T.C.
MİLLÎ EĞİTİM BAKANLIĞI

BİLİŞİM TEKNOLOJİLERİ

METOTLAR
482BK0121

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR iii
GİRİŞ 1
ÖĞRENME FAALİYETİ–1 2
1. METOTLAR 2
1.1. Metot Kavramı 2
1.2. Metot Tanımlama 3
1.3. Metotlarda Parametre Kullanımı 5
1.4. Metotlarla ilgili Önemli Özellikler 9
1.5. Özyineli (Rekürsif-Recursive) Metotlar 14
1.6. Main( ) Metodu 15
UYGULAMA FAALİYETİ 20
ÖLÇME VE DEĞERLENDİRME 21
ÖĞRENME FAALİYETİ–2 22
2. HAZIR METOTLAR 22
2.1. Metinsel (String) Metotları 22
2.1.1. Compare( ) 23
2.1.2. Concat( ) 25
2.1.3. Copy( ) 26
1.2.4. Format( ) 26
2.1.5. IsNullOrEmpty( ) 32
2.1.6. CompareTo ( ) 33
2.1.7. Contains( ) 34
2.1.8. CopyTo( ) 35
2.1.9. EndsWith( ) 37
2.1.10. IndexOf( ) 38
2.1.11. Insert(int baslangic,string value) 40
2.1.12. LastIndexOf( ) 41
2.1.13. PadLeft ( ) 43
2.1.14. PadRight ( ) 44
2.1.15. Remove ( ) 46
2.1.16. Replace ( ) 47
2.1.17. Split ( ) 49
2.1.18. StartsWith ( ) 49
2.1.19. Substring ( ) 50
2.1.20. ToLower ( ) 51
2.1.21. ToUpper ( ) 52
2.2. Matematiksel (Math) Metotları 52
2.2.1. Abs( ) 52
2.2.2. BigMul( ) 53
2.2.3. Ceiling( ) 53
2.2.4. DivRem( ) 54
2.2.5. Max( ) 54
2.2.6. Min( ) 55
2.2.7. Pow( ) 56
2.2.8. Round( ) 56

2.2.9. Sign( ) 57
2.2.10. Sqrt( ) 57
2.2.11. Cos( ) 59
2.2.12. Sin( ) 59
2.2.13. Tan( ) 60
2.2.14. Acos( ) 61
2.2.15. Asin( ) 61
2.2.16. Atan( ) 61
2.3. Tarih/Saat (DateTime) Metotları 61
2.3.1. MinValue 61
2.3.2. MaxValue 62
2.3.3. Today 62
2.3.4. Now 63
2.3.5. DateTime.Compare( ) 65
2.3.6. DateTime.DaysInMonth( ) 65
2.3.7. DateTime.IsLeapYear( ) 66
2.3.8. DateTime.Parse( ) 66
2.3.9. Subtract( ) 67
2.3.10. AddDays( ) 68
2.3.11. AddMonths( ) 68
2.3.12. AddYears( ) 68
2.3.13. AddHours( ) 68
2.3.14. AddMinutes( ) 69
2.3.15. AddSeconds( ) 69
2.3.16. AddMilliseconds( ) 69
UYGULAMA FAALİYETİ 70
ÖLÇME VE DEĞERLENDİRME 71
MODÜL DEĞERLENDİRME 72
CEVAP ANAHTARLARI 73
KAYNAKÇA 74

AÇIKLAMALAR

KOD	**482BK0121**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Metotlar**
MODÜLÜN TANIMI	Bu modül temel programlama işlemlerinde metotlar yazabilmenin öğrenildiği bir öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Bu modülün ön koşulu yoktur.
YETERLİK	Metotlarla çalışmak
MODÜLÜN AMACI	**Genel Amaç** Bu modül ile gerekli ortam sağlandığında; temel programlama işlemlerinde metotlar yazabileceksiniz. **Amaçlar** **1.** Metotları kullanabileceksiniz. **2.** Hazır metotları kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Bilgisayar laboratuarı **Donanım:** Bilgisayar, Programlama Yazılımı
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış testi, boşluk doldurma vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

Programlama Temelleri dersinin bu modülünde sizler, programlamanın bir diğer temel yapı taşı olan "Metotları" öğreneceksiniz.

Bilgisayar programcılığına giden bu yolda, önemli bir adımı da bu modülü öğrenerek atacaksınız.

Program yazarken belli bir işi yapan kod bloğunu bir kaç kez kullanmak gerekebilir. Bunun için aynı kod bloğunu tekrar yazmak yerine bunu bir metot olarak hazırlarız ve ihtiyaç duyduğumuz yerde metodu ismi ile çağırarak çalıştırabiliriz. Bu bize daha az kod yazma imkânı sağlayıp zaman kazandırdığı gibi, olası değişikliklerde de daha az hata yapmamızı sağlar.

Bu modülde birçok programlama dilinin temel kavramlarından olan metotları detayları ile inceleyeceğiz. Metotların tanımları ve kullanımı, parametrelerin özellikleri örneklerle işlenecektir.

ÖĞRENME FAALİYETİ–1

AMAÇ

Bu modül ile gerekli ortam sağlandığında; temel programlama işlemlerinde metotlar yazabilecek ve bunları programlarınızda kullanabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Çeşitli programlama dillerindeki alt programlar kavramlarını araştırınız.
- Fonksiyonların çalışma mantığı hakkında ön bilgi edininiz.
- Parametre, geri dönüş değeri nedir? Araştırınız.

1. METOTLAR

Programların hazırlanması esnasında, aynı işlemi gerçekleştiren program parçalarına programın birçok yerinde ihtiyaç duyulabilir. Bu ihtiyaçlar, metotlar yazılarak giderilir. Eğer metotlar kullanılmazsa; programda aynı kodu defalarca yazmamız gerekebilir ve program kodlarının okunması zorlaşır. Aynı zamanda kaynak kodun gereksiz uzamasına sebep olur. Bunun için programın birçok yerinde ihtiyaç duyulan ve aynı işlemleri yapan program parçaları metotlar olarak hazırlanırlar.

1.1. Metot Kavramı

Programların herhangi bir yerinde kullanılmak için belirli bir işi yerine getirmek amacıyla tasarlanmış alt programlara metot denilir. Metotlar belirli bir işi yapması için geliştirilirler.

Bir sefere mahsus yazılan bu kod parçaları programın akışı içerisinde defalarca çağrılarak kullanılabilirler.

Metotların amacı; programın yapısal olmasını sağlamak ve birbiriyle ilgili komutları veya programın bir bölümünü istenen isim altında toplamaktır. Bu şekilde programın okunması kolaylaşmakta ve yapısal bir görünüm kazanmaktadır.

Bir metot, bir veya daha fazla ifade içerebilir. İyi yazılmış bir programda, her metot yalnızca tek bir görev yürütür.

Metotlar tek başına çalışabilen yapılar değillerdir. Ancak ana program içerisinden çağrılarak çalıştırılırlar.

1.2. Metot Tanımlama

Her metodun bir ismi vardır ve program içerisinde metot çağrılırken bu isim kullanılır.

Bir metodun iş yapabilmesi için kendi çağıran metottan aldığı bilgilere **parametre**, kendisini çağıran fonksiyona döndürdüğü değere de **metot geri dönüş değeri** (return value) denir.

Metotlar genellikle şu şekilde tanımlanırlar;

- **Tanımlanması:**

```
erişim dönüş-tipi isim(parametre-listesi)
{
 //    metodun gövdesi;
}
```

Erişim: Bu metoda, programın diğer bölümlerinin nasıl erişebileceğini belirleyen bir erişim niteleyicisidir. Bunun kullanımı isteğe bağlıdır. Eğer herhangi bir erişim belirteci kullanılmazsa varsayılan olarak sınıfa özel (**private**) olarak belirlenir. Private olarak kullanıldığında yalnızca metodun yazıldığı sınıf içerisinden çağrılabilmesini öngörür. Eğer programın içerisinde bulunan diğer kodlar içerisinden de bu metot çağrılabilsin isteniyorsa, erişim belirteci **public** olarak belirtilmelidir. Nesne yönelimli programlama dillerinde metotlar, tanımlandıkları sınıf adı ile birlikte çağrılırken eğer metot, programın ana metodu (Main()) içerisinden çağrılacaksa **static** olarak tanımlanır ve sınıf adını yazmaya gerek kalmadan çağrılır.

Dönüş-tipi: Bu metodun çalıştırıldıktan sonra programda çağrıldığı noktaya döndürdüğü verinin tipini belirlediğimiz kısımdır. Eğer metot bir değer döndürmeyecekse dönüş-tipi **void** olarak belirtilmelidir.

İsim: Metodunun isminin belirtildiği kısımdır. Metodumuza isim verirken yapacağı iş ile alakalı bir isim vermek hem metodun ne işe yaradığıyla ilgili bize bilgi verecektir, hem de bizden başka aynı programı kodlayacak kimselere yol gösterecektir. Metoda isim verirken aynı değişken isimleri tanımlarken kullandığımız kuralları yine göz önünde bulundurmalıyız. Geri dönüş tiplerinin veya parametre-listesinin farklı olması durumunda aynı isme sahip birden fazla metot olabilir.

Parametre-Listesi: Virgül (**,**) ile ayrılmış tip ve tanımlayıcı çiftlerden oluşan bir listedir. Parametreler, metot çağrıldığında, metodun kullanması için gönderilen bilgilerdir. Eğer metot hiç parametre kullanmayacaksa parametre listesi de boş olur.

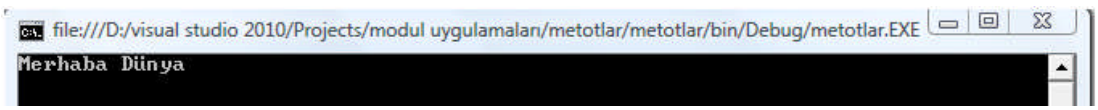
Örneklerle metot tanımlamalarını inceleyelim;

Örnek 0-1: Geri dönüş değeri ve parametre-listesi boş olan, ekrana "Merhaba Dünya" yazdıran metodu tanımlayıp program içerisinde kullanımına bir örnek veriniz.

```
static void MerhabaDunyaYazdir()
{
    Console.WriteLine("Merhaba Dünya");
}

static void Main(string[] args)
{
    MerhabaDunyaYazdir();
}
```

Yukarıdaki kod parçaları çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız.



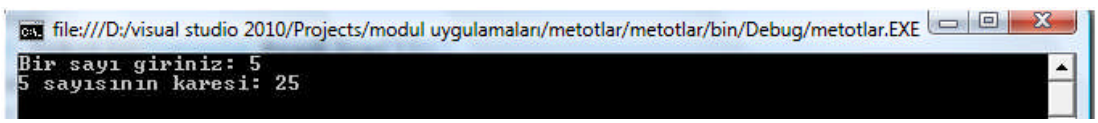
Resim 0.1: Parametresiz ve Geri Dönüş Türü Void Olan Metot Tanımlama

Örnek 0-2: Klavyeden girilen bir tam sayının karesini bulan metodu ve bu metodun program içerisinde kullanımını gösteren programın kodunu yazınız.

```
static int KareAl(int sayi)
{
    int karesi = sayi * sayi;
    return karesi;
}

static void Main(string[] args)
{
    Console.Write("Bir sayı giriniz: ");
    int s1,sonuc;
    s1=Convert.ToInt32(Console.ReadLine());
    sonuc = KareAl(s1);
    Console.WriteLine("{0} sayısının karesi: {1}",s1,sonuc);
}
```

Yukarıdaki kod parçaları çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız.



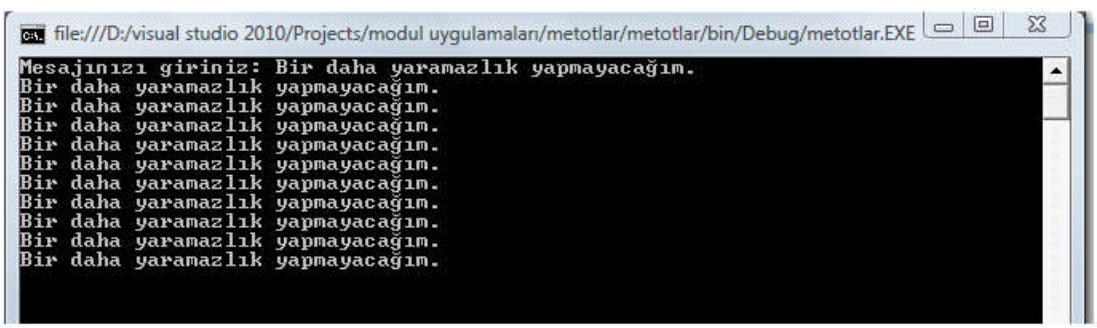
Resim 0.2: Geri Dönüş Değeri ve Parametre-Listesi Olan Metot Tanımlama

Örnek 0-3: Klavyeden girilen bir mesajı ekrana 10 defa yazdıran metodun kodunu yazınız.

```
static void MesajYaz(string msj)
{
    for (int i = 1; i <= 10; i++)
        Console.WriteLine(msj);
}

static void Main(string[] args)
{
    Console.Write("Mesajınızı giriniz: ");
    string mesaj;
    mesaj=Console.ReadLine();
    MesajYaz(mesaj);
}
```

Yukarıdaki kodları incelediğimiz zaman; klavyeden girilen yazı mesaj isimli değişken içerisine aktarılmakta ve daha sonra MesajYaz() isimli metota gönderilmektedir. MesajYaz() metodu ise kendisine parametre olarak verilen string türdeki mesajı ekrana 10 defa yazmaktadır. Çalıştırıldığında aşağıdaki gibi bir ekran çıktısıyla karşılaşırız.



Resim 0.3: Geri Dönüş Türü Void olan Parametreli Metot Uygulaması

1.3. Metotlarda Parametre Kullanımı

Parametrenin tanımını ve kullanımını daha önce metotların tanımlanması sırasında parametre listelerini oluştururken gördük.

Parametre-listeleri, tek bir türde verileri içeren bir liste olabileceği gibi, farklı türlerde de veriler içerebilen listelerdir.

Parametreler veri türünde olabileceği gibi nesneler de parametre olarak bir metoda gönderilebilirler.

Her bir parametre aralarına virgül kullanılarak birbirinden ayrılırlar. Aynı veri türüne sahip parametrelerin her biri için değişken isimlerinden önce ayrı ayrı veri türleri de yazılmak zorundadır.

Çeşitli veri türlerini parametre olarak metotlarımıza nasıl gönderdiğimizi örneklerle inceleyelim;

Örnek 0-4: Klavyeden girilerek parametre olarak gönderilen bir sayının, asal sayı olup olmadığını kontrol eden, eğer sayı asal ise **true**, değilse **false** değeri döndüren metodu yazınız.

```
    static bool AsalMi(int s)
    {
        bool durum=false;
        for (int i = 2; i < s / 2 + 1; i++)
        {
            if (s % 2 == 0)
                durum=false;
            else
                durum=true;
        }
        return durum;
    }

    static void Main(string[] args)
    {
        int sayi = 0;
        bool drm;
        Console.Write("Bir sayı giriniz: ");
        sayi=Convert.ToInt32(Console.ReadLine());
        drm=AsalMi(sayi);
        if (drm == true)
            Console.WriteLine("{0} sayısı asaldır.",sayi);
        else
            Console.WriteLine("{0} sayısı Asal değildir.",sayi);
    }
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
16	
43	
66	
89	

Örnek 0-5: Parametre olarak gönderilen kullanıcı adı ve şifreyi kontrol eden, önceden belirlenmiş olan bir kullanıcı adı ve şifreyle karşılaştıran metodun kodlarını yazınız.

```
    static void KullaniciKontrol(string kAdi, string psw)
    {
       if ((kAdi == "Admin") || (kAdi == "ADMİN") || (kAdi ==
"admin"))
```

```
        {
            if (psw == "123rty")
               Console.WriteLine("Tebrikler Kullanıcı ve Şifreniz
Doğru");
            else
               Console.WriteLine("Şifrenizi Hatalı Girdiniz");
        }
        else
        {
            Console.WriteLine("Kullanıcı adınız hatalı.");
        }
    }
    static void Main(string[] args)
    {
        string kullaniciAdi, sifre;
        Console.Write("Lütfen kullanıcı adınızı giriniz: ");
        kullaniciAdi = Console.ReadLine();
        Console.Write("Lütfen şifrenizi giriniz: ");
        sifre = Console.ReadLine();
        KullaniciKontrol(kullaniciAdi, sifre);
    }
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		**Ekran Çıktısı**
Kullanıcı Adı	**Şifre**	
Admin	123RTY	
Yönetici	123rty	
Admin	123rty	
admin	123rty	
ADMİN	123rty	

Örnek 0-6: Klavyeden girilen değerler arasında rastgele sayı üreten ve bu değerleri 10 elemanlı bir dizi içerisine atayan **SayiUret( )** isimli bir metot yazınız. Dizinin elemanlarını ekrana yazdıran **DiziYazdir( )** isimli bir metot daha yazarak elemanları ekrana yazdırınız. Daha sonra bu dizi içerisindeki en büyük sayı değerini bulan **EnBuyuk( )** isimli, en küçük değeri bulan **EnKucuk( )** isimli iki metot daha yazınız. EnBuyuk ve EnKucuk metotlarından dönen sayıları ekrana yazdıran programın kodlarını yazınız.

Bu kısımda rastgele sayılar üretilip parametre olarak gönderilen **dizi** isimli diziye değerler aktarılıyor ve dizi ana programa geri döndürülüyor;

```
    static int[] SayiUret(int bas, int bit,int[] dizi)
    {
        int tutulan = 0;
        Random rnd = new Random();
```

```
        for (int i = 0; i < 10; i++)
        {
            tutulan = rnd.Next(bas, bit);
            dizi[i] = tutulan;
        }
        return dizi;
    }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki değerler ekrana yazdırılıyor;

```
    static void DiziYazdir(int[] dizi1)
    {
        Console.WriteLine("----------------");
        Console.WriteLine("Tutulan sayılar:");
        foreach (int i in dizi1)
            Console.WriteLine(i);
        Console.WriteLine("----------------");
    }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en büyük değer bulunup ana programa geri döndürülüyor;

```
    static int EnBuyuk(int[] dizi2)
    {
        int ebs=0;//en küçük değer
        foreach (int s in dizi2)
        {
            if (s > ebs) //eğer sayı ebs'den büyükse
                ebs = s; //yeni ebs, sayının değeri olur
        }
        return ebs;
    }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en küçük değer bulunup ana programa geri döndürülüyor;

```
    static int EnKucuk(int[] dizi3)
    {
        int eks = 100;//en büyük değer
        foreach (int x in dizi3)
        {
            if (x < eks) //eğer sayı ebs'den küçükse
                eks = x; //yeni eks, sayının değeri olur
        }
        return eks;
    }
```

Ana programımız;

```
    static void Main(string[] args)
    {
       int[] sayilar=new int[10];
       int baslangic, bitis;
       Console.Write("Başlangıç değerini giriniz: ");
       baslangic = Convert.ToInt32(Console.ReadLine());

       Tekrar: Console.Write("Bitiş değerini giriniz: ");
       bitis = Convert.ToInt32(Console.ReadLine());

       if (bitis <= baslangic)
       {
           Console.WriteLine("Bitiş değeri başlangıçtan ({0}) küçük
ya da eşit olamaz tekrar deneyiniz.",baslangic);
            goto Tekrar;
        }
        //Rastgele sayılar üretilip diziye aktarılıyor
        sayilar = SayiUret(baslangic, bitis, sayilar);
        //Dizi ekrana yazdırılıyor
        DiziYazdir(sayilar);
        //En büyük değer bulunuyor
        int mak = EnBuyuk(sayilar);
        //En küçük değer bulunuyor
        int min = EnKucuk(sayilar);

        //Sonuçlar ekrana yazdırılıyor
        Console.WriteLine("En büyük sayı: " + mak);
        Console.WriteLine("En küçük sayı: " + min);
    }
```

1.4. Metotlarla ilgili Önemli Özellikler

Metotlarla ilgili bilinmesi gereken bazı önemli özellikler şunlardır;

- Metotlara isim verilirken aynı değişkenlere isim verirken uyduğumuz kurallara uymamız gerekir. **Main( )** ismi programımızın çalışmasını başlatan ana metodun ismi olduğu için bu ismi metot ismi olarak veremeyiz.

- Aynı isime sahip farklı geri dönüş tiplerine veya farklı parametre-listesine sahip metotlar oluşturabiliriz.

```
    static int Topla(int sayi1, int sayi2, int sayi3)
    {
        int toplam;
        toplam = sayi1 + sayi2 + sayi3;
        return toplam;
    }
    static int Topla(int sayi1, int sayi2)
    {
```

```
        int toplam;
        toplam = sayi1 + sayi2;
        return toplam;
    }
    static void Topla(int sayi1)
    {
        Console.Write("Parametresiz metodun sonucu= {0}", sayi1);
    }
    static void Main(string[] args)
    {
        int sonuc, s1, s2, s3;
        s1 = 43;
        s2 = 16;
        s3 = 66;
        sonuc = Topla(s1, s2, s3);
        Console.WriteLine("3 parametreli metodun sonucu= {0}",
sonuc);
        sonuc = Topla(s1, s2);
        Console.WriteLine("2 parametreli metodun sonucu= {0}",
sonuc);
        Topla(s1);
    }
```

Bu yöntem pek de tavsiye edilen bir yöntem değildir. Bu şekilde aynı isme sahip farklı metotlar oluştururken çok dikkatli olmalıyız.

- Metotlar çağrılırken, başlangıçta belirlenen parametre sayısından ne az ne de çok sayıda parametre girmeliyiz. Eğer metodumuz 2 parametre ile işlem yapıyorsa, biz bu metoda 1 veya 3 adet parametre gönderemeyiz. Aksi takdirde hata mesajı alırız.

```
    static int Topla(int sayi1, int sayi2)
    {
        int toplam;
        toplam = sayi1 + sayi2;
        return toplam;
    }

    static void Main(string[] args)
    {
        int sonuc,s1,s2,s3;
       s1=43;
       s2=16;
       s3=66;

        sonuc = Topla(s1, s2, s3);
        sonuc = Topla(s1);

      Console.WriteLine ("Sonuç= {0}",sonuc);
    }
```

Yukarıdaki kodları incelediğimizde; **Topla** isimli metoda ait parametre listesinde iki adet parametre alabileceği tanımlanmış. Ancak ilk koyu renkli satırdan da göreceğimiz üzere metot çağrılırken 3 parametre gönderildiğinde veya bir sonraki satırdaki gibi tek parametre gönderildiğinde hata mesajı alırız.

- Metotların geri dönüş değerleri vardır. Geri dönüş değeri olmayacak olan metotlarda geri dönüş tipi **void** olarak belirtilir ve **return** anahtar kelimesinin bu türdeki metotlarda kullanımına izin verilmez.

```
static void Topla(int sayi1, int sayi2)
{
    int toplam;
    toplam = sayi1 + sayi2;
    return toplam;
}
```

Yukarıdaki kodları incelediğimizde; geri dönüş tipi belirtilmeyen (**void** olarak tanımlanan) metottan **return** anahtar kelimesi kullanılarak geriye bir değer döndürülmeye çalışıldığında hata mesajı alırız.

- Geri dönüş türü void olan metotlar, herhangi bir değişken içerisine atanamazlar.

```
static void Topla(int sayi1, int sayi2)
{
    int toplam;
    toplam = sayi1 + sayi2;
    Console.WriteLine(toplam);
}

static void Main(string[] args)
{
    int sonuc,s1,s2,s3;
   s1=43;
   s2=16;
   s3=66;
    sonuc = Topla(s1, s2);
    Console.WriteLine ("Sonuç= {0}",sonuc);
}
```

Yukarıdaki kodları incelediğimizde; geri dönüş değeri bulunmayan Topla isimli metot, - koyu renkli satırdan da görüleceği üzere - **sonuc** isimli değişkene atama işlemi yapılmaya çalışılırsa hata mesajı alırız.

- Metotların geri dönüş değerleri herhangi bir veri türünde olabilir. Metot içerisindeki bir değer **return** anahtar sözcüğüyle metodun çağrıldığı yere geri döndürülür. Burada metodun geri dönüş tipine uyumlu bir değişken içerisine atanmalıdır. Aksi takdirde tür uyumsuzluğundan dolayı hata mesajı alırız.

```
static float Topla(int sayi1, int sayi2)
{
    float toplam;
    toplam = sayi1 + sayi2;
    return toplam;
}
static void Main(string[] args)
{
    int sonuc,s1,s2,s3;
   s1=43;
   s2=16;
   s3=66;
    sonuc = Topla(s1, s2);
    Console.WriteLine ("Sonuç= {0}",sonuc);
}
```

Yukarıdaki kodları incelediğimizde; **int** türünde tanımlanmış olan **sonuc** değişkeni içerisine **float** türünde tanımlanmış bir metodun geri dönüşü atanmaya çalışılırsa hata mesajı alırız.

- Bir metot parametre almadan da tanımlanabilir. Bu şekilde tanımlanan bir metoda parametre gönderilmez. Parametre-listesi parantezleri boş bırakılır.

```
static void yazdir()
{
    Console.Write("Merhaba Dünya");
}
static void Main(string[] args)
{
    yazdir();
}
```

- Metotlar tanımlanırken oluşturulan parametre-listesindeki tüm parametreler virgül (,) ile birbirinden ayrılmalıdır. Tek bir tür yazıp virgülle değişken isimlerini ayıramayız.

```
static float Topla(int sayi1,sayi2)
{
    float toplam;
    toplam = sayi1 + sayi2;
    return toplam;
}
```

Yukarıdaki gibi bir parametre listesi tanımlamaya çalışırsak hata mesajı alırız. Her bir parametreyi virgülle ayırarak tek tek tanımlamamız gerekir.

```
static float Topla(int sayi1, int sayi2)
{
    float toplam;
    toplam = sayi1 + sayi2;
    return toplam;
}
```

- Parametre-listesinde tanımlanan değişkenlerin isimleri, metot içerisinde tanımlanacak başka bir değişkende tekrar kullanılamaz.

```
static float Topla(int sayi1, int sayi2)
{
    float toplam;
    int sayi1,sayi2;
    toplam = sayi1 + sayi2;
    return toplam;
}
```

Yukarıdaki gibi bir parametre listesinde tanımlanmış **sayi1** ve **sayi2** değişken isimlerini, metot içerisinde tekrardan her ne veri türünde olursa olsun yeniden kullanamayız.

- Bir metot içerisinde başka bir metot tanımlanamaz. Ancak başka bir metot çağrılabilir.

```
static void yazdir()
        {
            static void MerhabaDunyaYazdir()
            {
                Console.Write("Merhaba Dünya");
            }
        }
```

Yukarıdaki kullanım hatalı bir kullanımdır. Metot içerisinde metot tanımlaması yapılamaz. Ancak aşağıdaki örnek gibi metot içerisinden başka bir metot çağrılabilir.

```
static void MerhabaDunyaYazdir()
{
    Console.Write("Merhaba Dünya");
}

static void Yazdir()
{
    MerhabaDunyaYazdir();
}
```

Yazdir isimli metot içerisinden **MerhabaDunyaYazdir** isimli metodu yukarıdaki gibi çağırabiliriz.

- Metotların içerisinde tanımlanan tüm değişkenler metot dışında kullanılamazlar, geçersiz olurlar.

```
static float Topla(int sayi1, int sayi2)
{
    int toplam;
    toplam = sayi1 + sayi2;
    return toplam;
}
```

```
    static void Main(string[] args)
    {
        int s1, s2;
        s1 = 43;
        s2 = 16;
        toplam = Topla(s1, s2);
        Console.WriteLine("Sonuç: "+toplam);
    }
```

Yukarıdaki kodlar incelendiğinde; **Topla** isimli metot içerisinde tanımlanan float türündeki **toplam** isimli değişkene metot dışında tekrar erişilmek istenirse hata mesajı alırız.

1.5. Özyineli (Rekürsif-Recursive) Metotlar

Bir metodun kendi kendini çağırmasına ***yinelenme*** (recursion), kendi kendini çağıran metotlara da ***yinelenen*** veya ***özyineli*** (recursive) ***metotlar*** denir.

Bir metodun kendi kendini çağırması, zaman zaman da olsa program yazarken ihtiyaç duyulan bir olaydır. Yinelenen metotlar tasarlanırken çok dikkatli olunmalıdır. Aksi takdirde sonsuz bir döngü içerisine girilebilir. Bu döngünün bir şekilde sonlandırılması gerekmektedir.

Yinelenen metotlara basit bir örnekle giriş yapalım;

Örnek 0-7: 1'den klavyeden girilen sayıya (**n**) kadar olan sayıların toplamını hesaplayan programın kodunu yazınız.

```
    static int Topla(int n)
    {
        if (n == 0)
            return 0;
        return n + Topla(n - 1);
    }

    static void Main(string[] args)
    {
        int sonuc = Topla(sayi);
        Console.WriteLine("1'den {0} sayısına kadar olan sayıların
toplamı = {1}", sayi, sonuc);
    }
```

Hemen hemen bütün bilgisayar programlama kitaplarında yinelenen metotlarla ilgili klasik olarak bir sayının faktöriyelini hesaplayan programlara yer verilir. Bizlerde bu geleneği bozmayalım ve bir önceki örnekteki mantıkla klavyeden girilen bir sayının faktöriyelini hesaplayan özyineli bir metot hazırlayalım.

Örnek 0-8: Klavyeden girilen sayının faktöriyelini hesaplayan programın kodunu yazınız.

```
    static double Faktoriyel(int n)
    {
        if (n == 0)
            return 1;
        return n * Faktoriyel(n - 1);
    }
```

Faktoriyel( ) isimli metot incelendiğinde; return anahtar kelimesiyle birlikte Faktoriyel( ) metodu parametre (**n**) değeri 1 azaltılarak tekrar tekrar çağrılıyor. Ne zaman ki parametre (**n**) değeri 0'a erişti, o zaman metot içerisinden çıkılıyor.

```
    static void Main(string[] args)
    {
       tekrar:
           Console.Write("Bir sayı giriniz: ");
           int sayi= Convert.ToInt32(Console.ReadLine());
       if (sayi < 0)
       {
           Console.WriteLine("Negatif Sayıların Faktöriyeli
Hesaplanmaz...");
          goto tekrar;
      }
      else
      {
          double sonuc = Faktoriyel(sayi);
          Console.WriteLine("{0}! = {1}", sayi, sonuc);
      }
    }
```

Yinelenen metotları bir bakıma iç içe geçen bir radyo anteni veya teleskop gibi kimi zaman uzayıp, kimi zaman kısalabildiğini düşünebiliriz.

1.6. Main( ) Metodu

Metotlar modülümüzün başından bu yana örneklerimizde hep Main( ) isminde bir metot içerisinde ana programlarımızın yazımını gerçekleştirdik. Peki nedir bu Main( ) metodu?

Aslında Main( ) metodu şimdiye kadar yazdığımız veya kullandığımız metotlardan pek de farkı olmayan bir metot türüdür. Tek ve en önemli farkı Main( ) metodunun ana programın başlamasını sağlayan nokta olmasıdır. İşte bu yüzden Main( ) metodu diğer metotlara göre daha özeldir.

Şimdiye kadar Main( ) metodunu kullanırken herhangi bir geri dönüş değeri kullanmadık, hep geri dönüşünü **void** olarak belirledik. Ancak bazı durumlarda Main( ) metodunun **void** dışında **int** türünde bir geri dönüş değeri de kullanılır. Bu geri dönüş değeri, aslında biz kullanıcıların pek de işine yarayacak bir geri dönüş değeri değildir. Bu değer, genellikle (bütün programların üzerinde çalıştığı) işletim sisteminin, yazılan programın nasıl sonlandırıldığıyla ilgili bilgi almasını sağlayacak bir değerdir.

int türünde bir değer döndüren Main( ) metodu şu şekilde tanımlanır;

```
static int Main()
{
    ...
}
```

Burada dikkat ederseniz Main( ) metodunun geri dönüş değeri **void** yerine **int** türüne sahiptir. Bu **int** türündeki değer genellikle programın nasıl sonlandırıldığı bilgisini işletim sistemine göndermeye yarar.

Eğer dönüş değeri;

- Sıfır (0) ise program normal bir şekilde sonlandırılmış,
- Sıfırdan farklı ise programın bir hata sebebiyle sonlandırılmış,

olduğunu belirtir.

Main( ) metodunun geri dönüş değerinin olmasının yanında, bazı durumlarda da parametre alması mümkündür.

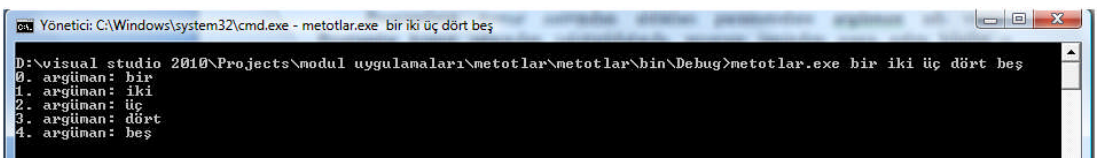
Programların *komut satırı*ndan aldıkları parametrelere ***argüman*** adı verilir. Programlar komut satırından çalıştırıldığında, program isminden sonra gelen bilgiler o programın argümanlarıdır.

Örneğin aşağıdaki resimde **metotlar.exe** isimli program, komut satırından çağrılırken, program isminden sonra bir takım argümanlar eklenmiş ve bunlar Main( ) metodu içerisinde sırayla ekrana yazdırılmış.

Komut satırı:

```
D:\...........\bin\Debug\metotlar.exe bir iki üç dört beş
```

Buradaki yol sizin bilgisayarınızda çalıştığınız klasörlere göre elbette ki değişecektir.



Resim 0-4. Main( ) Metodunun Argümanlarının Listelenmesi

Yukarıdaki gibi Main( ) metodunun argümanlarının listelenmesi için aşağıdaki gibi bir kod yazmak yeterli olacaktır;

```
static void Main(string[] args)
{
    int i=0;
    foreach (string a in args)
    {
        Console.WriteLine(i + ". argüman: " + a);
        i++;
    }
}
```

Örnek 0-9: Komut satırını kullanarak iki sayının toplamını bulan programı yazınız.

```
static int Main(string[] args)
{
    int argumanAdedi = args.Length;

    if (argumanAdedi == 0 || argumanAdedi == 1 )
    {
        Console.WriteLine("Eksik Parametre Girildi");
        return 1;
    }

    if (argumanAdedi > 2)
    {
        Console.WriteLine("Fazla Parametre Girildi");
        return 1;
    }

    int sayi1 =Convert.ToInt32(args[0]);
    int sayi2 =Convert.ToInt32(args[1]);

    int sonuc = sayi1 + sayi2;
    Console.WriteLine("Girilen sayıların toplamı: "+sonuc);
    return 0;
 }
```

Yukarıdaki kodları inceleyelim;

```
static int Main(string[] args)
```

Satırı ile Main( ) metodu tanımlanıyor. Bu metoda string türünde bir dizi (**args**) parametre olarak gönderiliyor.

```
int argumanAdedi = args.Length;
```

Satırı ile de gönderilen parametre sayısı argumanAdedi değişkenine aktarılıyor.

```
if (argumanAdedi == 0 || argumanAdedi == 1 )
```

ve

```
if (argumanAdedi > 2)
```

Programımızla 2 adet sayının toplamını bulacağımız için argüman sayısının 2'den az mı yoksa çok mu olduğunu kontrol etmemiz gerekiyor. Eğer yukarıdaki kontrollerde argüman sayısı 2'den az veya çok ise `return 1;` ile işletim sistemine yeterli argüman girilmediğini bildiren hata mesajları gönderiliyor.

Eğer girilen argüman sayısı 2 ise aşağıdaki satırlar çalıştırılır ve gerekli işlemler gerçekleştirilir.

```
    int sayi1 =Convert.ToInt32(args[0]);//1. argüman
    int sayi2 =Convert.ToInt32(args[1]);//2. argüman
    int sonuc = sayi1 + sayi2;
    Console.WriteLine("Girilen sayıların toplamı: "+sonuc);
    return 0; //Hatasız bir şekilde sonlandırıldığını belirtir
```

Bu programımız çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız.

Resim 0-5. Main( ) Metoduyla Komut Satırından İşlem Yapma

Metotlar.exe isimli programımız komut satırından çalıştırıldığında;

- Hiç parametre girilmezse veya sadece 1 parametre girilirse "Eksik Parametre Girildi" mesajıyla karşılarız.
- 3 veya daha fazla sayıda parametre girilirse "Fazla Sayıda Parametre Girildi" mesajıyla karşılaşırız.
- 2 parametre girişi gerçekleştirirsek, programımız girilen bu iki sayının toplamını ekranımıza yazdırır.

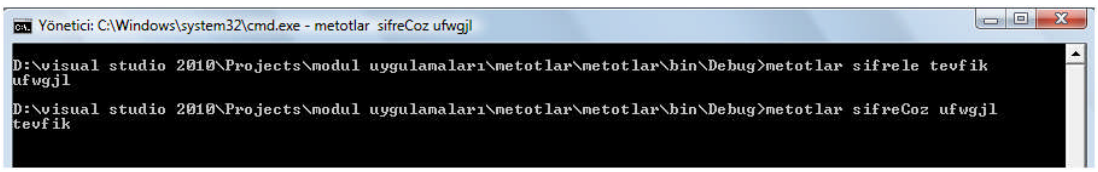
Örnek 0-10: Klavyeden komut satırına girilen metni şifreleyen ve şifreyi çözen programı yazınız.

```
    static int Main(string[] args)
    {
        int argumanAdedi = args.Length;
        if (argumanAdedi == 0 || argumanAdedi == 1)
        {
            Console.WriteLine("Eksik Parametre Girildi");
            return 1;
        }
        if (argumanAdedi > 2)
        {
            Console.WriteLine("Fazla Parametre Girildi");
            return 1;
        }
        string anahtarKelime = args[0];
        string kelime= args[1];
        for (int i = 0; i < kelime.Length; i++)
        {
            if (anahtarKelime == "sifrele")
                Console.Write((char)(kelime[i] + 1));
            else if (anahtarKelime == "sifreCoz")
```

```
                Console.Write((char)(kelime[i] - 1));
            }
            Console.ReadLine();
            return 0;
        }
```

Şifreleme işlemi olarak yaptığımız şey; klavyeden girilen metin içerisindeki harfleri tek tek ele alıp, o harfi, kendinden bir sonra gelen harfe dönüştürmektir. Şifre çözme işlemi ile de şifrelenmiş kelimenin harflerini geri eski hallerine getirmek, yani ilgili harften bir önceki harfe dönüştürmektir.

Yukarıdaki kodları çalıştırdığımızda aşağıdaki gibi bir ekran görüntüsü alırız.



Resim 0-6. Metin Şifreleme

Programda ilk olarak, **sifrele** parametresiyle birlikte "**tevfik**" kelimesi gönderdik sonuç olarak "**ufwgjl**" kelimesini bize geri döndürdü. Daha sonra **sifreCoz** parametresi ile "**ufwgjl**" kelimesini gönderdiğimizde, geri "**tevfik**" kelimesini döndürdü.

Sizler de program vasıtasıyla şifreleme algoritmasının üzerinde değişiklik yaparak, kendi kelimelerinizi şifreleyebilir, daha sonra da çözebilirsiniz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➢ Klavyeden girilen sayıların tek mi çift mi olduğunu bulan ve sayı tek ise geri "Sayı Tektir.", çiftse "Sayı Çifttir." Mesajını döndüren metodu yazınız.	➢ Bir sayının 2'ye bölümünden kalan sıfır(0) ise sayı çifttir. ➢ Mod alma (%) işlemi kullanınız.
➢ Parametre olarak gönderilen metni şifreleyen ve geri şifrelenmiş halini döndüren, ➢ Parametre olarak gönderilen şifrelenmiş metnin şifresini çözen ve geri çözülmüş halini döndüren, ➢ Metotları yazınız.	➢ Şifreleme işlemi, kelimedeki harflerin alfabeye göre 1 ötelenmesi ile ➢ Şifre çözme işlemi de kelimedeki harflerin alfabeye göre 1 geri alınmasıyla Gerçekleştirilecektir. ***Örneğin:*** *A harfi şifrelendiğinde B harfi olacaktır.*
➢ AlanHesapla isimli metot, klavyeden kenar uzunlukları girilen bir dikdörtgenin alanını hesaplayıp ekrana yazdıran bir metottur. Bu metodu yazınız.	➢ Metoda uzun kenar ve kısa kenar olmak üzere 2 parametre gönderilecektir. ➢ Metottan geriye dikdörtgenin alan değeri döndürülecektir.
➢ Klavyeden 3 kenar uzunluğu girilen bir üçgenin, dik üçgen olup olmadığını kontrol eden metodu yazınız.	➢ Uzun kenarının karesi, kısa kenarlarının karelerinin toplamına eşit olan üçgenler dik üçgenlerdir.
➢ Matematikteki Fibonacci sayı dizisinin ilk 10 elemanını bulan özyineli (rekürsif) metodu yazınız.	➢ Fibonacci sayı dizisi: kendisi ve kendisinden önce gelen sayının toplamının kendisinden sonra gelen sayıya eşit olduğu sayılar dizisidir. 0-1-1-2-3-5-8-13…… ***Fibonacci(n)= Fibonacci(n-1)+ Fibonacci(n-2)***
➢ Parametre olarak gönderilen gün sayısının kaç yıl, kaç ay, kaç gün olduğunu hesaplayan metodu yazınız.	➢ 1 yıl = 360 gün ➢ 1 ay = 30 gün Baz alınacaktır. Örnek: 1943 gün = 5 yıl 4 ay 23 gün
➢ Erkeklerin 25 yıl, kadınların da 20 yıl çalıştıktan sonra emekli olacağı varsayıldığında, klavyeden girilen cinsiyet ve sigorta prim gün sayısına göre kişinin emekliliğine ne kadar süresi kaldığını bulan metodunu yazınız.	➢ Bir önceki örnekteki metoda cinsiyet parametresini de ekleyerek, buna göre sonucu ekrana yazdırınız. Örneğin: Erkek ve 6643 gün parametreleri girildiğinde "Emekliliğinize 6 yıl 6 ay 17 gün kaldı" şeklinde sonuç döndürecek.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Parametre-listesi bir metottan geri değer döndürmek için kullanılır.
2. ( ) Bir metodun çağrıldığı zaman aldığı ve metot içerisinde kullanacağı değişkenler listesine parametre-listesi denir.
3. ( ) Parametre listesi int türünde değişkenler içeren bir tür dizidir.
4. ( ) Parametre listesindeki değişkenler aynı türde olmak zorundadırlar.
5. ( ) Eğer herhangi bir erişim belirteci kullanılmazsa varsayılan olarak sınıfa özel (private) olarak belirlenir.
6. ( ) Eğer metot bir değer döndürmeyecekse dönüş-tipi void olarak belirtilmelidir.
7. ( ) Bir metot hiç parametre kullanmayacaksa, parametre-listesi boş bırakılır.
8. ( ) Parametre listesindeki değişkenler birbirinden noktalı virgül (;) ile ayrılırlar.
9. ( ) Main( ) isimli metot, programımızın çalışmasını başlatan metottur.
10. ( ) Bir metot kendi kendini hiçbir şartta çağıramaz.
11. ( ) Aynı isme sahip metotlar oluşturamayız.
12. ( ) Parametre-listesinde belirtilen bir değişken ismi, metot içerisinde başka bir veri türünde tekrar kullanılabilir.
13. ( ) Metotlarda geri dönüş değeri out anahtar kelimesiyle gönderilir.
14. ( ) Geri dönüş türü void olan metotlar, herhangi bir değişken içerisine aktarılamazlar.
15. ( ) Metot içerisinde başka bir metot tanımlaması yapılabilir.
16. ( ) Metot içerisinde başka bir metot çağrılabilir.
17. ( ) Özyineli metotların dönüş değeri olamaz.
18. ( ) Main( ) metodunun geri dönüş değeri void veya int türünde olabilir.
19. ( ) Main( ) metodu hiç parametre almaz.
20. ( ) Bir program içerisinde birden fazla Main( ) metodu kullanabiliriz.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Bu modül ile hazır metotları kullanabilecek ve programlarınızda uygulayabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız

- String değişken türleri nasıl tanımlanır? Araştırınız.
- Sayılarla kullanabildiğimiz değişken türleri araştırınız.
- Tarih/Zaman ifadeleri ile birlikte kullandığımız değişken türlerini araştırınız.

2. HAZIR METOTLAR

Programlama dili kütüphaneleri içerisinde önceden tanımlanmış ve programcıların işlerini kolaylaştıran bir takım hazır metotlar vardır.

Bu modülümüz içerisinde metinsel (String), matematiksel, tarih ve zaman işlemlerinde sıkça kullanacağımız metotlarını inceleyeceğiz.

2.1. Metinsel (String) Metotları

Programlama dili içerisindeki String sınıfı altında bulunan ve metinsel (String) ifadelerle ilgili bir takım işlemleri daha kolay yapabilmek için bir takım hazır metotlar vardır. Metinsel metotlardan sık kullanılanlar şunlardır;

- String sınıfı ile çağırılan metotlar;

  Compare, Concat, Copy, Format, IsNullOrEmpty

- String ifade ile birlikte çağrılan metotlar;

  CompareTo, Contains, CopyTo, EndsWith, IndexOf, Insert, LastIndexOf, PadLeft, PadRight, Remove, Replace, Split, StartsWith, Substring, ToLower, ToUpper

2.1.1. Compare( )

Parametre olarak verilen iki string ifadeyi karşılaştırır ve geriye int türünde bir veri döndürür. Eğer dönüş değeri sıfır (0) ise iki metin birbirine eşittir. Aksi takdirde parametre olarak verilen metinleri ilk harflerinden itibaren tek tek karşılaştırır ve farklılığın olduğu ilk harflerin alfabedeki sıralarına göre -1 veya 1 sayı değerlerini döndürür.

- **Kullanımı:**

```
int donusDegeri=String.Compare(metin1,metin2);
```

Aşağıdaki tabloda metinlerin karşılaştırma durumları ve geri dönüş değerleri verilmektedir;

Durum	Dönüş Değeri
metin1>metin2	1
metin1=metin2	0
metin1<metin2	-1

Burada metinlerin büyüktür/küçüktür karşılaştırmaları, harflerin alfabetik sırasıyla ilgilidir.

Örnek 0-1: Klavyeden girilen iki metnin karşılaştırılmasını yapan programın kodlarını yazınız.

```
    Console.Write("1. metni giriniz: ");
    string metin1 = Console.ReadLine();
    Console.Write("2. metni giriniz: ");
    string metin2 = Console.ReadLine();
    int donusDegeri = String.Compare(metin1, metin2);

    switch (donusDegeri)
    {
        case -1:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinden
farklıdır. \nGeri Dönüş değeri: {2}",metin1,metin2,donusDegeri);
           break;
        case 0:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinin
aynısıdır. \nGeri Dönüş değeri: {2}", metin1, metin2, donusDegeri);
            break;
        case 1:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinden
farklıdır. \nGeri Dönüş değeri: {2}", metin1, metin2, donusDegeri);
            break;
    }
```

Yukarıdaki programı çalıştırdıktan sonra aşağıdaki tabloda verilen değerleri tek tek deneyerek String.Compare( ) metoduyla geri dönen değerleri, tabloda boş bırakılan alanlara yazınız.

Girilecek Değerler		**Geri Dönüş Değeri**
metin1	**metin2**	
Tavşanlı	tavşanlı	
Tavşanlı	Tavsanlı	
Tavşanlı	Tavşanlı	
Tavşanlı	Davşanlı	
Tavşanlı	Tavışanlı	

Eğer metin karşılaştırmalarında büyük/küçük harfe dikkat edilsin istemiyorsak Compare( ) metodunun bir başka kullanımı olan Compare(metin1,metin2,boolean) formunu kullanmamız gerekir.

- **Kullanımı:**

```
bool buyukKucuk=true;
int donusDegeri=String.Compare(metin1,metin2,buyukKucuk);
```

Burada **bool** türündeki değişkenin değeri **true** ise **Compare( )** metodu büyük/küçük harfe bakmasızın iki kelimeyi karşılaştırır. Eğer **false** değeri gönderilirse, bu durumda karşılaştırma işlemini büyük/küçük harfe biçimde gerçekleştirir.

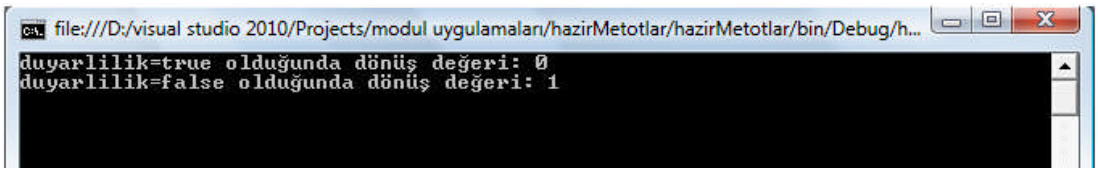
Örnek 0-2: "Tavşanlı" ve "tavşanlı" metinlerini büyük/küçük harf duyarlı olarak ve büyük/küçük harf duyarsız olarak iki şekilde karşılaştırılmasını yapan programın kodlarını yazınız.

```
string metin1="MOYMUL";
string metin2="moymul";

// Büyük/Küçük harf duyarsız
bool duyarlilik=true;
int donusDegeri = String.Compare(metin1, metin2,duyarlilik);
Console.WriteLine("duyarlilik=true olduğunda dönüş değeri: " +
donusDegeri);

// Büyük/Küçük harf duyarlı
duyarlilik=false;
donusDegeri = String.Compare(metin1, metin2,duyarlilik);
Console.WriteLine("duyarlilik=false olduğunda dönüş değeri: " +
donusDegeri);
```

Yukarıdaki program çalıştırıldığında aşağıdaki gibi bir ekran çıktısı elde ederiz.



Resim 0.1: Compare Metodu Büyük/Küçük Harf Duyarlılığı

Bu programdaki metin değerlerini değiştirerek farklı büyük küçük harf duyarlılıklarını inceleyiniz.

2.1.2. Concat( )

Parametre olarak verilen nesneleri string türünde birbirine peşisıra ekler ve geriye string türünde bir değer döndüren String metodudur.

- **Kullanımı:**

```
string donenMetin=String.Concat(parametre-listesi);
```

Örnek 0-3: Klavyeden girilen iki metni birleştiren programın kodlarını yazınız.

```
Console.Write("1. metni giriniz: ");
string metin1 = Console.ReadLine();
Console.Write("2. metni giriniz: ");
string metin2 = Console.ReadLine();
string birlestirilen = String.Concat(metin1, metin2);
Console.WriteLine("{0} ve {1} kelimelerinin birleştirilmiş
hali: {2}",metin1,metin2,birlestirilen);
```

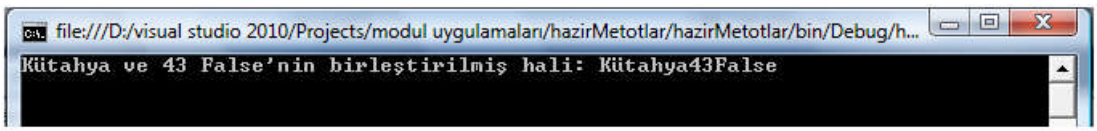
Yukarıdaki kodlar çalıştırıldığında ve metin1 olarak "Linyit", metin2 olarak da "spor" kelimeleri girildiğinde birleştirilen kelime "Linyitspor" olarak karşımıza çıkacaktır.

String.Concat( ) ile farklı türlerdeki verileri de birleştirme şansımız vardır. Aşağıdaki örnekte hem metin, hem sayı, hem de boolean türündeki verileri birleştirip, geriye string türünde bir veri elde etme işlemini inceleyeceğiz.

Örnek 0-4: Metin, sayı ve boolean türünde verilerin birleştirilmesini sağlayan programın kodlarını yazınız.

```
string metin = "Kütahya";
int sayi = 43;
bool durum = false;
string birlestirilen = String.Concat(metin, sayi, durum);
Console.WriteLine("{0} ve {1} {2}'nin birleştirilmiş hali:
{3}", metin, sayi,durum,birlestirilen);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran çıktısı karşımıza çıkar;



Resim 0.2: String.Concat( ) Metodu ile Farklı Türlerdeki Verilerin Birleştirilmesi

2.1.3. Copy( )

Parametre olarak verilen string türündeki metnin bir kopyasını almaya yarayan String metodudur.

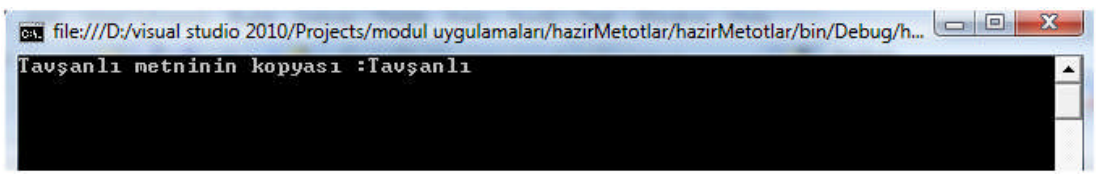
- **Kullanımı:**

```
string kopyaMetin=String.Copy(metin);
```

Örnek 0-5: Bir metinin kopyasını alan programı yazınız.

```
string metin = "Tavşanlı";
string kopyaMetin=String.Copy(metin);
Console.WriteLine("{0} metninin kopyası :{1}", metin,
kopyaMetin );
```

Yukarıda kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız.



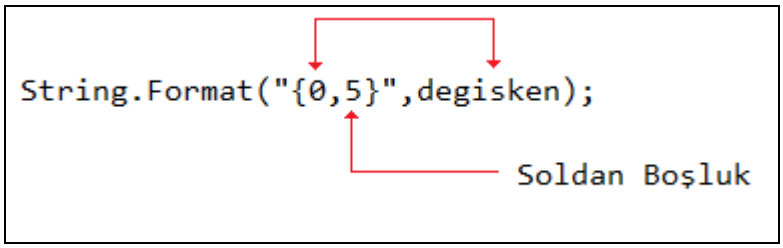
Resim 0-3. String.Copy( ) Metodu ile String Kopyalama

1.2.4. Format( )

Programlama esnasında bazı ifadeleri belirli bir biçim içerisinde yazmamız istenirse String.Format( ) metodu kullanılır.

Bu metot geriye **string** türünde bir veri döndürür.

- Örneğin **metinsel ifadelerin** belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;



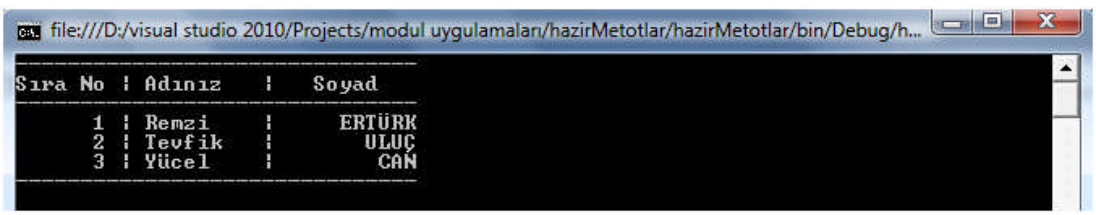
Resim 0.4: String.Format( ) Metodu Kullanımı

Yukarıdaki resmi incelediğimizde String.Format( ) metodunun kullanımında, küme parantezleri ({ }) içerisindeki ilk değer, degisken isimli değişken içerisindeki değeri referans göstermektedir. İkinci değer ise değişkenin içeriğinin, ekranda kaç karakterlik alan kaplayacağını (Örn: Bu değer 5 ise ekranda 6 karakterlik, -7 ise 8 karakterlik yer kaplar) belirtir. Bu değerin pozitif olması, değişken değerinin ayrılan alanının sağına hizalı olacağını, negatif olması ise soluna hizalı olacağını belirler.

Örnek 0-6: Sıra No, Adınız, Soyadınız şeklinde başlıkları olan ve içeriği dolu olan bir tablo hazırlayıp, ekrana yazdırınız.

```
    Console.WriteLine("------------------------------");
    Console.WriteLine("Sıra No | Adınız    |   Soyad");
    Console.WriteLine("------------------------------");
    Console.WriteLine(String.Format("{0,7} | {1,-8} | {2,10}", 1,
"Remzi", "ERTÜRK"));
    Console.WriteLine(String.Format("{0,7} | {1,-8} | {2,10}", 2,
"Tevfik", "ULUÇ"));
    Console.WriteLine(String.Format("{0,7} | {1,-8} | {2,10}", 3,
"Yücel", "CAN"));
    Console.WriteLine("------------------------------");
```

Yukarıdaki kodları çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı ile karşılaşırız.



Resim 0.5: String.Format( ) Metodu ile Metin Biçimleme

- Örneğin **int türündeki sayısal ifadelerin** belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

```
    String.Format("{0:00000}", 15);          // "00015"
```

ifadesiyle 15 sayısı ekrana başına 3 adet 0 eklenerek toplamda 5 karakter olarak yazılır.

```
String.Format("{0:00000}", -15);          // "-00015"
```

ifadesiyle -15 sayısı ekrana başına 3 adet 0 eklenerek toplamda 5 karakter olarak yazılır.

```
String.Format("{0,5}", 15);               // "   15"
```

ifadesiyle 15 sayısı ekrana başına 3 adet boşluk eklenerek toplamda 5 karakterlik bir alana sağa hizalı olarak yazılır.

```
String.Format("{0,-5}", 15);              // "15   "
```

ifadesiyle 15 sayısı ekrana başına 3 adet boşluk eklenerek toplamda 5 karakterlik bir alana sola hizalı olarak yazılır.

```
String.Format("{0,5:000}", 15);           // "  015"
```

ifadesiyle 15 sayısı ekrana başına bir adet 0 ve iki adet boşluk eklenerek toplamda 5 karakterlik bir alana sağa hizalı olarak yazılır.

```
String.Format("{0,-5:000}", 15);          // "015  "
```

ifadesiyle 15 sayısı ekrana başına bir adet 0 ve iki adet boşluk eklenerek toplamda 5 karakterlik bir alana sola hizalı olarak yazılır.

String.Format( ) metodunun sıfır ve negatif sayılar için özel formları vardır. Sayılar biçimlenirken kullanılan noktalı virgül (;) ile formatı 3 bölüme ayırırız. Buradaki ilk bölüm sayının değerini, ikinci bölüm negatif sayıların biçimini, üçüncü bölüm ise sıfırın ekrana nasıl yazılacağının biçimini belirlememize yardımcı olur.

```
String.Format("{0:#;eksi #}", 15); // "15"
String.Format("{0:#;eksi #}", -15); // "eksi 15"
String.Format("{0:#;eksi #;Sıfır}", 0); // "Sıfır"
```

Sayıları isteğe bağlı biçimlendirmek istersek (*örneğin bir telefon numarasını alan kodu ve telefon numarası ayrı ayrı yazılsın istiyorsak*), biçimlendirme işleminde diyez (#) işareti ile formatımızı belirleriz;

```
String.Format("{0:### ## ##}", 1234567); // 123 45 67
String.Format("{0:(#) ###-##-##}", 12345678); //(1) 234-56-78
```

- Örneğin **double türündeki sayısal ifadelerin** belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

Ondalıklı sayılarda virgülden (programlamada nokta) sonra kaç basamak görünsün istiyorsak köşeli parantezler ({ }) içerisindeki biçimleme kısmında noktadan sonra o kadar sıfır (0) koymamız gerekir;

```
String.Format("{0:0.00}", 123.4567);      // "123.46"
String.Format("{0:0.00}", 123.4);         // "123.40"
String.Format("{0:0.00}", 123);           // "123.00"
```

Eğer ondalıklı sayının en fazla kaç basamağının ekranda çıkmasını istiyorsak, bu sefer sıfır yerine o kadar sayıda diyez (#) işareti kullanmamız gerekir;

```
String.Format("{0:0.##}", 123.4567);      // "123.46"
String.Format("{0:0.##}", 123.4);         // "123.4"
String.Format("{0:0.##}", 123);           // "123"
```

Ondalıklı sayılarda virgülden önce kaç basamak görüntülemek istiyorsak biçimlendirme yaparken, noktadan önce kaç basamak istiyorsak o kadar sıfır (0) kullanmamız gerekir;

```
String.Format("{0:000.0}", 123.4567);       // "123.5"
String.Format("{0:000.0}", 23.4567);        // "023.5"
String.Format("{0:00.00}", 3.4567);         // "03.46"
String.Format("{0:00.00}", -3.4567);        // "-03.46"
```

Eğer sayıların görüntülenmesinde bin ayracı kullanılmak isteniyorsa;

```
String.Format("{0:0,0.0}", 12345.678);      // "12,345.7"
String.Format("{0:0,0.00}", 12345.678);     // "12,345.68"
String.Format("{0:0,0}", 12345.678);        // "12,346"
```

0 ile 1 arasındaki ondalıklı sayıların gösterimi iki şekilde olur. Birincisinde sayının tam kısmı 0 ve noktadan sonra ondalıklı kısmı gelir (*Örn: 0.123 şeklinde*), bir diğer gösterim şeklinde ise sayının tam kısmı yazılmaz sadece nokta ve sonrasındaki ondalıklı kısım yazılır (*Örn: .123 şeklinde*).

İşte bu durumlardaki sayıların gösterimi ise şu şekilde gerçekleştirilir;

```
String.Format("{0:0.0}", 0.0);            // "0.0"
String.Format("{0:0.#}", 0.0);            // "0"
String.Format("{0:#.0}", 0.0);            // ".0"
String.Format("{0:#.#}", 0.0);            // ""
```

Bütün bu formların dışında sayılarımızı aşağıdaki gibi istediğimiz metinler ile birlikte yazmamız da mümkündür;

```
String.Format("{0:sonuç 0.0}", 12.3);   // "sonuç 12.3"
String.Format("{0:x0x.yy0yy}", 12.3);   // "x12x.yy3yy"
```

- Örneğin **tarih/saat türündeki** ifadelerin belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

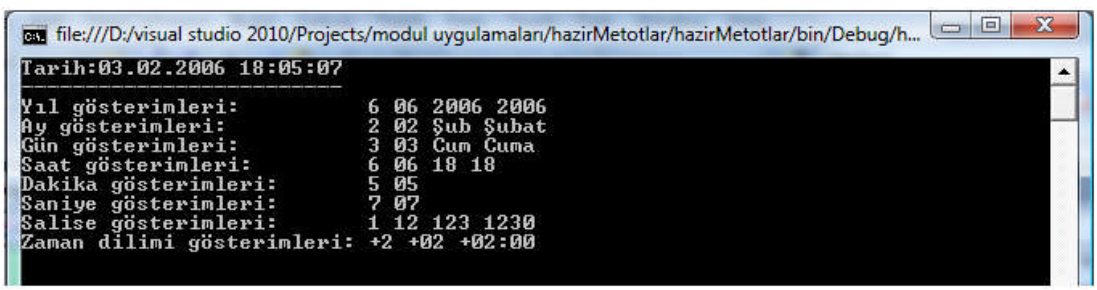
Tarih/Zaman ifadelerini belirtmek için önceden belirlenmiş bazı anahtar harfler vardır. Bunlar;

y: Yıl,**M**: Ay,	**d**: Gün,	
h: 12'lik sistemde saat,	**H**: 24'lük sistemde saat	**m**: Dakika
s: Saniye	**f**: Salise	**z**: Zaman dilimi

Örnek 0-7: 14.02.2006 18:05:07.123 zamanına ait değerlerin gösterimleri şu şekilde sağlanır;

```
     DateTime dt = new DateTime(2006, 2, 3, 18, 5, 7, 123);
     Console.WriteLine("Tarih:"+dt);
     Console.WriteLine("-------------------------");
     Console.WriteLine("Yıl gösterimleri:
"+String.Format("{0:y yy yyy yyyy}", dt));
     Console.WriteLine("Ay gösterimleri:           " +
String.Format("{0:M MM MMM MMMM}", dt));
     Console.WriteLine("Gün gösterimleri:          " +
String.Format("{0:d dd ddd dddd}", dt));
     Console.WriteLine("Saat gösterimleri:         " +
String.Format("{0:h hh H HH}", dt));
     Console.WriteLine("Dakika gösterimleri:       " +
String.Format("{0:m mm}", dt));
     Console.WriteLine("Saniye gösterimleri:       " +
String.Format("{0:s ss}", dt));
     Console.WriteLine("Salise gösterimleri:       " +
String.Format("{0:f ff fff ffff}", dt));
     Console.WriteLine("Zaman dilimi gösterimleri: " +
String.Format("{0:z zz zzz}", dt));
```

Yukarıdaki kodlar çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-6. Tarih/Zaman Gösterim Biçimleri -1

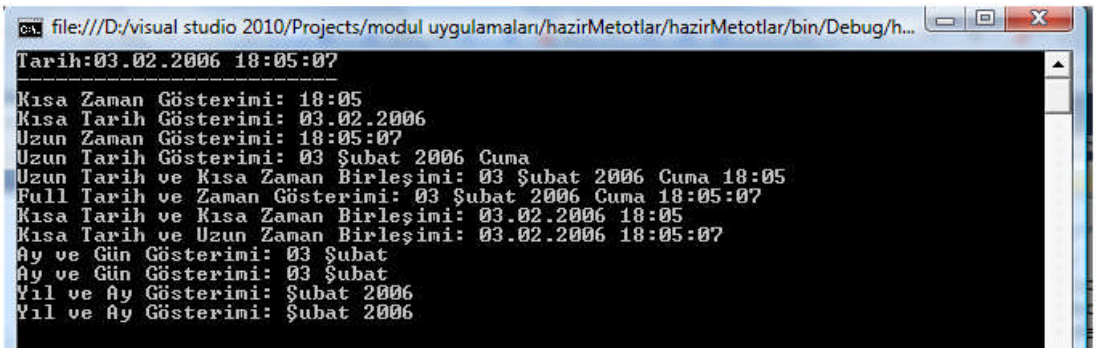
Tarih/zaman gösterimlerinin biçimlendirilmesinde kullanılan bir diğer yol da; daha önceden tanımlanmış belirteçler ile değerlerin ekrana yazdırılmasıdır. Bu değerler aşağıdaki tabloda verilmiştir.

Belirteç	Tarih/Zaman Özelliği	Gösterim Örneği
t	Kısa zaman gösterimi	h:mm
d	Kısa tarih gösterimi	M/d/yyy
T	Uzun zaman gösterimi	h:mm:ss
D	Uzun tarih gösterimi	dddd,MMMM dd, yyyy
f	D ve t' nin birleşimi	dddd, MMMM dd, yyyy h:mm
F	D ve T'nin birleşimi	dddd, MMMM dd, yyyy h:mm:ss
g	d ve t'nin birleşimi	M/d/yyyy h:mm
G	d ve T'nin birleşimi	M/d/yyyy h:mm:ss
m,M	Ay ve gün gösterimi	MMMM dd
y,Y	Yıl ve ay gösterimi	MMMM, yyyy

Örnek 0-8: 14.02.2006 18:05:07.123 zamanını yukarıda verilen biçimlere göre gösterimini tek tek ekrana yazdıran programın kodunu yazınız.

```
     DateTime dt = new DateTime(2006, 2, 3, 18, 5, 7, 123);
     Console.WriteLine("Tarih:" + dt);
     Console.WriteLine("------------------------");
     Console.WriteLine("Kısa Zaman Gösterimi: " +
String.Format("{0:t}", dt));
     Console.WriteLine("Kısa Tarih Gösterimi: " +
String.Format("{0:d}", dt));
     Console.WriteLine("Uzun Zaman Gösterimi: " +
String.Format("{0:T}", dt));
     Console.WriteLine("Uzun Tarih Gösterimi: " +
String.Format("{0:D}", dt));
     Console.WriteLine("Uzun Tarih ve Kısa Zaman Birleşimi: " +
String.Format("{0:f}", dt));
     Console.WriteLine("Full Tarih ve Zaman Gösterimi: " +
String.Format("{0:F}", dt));
     Console.WriteLine("Kısa Tarih ve Kısa Zaman Birleşimi: " +
String.Format("{0:g}", dt));
     Console.WriteLine("Kısa Tarih ve Uzun Zaman Birleşimi: " +
String.Format("{0:G}", dt));
     Console.WriteLine("Ay ve Gün Gösterimi: " +
String.Format("{0:m}", dt));
     Console.WriteLine("Ay ve Gün Gösterimi: " +
String.Format("{0:M}", dt));
     Console.WriteLine("Yıl ve Ay Gösterimi: " +
String.Format("{0:y}", dt));
     Console.WriteLine("Yıl ve Ay Gösterimi: " +
String.Format("{0:Y}", dt));
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-7. Tarih/Zaman Gösterim Biçimleri -2

2.1.5. IsNullOrEmpty( )

Parametre olarak verilen string türündeki değişkenin içeriğinin boş mu olduğunu kontrol eden metottur. Eğer değişkenin içeriği boşsa geriye bool türünde **true** değeri döndürür. Eğer değişkene herhangi bir değer ataması yapılmışsa geriye **false** değerini döndürür.

- **Kullanımı:**

```
string metin="";
bool sonuc=String.IsNullOrEmpty(metin);
```

Örnek 0-9: Kullanıcı adı ve şifre girişi yapılan bir giriş ekranında kullanıcı adı veya şifre boş girilirse uyarı veren, boş girilmemişse girilen değerleri ekrana yazdıran programın kodlarını yazınız.

```
Console.Write("Kullanıcı adınızı giriniz: ");
string kullaniciAdi = Console.ReadLine();
Console.Write("Şifrenizi giriniz: ");
string sifre = Console.ReadLine();
bool kaBosmu=String.IsNullOrEmpty(kullaniciAdi);
bool sfrBosmu=String.IsNullOrEmpty(sifre);
if(kaBosmu==true)
    Console.WriteLine("Kullanıcı adını boş geçemezsiniz.");
else
    Console.WriteLine("Girmiş olduğunuz kullanıcı adı:" +
kullaniciAdi);
if (sfrBosmu == true)
    Console.WriteLine("Şifreyi boş geçemezsiniz.");
else
    Console.WriteLine("Girmiş olduğunuz şifre:" + sifre);
```

Bu kısımdan sonra anlatılacak olan metinsel metotlar direkt olarak String sınıfı üzerinden değil, string değişken üzerinden çağrılacak olan metotlardır.

2.1.6. CompareTo ( )

Çağrıldığı string ifade ile parametre olarak verilen string ifadeyi karşılaştırır ve iki ifade de birbirine eşitse geriye int türünde sıfır (0) değerini döndürür. Aksi takdirde metinleri ilk harflerinden itibaren tek tek karşılaştırır ve farklılığın olduğu ilk harflerin alfabedeki sıralarına göre -1 veya 1 sayı değerlerini döndürür.

Kullanımı ve çalışma prensibi daha önce gördüğümüz String.Compare( ) metoduyla hemen hemen aynıdır.

- **Kullanımı:**

```
int donusDegeri=metin1.CompareTo(metin2);
```

Aşağıdaki tabloda metinlerin karşılaştırma durumları ve geri dönüş değerleri verilmektedir;

Durum	Dönüş Değeri
metin1>metin2	1
metin1=metin2	0
metin1<metin2	-1

Daha önceden de belirttiğimiz gibi metinlerin büyüktür/küçüktür karşılaştırmaları, harflerin alfabetik sırasıyla ilgilidir.

Örnek 0-10: Klavyeden girilen iki metnin karşılaştırılmasını yapan programın kodlarını yazınız.

```
    Console.Write("1. metni giriniz: ");
    string metin1 = Console.ReadLine();
    Console.Write("2. metni giriniz: ");
    string metin2 = Console.ReadLine();
    int donusDegeri = metin1.CompareTo(metin2);

    switch (donusDegeri)
    {
        case -1:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinden
farklıdır. \nGeri Dönüş değeri: {2}",metin1,metin2,donusDegeri);
           break;
        case 0:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinin
aynısıdır. \nGeri Dönüş değeri: {2}", metin1, metin2, donusDegeri);
            break;
        case 1:
            Console.WriteLine("{0} ve {1} kelimeleri birbirinden
farklıdır. \nGeri Dönüş değeri: {2}", metin1, metin2, donusDegeri);
            break;
    }
```

Yukarıdaki programı çalıştırdıktan sonra aşağıdaki tabloda verilen değerleri tek tek deneyerek CompareTo( ) metoduyla geri dönen değerleri, tabloda boş bırakılan alanlara yazınız.

Girilecek Değerler		Geri Dönüş Değeri
metin1	**metin2**	
Bilişim	Bilişim	
bilişim	bilisim	
BİLİŞİM	BILISIM	
bİLİŞİM	Bilişim	
Bilisim	Bilisim	

Yukarıda verilen değerleri istediğiniz şekilde değiştirerek metodun çalışmasını iyice pekiştiriniz.

2.1.7. Contains( )

Birlikte çağrıldığı metinsel ifade içerisinde parametre olarak verilen char türündeki karakteri veya yine parametre olarak verilen string türündeki metinsel ifadeyi arar ve geriye bool türünde bir değer döndürür.

- **Kullanımı:**

Metinsel ifade içerisinde karakter arama;

```
char karakter= '';
bool donusDegeri=metin1.CompareTo(karakter);
```

Metinsel ifade içerisinde string arama;

```
string aranan= "";
bool donusDegeri=metin1.CompareTo(aranan);
```

Örnek 0-11: Klavyeden girilen metin içerisinde rakamsal ifade olup olmadığını kontrol eden bir metot yazıp, kullanıcıyı uyaran programın kodlarını yazınız.

```
 static void Main(string[] args)
{
   Console.Write("Kelime giriniz: ");
   string ifade = Console.ReadLine();
   bool sonuc = rakamVarMi(ifade);
   if (sonuc)
      Console.WriteLine("Girmiş olduğunuz metin rakamsal
ifadeler içeriyor");
   else
```

```
           Console.WriteLine("Girmiş olduğunuz metin rakamsal
ifadeler içermiyor");
       }
     static bool rakamVarMi(string metin)
     {
         if (metin.Contains('0'))
             return true;
         else if (metin.Contains('1'))
             return true;
         else if (metin.Contains('2'))
             return true;
         else if (metin.Contains('3'))
             return true;
         else if (metin.Contains('4'))
             return true;
         else if (metin.Contains('5'))
             return true;
         else if (metin.Contains('6'))
             return true;
         else if (metin.Contains('7'))
             return true;
         else if (metin.Contains('8'))
             return true;
         else if (metin.Contains('9'))
             return true;
         else
             return false;
     }
```

Yukarıdaki uygulamayı rakamsal ifade içeren ve içermeyen çeşitli kelime girişleriyle deneyip pekiştiriniz.

Örnek 0-12: Klavyeden girilen metin içerisinde, yine klavyeden girilen bir metni arayan programın kodlarını yazınız.

```
     Console.Write("Metin giriniz: ");
     string ifade = Console.ReadLine();
     Console.Write("Aranan metni giriniz: ");
     string aranan = Console.ReadLine();
     if (ifade.Contains(aranan))
         Console.WriteLine("({0}) ifadesi içerisinde ({1}) kelimesi
mevcuttur.", ifade, aranan);
     else
         Console.WriteLine("({0}) ifadesi içerisinde ({1}) kelimesi
yoktur.", ifade, aranan);
```

2.1.8. CopyTo( )

Bu metot kaynakBaslangicIndexi (int türünde), hedefDizisi (char dizisi türünde), hedefBaslangicIndexi (int türünde) ve miktar (int türünde) olmak üzere 4 parametre alır.

Birlikte çağrıldığı metinsel ifadenin;
- Parametre olarak verilen int türündeki kaynak başlangıç indeksinden itibaren,
- Parametre olarak verilen char[] dizisinin içerisine,
- Parametre olarak verilen hedef başlangıç indeksinden itibaren,
- Parametre olarak verilen sayıda karakteri kopyalamaya yarayan metottur.

➢ **Kullanımı:**

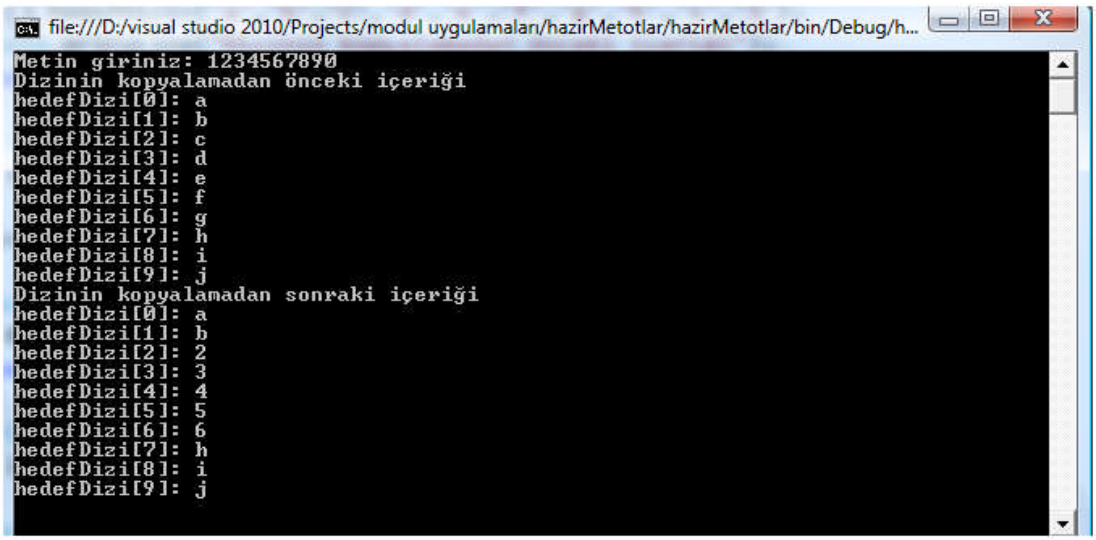
```
string metin1= "";
char[] hedefDizisi={,,,,,} ;
int kayBasInd, hedBasInd, adet;
metin1.CopyTo(kayBasInd,hedefDizisi,hedBasInd,adet);
```

Örnek 0-13: Klavyeden girilen 10 harfli bir metnin içeriğini 2.karakterinden başlayarak 5 karakterini hedefDizi isimli char türündeki 10 elemanlı bir dizinin 3.elemanından başlayarak kopyalayınız. hedefDizi isimli dizinin içeriğini kopyalamadan önce ve sonra ekrana yazdırarak değişimi gözlemleyiniz.

```
Console.Write("Metin giriniz: ");
string metin = Console.ReadLine();
int i = 0;
char[] hedefDizi={'a','b','c','d','e','f','g','h','i','j'};
Console.WriteLine("Dizinin kopyalamadan önceki içeriği");
foreach (char harf in hedefDizi)
{
    Console.WriteLine("hedefDizi[{0}]: {1}", i, harf);
     i++;
 }
 metin.CopyTo(1, hedefDizi, 2, 5);

 Console.WriteLine("Dizinin kopyalamadan sonraki içeriği");
 i = 0;
 foreach (char harf in hedefDizi)
 {
     Console.WriteLine("hedefDizi[{0}]: {1}", i, harf);
     i++;
 }
```

Yukarıdaki kod parçası çalıştırıldığında ve klavyeden "1234567890" verisi girilerek çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-8. CopyTo Metodu ile Bir Dizinin İçeriğini Değiştirme

2.1.9. EndsWith( )

Birlikte çağrıldığı metinsel ifade parametre olarak verilen string türündeki ifade ile bitip bitmediğini kontrol eden metottur. Geriye bool türünde bir değer döndürür. Eğer metin parametre olarak verilen ifade ile bitiyorsa geriye **true** değerini döndürür. Eğer metin parametre olarak verilen ifade ile bitmiyorsa geriye **false** değerini döndürür.

- **Kullanımı:**

```
metin.EndsWith(ifade);
```

Örnek 0-14: Klavyeden girilen metin sesli harf ile bitip bitmediğini kontrol eden ve geriye bool türünde bir değer döndüren metot tanımlayınız.

```
    static void Main(string[] args)
    {
        Console.Write("Metin giriniz: ");
        string ifade= Console.ReadLine();
        if (bitisSesliMi(ifade))
            Console.WriteLine("Klavyeden girilen metin sesli harf
ile bitiyor");
        else
            Console.WriteLine("Klavyeden girilen metin sesli harf
ile bitmiyor");
    }

    static bool bitisSesliMi(string metin)
    {
        if (metin.EndsWith("a"))
            return true;
        else if (metin.EndsWith("e"))
```

```
            return true;
        else if (metin.EndsWith("ı"))
            return true;
        else if (metin.EndsWith("i"))
            return true;
        else if (metin.EndsWith("o"))
            return true;
        else if (metin.EndsWith("ö"))
            return true;
        else if (metin.EndsWith("u"))
            return true;
        else if (metin.EndsWith("ü"))
            return true;
        else
            return false;
    }
```

Yukarıdaki uygulamayı çeşitli kelime girişleri ile deneyerek pekiştiriniz.

2.1.10. IndexOf( )

Bu metodun birden fazla kullanım şekli vardır.

2.1.10.1. IndexOf(char)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen karakteri arar ve geriye bu karakterin metin içerisinde **ilk** bulunduğu karakter sırasını döndürür. Metnin ilk karakterinin indeks numarasının sıfır (0) olduğunu unutmayınız.

Eğer aranan karakter kelime içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan karakterin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
    int indeks=metin.IndexOf(char);
```

Örnek 0-15: IndexOf(char) metodunun kullanımı;

```
    string metin = "Bilişim teknolojileri";
    Console.WriteLine(metin.IndexOf('T'));     //    -1
    Console.WriteLine(metin.IndexOf('e'));     //    9
    Console.WriteLine(metin.IndexOf('i'));     //    1
    Console.WriteLine(metin.IndexOf('z'));     //    -1
```

2.1.10.2. IndexOf(string)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen string ifadeyi arar ve geriye bu ifadenin, metin içerisinde ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.IndexOf(string);
```

Örnek 0-16: IndexOf(string) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.IndexOf("bilişim"));     //   -1
Console.WriteLine(metin.IndexOf("Bilişim"));     //   0
Console.WriteLine(metin.IndexOf("loji"));        //   13
Console.WriteLine(metin.IndexOf("Bilisim"));     //   -1
```

2.1.10.3. IndexOf(char deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen karakteri, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.IndexOf(char deger,int baslangic);
```

Örnek 0-17: IndexOf(char deger,int baslangic) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.IndexOf('T',3));         //   -1
Console.WriteLine(metin.IndexOf('e',10));        //   18
Console.WriteLine(metin.IndexOf('i',4));         //   5
Console.WriteLine(metin.IndexOf('o',18));        //   -1
```

2.10.1.4. IndexOf(string deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen metinsel ifadeyi, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.IndexOf(string deger,int baslangic);
```

Örnek 0-18: IndexOf(string deger,int baslangic) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.IndexOf("bilişim",0));    //   -1
Console.WriteLine(metin.IndexOf("Bilişim",1));    //   -1
Console.WriteLine(metin.IndexOf("loji",3));       //   13
Console.WriteLine(metin.IndexOf("il",2));         //   16
```

2.1.11. Insert(int baslangic,string value)

Parametre olarak verilen **int** türündeki başlangıç indeksinden başlayarak, yine parametre olarak verilen metinsel ifadeyi, çağrıldığı metnin içerisine eklemeye yarayan metottur. Geriye **string** türünde metinsel bir ifade döndürür.

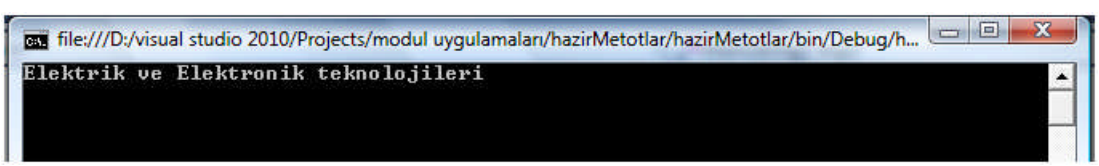
- **Kullanımı:**

```
string yeniMetin=metin.Insert(int baslangic, string eklenecek);
```

Örnek 0-19: Insert(int baslangic, string deger) metodunun kullanımı;

```
string metin = "Elektrik teknolojileri";
string yeniMetin=metin.Insert(9, "ve Elektronik ");
Console.Write(yeniMetin);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi ekran çıktısı ile karşılaşırız;



Resim 0-9. Insert( ) Metodu ile Metin İçerisine Metin Ekleme

2.1.12. LastIndexOf( )

Bu metodun da IndexOf metotdu gibi birden fazla kullanım şekli vardır.

2.1.12.1. LastIndexOf (char)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen karakteri arar ve geriye bu karakterin metin içerisinde **son** bulunduğu karakter sırasını döndürür. Metnin ilk karakterinin indeks numarasının sıfır (0) olduğunu unutmayınız.

Eğer aranan karakter kelime içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan karakterin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.LastIndexOf(char);
```

Örnek 0-20: LastIndexOf(char) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.LastIndexOf('T'));      //    -1
Console.WriteLine(metin.LastIndexOf('e'));      //    18
Console.WriteLine(metin.LastIndexOf('i'));      //    20
Console.WriteLine(metin.LastIndexOf('z'));      //    -1
```

2.1.12.2. LastIndexOf (string)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen string ifadeyi arar ve geriye bu ifadenin, metin içerisinde son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.LastIndexOf(string);
```

Örnek 0-21: LastIndexOf(string) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.LastIndexOf("bilişim"));      //    -1
Console.WriteLine(metin.LastIndexOf("Bilişim"));      //    0
Console.WriteLine(metin.LastIndexOf("il"));           //    16
Console.WriteLine(metin.LastIndexOf("Bilisim"));      //    -1
```

2.1.12.3. LastIndexOf (char deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen karakteri, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.LastIndexOf(char deger,int baslangic);
```

Örnek 0-22: LastIndexOf(char deger,int baslangic) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.LastIndexOf('T',3));          //    -1
Console.WriteLine(metin.LastIndexOf('e',10));         //    9
Console.WriteLine(metin.LastIndexOf('i',4));          //    13
Console.WriteLine(metin.LastIndexOf('o',18));         //    14
```

2.1.12.4. LastIndexOf(string deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen metinsel ifadeyi, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.LastIndexOf(string deger,int baslangic);
```

Örnek 0-23: LastIndexOf(string deger,int baslangic) metodunun kullanımı;

```
string metin = "Bilişim teknolojileri";
Console.WriteLine(metin.LastIndexOf("bilişim",0));      //   -1
Console.WriteLine(metin.LastIndexOf("Bilişim",1));      //   -1
Console.WriteLine(metin.LastIndexOf("im",15));          //   5
Console.WriteLine(metin.LastIndexOf("il",2));           //   1
```

2.1.13. PadLeft ( )

PadLeft metodunun 2 farklı kullanımı vardır.

2.1.13.1. PadLeft(int deger)

PadLeft( ) metodunun bu kullanımında, birlikte çağrıldığı metne parametre olarak verilen değer kadar karakterlik bir alan ayırır ve metni sağa hizalanmış yeni bir metinsel ifade geriye döndürür.

Hizalamanın görülebilmesi için parametre olarak verilen değerin metnin karakter uzunluğundan fazla olduğundan emin olunuz.

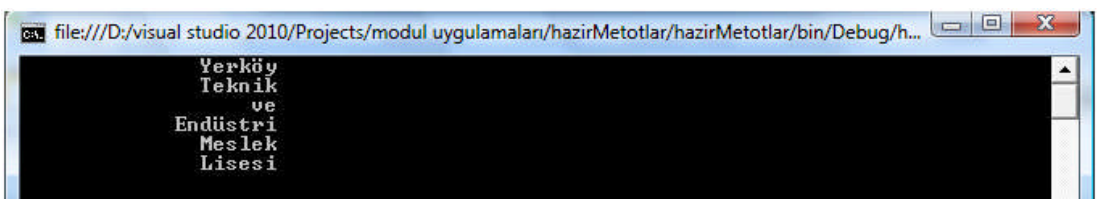
- **Kullanımı:**

```
string yeniMetin=metin.PadLeft(int deger);
```

Örnek 0-24: PadLeft(int deger) metodunun kullanımı;

```
string metin = "Yerköy";
string metin2 = "Teknik";
string metin3 = "ve";
string metin4 = "Endüstri";
string metin5 = "Meslek";
string metin6 = "Lisesi";
Console.WriteLine(metin.PadLeft(20));
Console.WriteLine(metin2.PadLeft(20));
Console.WriteLine(metin3.PadLeft(20));
Console.WriteLine(metin4.PadLeft(20));
Console.WriteLine(metin5.PadLeft(20));
Console.WriteLine(metin6.PadLeft(20));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-10. PadLeft(int deger) Metodunun Kullanımı

2.1.13.2. PadLeft(int deger,char karakter)

PadLeft( ) metodunun bu kullanımında da bir önceki kullanımda olduğu gibi birlikte çağrıldığı metne parametre olarak verilen değer kadar karakterlik bir alan ayırır ve metni sağa hizalanmış yeni bir metinsel ifade geriye döndürür. Ancak bir önceki kullanımda metin hizalanırken metnin sol tarafı boşluk karakteriyle dolduruluyordu. Bu kullanımda ise parametre olarak verilen karakter bu doldurma işlemi için kullanılır.

Hizalamanın görülebilmesi için parametre olarak verilen değerin metnin karakter uzunluğundan fazla olduğundan emin olunuz.

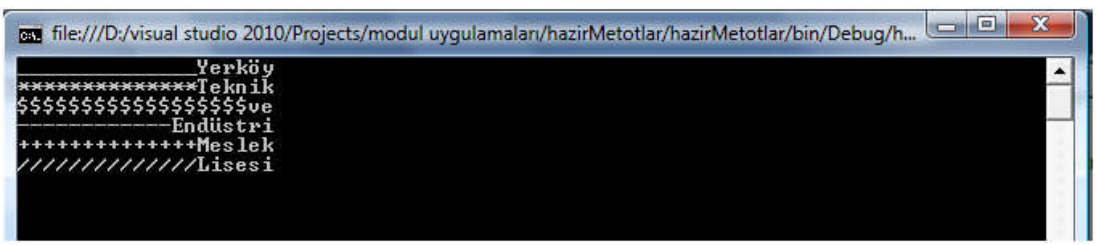
- **Kullanımı:**

```
string yeniMetin=metin.PadLeft(int deger,char karakter);
```

Örnek 0-25: PadLeft(int deger,char karakter) metodunun kullanımı;

```
string metin = "Yerköy";
string metin2 = "Teknik";
string metin3 = "ve";
string metin4 = "Endüstri";
string metin5 = "Meslek";
string metin6 = "Lisesi";
Console.WriteLine(metin.PadLeft(20,  '_'));
Console.WriteLine(metin2.PadLeft(20, '*'));
Console.WriteLine(metin3.PadLeft(20, '$'));
Console.WriteLine(metin4.PadLeft(20, '-'));
Console.WriteLine(metin5.PadLeft(20, '+'));
Console.WriteLine(metin6.PadLeft(20, '/'));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-11. PadLeft(int deger,char karakter) Metodunun Kullanımı

Yukarıdaki resimden de görüleceği üzere hizalama işleminde boşlukların doldurulması çeşitlik karakterler ile gerçekleştirilmiştir.

2.1.14. PadRight ( )

PadRight metodunun 2 farklı kullanımı vardır.

2.1.14.1. PadRight (int deger)

PadRight( ) metodunun bu kullanımında, birlikte çağrıldığı metne parametre olarak verilen değer kadar karakterlik bir alan ayırır ve metni sola hizalanmış yeni bir metinsel ifade geriye döndürür.

PadLeft( ) metodunun kullanımıyla tamamen aynı olan bu metot ile metinsel ifade bu kez sola hizalı olarak yeni bir metinsel ifade geriye döndürür.

Hizalamanın görülebilmesi için parametre olarak verilen değerin metnin karakter uzunluğundan fazla olduğundan emin olunuz.

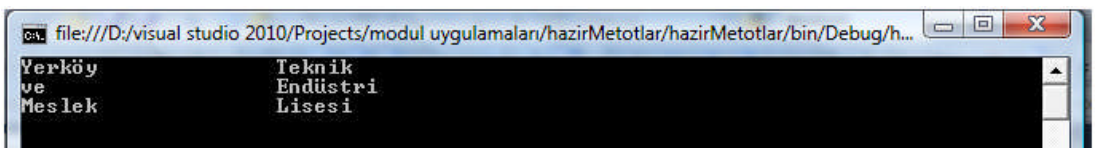
- **Kullanımı:**

```
string yeniMetin=metin.PadRight(int deger);
```

Örnek 0-26: PadRight(int deger) metodunun kullanımı;

```
string metin = "Yerköy";
string metin2 = "Teknik";
string metin3 = "ve";
string metin4 = "Endüstri";
string metin5 = "Meslek";
string metin6 = "Lisesi";
Console.WriteLine(metin.PadRight(20) + metin2.PadRight(20));
Console.WriteLine(metin3.PadRight(20) + metin4.PadRight(20));
Console.WriteLine(metin5.PadRight(20) + metin6.PadRight(20));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-12. PadRight(int deger) Metodunun Kullanımı

2.1.14.2. PadRight(int deger,char karakter)

PadRight( ) metodunun bu kullanımında da bir önceki kullanımda olduğu gibi birlikte çağrıldığı metne parametre olarak verilen değer kadar karakterlik bir alan ayırır ve metni sola hizalanmış yeni bir metinsel ifade geriye döndürür.

Ancak bir önceki kullanımda metin hizalanırken metnin sağ tarafı boşluk karakteriyle dolduruluyordu. Bu kullanımda ise parametre olarak verilen karakter bu doldurma işlemi için kullanılır.

Hizalamanın görülebilmesi için parametre olarak verilen değerin metnin karakter uzunluğundan fazla olduğundan emin olunuz.

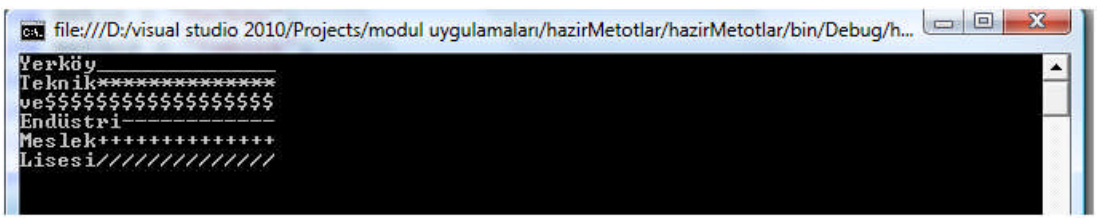
- **Kullanımı:**

```
string yeniMetin=metin.PadRight(int deger,char karakter);
```

Örnek 0-27: PadRight(int deger,char karakter) metodunun kullanımı;

```
string metin = "Yerköy";
string metin2 = "Teknik";
string metin3 = "ve";
string metin4 = "Endüstri";
string metin5 = "Meslek";
string metin6 = "Lisesi";
Console.WriteLine(metin.PadRight(20,  '_'));
Console.WriteLine(metin2.PadRight(20, '*'));
Console.WriteLine(metin3.PadRight(20, '$'));
Console.WriteLine(metin4.PadRight(20, '-'));
Console.WriteLine(metin5.PadRight(20, '+'));
Console.WriteLine(metin6.PadRight(20, '/'));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-13. PadRight(int deger,char karakter) Metodunun Kullanımı

Yukarıdaki resimden de görüleceği üzere hizalama işleminde boşlukların doldurulması çeşitlik karakterler ile gerçekleştirilmiştir.

2.1.15. Remove ( )

Remove metodunun 2 farklı kullanımı vardır.

2.1.15.1. Remove (int deger)

Birlikte çağrıldığı metnin, parametre olarak verilen değerinin bulunduğu indeks değerinden itibaren sonuna kadar olan kısmını siler. Silinme işleminden arta kalan metni geriye döndürür.

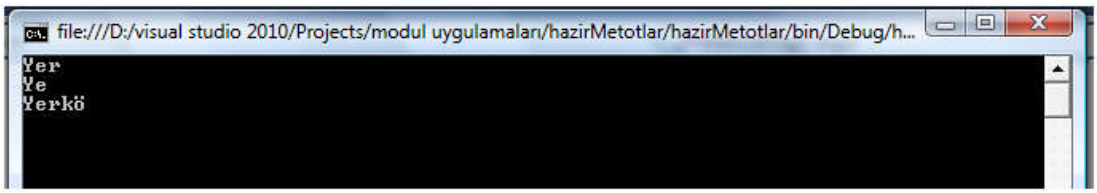
- **Kullanımı:**

```
string yeniMetin=metin.Remove(int deger);
```

Örnek 0-28: Remove(int deger) metodunun kullanımı;

```
string metin = "Yerköy";
Console.WriteLine(metin.Remove(3));
Console.WriteLine(metin.Remove(2));
Console.WriteLine(metin.Remove(5));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-14. Remove(int deger) Metodu ile Metin Kırpma İşlemi

2.1.15.2. Remove (int deger,int adet)

Birlikte çağrıldığı metnin, parametre olarak verilen değerinin bulunduğu indeks değerinden itibaren yine parametre olarak verilen adet kadar olan kısmını siler. Silinme işleminden arta kalan metni geriye döndürür.

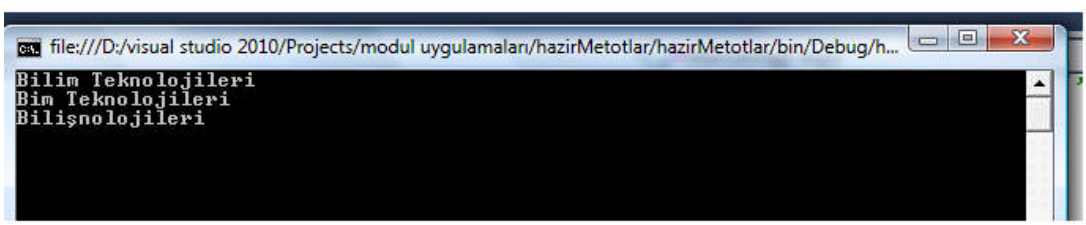
- **Kullanımı:**

```
string yeniMetin=metin.PadRight(int deger,int adet);
```

Örnek 0-29: Remove(int deger,int adet) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine(metin.Remove(3, 2));
Console.WriteLine(metin.Remove(2, 4));
Console.WriteLine(metin.Remove(5, 6));
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-15. Remove(int deger,int adet) Metodu ile Metin Kırpma İşlemi

2.1.16. Replace ( )

Replace metodunun 2 farklı kullanımı vardır.

2.1.16.1. Replace (char eski, char yeni)

Birlikte çağrıldığı metin içerisinde, ilk parametredeki karakterleri, ikinci parametredeki karakter değerleriyle değiştiren metottur. Geriye değiştirme işleminin gerçekleştirildiği string türünde bir ifade döndürür.

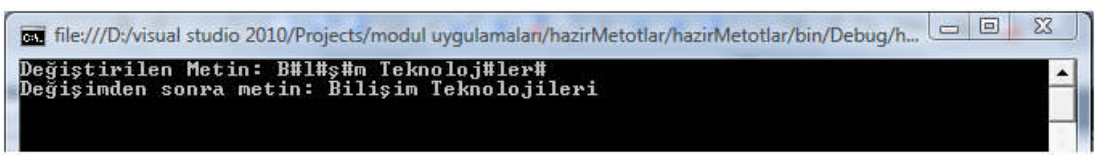
- **Kullanımı:**

```
string yeniMetin=metin.Replace(char eski,char yeni);
```

Örnek 0-30: Replace(int deger,int adet) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine("Değiştirilen: "+metin.Replace('i', '#'));
Console.WriteLine("Değişimden sonra: "+metin);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-16. Replace(char eski, char yeni) Metodu ile Karakter Değiştirme

2.1.16.2. Replace (string eski, string yeni)

Replace metodunun bir önceki kullanımından farkı parametre olarak bu kez char türünde karakterler yeni string türünde metinsel ifade almasıdır.

- **Kullanımı:**

```
string yeniMetin=metin.Replace(string eski, string yeni);
```

Örnek 0-31: Replace(int deger,int adet) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine("Değiştirilen: "+metin.Replace("Bilişim",
"Metal"));
Console.WriteLine("Değişimden sonra: "+metin);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;

Resim 0-17. Replace(string eski, string yeni) Metodu ile Metin Değiştirme

2.1.17. Split ( )

Split( ) metodu, çağrıldığı metni istenilen karakterden itibaren parçalara bölmek için kullanılan bir metottur. Eğer istenilen karakter mevcut metin ifadesi içerisinde yer alıyorsa, Split( ) metodu metni karakterlerden öncesi ve sonrası şeklinde parçalara ayırır ve bu parçaları string türünde bir dizi içerisinde saklar. Geriye de bu string[] türündeki diziyi döndürür.

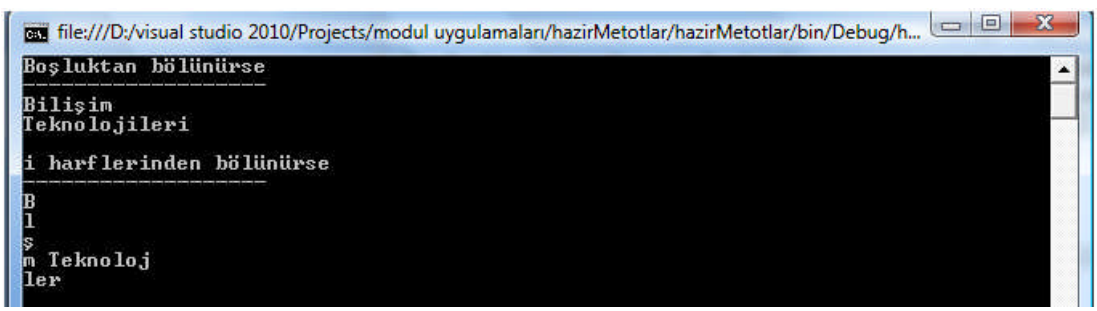
- **Kullanımı:**

```
string[] dizi=metin.Split(char karakter);
```

Örnek 0-32: Split(char karakter) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine("Boşluktan bölünürse");
Console.WriteLine("--------------------");
foreach(string harf in metin.Split(' '))
    Console.WriteLine(harf);
Console.WriteLine(" ");
Console.WriteLine("i harflerinden bölünürse");
Console.WriteLine("--------------------");
foreach (string harf in metin.Split('i'))
    Console.WriteLine(harf);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-18. Split(char karakter) Metodu ile Metin Parçalama

2.1.18. StartsWith ( )

Birlikte çağrıldığı metinsel ifade parametre olarak verilen string türündeki ifade ile başlayıp başlamadığını kontrol eden metottur. Geriye bool türünde bir değer döndürür. Eğer metin parametre olarak verilen ifade ile başlıyorsa geriye **true** değerini döndürür. Eğer metin parametre olarak verilen ifade ile başlamıyorsa geriye **false** değerini döndürür.

- **Kullanımı**:

```
metin.StartsWith(ifade);
```

Örnek 0-33: Klavyeden girilen kullanıcı adının rakamla başlayıp başlamadığını kontrol eden, eğer kullanıcı adı rakam ile başlıyorsa uyarı mesajı veren bir metot tanımlayınız.

```
static void Main(string[] args)
{
    Console.Write("Kullanıcı adı belirleyiniz: ");
    string kAdi= Console.ReadLine();
    if (KullaniciAdiKontrol(kAdi))
        Console.WriteLine("Kullanıcı adı tanımınız başarılı");
    else
        Console.WriteLine("Kullanıcı adı sayı ile başlayamaz");
}

static bool KullaniciAdiKontrol(string kAdi)
{
    if (kAdi.StartsWith("1"))
        return false;
    else if (kAdi.StartsWith("2"))
        return false;
    else if (kAdi.StartsWith("3"))
        return false;
    else if (kAdi.StartsWith("4"))
        return false;
    else if (kAdi.StartsWith("5"))
        return false;
    else if (kAdi.StartsWith("6"))
        return false;
    else if (kAdi.StartsWith("7"))
        return false;
    else if (kAdi.StartsWith("8"))
        return false;
    else if (kAdi.StartsWith("9"))
        return false;
    else if (kAdi.StartsWith("0"))
        return false;
    else
        return true;
}
```

Yukarıdaki uygulamayı çeşitli kelime girişleri ile deneyerek pekiştiriniz.

2.1.19. Substring ( )

Substring( ) metodunun iki kullanımı vardır.

2.1.19.1. Substring (int indeks)

Birlikte çağrıldığı metni parametre olarak verilen indeks değerinden itibaren keser ve arta kalan metni geriye string türünde döndüren metottur.

- **Kullanımı:**

```
string yeniMetin=metin.Substring(int indeks);
```

Örnek 0-34: Substring(int indeks) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine(metin.Substring(3));     //işim Teknolojileri
Console.WriteLine(metin.Substring(8));     //Teknolojileri
Console.WriteLine(metin.Substring(14));    //ojileri
```

2.1.19.2. Substring (int indeks, int uzunluk)

Substring metodunun bu kullanımında ise ilk parametre indeks değerini, ikinci parametre ise kaç karakter uzunluğunda bir metnin kesileceğini belirtir.

- **Kullanımı:**

```
string yeniMetin=metin.Substring(int indeks,int uzunluk);
```

Örnek 0-35: Substring (int indeks, int uzunluk) metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine(metin.Substring(3,4));   //işim
Console.WriteLine(metin.Substring(8,3));   //Tek
Console.WriteLine(metin.Substring(14,1));  //o
```

2.1.20. ToLower ( )

Birlikte çağrıldığı metninin tüm karakterlerini küçük harfe dönüştürerek yeni bir metin geriye döndürür.

- **Kullanımı:**

```
string yeniMetin=metin.ToLower();
```

Örnek 0-36: ToLower() metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine(metin.ToLower());  //bilişim teknolojileri
string metin2= "yERKÖy tEKNİK Ve ENdüsTRİ";
Console.WriteLine(metin2.ToLower()); //yerköy teknik ve endüstri
```

2.1.21. ToUpper ( )

ToLower( ) metodunun tam tersi şeklinde çalışır ve birlikte çağrıldığı metninin tüm karakterlerini büyük harfe dönüştürerek yeni bir metin geriye döndürür.

- **Kullanımı:**

```
string yeniMetin=metin.ToUpper();
```

Örnek 0-37: ToUpper() metodunun kullanımı;

```
string metin = "Bilişim Teknolojileri";
Console.WriteLine(metin.ToUpper());  //BİLİŞİM TEKNOLOJİLERİ
string metin2= "yERKÖy tEKNİK Ve ENdüsTRİ";
Console.WriteLine(metin2.ToUpper()); //YERKÖY TEKNİK VE ENDÜSTRİ
```

2.2. Matematiksel (Math) Metotları

Programlama dili içerisindeki Math sınıfı altında bulunan ve matematiksel bazı işlem ve fonksiyonları daha kolay yapabilmek için bir takım hazır metotlar vardır.

Matematiksel metotlardan sık kullanılanlar şunlardır;

- Abs	- BigMul	- Ceiling	- DivRem
- Max	- Min	- Pow	- Round
- Sign	- Sqrt	- Cos	- Sin
- Tan	- Acos	- Asin	- Atan

2.2.1. Abs( )

Abs( ) metodu parametre olarak verilen sayının mutlak değerini veren metottur. Parametre olarak farklı sayı türlerinde değerler alabilir ve aldığı değerin türünde bir değer geri döndürür.

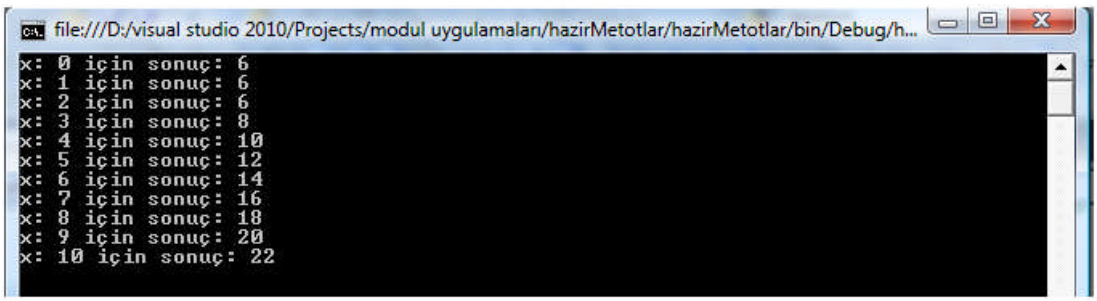
- **Kullanımı:**

```
int mutlakDeger=Math.Abs(int sayi);
decimal mutlakDeger=Math.Abs(decimal sayi);
double mutlakDeger=Math.Abs(double sayi);
float mutlakDeger=Math.Abs(float sayi);
long mutlakDeger=Math.Abs(long sayi);
short mutlakDeger=Math.Abs(short sayi);
sbyte mutlakDeger=Math.Abs(sbyte sayi);
```

Örnek 0-38: $|x - 2| + 2 + |2 + x|$ ifadesinin sonucunu x'in 0'dan 10'a kadar olan değerleri için tek tek ekrana yazdıran programın kodunu yazınız.

```
int sonuc=0;
for (int x = 0; x <= 10; x++)
{
    sonuc = Math.Abs(x - 2) + 2 + Math.Abs(2 + x);
    Console.WriteLine("x: {0} için sonuç: {1}",x,sonuc);
}
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-19. Math.Abs( ) Metodunun Kullanımı

2.2.2. BigMul( )

Parametre olarak verilen iki **int** türündeki sayının çarpımını **long** türünde veren metottur.

- **Kullanımı:**

```
long sonuc=Math.BigMul(int a, int b);
```

Örnek 0-39: Math.BigMul( ) metodunun kullanımı

```
long sonuc=Math.BigMul(2,4);          // 8
sonuc=Math.BigMul(43,2);              // 86
sonuc=Math.BigMul(9,80);              // 720
sonuc=Math.BigMul(100,10);            // 1000
```

2.2.3. Ceiling( )

Parametre olarak verilen **double** türündeki ondalıklı sayıdan büyük, en küçük tam sayının değerini veren metottur.

- **Kullanımı:**

```
decimal sonuc=Math.Ceiling(decimal sayi1);
double sonuc2=Math.Ceiling(double sayi2);
```

Örnek 0-40: Math.Ceiling( ) metodunun kullanımı

```
double sayi= 2.00;
double sonuc=Math.Ceiling(sayi);          // 2
sayi= 2.01;
sonuc=Math.Ceiling(sayi);                 // 3
sayi= 2.50;
sonuc=Math.Ceiling(sayi);                 // 3
sayi= 2.99;
sonuc=Math.Ceiling(sayi);                 // 3
```

2.2.4. DivRem( )

Parametre olarak verilen ilk iki sayının bölme işlemini yapar ve geriye bölme işleminin sonucunu döndüren ve 3. parametre olarak verilen değişkene de bölme işleminin kalanını aktaran metottur.

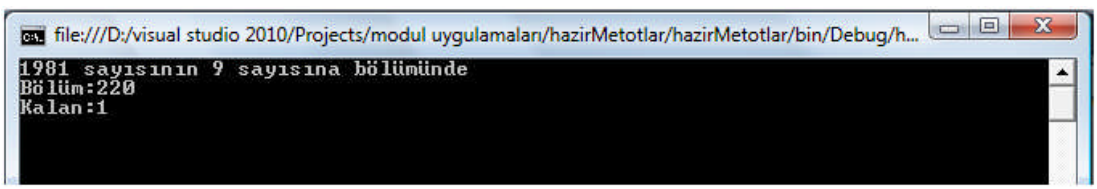
- **Kullanımı:**

```
int bolum=Math.DivRem(int bolunen, int bolen,out int kalan);
long bolum=Math.DivRem(long bolunen,long bolen,out long kalan);
```

Örnek 0-41: Math.DivRem( ) metodu ile bölme işlemi

```
int bolunen = 1981;
int bolen = 9;
int kalan = 0;
int bolum = Math.DivRem(bolunen, bolen, out kalan);
Console.WriteLine("{0} sayısının {1} sayısına bölümünde",
bolunen, bolen);
Console.WriteLine("Bölüm:{0}", bolum);
Console.WriteLine("Kalan:{0}", kalan);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-20. Math.DivRem( ) ile Bölme İşlemi

2.2.5. Max( )

Parametre olarak verilen iki sayıdan büyük olanı geriye döndüren metottur. Bütün sayı türleri tarafından desteklenen bir metot çeşididir.

➢ **Kullanımı:**

```
byte maksimum=Math.Max(byte sayi1,byte sayi2);
decimal maksimum=Math.Max(decimal sayi1,decimal sayi2);
double maksimum=Math.Max(double sayi1,double sayi2);
float maksimum=Math.Max(float sayi1,float sayi2);
int maksimum=Math.Max(int sayi1,int sayi2);
long maksimum=Math.Max(long sayi1,long sayi2);
sbyte maksimum=Math.Max(sbyte sayi1,sbyte sayi2);
short maksimum=Math.Max(short sayi1,short sayi2);
unit maksimum=Math.Max(unit sayi1,unit sayi2);
ulong maksimum=Math.Max(ulong sayi1,ulong sayi2);
ushort maksimum=Math.Max(ushort sayi1,ushort sayi2);
```

Örnek 0-42: Math.Max( ) metodu ile klavyeden girilen sayılardan büyüğünü bulma

```
Console.Write("1. sayıyı giriniz:");
int sayi1 = Convert.ToInt32(Console.ReadLine());
Console.Write("2. sayıyı giriniz:");
int sayi2 = Convert.ToInt32(Console.ReadLine());
int maksimum = Math.Max(sayi1, sayi2);
Console.WriteLine("{0} ve {1} sayılarından en büyüğü {2}
sayısıdır.", sayi1, sayi2, maksimum);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;

Resim 0-21. Math.Max( ) ile En Büyük Sayıyı Bölme

2.2.6. Min( )

Parametre olarak verilen iki sayıdan küçük olanı geriye döndüren metottur. Bütün sayı türleri tarafından desteklenen bir metot çeşididir.

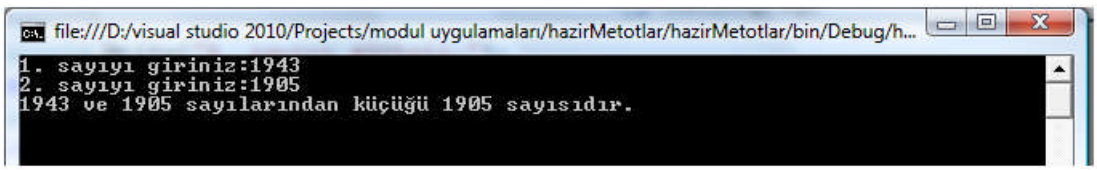
➢ **Kullanımı:**

```
byte minimum=Math.Min(byte sayi1,byte sayi2);
decimal minimum=Math.Min(decimal sayi1,decimal sayi2);
double minimum=Math.Min(double sayi1,double sayi2);
float minimum=Math.Min(float sayi1,float sayi2);
int minimum=Math.Min(int sayi1,int sayi2);
long minimum=Math.Min(long sayi1,long sayi2);
sbyte minimum=Math.Min(sbyte sayi1,sbyte sayi2);
short minimum=Math.Min(short sayi1,short sayi2);
unit minimum=Math.Min(unit sayi1,unit sayi2);
ulong minimum=Math.Min(ulong sayi1,ulong sayi2);
ushort minimum=Math.Min(ushort sayi1,ushort sayi2);
```

Örnek 0-43: Math.Min( ) metodu ile klavyeden girilen sayılardan küçüğünü bulma

```
    Console.Write("1. sayıyı giriniz:");
    int sayi1 = Convert.ToInt32(Console.ReadLine());
    Console.Write("2. sayıyı giriniz:");
    int sayi2 = Convert.ToInt32(Console.ReadLine());
    int minimum = Math.Min(sayi1, sayi2);
    Console.WriteLine("{0} ve {1} sayılarından küçüğü {2}
sayısıdır.", sayi1, sayi2, minimum);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



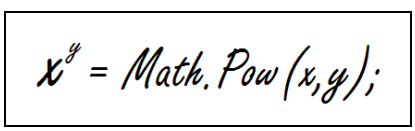
Resim 0-22. Math.Min( ) ile En Büyük Sayıyı Bölme

2.2.7. Pow( )

Parametre olarak verilen ilk sayının, yine parametre olarak verilen ikinci sayı kadar üssünü hesaplayan metottur.

- **Kullanımı:**

```
    double usluSayi=Math.Pow(double x, double y);
```



Resim 0-23. Math.Pow( ) Kullanımı

Örnek 0-44: Math.Pow( ) metodunun kullanımı

```
    double usluSayi=Math.Pow(3,3);          //    27
    double usluSayi=Math.Pow(2,16);         //    65536
    double usluSayi=Math.Pow(12,0);         //    1
    double usluSayi=Math.Pow(5,-2);         //    0.04
    double usluSayi=Math.Pow(-10,-2);       //    0.01
```

2.2.8. Round( )

Parametre olarak verilen sayıyı en yakın tam sayıya yuvarlayan metottur.

- **Kullanımı:**

```
    double yuvarlanmis=Math.Round(double sayi);
```

Örnek 0-45: Math.Round( ) metodunun kullanımı

```
     double yuvarlanmis=Math.Round(3.14);       //    3
     double yuvarlanmis=Math.Round(3.499);      //    3
     double yuvarlanmis=Math.Round(3.5);        //    4
     double yuvarlanmis=Math.Round(3.9999);     //    4
```

2.2.9. Sign( )

Parametre olarak verilen sayının işaretini verir. Sayı pozitif ise 1, negatif ise -1, sayı sıfıra eşitse de geriye 0 değerini döndüren metottur.

- **Kullanımı:**

```
     int isaret=Math.Sign(int sayi);
```

Örnek 0-46: Math.Sign( ) metodunun kullanımı

```
     int isaret=Math.Sign(26638);               //     1
     int isaret=Math.Sign(-26638);              //    -1
     int isaret=Math.Sign(0);                   //     0
```

2.2.10. Sqrt( )

Parametre olarak verilen **double** türündeki sayının karekök değerini **double** türünde geriye döndüren metottur.

- **Kullanımı:**

```
     double karekok=Math.Sqrt(double sayi);
```

Örnek 0-47: Klavyeden 2 kenar uzunluğu girilen dik üçgenin hipotenüsünün uzunluğunu hesaplayan programın kodunu yazınız.

```
     Console.Write("1. kenar uzunluğunu giriniz: ");
     double kenar1= Convert.ToDouble(Console.ReadLine());
     Console.Write("2. kenar uzunluğunu giriniz: ");
     double kenar2 = Convert.ToDouble(Console.ReadLine());
     double kenarlarinKareToplami = Math.Pow(kenar1, 2) +
Math.Pow(kenar2, 2);
     double kenar3= Math.Sqrt(kenarlarinKareToplami);
     Console.WriteLine("Hipotenüsün uzunluğu: "+kenar3);
```

Uyarı: Trigonometrik fonksiyonlarda açı değerleri radyan cinsinden verilmelidir.

Derece	0°	30°	45°	60°	90°	180°	270°	360°
Radyan	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π

Dilerseniz trigonometrik metotlara geçmeden önce verilen derece cinsinden verilen açı değerini radyana dönüştüren ve radyan değeri verilen açının derece cinsinden değerini veren basit metotlar yazalım ve bundan sonraki örneklerimizde bu metottan faydalanalım.

Örnek 0-48: Derece cinsinden açı değerini radyan cinsinden açı değerine dönüştüren metodun yazımı;

```
static double radyanaDonustur(double derece)
{
    double radyan = 0;
    double piSayisi = Math.PI;
    radyan = piSayisi * (derece / 180);
    return radyan;
}
```

Yukarıdaki şekilde yazdığımız metodu aşağıdaki gibi çağırarak programlarımız içerisinde kullanabiliriz.

```
double radyanDegeri=radyanaDonustur(double aci)
```

Örnek 0-49: Radyan cinsinden verilen açı değerini derece cinsinden açı değerine dönüştüren metodun yazımı;

```
static double aciyaDonustur(double radyan)
{
    double derece = 0;
    double piSayisi = Math.PI;
    derece = (radyan / piSayisi) * 180;
    return derece;
}
```

Yukarıdaki şekilde yazdığımız metodu aşağıdaki gibi çağırarak programlarımız içerisinde kullanabiliriz.

```
double dereceDegeri=aciyaDonustur(double radyan)
```

Matematiksel metotların içerisinde yer alan ve trigonometrik işlemlerde sıkça kullandığımız diğer metotlar ise şunlardır;

2.2.11. Cos( )

Parametre olarak verilen radyan açı değerinin kosinüs değerini veren metottur.

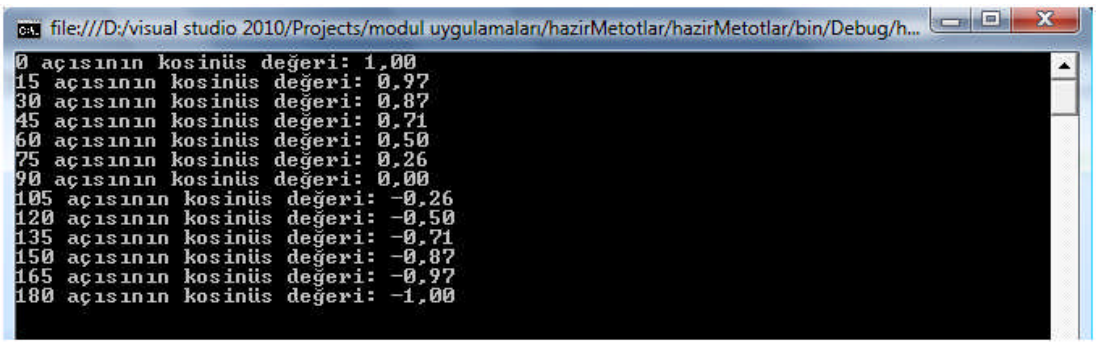
- **Kullanımı:**

```
double kosinus=Math.Cos(double aci);
```

Örnek 0-50: 0-180 arası 15 ve katlarının kosinüs değerini ekrana yazdıran programın kodlarını yazınız.

```
for (double aci = 0; aci <= 180; aci = aci + 15)
{
    double radyanDegeri = radyanaDonustur(aci);
    double kosinus = Math.Cos(radyanDegeri);
    Console.WriteLine("{0} açısının kosinüs değeri: {1:0.00}",
aci, kosinus);
}
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-24. Math.Cos ile Açı Değerlerinin Kosinüsünü Hesaplama

2.2.12. Sin( )

Parametre olarak verilen radyan açı değerinin sinüs değerini veren metottur.

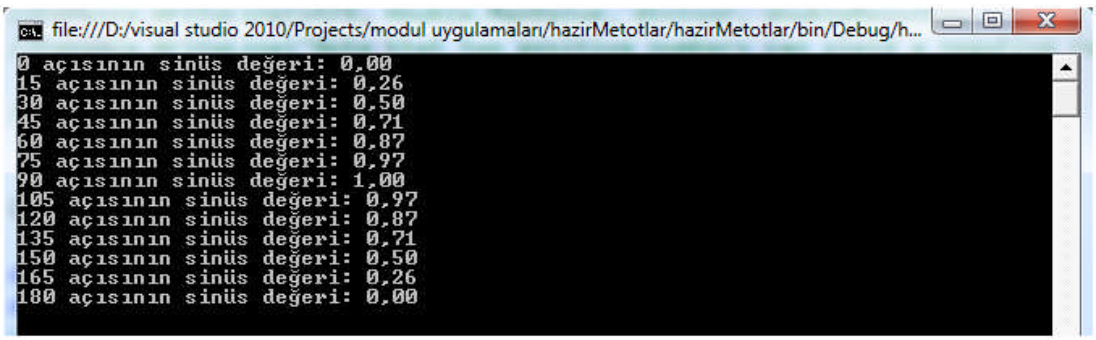
- **Kullanımı:**

```
double sinus=Math.Sin(double aci);
```

Örnek 0-51: 0-180 arası 15 ve katlarının sinüs değerini ekrana yazdıran programın kodlarını yazınız.

```
for (double aci = 0; aci <= 180; aci = aci + 15)
{
    double radyanDegeri = radyanaDonustur(aci);
    double sinus = Math.Sin(radyanDegeri);
    Console.WriteLine("{0} açısının sinüs değeri: {1:0.00}",
aci, sinus);
}
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-25. Math.Sin ile Açı Değerlerinin Sinüsünü Hesaplama

2.2.13. Tan( )

Parametre olarak verilen radyan açı değerinin tanjant değerini veren metottur.

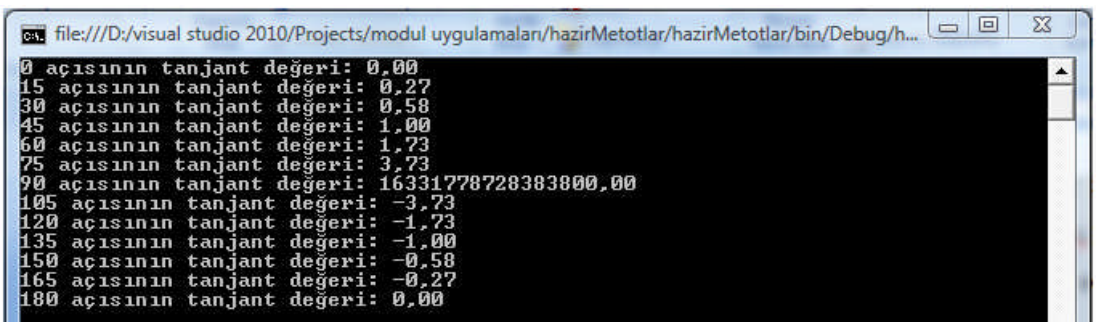
- **Kullanımı:**

```
double tanjant=Math.Sin(double aci);
```

Örnek 0-52: 0-180 arası 15 ve katlarının tanjant değerini ekrana yazdıran programın kodlarını yazınız.

```
for (double aci = 0; aci <= 180; aci = aci + 15)
{
    double radyanDegeri = radyanaDonustur(aci);
    double tanjant = Math.Tan(radyanDegeri);
    Console.WriteLine("{0} açısının tanjant değeri: {1:0.00}",
aci, tanjant);
}
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-26. Math.Tan ile Açı Değerlerinin Tanjantını Hesaplama

2.2.14. Acos( )

Parametre olarak verilen kosinüs değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double kosinusAcisi=Math.Acos(double kosinus);
```

2.2.15. Asin( )

Parametre olarak verilen sinüs değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double sinusAcisi=Math.Asin(double sinus);
```

2.2.16. Atan( )

Parametre olarak verilen tanjant değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double tanjantAcisi=Math.Atan(double tanjant);
```

2.3. Tarih/Saat (DateTime) Metotları

Programlama dili içerisinde, tarih ve zamanlar ile ilgili işlemler yaparken bir takım işleri daha kolay yapabilmemiz için önceden tanımlanmış Tarih/Zaman metotlarını kullanırız. Tarih/Zaman (DateTime) metotlarından sık kullanılanlar şunlardır;

- DateTime sınıfı ile çağırılan metotlar;

  Compare, DaysInMonth, IsLeapYear, Parse,

- DateTime türünde bir ifade ile birlikte çağrılan metotlar;

  Subtract, AddDays, AddMonths, AddYears, AddHours, AddMinutes, AddSeconds, AddMilliseconds

Tarih / Zaman metotlarının ayrıntılarını incelemeden önce DateTime sınıfı altında yeralan ve sıkça kullanacağımız üyeleri tanıyalım.

2.3.1. MinValue

Bu özellik ile DateTime yapısı ile kullanabileceğimiz en küçük tarih-saat bilgisine erişebiliriz.

➢ **Kullanımı**:

```
DateTime enKucuk=DateTime.MinValue;
```

Bu sabit özellik çağrıldığında geriye veri türü DateTime olan "01.01.0001 00:00:00" değerini döndürür. Bu değer değiştirilemeyen "Salt Okunur" bir veridir.

2.3.2. MaxValue

Bu özellik ile DateTime yapısı ile kullanabileceğimiz en büyük tarih-saat bilgisine erişebiliriz.

➢ **Kullanımı:**

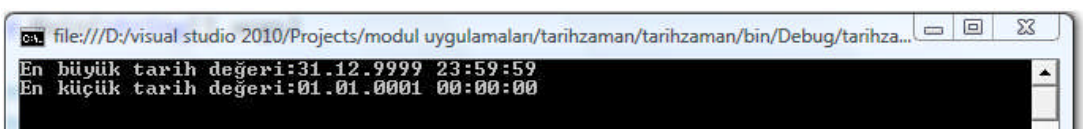
```
DateTime enBuyuk=DateTime.MaxValue;
```

Bu sabit özellik çağrıldığında geriye veri türü DateTime olan "31.12.9999 23:59:59" değerini döndürür. Bu değer de aynı MinValue gibi değiştirilemeyen "Salt Okunur" bir veridir.

Örnek 0-53: DateTime sınıfı ile kullanılabilen en büyük ve en küçük tarihlerin ekrana yazdırınız;

```
DateTime enKucuk=DateTime.MinValue;
DateTime enBuyuk=DateTime.MaxValue;
Console.WriteLine("En büyük tarih değeri:{0} ", enBuyuk);
Console.WriteLine("En küçük tarih değeri:{0} ", enKucuk);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-27. Sistemdeki En Büyük ve En Küçük Tarihler

2.3.3. Today

Bu özellik ile DateTime yapısı ile birlikte kullanılır ve bugünün tarihini bize DateTime türünde geri döndürür.

➢ **Kullanımı:**

```
DateTime bugun=DateTime.Today;
```

Programın çalıştırıldığı sistemin tarih değerini gösterir.

2.3.4. Now

Bu özellik de DateTime yapısı ile birlikte kullanılır ve çağrıldığı andaki hem tarih hem de saat bilgisini bize DateTime türünde geri döndürür.

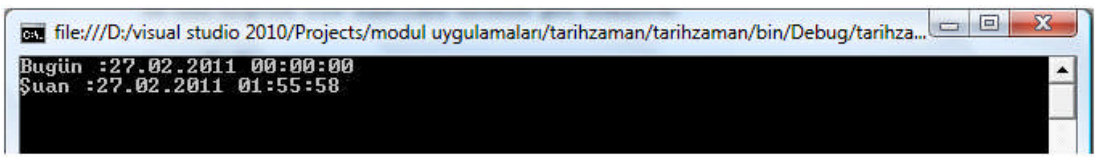
- **Kullanımı:**

```
DateTime simdi=DateTime.Now;
```

Örnek 0-54: DateTime sınıfı ile kullanılabilen Now ve Today özelliklerinin kullanımı

```
DateTime bugun=DateTime.Today;
DateTime simdi=DateTime.Now;
Console.WriteLine("Bugün :{0} ", bugun);
Console.WriteLine("Şuan :{0} ", simdi);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-28. Today ve Now Kullanımı

DateTime sınıfından türetilmiş bir nesne ile çalışırken sıkça kullanacağımız özellikler aşağıdaki tabloda açıklamalarıyla birlikte verilmiştir;

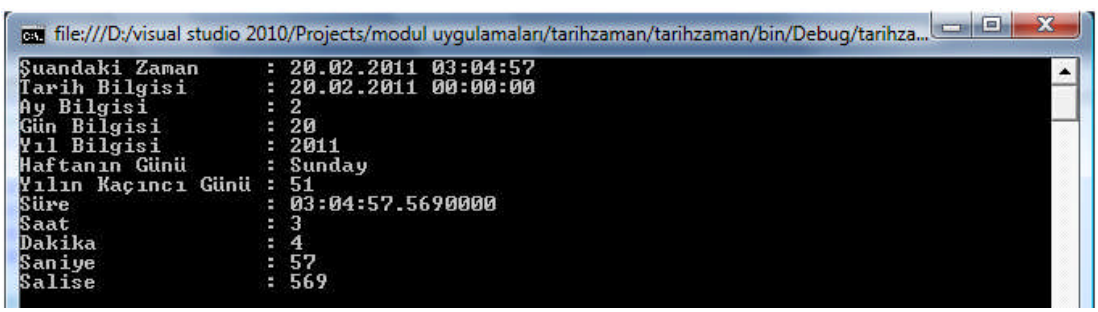
Özellik Adı	Açıklama	Geri Dönüş Türü
Date	Nesneye ilişkin saat dışındaki bilgiyi verir.	DateTime
Month	Nesnenin ay bilgisini verir.	int
Day	Nesnenin gün bilgisini verir.	int
Year	Nesnenin yıl bilgisini verir.	int
DayOfWeek	Haftanın günü, Nesnenin haftanın kaçıncı günü olduğu bilgisini verir.	DayOfWeek
DayOfYear	Nesnenin yılın kaçıncı günü olduğu bilgisini verir.	int
TimeOfDay	Nesneye ait saat bilgisini verir.	TimeSpan
Hour	Nesnenin saat bilgisini verir.	int
Minute	Nesnenin dakika bilgisini verir.	int
Second	Nesnenin saniye bilgisini verir.	int
Millisecond	Nesnenin milisaniye bilgisini verir.	int

Örnek 0-55: DateTime sınıfı ile kullanılabilen özelliklerin kullanımı ve ekrana yazdırılması

```
     DateTime zaman = DateTime.Now;
     DateTime tarih = zaman.Date;
     int ay=zaman.Month;
     int gun=zaman.Day;
     int yil=zaman.Year;
     DayOfWeek haftaninGunu=zaman.DayOfWeek;
     int yilinKacinciGunu=zaman.DayOfYear;
     TimeSpan sure=zaman.TimeOfDay;
     int saat=zaman.Hour;
     int dakika=zaman.Minute;
     int saniye=zaman.Second;
     int salise = zaman.Millisecond;

     Console.WriteLine("Şuandaki Zaman     : {0}", zaman);
     Console.WriteLine("Tarih Bilgisi      : {0}", tarih);
     Console.WriteLine("Ay Bilgisi         : {0}", ay);
     Console.WriteLine("Gün Bilgisi        : {0}", gun);
     Console.WriteLine("Yıl Bilgisi        : {0}", yil);
     Console.WriteLine("Haftanın Günü      : {0}", haftaninGunu);
     Console.WriteLine("Yılın Kaçıncı Günü : {0}",
yilinKacinciGunu);
     Console.WriteLine("Süre               : {0}", sure);
     Console.WriteLine("Saat               : {0}", saat);
     Console.WriteLine("Dakika             : {0}", dakika);
     Console.WriteLine("Saniye             : {0}", saniye);
     Console.WriteLine("Salise             : {0}", salise);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-29. DateTime Nesnesinin Özelliklerinin Kullanımı

DateTime Sınıfından türetilen nesnelerle birlikte kullanılan özelliklerden sonra şimdi de DateTime sınıfı altında yer alan metotlarımıza göz atalım.

2.3.5. DateTime.Compare( )

Bu metot ile DateTime türünde parametre olarak verilen iki değer karşılaştırılır ve geriye int türünde bir değer döndürür.

Eğer karşılaştırmada;

- 1. parametredeki tarih değeri, 2. parametredeki tarih değerinden daha eski bir tarih değeri ise sonuç -1,
- 2. parametredeki tarih değeri, 1. parametredeki tarih değerinden daha eski bir tarih değeri ise sonuç 1,
- 1. parametredeki tarih değeri ile 2. parametredeki tarih değeri birbirine eşitse sonuç 0,

olarak geriye döner.

- **Kullanımı:**

```
int sonuc=DateTime.Compare(DateTime tarih1, DateTime tarih2);
```

Örnek 0-56: 31/12/1990 tarihinden önce doğmuş kişilerin başvuru yapabildikleri bir sınava, klavyeden doğum tarihi bilgileri girilen kişinin, sınava girip giremeyeceğini kontrol eden programın kodlarını yazınız.

```
    DateTime tarih1 = new DateTime(1990, 12, 31);
    Console.Write("Doğum tarihinizi GG/AA/YYYY şeklinde giriniz:
");
    //Klavyeden girilen string türündeki doğum tarihi DateTime
türüne dönüştürülüyor.
    DateTime dogumTarihi = Convert.ToDateTime(Console.ReadLine());
    int sonuc = DateTime.Compare(tarih1,dogumTarihi);
    if(sonuc==-1)
        Console.Write("Üzgünüz, sınava giremezsiniz...");
    else
        Console.Write("Tebrikler, sınava girebilirsiniz...");
```

2.3.6. DateTime.DaysInMonth( )

Bu metot ile **int** türünde parametre olarak verilen yıl ve ay bilgilerine denk gelen ayın kaç günden oluştuğunu, geriye int türünde bir değer döndürür.

- **Kullanımı:**

```
int gunSayisi=DateTime.Compare(int yil, int ay);
```

```
int gunSayisi = DateTime.DaysInMonth(2008,2);          // 29
int gunSayisi = DateTime.DaysInMonth(2009,8);          // 31
int gunSayisi = DateTime.DaysInMonth(2010,4);          // 30
int gunSayisi = DateTime.DaysInMonth(2011,2);          // 28
```

2.3.7. DateTime.IsLeapYear( )

Bu metot ile **int** türünde parametre olarak verilen yılın, "***artık yıl***" (şubat ayının 29 günden, yılın toplam 366 günden oluştuğu) olup olmadığını kontrol eder, geriye **bool** türünde bir değer döndürür. Eğer yıl artık yılsa geriye **true**, değilse **false** değeri döndürür.

- **Kullanımı:**

```
bool artikYilMi=DateTime.IsLeapYear(int yil);
```

```
bool artikYilMi=DateTime.IsLeapYear(2008);            // true
bool artikYilMi=DateTime.IsLeapYear(2010);            // false
bool artikYilMi=DateTime.IsLeapYear(2012);            // true
bool artikYilMi=DateTime.IsLeapYear(1996);            // true
```

2.3.8. DateTime.Parse( )

Bu metot ile **string** türünde parametre olarak verilen metni DateTime türündeki tarih ve saat bilgisine dönüştürür.

- **Kullanımı:**

```
DateTime tarih=DateTime.Parse(string metin);
```

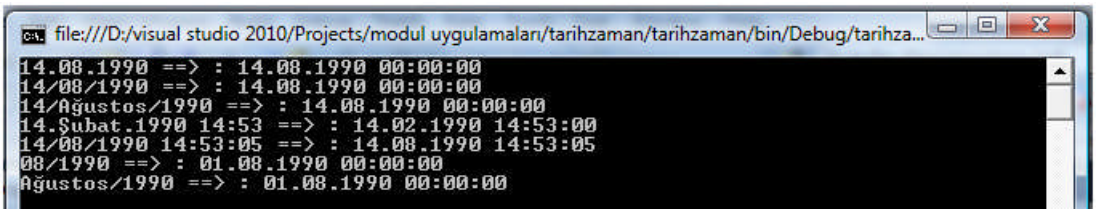
Örnek 0-57: DateTime.Parse( ) ile metinlerin tarihe dönüştürülmesi

```
string metin1 = "14.08.1990";
string metin2 = "14/08/1990";
string metin3 = "14/Ağustos/1990";
string metin4 = "14.Şubat.1990 14:53";
string metin5 = "14/08/1990 14:53:05";
string metin6 = "08/1990";
string metin7 = "Ağustos/1990";

DateTime tarih1 = DateTime.Parse(metin1);
DateTime tarih2 = DateTime.Parse(metin2);
DateTime tarih3 = DateTime.Parse(metin3);
DateTime tarih4 = DateTime.Parse(metin4);
DateTime tarih5 = DateTime.Parse(metin5);
DateTime tarih6 = DateTime.Parse(metin6);
DateTime tarih7 = DateTime.Parse(metin7);

Console.WriteLine("{0} ==> : {1}" , metin1, tarih1);
Console.WriteLine("{0} ==> : {1}" , metin2, tarih2);
Console.WriteLine("{0} ==> : {1}" , metin3, tarih3);
Console.WriteLine("{0} ==> : {1}" , metin4, tarih4);
Console.WriteLine("{0} ==> : {1}" , metin5, tarih5);
Console.WriteLine("{0} ==> : {1}" , metin6, tarih6);
Console.WriteLine("{0} ==> : {1}" , metin7, tarih7);
```

Yukarıdaki kod parçası çalıştırıldığında aşağıdaki gibi bir ekran çıktısı ile karşılaşırız;



Resim 0-30. DateTime.Parse( ) ile Metni Tarihe Dönüştürme

Yukarıdaki örnekte çeşitli biçimlerde metin olarak verilen tarih/saat bilgilerinin nasıl sonuçlar verdiğini inceleyiniz.

Bu kısımdan sonra anlatılacak olan tarih/zaman metotları direkt olarak DateTime sınıfı üzerinden değil, DateTime sınıfından türetilmiş nesneler üzerinden çağrılacak olan metotlardır.

DateTime Sınıfından türetilen nesnelerle birlikte kullanılan metotlarda şunlardır;

2.3.9. Subtract( )

Bu metot ile DateTime türünde türetilmiş bir nesnenin değerinden, parametre olarak verilen **DateTime** veya **TimeSpan** türündeki değer çıkartılır ve geriye **TimeSpan** türünde bir değer döndürür.

- **Kullanımı:**

```
TimeSpan yeniTarih=eskiTarih.Subtract(DateTime cikarilanTarih);
```

TimeSpan türünde dönüş yapılan değer üzerinden;
- **TotalDays** özelliği ile toplam gün sayısını,
- **TotalHours** özelliği ile toplam saati,
- **TotalMinutes** özelliği ile toplam dakikayı,
- **TotalSeconds** özelliği ile toplam saniyeyi
- **TotalMilliseconds** özelliği ile toplam saliseyi

görebiliriz.

Örnek 0-58: Şuandaki zamandan doğum tarihinizi çıkartarak ne kadar süredir hayatta olduğunuzu hesaplayınız.

```
    DateTime bugun= DateTime.Now;
    Console.Write("Doğum tarihinizi giriniz: ");
    DateTime dogumTarihiniz = DateTime.Parse(Console.ReadLine());
    TimeSpan sure = bugun.Subtract(dogumTarihiniz);
    Console.WriteLine("Dünya üzerinde geleli {0} gün olmuş.",
sure.TotalDays);
```

```
    Console.WriteLine("Dünya üzerinde geleli {0} saat olmuş.",
sure.TotalHours);
    Console.WriteLine("Dünya üzerinde geleli {0} dakika olmuş.",
sure.TotalMinutes);
    Console.WriteLine("Dünya üzerinde geleli {0} saniye olmuş.",
sure.TotalSeconds);
    Console.WriteLine("Dünya üzerinde geleli {0} salise olmuş.",
sure.TotalMilliseconds);
```

2.3.10. AddDays( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar gün eklenir ve geriye **DateTime** türünde bir değer döndürür.

- **Kullanımı:**

```
    DateTime yeniTarih=eskiTarih.AddDays(double gunSayisi);
```

2.3.11. AddMonths( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar ay eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

- **Kullanımı:**

```
    DateTime yeniTarih=eskiTarih.AddDays(double ay);
```

2.3.12. AddYears( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar yıl eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

- **Kullanımı:**

```
    DateTime yeniTarih=eskiTarih.AddDays(double yil);
```

2.3.13. AddHours( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar saat eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

- **Kullanımı:**

```
    DateTime yeniTarih=eskiTarih.AddDays(double saat);
```

2.3.14. AddMinutes( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar dakika eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

- **Kullanımı:**

```
DateTime yeniTarih=eskiTarih.AddDays(double dakika);
```

2.3.15. AddSeconds( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar saniye eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

- **Kullanımı:**

```
DateTime yeniTarih=eskiTarih.AddDays(double saniye);
```

2.3.16. AddMilliseconds( )

Bu metot ile DateTime türünde türetilmiş bir nesneye, parametre olarak verilen **double** türündeki değer kadar saniye eklenir ve geriye **DateTime** türünde yeni bir tarih döndürür.

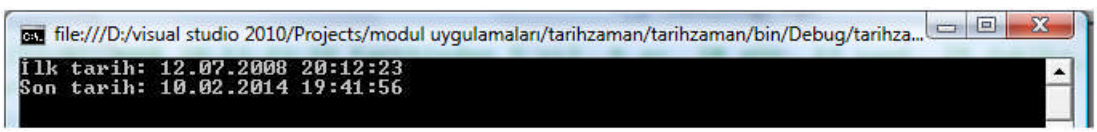
- **Kullanımı:**

```
DateTime yeniTarih=eskiTarih.AddDays(double saniye);
```

Örnek 0-59: 12/07/2008 20:12:23:33 tarihinden 5 yıl, 6 ay, 28 gün, 23 saat, 29 dakika, 33 saniye ve 43 salise sonrasının tarihi nedir?

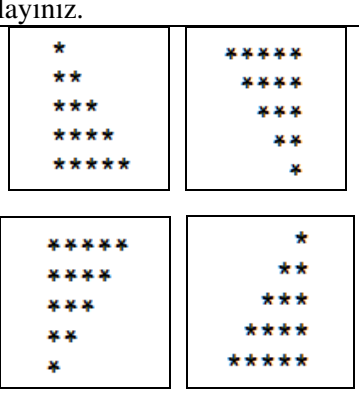
```
DateTime ilkTarih = new DateTime(2008,07,12,20,12,23,33);
DateTime bitisTarihi = ilkTarih.AddYears(5);
bitisTarihi = bitisTarihi.AddMonths(6);
bitisTarihi = bitisTarihi.AddDays(28);
bitisTarihi = bitisTarihi.AddHours(23);
bitisTarihi = bitisTarihi.AddMinutes(29);
bitisTarihi = bitisTarihi.AddSeconds(33);
bitisTarihi = bitisTarihi.AddMilliseconds(43);
Console.WriteLine("İlk tarih: "+ ilkTarih);
Console.WriteLine("Son tarih: "+ bitisTarihi);
```

Yukarıdaki kodları çalıştırdığımız zaman aşağıdaki gibi bir ekran görüntüsüyle karşılaşırız;



Resim 0-31. Ekleme Metotlarının Toplu Kullanımı

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➢ Sıra No, Ad, Soyad ve Telefon numarası sütun başlıklarına sahip bir tablo hazırlayınız.	➢ Örnek 2.6'daki gibi String.Format( ) metodundan faydalanabilirsiniz.
 ➢ Yukarıdaki şekilleri PadLeft, PadRight metotlarını kullanarak ekrana tek tek yazdırınız.	➢ PadLeft( ) ve PadRight( ) metotlarının kullanımlarını ve çalışmalarını inceleyiniz. ➢ 1'den 5'e döngünüzü kurunuz. ➢ PadLeft( ) veya PadRight( ) metotlarından hangisini kullanacağınıza karar veriniz. ➢ İfadenize ilgili biçimi verdikten sonra ekrana yazdırınız. ➢ 4 şekilde de yukarıdaki adımlardan faydalanınız.
➢ Klavyeden girilen metin içerisindeki Türkçe karakterleri (ç,ğ,ı,ö,ş,ü, Ç,Ğ,İ,Ö,Ş,Ü) sırasıyla (c,g,i,o,s,u, C,G,I,O,S,U) harfleriyle değiştiren metodu yazınız.	➢ Parametresi ve geri dönüş değeri string olan yeni bir metot tanımlayınız. ➢ Kelime içerisindeki harfleri tek tek kontrol ediniz. ➢ Switch-Case yapısını kullanarak Türkçe karakterleri tespit edip, daha sonra bunları yeni değerleriyle değiştiriniz. ➢ Yeni metni string türünde geri gönderiniz.
$x = \frac{-b + \lvert\sqrt{b^2 - 4ac}\rvert}{2a}$ ➢ Şeklindeki bir formülün a,b ve c değerlerini klavyeden girerek, x değerini hesaplayan programın kodunu yazınız.	➢ Çarpma işlemi için BigMul( ) metodunu, ➢ Karekök almak için Sqrt( ) metodunu, ➢ Üs almak için Pow( ) metodunu, ➢ Mutlak değeri hesaplamak için Abs( ) metodunu kullanınız.
➢ Doğum günü, doğum ayı, doğum yılı bilgileri girildikten sonra, bugüne kadar geçen süreyi ve haftanın hangi gününde doğduğunu hesaplayan programı yazınız	➢ Bugünün tarihinden doğum tarihini çıkartınız ve gecen gün sayısını hesaplayınız. ➢ DayOfWeek metoduyla doğduğu günü bulunuz.

ÖLÇME VE DEĞERLENDİRME

Bu faaliyet kapsamında kazandığınız bilgileri, aşağıdaki soruları cevaplayarak belirleyiniz.

Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. İki metinsel ifadeyi karşılaştırmaya yaran metot aşağıdakilerden hangisidir?
A) Compare( ) B) Concat( ) C) Copy( ) D) Format( )

2. Metinsel ifadenin içerisinde başka bir ifadeyi aramak için kullanılan metot aşağıdakilerden hangisidir?
A) Concat( ) B) Substring( ) C) Contains( ) D) CopyTo( )

3. Rakamlardan oluşan metinsel bir ifadeyi belirli bir formda yazmak için kullanılan metinsel metot aşağıdakilerden hangisidir?
A) Insert( ) B) Concat( ) C) IndexOf( ) D) Format( )

4. Aşağıdakilerden hangisi trigonometrik işlemler için kullanılan bir metottur?
A) Sign( ) B) Pow( ) C) Sin( ) D) BigMul( )

5. Aşağıdakilerden hangisi bir sayının karekökünü almak için kullanılan bir metottur?
A) Sqrt( ) B) Cos( ) C) Atan( ) D) Sign( )

6. Aşağıdakilerden hangisi ondalıklı bir sayıyı yuvarlamak için kullanılan bir metottur?
A) Round( ) B) Abs( ) C) Tan( ) D) BigMul( )

7. Aşağıdakilerden hangisi IsLeapYear( ) metoduna parametre olarak gönderildiğinde, geriye true değeri döner?
A) 1923 B) 1453 C) 2012 D) 1881

8. Aşağıdakilerden bugünün tarihini ve saatini veren özelliktir?
A) DateTime.Today B) DateTime.Now
C) DateTime.MaxValue D) DateTime.MinValue

9. Aşağıdakilerden hangisi iki tarih arasındaki farkı bulmaya yarayan metottur?
A) Parse B) Subtract C) AddDays D) AddYears

10. Aşağıdakilerden hangisinin geri dönüş değerinin türü int türündedir?
A) DateTime.Date B) DateTime.DayOfWeek
C) DateTime.TimeOfDay D) DateTime.Day

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru "Modül Değerlendirme"ye geçiniz.

MODÜL DEĞERLENDİRME

Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

1. ( ) Eğer metot bir değer döndürmeyecekse dönüş-tipi string olarak belirtilmelidir.
2. ( ) Bir metot hiç parametre kullanmayacaksa, parametre-listesine void yazılır.
3. ( ) Parametre listesindeki değişkenler birbirinden virgül (,) ile ayrılırlar.
4. ( ) Main( ) isimli metot, programımızın çalışmasını başlatan metottur.
5. ( ) Aynı isme sahip metotlar oluşturamayız.
6. ( ) Bir metot kendi kendini çağırabilir. Bu tip metotlara da özyineli metotlar denir.
7. ( ) Parametre-listesi farklı fakat isimleri aynı metotlar oluşturabiliriz.
8. ( ) Main( ) metodu bir program içerisinde yalnızca bir kez yazılır.
9. ( ) Özyineli metotlar, geri dönüş değeri olmayan metotlardır.
10. ( ) Metotlarda geri dönüş değeri return anahtar kelimesiyle gönderilir.
11. ( ) Bir string ifadenin boş olup olmadığını kontrol etmek için IsNullOrEmpty( ) metodu kullanılır.
12. ( ) CompareTo( ) metoduyla karşılaştırılan iki metin eşitse, geriye 1 değeri döner.
13. ( ) IndexOf( ) metodu sadece verilen karaktere göre işlem yapar.
14. ( ) Bir sayının mutlak değerini Abs( ) metoduyla buluruz.
15. ( ) Bir sayının üssünü almak için BigMul( ) metodu kullanılır.
16. ( ) Bir sayının negatif veya pozitif olduğunu Sign( ) metodu ile buluruz.
17. ( ) Trigonometrik metotlarda açı değeri derece olarak verilir.
18. ( ) Atan( ) metodu, parametre olarak verilen açının tanjant değerini hesaplar.
19. ( ) DayOfWeek özelliği haftanın günü bilgisini verir.
20. ( ) Subtract( ) metodu geriye TimeSpan türünde bir değer döndürür.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1	**Y**	**11**	**Y**
2	**D**	**12**	**Y**
3	**Y**	**13**	**Y**
4	**Y**	**14**	**D**
5	**D**	**15**	**Y**
6	**D**	**16**	**D**
7	**D**	**17**	**Y**
8	**Y**	**18**	**D**
9	**D**	**19**	**Y**
10	**Y**	**20**	**Y**

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1	**A**
2	**C**
3	**D**
4	**C**
5	**A**
6	**A**
7	**C**
8	**B**
9	**B**
10	**D**

MODÜL DEĞERLENDİRME SORULARI CEVAP ANAHTARI

1	**Y**	**11**	**D**
2	**Y**	**12**	**Y**
3	**D**	**13**	**Y**
4	**D**	**14**	**D**
5	**Y**	**15**	**Y**
6	**D**	**16**	**D**
7	**D**	**17**	**Y**
8	**D**	**18**	**Y**
9	**Y**	**19**	**D**
10	**D**	**20**	**D**

KAYNAKÇA

- ALGAN Sefer, **Her Yönüyle C#**, Pusula Yayıncılık, 1.Baskı, İstanbul, Türkiye, (2003)
- SCHILDT Herbert, **Herkes İçin C#**, Alfa Yayınevi, 1.Baskı, İstanbul, Türkiye, (2002)
- ASLAN KAAN, **A'dan Z'ye C Kılavuzu**, Pusula Yayıncılık, 8.Baskı, İstanbul, Türkiye, (**2002**)
- Butow, E., Ryan, T.: **"Your Visual Blueprint For Building .NET Application"**
- MSDN : **"Introduction to C# Programming for the Microsoft® .NET Platform (Prerelease)Workbook"**
- Hejlsberg, A., Wiltamuth, S.: **"C# Language Referance"**
- Turtschi, A., Werry J., Hack, G., Albahari, J., Nandu S.: **"C#.NET Web Developer's Guide"**, Syngress Publishing, Inc., Rockaland, USA (2002)
- Microsoft: **"C# Language Specification"**
- MSDN Yardım Dokümanları