Taşıma
Klasör Varlığını Kontrol Etme
Klasör Bilgilerini Elde Etme
Klasör Listeleme
Dosya İşlemleri
Dosya Oluşturma
Dosya Silme
Dosya Kopyalama
Dosya Taşıma
Dosya Varlığını Kontrol Etme
Dosya Bilgilerini Elde Etme
Dosya Okuma ve Yazma İşlemleri
FileStream Sınıfı ve Metotları
StreamReader Sınıfı ve Metotları
StreamWriter Sınıfı ve Metotları

# GİRİŞ

Sabit disk, CD gibi belleklere kaydedilen veri ya da bilgiler "dosya" olarak adlandırılır. Verilerin organizasyonu sağlamak, dosyaların karışmasını önlemek, aranan bilgiye daha çabuk ulaşabilmek için ise klasör denilen yapılardan faydalanılır. Örneğin bilgisayardaki her bir resim dosya olarak, bu resimlerin bir arada tutulduğu yapı ise klasör olarak adlandırılır.

Sınırlı sayıdaki bilgilerin disk ortamında saklanması, bu bilgilere erişilmesi, üzerinde değişiklik yapılması gibi işlemler için veri tabanı oluşturmak yerine dosyalama işlemleri kullanılabilir. Dosya veya klasör işlemleri programlama dilleri ile yapılabilir.

Programlama dillerinde yer alan girdi çıktı işlemlerine İngilizcede Input/Output kelimesinin kısaltması olan I/O işlemleri denir. Klasör ve dosyalama işlemleri de bir girdi çıktı işlemi olduğu için C# dilinde girdi çıktı işlemleri için kullanılan "System.IO" kütüphanesi kullanılır (Petekçi, 2020)**.** Klasör işlemleri için "System.IO" kütüphanesinde yer alan "Directory" kullanılırken, dosya işlemleri için bu kütüphane içinde yer alan "File" sınıfı kullanılmaktadır.

"System.IO" kütüphanesi bir çok metoda sahiptir. Bu ünitede C# dili kullanılarak dosya ve klasör işlemlerini gerçekleştirmek için gerekli olan kütüphane, sınıf ve metotlar hakkında bilgi verip ilgili metotlar kullanılarak klasör ve dosya oluşturma, silme, listeleme, varlığını kontrol etme ve taşıma işlemlerini örnekler üzerinden detaylı olarak inceleyeceğiz. Son olarak dosya üzerinde okuma ve yazma işlemlerinin nasıl yapıldığı, bu işlemler için kullanılan metotlar ve özellikler hakkında bilgi verip örneklemeler yapacağız.

# DOSYA ve KLASÖR İŞLEMLERİ

Giriş kısmında da bahsedildiği gibi dosyalar, klasörler içerisinde yer alırlar. Bu yüzden ilk olarak klasör işlemlerinden bahsedilip daha sonra dosya işlemleri anlatılacaktır.

Klasör ve dosya işlemleri için System.IO kütüphanesinden yer alan sınıflar ve bu sınıflara ait metot ve fonksiyonlar kullanılmaktadır. Bu sınıfların derleyici tarafından tanınması için programın en başına *using System.IO* kütüphanesinin eklenmesi gerekmektedir.

## Klasör İşlemleri

C#'ta klasör işlemleri System.IO namespace'de bulunan *Directory* sınıfı ile yapılmaktadır.

### Klasör oluşturma

Klasör oluşturmak için Directory sınıfının "CreateDirectory" metodu kullanılır.

- Directory.CreateDirectory("klasör ismi);

C#'ta klasör veya dosya işlemleri yapılmadan önce programın başına using.System ve using.System.IO kodları eklenmelidir. Böylece I/O kütüphaneleri kod içine eklenmiş olur.

Programın çalıştığı dizinde belirtilen klasör isminde bir klasör oluşturulur. Çalışılan dizin dışında bir yerde klasör oluşturulmak istenirse "@" işaretinden sonra klasör yolunun belirtilmesi gerekir.

- Directory.CreateDirectory(@"C:\klasör ismi");
  C dizininde belirtilen klasör isminde bir klasör oluşturulur.

## Klasör silme

Klasör silmek için için Directory sınıfı ile birlikte "Delete" metodu kullanılır. Eğer klasör boş ise aşağıdaki örnekte de görüldüğü gibi Delete metodu ile klasör kolayca silinebilir. Dosyanın boş ya da dolu olduğuna bakılmadan silinmek isteniyorsa silinecek klasör yolu belirtildikten sonra *true* komutunun eklenmesi gerekir.

- Directory.Delete(@"C:\klasör ismi");
  C dizini altında içerisi boş olan klasör silinir.
- Directory.Delete(@"C:\ataaof", true);
  C dizini altında ki klasör dolu veya boş olduğu kontrol edilmeden silinir.

## Klasör taşıma

Klasör ve dosya taşıma işlemi için "Move" metodu kullanılır. Klasör taşıma işleminde dikkat edilmesi gereken nokta *klasör taşınırken kaynak ve hedef klasörün aynı dizinde olması gerekir*. Dosya taşıma işlemi için böyle bir kural geçerli değildir.

- Directory.Move(@"C:\ataaof", @"C:\Desktop\ataaof");
  Klasörü birinci parametredeki dizinden alıp, ikinci parametredeki dizine taşır.

## Klasör varlığını kontrol etme

Herhangi bir klasörün belirtilen yolda olup olmadığı "Exists" metodu ile kontrol edilir. Exists metıdu geriye true ya da false değeri döndürür.

## Klasör bilgilerini elde etme

Klasör oluşturma ya da değiştirme tarihi gibi klasör üzerinde yapılan işlemler hakkında bilgiye ihtiyaç duyulabilir. Bu durumda klasör kodları ile zaman kodları birlikte kullanılarak ihtiyaç duyulan bilgiler elde edilebilir.

- DateTime oluşturma_tarihi=Directory.GetCreationTime("klasör ismi");
  Klasörün oluşturulduğu tarih ve saat bilgisi elde edilir.
- DateTime erisim_zamani=Directory.GetLastAccessTime("klasör ismi");
  Klasöre en son erişilen zaman bilgisi elde edilir.
- DateTime veri_yazma=Directory.GetLastWriteTime("klasör ismi");
  Klasöre en son veri yazılan zaman bilgisi elde edilir.

"Directory.GetCurrentDirectory()" metodu aktif dizin bilgisini verir.

## Klasörleri listeleme

Klasör üzerinde işlemler yapılırken "@" işareti ile klasör adres bilgisi belirtilir. Aksi durumda programın çalışıldığı dizindeki klasörler belirtilmiş olur.

Belirtilen adresteki tüm klasörlerin isimlerini elde etmek için "GetDirectory" metodu kullanılır.

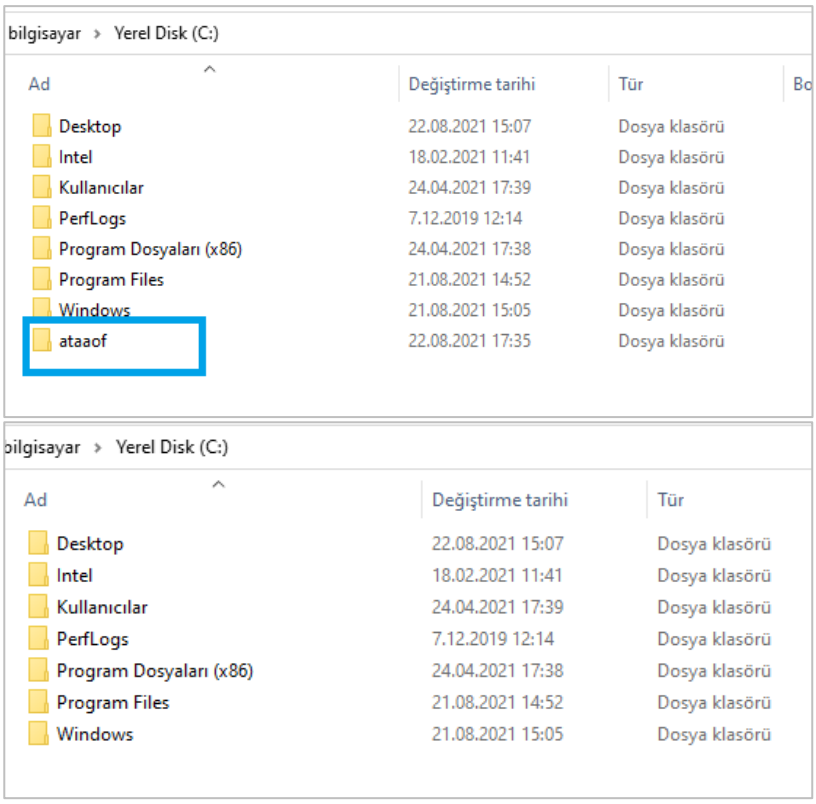
- string[] klasorler = Directory.GetDirectories("C: \Desktop");

Bu kod ile masa üstündeki klasörler elde edilmiş olur. Belirlenen bir klasördeki dosyaları listelemek için Directory metodu "*GetFiles*" ile birlikte kullanılır**.**

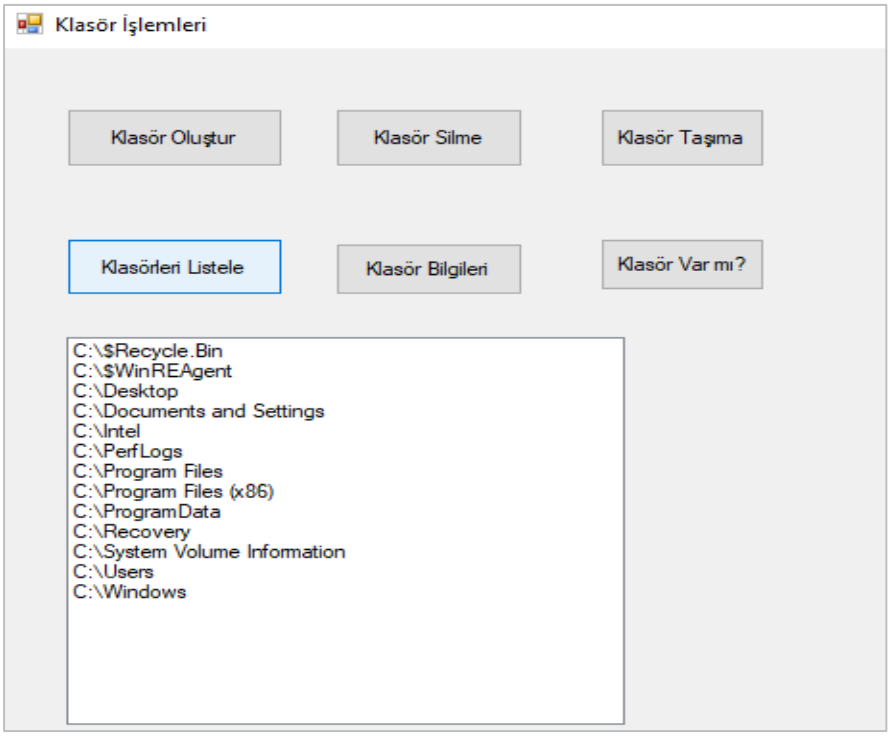
- string[] dosyalar = System.IO.Directory.GetFiles("D:");

Klasör üzerinde yapılabilecek işlemleri gösteren örnek bir uygulama;

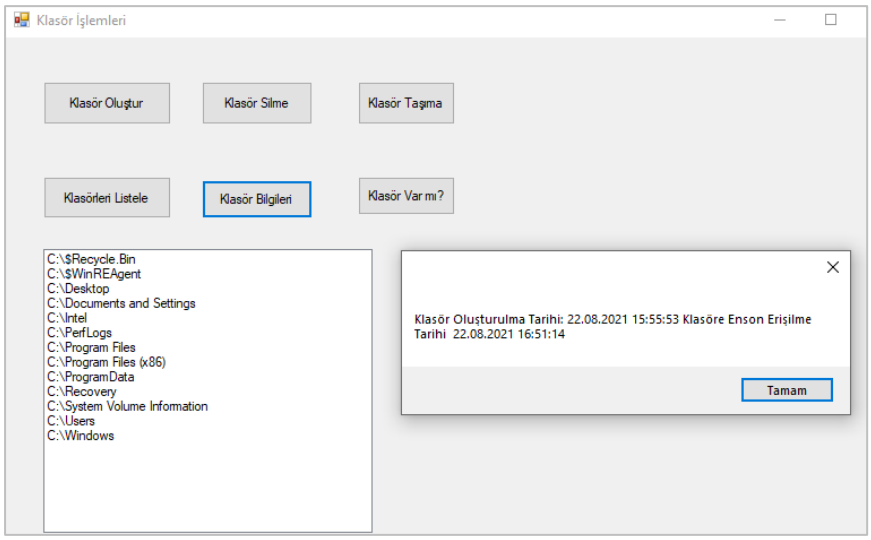
```
private void KlasorOlustur_Click(object sender, EventArgs e)
{
          // Klasör oluşturuldu.
          Directory.CreateDirectory(@"C:\ataaof");
}
private void KlasorSil_Click(object sender, EventArgs e)
{
          Directory.Delete(@"C:\ataaof"); //Silme işlemi yapıldı
}
private void Klasorler_Click(object sender, EventArgs e)
{
          //Dizindeki klasörler
          string[] klasorler = Directory.GetDirectories(@"C:\");
          for (int j = 0; j < klasorler.Length; j++)
          {
                    listBox1.Items.Add(klasorler[j]);
          }
}
private void KlasorTasima_Click(object sender, EventArgs e)
{
          //Klasör taşındı
          Directory.Move(@"C:\ataaof", @"C:\Users\İshak Metehan SİS\Desktop\ataaof");
}
private void KlasorBilgisi_Click(object sender, EventArgs e)
{
          //Klasör bilgileri elde edildi
          DateTime oluşturma_tarihi = Directory.GetCreationTime("ataaof");
          DateTime erisim_zamani = Directory.GetLastAccessTime("ataaof");
          MessageBox.Show("Klasör Oluşturulma Tarihi:"+" "+
          oluşturma_tarihi.ToString()+" "+"Klasöre Enson Erişilme Tarihi" +" "+" "+
          erisim_zamani.ToString());
}
private void KlasorVarmi_Click(object sender, EventArgs e)
{
          //Klasör varsa
          if (System.IO.Directory.Exists(@"C:\Users\ İshak Metehan SİS\Desktop\ataaof"))
                    MessageBox.Show("Klasör var.");
          Else //Klasör yoksa
                    MessageBox.Show("Klasör yok.");
}
```



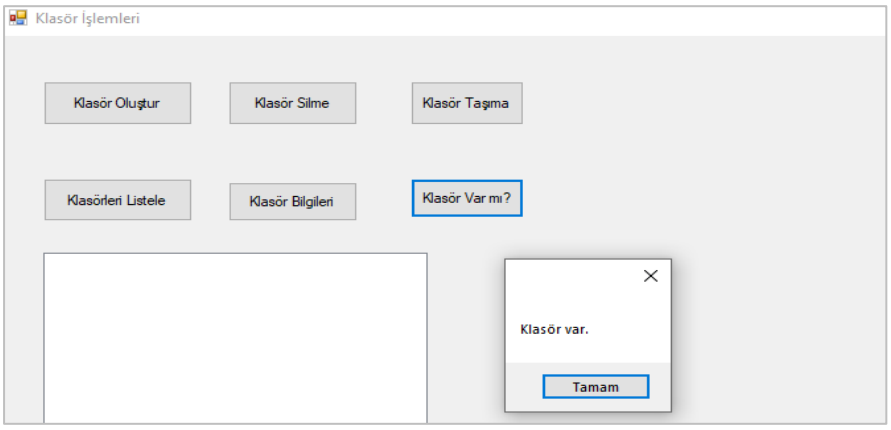

**Şekil 12.1.** Klasör Oluşturma ve Silme




**Şekil 12.2.** Klasör Listeleme




**Şekil 12.3.** Klasör Bilgileri




**Şekil 12.4.** Klasör Varlığını Kontrol Etme

## Dosya İşlemleri

Görsel programlamada dosya işlemleri System.IO isim uzayında bulunan *File* sınıfı ile yapılmaktadır.

### Dosya oluşturma

Dosya oluşturmak için File sınıfının "Create" metodu kullanılır.

- FileStream fs= File.Create(@"C:\dosyaismi.txt")

  Belirtilen dosya ismi ile yeni bir dosya oluşturulur.

### Dosya silme

Görsel programlamada dosya işlemleri File sınıfında bulunan "Delete" metodu ile yapılmaktadır. *Silinen dosyalar geri dönüşüm kutusuna gönderilmez.*

- File.Delete("C:\dosyaismi ")

  Belirtilen dosya silinir.

### Dosya kopyalama

Dosya kopyalama için "Copy" metodu kullanılır.

- File.Copy (@"D:\Eski Klasör\muzik.mp3", @"C:\Yeni Klasör\muzik.mp3");

Eski Klasör içerisinde yer alan müzik dosyası C dizininde yer alan Yeni Klasör içine kopyalanır.

## Dosya taşıma

Dosya taşıma için "Move" metodu kullanılır.

- File.Move(@"C:\Eski Klasör \metinbelgesi.txt", @"C:\Yeni Klasör metinbelgesi.txt");

İlk parametrede ki dosya, ikinci parametrede belirtilen adrese taşınır.

GetCreationTime(),GetLastAccessTime(),GetLastWriteTime() gibi metotlara adres olarak dosya adı ve uzantısı verildiğinde bu metotlar dosyalar için de geçerli olur.

## Dosya varlığını kontrol etme

Herhangi bir dosyanın olup olmadığı "Exists" metodu ile kontrol edilir. Exists metodu geriye true ya da false değeri döndürür.

## Dosya bilgilerini elde etme

Var olan bir dosyanın oluşturulma tarihi, değiştirilme tarihi ya da boyutu gibi bilgiler elde edilebilir. Daha önce ki bölümde klasör bilgilerini elde etmek için kullanılan GetCreationTime(),GetLastAccessTime(),GetLastWriteTime() gibi metotlar File sınıfı ile kullanıldığında dosyaya ait bilgileri verirler.

Ayrıca C# FileInfo sınıfını kullanarak da dosya ile ilgili bilgilere ulaşıp programlama üzerinde gerekli işlemler gerçekleştirilebilir (Öner, 2013).

Örnek

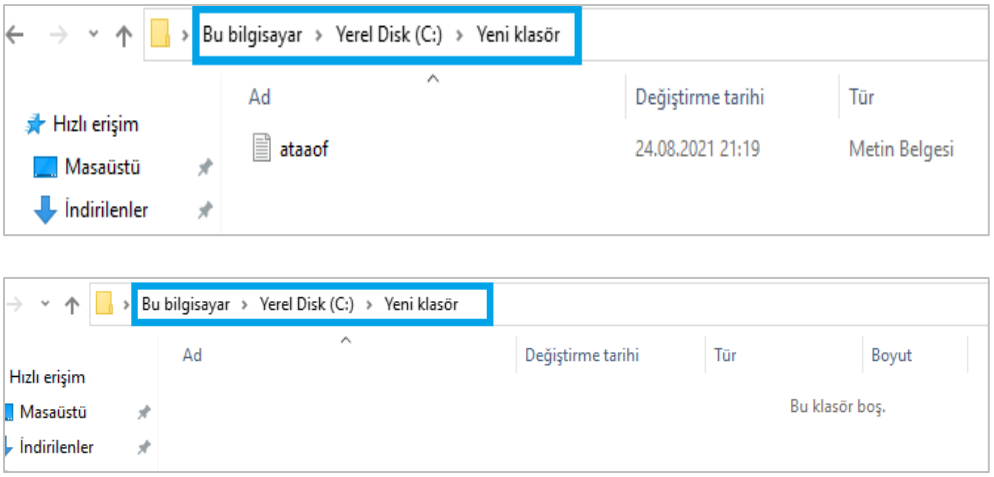
- FileInfo fi = new FileInfo("C:/dosya.txt");
  MessageBox.Show("Oluşturma Tarihi :" " + fi.CreationTime.ToString()+" "+ "Dosya Tam İsmi"+ " " + fi. fi.FullName.ToString()+" "+ "Dosya uzantısı:" + " "+ fi.Extension).ToString()););

Dosya üzerinde yapılabilecek işlemleri gösteren örnek bir uygulama;

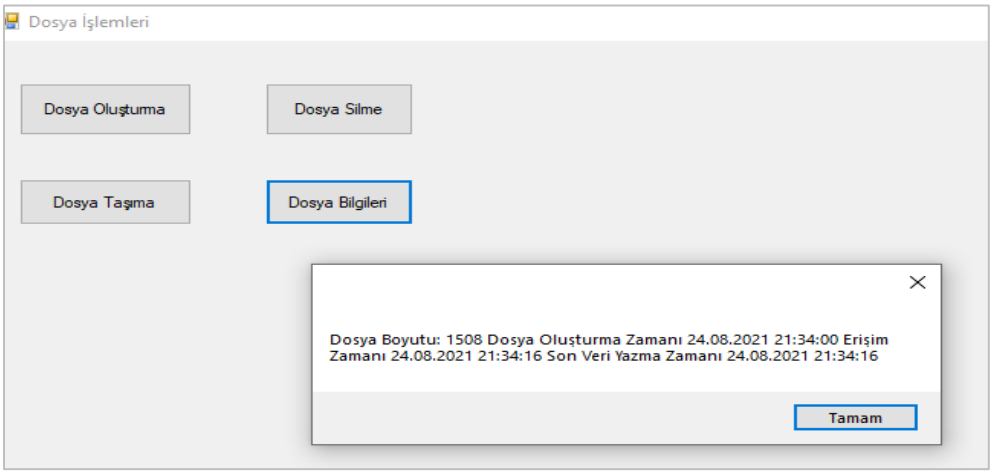
```
private void DosyaOlusturma_Click(object sender, EventArgs e)
{
        //Dosya oluşturuldu.
        FileStream fs = File.Create(@"C:\Yeni Klasör\ataaof.txt");
}
private void DosyaSilme_Click(object sender, EventArgs e)
{
        //Dosya silindi.
        File.Delete(@"C:\Yeni Klasör\ataaof.txt");
}
```

```
private void DosyaTasima_Click(object sender, EventArgs e)
{
        //Dosya taşındı.

        File.Move(@"C:\Users\İshak Metehan SİS\Desktop\EskiKlasör\ataaof.txt",
        @"C:\Users\İshak Metehan SİS\Desktop\YeniKlasör\ataaof.txt");
}
private void DosyaBilgileri_Click(object sender, EventArgs e)
{
         //Masaüstünde ataaof adlı dosyaya ait bilgiler çekilmiştir.
        FileInfo dosya = new System.IO.FileInfo(@"C:\Users\ İshak Metehan
        SİS\Desktop\ataaof.txt");
        DateTime olusmaZamani = File.GetCreationTime(@"C:\Users\ İshak Metehan
        SİS\Desktop\ataaof.txt");
        DateTime erisimZamani = File.GetLastAccessTime(@"C:\Users\ İshak Metehan
        SİS\Desktop\ataaof.txt");
        DateTime sonVeriYazma= File.GetLastWriteTime(@"C:\Users\ İshak Metehan
        SİS\Desktop\ataaof.txt");
        MessageBox.Show("Dosya Boyutu:"+ " "+ dosya.Length.ToString()+" " + "Dosya
        Oluşturma Zamanı"+" "+ olusmaZamani.ToString()+" "+ "Erişim Zamanı"+ " "+
        erisimZamani + " "+ "Son Veri Yazma Zamanı"+ " "+ sonVeriYazma.ToString());
}
```




**Şekil 12.5.** Dosya Oluşturma ve Silme




**Şekil 12.6.** Dosya Bilgileri

## Dosya Okuma ve Yazma İşlemleri

Dosya okuma ve yazma işlemleri içi FileStream, StreamReader ve StreamWriter sınıfları kullanılmaktadır.

### FileStream sınıfı ve metotları

FileStream sınıfı dosya oluşturmak için kullanılır.

- FileStream fS = new FileStream (string dosyaAdresi, FileMode dosya_modu);

İlk parametre dosyanın adresini ikinci parametre ise dosyanın ne amaçla açıldığını göstermektedir. FileStream sınıfı örnekte verildiği gibi sadece 2 parametreden oluşmamaktadır. FileAccess, FileShare, FileOptions gibi farklı parametreler alabilir.

- FileStream Stream(String, FileMode, FileAccess, FileShare, Int32, FileOptions, FileSecurity);
- **File Mode**

Dosya açılacak mı yeniden mi oluşturulacak bu seçenekleri belirler. FileMode seçenekleri Tablo 12.1'de gösterilmiştir (Gürsoy, 2012).

**Tablo 12.1.** FileMode Seçenekleri

| | **Açıklama** |
|---|---|
| **Open** | Dosyayı açar ve yazma işlemi mevcut içerik üzerine yapılır. |
| **Append** | Dosya var ise dosyanın sonuna ekleme yapar. Eğer dosya yoksa yeni dosya oluşturur. |
| **Create** | Yeni bir dosya oluşturulmak istendiğinde kullanılır. |
| **CreateNew** | Yeni bir dosya oluşturmak için kullanılır. Dosya mevcut ise hata verir. |
| **OpenOrCreate** | Belirtilen dosya varsa açılır, yoksa yeni dosya oluşturulur. |
| **Truncate** | Dosya açılarak içindeki bilgiler silinir. |

Append yalnızca FileAccess.Write ile birlikte kullanılabilir.

- **FileAcces**

Dosyaya nasıl erişileceğini belirler. FileAccess ile yapılacak işlemler Tablo 12.2'de yer almaktadır.

**Tablo 12.2.** FileAcces Seçenekleri

| | **Açıklama** |
|---|---|
| **Read** | Dosya okuma yetkisi ile erişim verir. Farklı bir işlem yapılamaz. |
| **Write** | Dosyaya sadece yazma işlemi için erişim verir. |
| **ReadWrite** | Dosya hem okuma hem de yazma işlemi yapma yetkisi ile açılır. |

- **FileShare**

FileShare ile dosyanın diğer strem'ler tarafından nasıl paylaşılacağı belirlenir. FileShare ile yapılacak işlemler Tablo 12.3'de yer almaktadır (Cesur, 2012).

**Tablo 12.3.** FileShare Seçenekleri

| | **Açıklama** |
|---|---|
| **None** | İşlem yapılan stream dışındaki diğer bir stream dosyaya yazma veya okuma için erişemez. |
| **Read** | İşlem yapılan streamler ya da diğerleri dosyaya okuma için erişim sağlayabilirler. |
| **Write** | İşlem yapılan streamler ya da diğerleri dosyaya yazma için erişim sağlayabilirler. |
| **ReadWrite** | İşlem yapılan streamler ya da diğerleri dosyaya okuma ve yazma için erişim sağlayabilirler. |

- **FileOptions**

FileOptions parametresi ile dosya seçenekleri belirlenir. FileOptions ile yapılacak işlemler Tablo 12.4'de yer almaktadır.

**Tablo 12.4.** FileOptions Seçenekleri

| | **Açıklama** |
|---|---|
| **None** | Hiçbir özellik yer almaz. |
| **WriteThrough** | Sistemin direkt disk üzerine yazılacağını belirtir. |
| **Asynchronous** | Dosya asenkron olarak okuma ve yazma işlemi yapacak şekilde ayarlanır. |
| **RandomAccess** | Dosyanın rastgele erişilebilecek durumda olduğu belirtilir. |
| **DeleteOnClose** | Kullanım dışı olan dosyanın otomatik olarak silineceğini belirtir. |
| **Encrypted** | Dosyanın şifrelendiğini belirtir. |

Ağdan, bilgisayardan veya fiziksel bir sürücüden gelen bit veya byte topluluklarına steam denir.

FileStream sınıfının yaygın kullanılan bazı özellikleri Tablo 12.5'te yer almaktadır (Yakar, 2017).

**Tablo 12.5** FileStream Özellikleri

| | **Açıklama** |
|---|---|
| **Length** | Aktif olan stream nesnesinin boyut bilgisini gösterir. |
| **Name** | Aktif olan stream nesnesinin isim bilgisini gösterir. |
| **Position** | Aktif olan stream nesnesinin o andaki pozisyon bilgisini gösterir. |
| **CanWrite** | Aktif stream nesnesinin yazılabilir olup olmadığını belirtir. |
| **CanRead** | Aktif stream nesnesinin okunabilir olup olmadığını belirtir. |
| **Handle** | İşletim sistemi tarafından stream için üretilen Handle numarasını belirtir. |
| **ReadTimeout** | Stream nesnesinin, zaman aşımından önce ne kadar süre okunabileceğini milisaniye cinsinden belirtir. |
| **WriteTimeout** | Stream nesnesinin, zaman aşımından önce ne kadar süre yazılabileceğini milisaniye cinsinden belirtir. |

| | |
|---|---|
| **IsAsync** | Aktif stream nesnesinin eşzamanlı ya da eş zamansız açılıp açılmadığını belirtir. |
| **CanSeek** | Aktif stream nesnesinin erişilebilirliğini ayarlar. |

FileStream sınıfının yaygın kullanılan bazı metotları Tablo 12.6'da yer almaktadır.

ReadByte metodu byte değil int türünde geri değer döndürür.

**Tablo 12.6.** FileStream Metotları

| | **Açıklama** |
|---|---|
| **BeginRead** | Bu metot ile okuma işlemi başlatılır. |
| **BeginWrite** | Bu metot ile yazma işlemi başlatılır. |
| **Seek** | Stream üzerinde yapılacak işlemin istenilen pozisyondan başlatılmasını sağlar. |
| **Flush** | Tampondaki bilgilerin boşaltılarak stream dosyasına yazılmasını sağlar. |
| **EndRead** | Okuma işlemi sonlandırılır. |
| **EndWrite** | Yazma işlemini sonlandırılır. |
| **Read** | Belirlenen byte kadar bilgi okur. |
| **ReadByte** | Tek bir byte okur, okuma işleminden sonra bir sonrakine geçer. |
| **Write** | Belirlenen byte kadar bilgi streame yazılır. Stream konumu yazılan byte kadar ilerletilir. |
| **WriteByte** | Tek bir byte yazar ve yazma işleminden sonra stream bir byte kayar. |
| **Close** | Stream kapatılır. |
| **Finalize** | Stream işleminde kullanılacak kaynaklar serbest bırakılır. |
| **Dispose** | Stream nesnesi tekrar açılmayacak ise bu metot ile silinebilir. |
| **GetType** | Aktif stream nesnesinin tipini bulur. |
| **ToString** | Aktif steream nesnesinin türünü string yapar. |
| **Unlock** | Stream nesnesini kilitleyerek diğer işlemler tarafından erişilmesini engeller. |
| **SetLength** | Stream nesnesinin uzunluğunu belirler. |

## StreamReader sınıfı ve metotları

StreamReader sınıfı txt, html, xml gibi uzantılı dosyalar içerisindeki metinleri okumak için kullanılır. StreamReader sınıfını kullanabilmek için ilk olarak *FileSteram* tanımlanır.

- FileStream fs= new FileStream(okunacakDosyaAdresi, FileMode.Open);
- StreamReader okuma= new StreamReader(fs);

SreamReader sınıfında kullanılan metotlar ve açıklamaları Tablo 12.7'de yer almaktadır (Chand, 2019).

Stream sınıfları kullanıldıktan sonra işlem bitince nesneler mutlaka kapatılmalıdır.

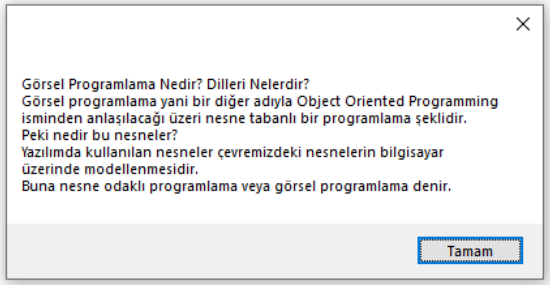
**Tablo 12.7.** StreamReader Metotları

| | **Açıklama** |
|---|---|
| **ReadLine** | Dosyayı satır satır okumak için kullanılır. Geri dönüş tipi stringtir. Dosyanın sonuna ulaşılınca "null" değeri döndürür. |
| **ReadToEnd** | Bu metot dosyanın sonuna kadar okuma yapılır. |
| **Peek** | Dosyada bir sonraki karakteri kontrol eder. Okunacak karakter bulamaz ise "-1" döndürür. |
| **Read** | Bu metot stremade ki sonraki karakteri okur. |
| **ReadAsync** | Bu metot stremade ki sonraki karakter ya da karakter topluluğunu eş zamansız olarak okur. |
| **GetType** | Nesnenin tipini alır. |
| **ToString** | Nesneyi string olarak değiştirir. |
| **Close** | Bu metot ile "StreamReader" kapatılır. |

ReadToEnd() metodu örnek kullanımı;

```
private void Form1_Load(object sender, EventArgs e)
{
    string dosya = "C:\\Users\\Desktop\\gorselprogramlama.txt";

    StreamReader sr = new StreamReader(dosya);
    string tumdosya = sr.ReadToEnd();
    MessageBox.Show(tumdosya);
    sr.Close();
}
```




**Şekil 12.7.** ReadToEnd Metodu

StreamWriter sınıfını kullanabilmek için ilk olarak FileSteram tanımlanır.

## StreamWriter sınıfı ve metotları

StreamWriter sınıfı ile dosyalara yazma işlemi yapılır.

- FileStream fs= new FileStream(yazılacakDosyaYolu, FileMode.Open);
- StreamWriter yazma= new StreamWriter(fs);

SreamWriter sınıfında kullanılan metotlar ve açıklamaları Tablo 12.8'de yer almaktadır (Chand, 2019).

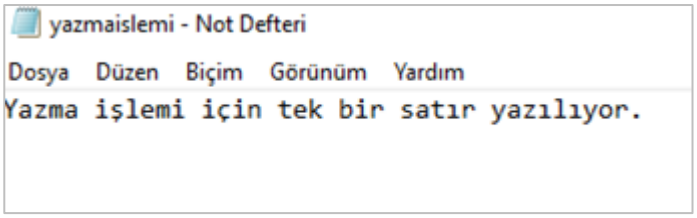
**Tablo 12.8.** StreamWriter Metotları

| | **Açıklama** |
|---|---|
| **Flush** | Mevcut "StreamWriter" için ara bellekte bekletilen verileri silerek ara belleği temizler. |
| **WriteLine** | Dosyaya verileri satır sonu işareti ekleyerek yazar. |
| **Write** | Dosyaya verileri satır sonu işareti eklemeden yazar. |
| **Close** | Bu metot ile "StreamWriter" kapatılır. |
| **GetType** | Nesnenin tipini alır. |
| **ToString** | Nesneyi string olarak değiştirir. |

WriteLine() metodu örnek kullanımı;

```
private void Form1_Load(object sender, EventArgs e)
{
        string dosya = "C:\\Users\\İshak Metehan Sis \\Desktop\\yazmaislemi.txt";
        FileStream stream = File.Create(dosya);
   StreamWriter sw = new StreamWriter(stream);
   //Tek satırlık .
   sw.WriteLine("Yazma işlemi için tek bir satır yazılıyor.");
        sw.Close();

}
```




**Şekil 12.8.** WriteLine Metodu

**Bireysel Etkinlik**

- Bir dosyanın içindeki veriyi satır satır okuyan C# form uygulmasını yazınız.

Özet

- DOSYA ve KLASÖR İŞLEMLERİ
- Sabit disk, CD gibi belleklere kaydedilen veri ya da bilgiler “dosya” olarak adlandırılır. Verilerin organizasyonu sağlamak, dosyaların karışmasını önlemek, aranan bilgiye daha çabuk ulaşabilmek için klasör denilen yapılardan faydalanılır. Örneğin bilgisayardaki her bir resim dosya, bu resimlerin bir arada tutulduğu yapı ise klasör olarak adlandırılır. Dosya veya klasör işlemleri programlama dilleri le yapılabilir. Programlama dillerinde yer alan girdi çıktı işlemlerine ingilizcede Input/Output kelimesinin kısaltması olan I/O işlemleri denir. Klasör ve dosyalama işlemleri de bir girdi çıktı işlemi olduğu için C# dilinde girdi çıktı işlemleri için kullanılan “System.IO” kütüphanesi kullanılır. Klasör işlemleri için “System.IO” kütüphanesinde de yer alan “Directory” kullanılırken, dosya işlemleri için bu kütüphane içinde yer alan “File” sınıfı kullanılmaktadır.
- Klasör İşlemleri
- Görsel programlamada klasör işlemleri namespacede bulunan Directory sınıfı ile yapılmaktadır
- Klasör oluşturma : Klasör oluşturmak için Directory sınıfının "CreateDirectory" metodu kullanılır.
- Klasör Silme: Klasör silmek için Directory sınıfı ile birlikte "Delete" metodu kullanılır.
- Klasör Taşıma: Klasör ve dosya taşıma işlemi için "Move" metodu kullanılır.
- Klasör varlığını kontrol etme : Herhangi bir klasörün belirtilen yolda olup olmadığı “Exists” metodu ile kontrol edilir.
- Klasör Bilgilerini Getirme: Klasörün oluşturma tarihi, erişilme zamanı ve klasör üzerinde son işlem yapılan tarih gibi bigiler sırası ile Directory.GetCreationTime, Directory.GeLastAccessTime, Directory.GetLastWriteTime metotları ile elde edilir.
- Klasör Listeleme: Belirtilen adresteki tüm klasörlerin isimlerini elde etmek için "GetDirectory" metodu kullanılır.
- Dosya İşlemleri
- C#'ta dosya işlemleri File sınıfı ile yapılmaktadır.
- Dosya oluşturma: Klasör oluşturmak için File sınıfının "Create" metodu kullanılır.
- Dosya silme:Dosya silmek için File sınıfının "Delete" metodu kullanılmaktadır.
- Dosya kopyalama:File sınıfının "Copy" metodu ile dosya kopyalama işlemi yapılır.
- Dosya taşıma: File sınıfının "Move" metodu ile taşıma işlemi yapılr.
- Dosya varlığını kontrol etme: Herhangi bir dosyanın olup olmadığı “Exists” metodu ile kontrol edilir.
- Dosya Bilgilerini Getirme: Dosya oluşturma tarihi, erişilme zamanı ve dosya üzerinde son işlem yapılan tarih gibi bigiler sırası ile File.GetCreationTime, FileLastAccessTime, File.GetLastWriteTime metotları ile elde edilir.

## Özet (devamı)

- Dosya Okuma ve Yazma İşlemleri
- Dosya okuma ve yazma işlemleri içi FileStream, StreamReader ve StreamWriter sınıfları kullanılmaktadır.
- FileStream sınıfı ve metotları
- FileStream sınıfı dosya oluşturmak için kullanılır. Dosya adı, FileShare, FileAccess, FileMode, FileOptions gibi parametreler alır. FileMode dosyanın açılış şeklini, FileAccess dosyaya nasıl erişileceğini, FileOptions dosya seçeneklerini, FileShare dosyanın nasıl paylaşıacağını belirler.
- StreamReader sınıfı ve metotları
- StreamReader sınıfı txt, html, xml gibi uzantılı dosyalar içerisindeki metinleri okumak için kullanılır. ReadLine, ReadToEnd, Peek, Read, GetType, ToString, Close gibi metotları bulunmaktadır.
- StreamWriter sınıfı ve metotları
- StreamWriter sınıfı ile dosyalara yazma işlemi yapılır. Flush, WriteLine, Write, Close gibi metotları mevcuttur.

# DEĞERLENDİRME SORULARI

1. Aşağıdakilerden hangisi C#'ta klasör ve dosya işlemleri yapabilmek için kullanılan kütüphanedir?
   a) System.Data
   b) System.Drawing
   c) System.IO
   d) System.Linq
   e) System.Text

2. Belirtilen dizindeki klasörlerin tamamını listelemek için aşağıdaki metotlardan hangisi kullanılır?
   a) GetDirectories()
   b) GetLastAccessTime()
   c) Exists ()
   d) GetCreationTime()
   e) GetLastWriteTime()

3. Aşağıdakilerden hangisi StreamReader() sınıfının metotlarından biri değildir?
   a) ReadToEnd()
   b) ReadLİne()
   c) Peek()
   d) Flush()
   e) Read()

4. İstenen dosyanın olup olmadığını kontrol eden metot aşağıdakilerden hangisidir?
   a) Create()
   b) Exists()
   c) Move()
   d) Copy()
   e) Delete()

5. Aşağıdakilerden hangisi FileStream() parametrelerinden biri değildir?
   a) FileMode.Create
   b) FileAccess.Write
   c) FileShShare.None
   d) FileOptions. Encrypted
   e) FileShare.Asynchronous

6. Aşağıda verilen program kodlarından hangisi D dizininde bulunan “ataaof” adlı klasörü içinde veri olup olmadığını kontrol etmeden siler?
   a) Directory.Delete(“ataaof");
   b) Directory.Delete(“ataaof",true);
   c) Directory.Delete(@"D:\ataaof",true);
   d) Directory.Delete(@"D:\ataaof, true”);
   e) Directory.Delete(@"C:\ataaof",true);

7. Dosya okuma yazma işleminden önce aşağıdaki sınıflardan hangisi eklenir?
   a) StreamReader
   b) StreamWriter
   c) Directory
   d) FileStream
   e) File

8. Aşağıdakilerden hangisi StreamWriter() sınıfının metotlarından biri değildir?
   a) Peek()
   b) WriteLine()
   c) Write()
   d) Close()
   e) GetType()

9. Aşağıdaki metotlardan hangisi belirtilen klasörün oluşturulma tarihini veriri?
   a) GetLastWriteTime ()
   b) GetCreationTime()
   c) GetDirectories()
   d) GetLastAccessTi()me
   e) GetType()

10. Dosyanın ilk satırını okumak isteyen bir kişi aşağıdaki metotlardan hangisini kullanmalıdır?
    a) ReadLine()
    b) ReadToEnd()
    c) Peek()
    d) Read ()
    e) ReadAsync()

**Cevap Anahtarı**
1.c, 2.a, 3.d, 4.b, 5.e, 6.c, 7.d, 8.a, 9.b, 10.a

# YARARLANILAN KAYNAKLAR

Cesur, K. (2012). FileStream Sınıfı. 21 Ağustos 2021 tarihinde http://www.kazimcesur.com/filestream/ adresinden erişildi.

Chand, M. (2019). StreamReader Sınıfı. 22 Ağustos 2021 tarihinde https://www.c-sharpcorner.com/article/working-with-c-sharp-streamreader/ adresinden erişildi.

Chand, M. (2019). StreamWriter Sınıfı. 22 Ağustos 2021 tarihinde https://www.c-sharpcorner.com/article/csharp-streamwriter-example/ adresinden erişildi.

Gürsoy, İ. (2012). FileStream Sınıfı. 20 Ağustos 2021 tarihinde https://www.ismailgursoy.com.tr/filestream-sinifi/ adresinden erişildi.

Öner, M. (2013). FileInfo Sınıfı. 19 Ağustos 2021 tarihinde https://www.muratoner.net/csharp/csharp-file-info-sinifini-kullanmak adresinden erişildi.

Petekçi, A. R. (2020). Dosya ve Klasör İşlemleri. 16 Ağustos 2021 tarihinde https://www.researchgate.net/publication/344209749_Programlama_Dillerinde Dosya ve Klasor Islemleri adresinden erişildi.

Yakar, C. (2017). Stream Kavramı. 18 Ağustos 2021 tarihinde https://www.cihanyakar.com/streamkavrami/ adresinden erişildi.