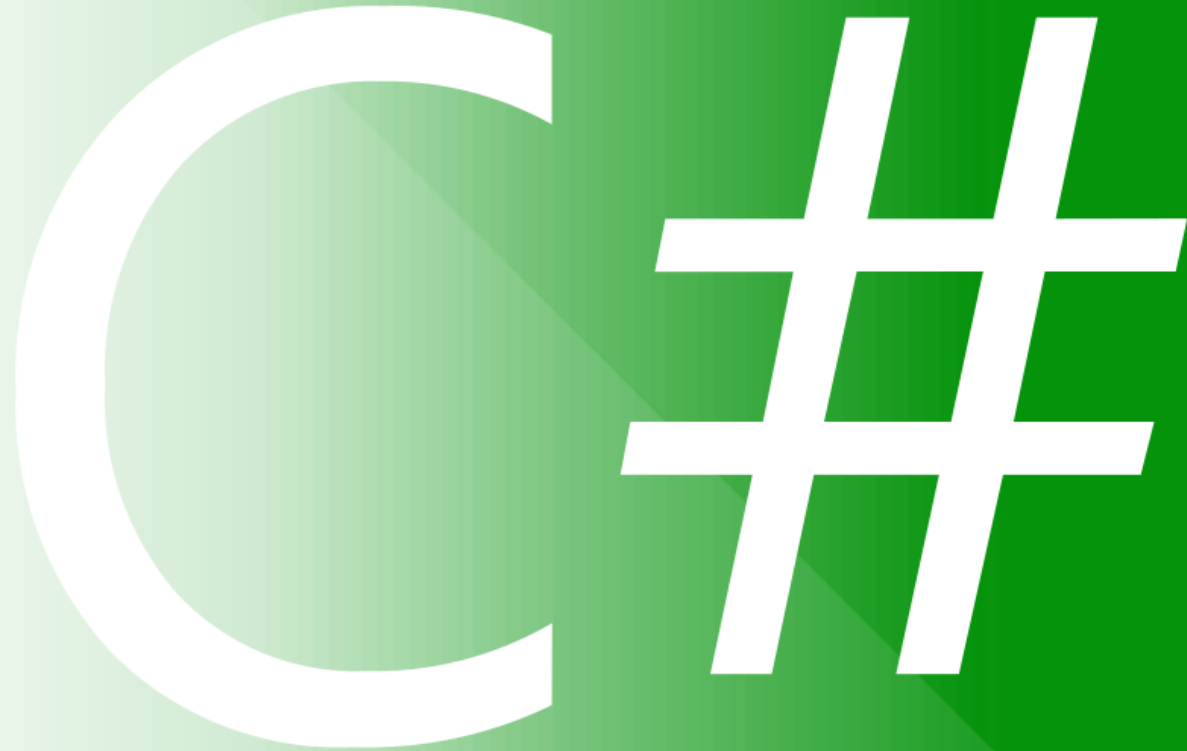




ISE 102- Nesneye Dayalı Programlama

Hafta 2 –
C# Temelleri

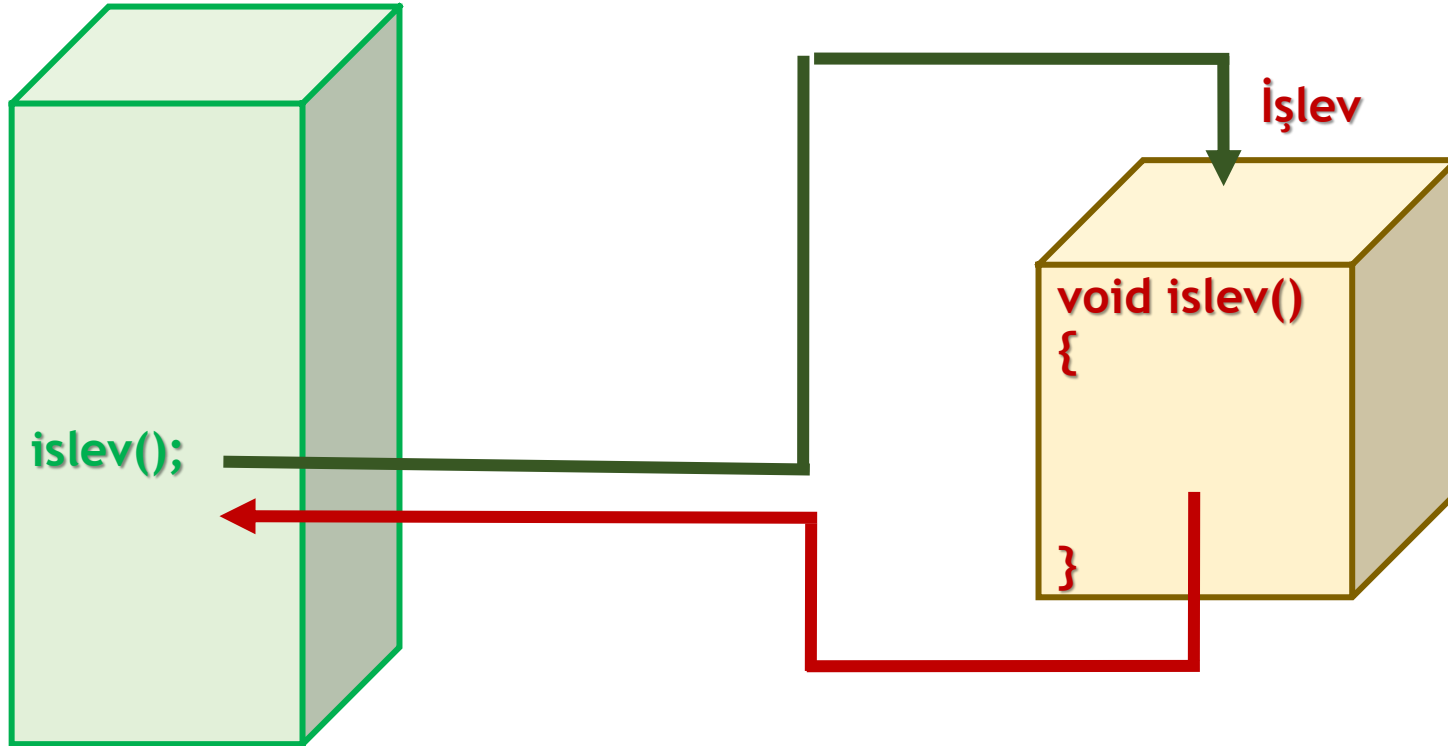


Metot ve Fonksiyonlar

- Programın herhangi bir yerinde bir işi yerine getirmek amacıyla tasarlanmış alt programlardır.
- Büyük problemler, küçük parçalara bölünerek kolay çözülür.
- Yazılımlar benzer modüller içerirler. Dolayısıyla program içerisinde tekrar eden işlemlerin her defasında yeniden yazılması engellenerek program geliştirme kolaylaştırılır.
- Hata ayıklama küçük ölçekte daha kolaydır.
- Küçük parçalara yoğunlaşmak kolaydır.
- Daha güvenli ve verimli kod üretimi sağlanır.
- Ayrıca program boyutları da nispeten küçülür.

Metot

İşlevi Çağırın Program



Metot Tanımlama

```
[erişim belirteçleri] <Geri dönüş tipi> Metot Adı (parametre_listesi) {  
    gerekli değişken tanımlamaları...;  
    metot gövdesi...;  
    return geriDonusDegeri / Degiskeni;  
}
```

- Erişim belirteci yazılmazsa varsayılan olarak bildirilen sınıfa özel <private> olur.
- **Metot adı**
 - Metodun adını belirler. Değişken isimlendirme kuralları aynen geçerlidir.
- **Parametre listesi**
 - Metot içerisinde kullanılacak giriş parametreleri tanımlanır.
- **Geri Dönüş Değeri / Değişkeni**
 - Geriye dönen değer / değişken belirtilir.
- **Dönüş Tipi**
 - Metottan geri dönen değerın tipi yazılır. Geriye değer dönmeyecek ise **void** yazılmalıdır. void kullanımı durumunda fonksiyon return komut satırı içermez. Fonksiyonda geri dönüş değeri/değişkeni **return** komut satırı ile belirtilir.

Örnekler

```
float aylıkMaas(float mesaiSaati, float saatUcreti)
```

```
float bol(int x, float f)
```

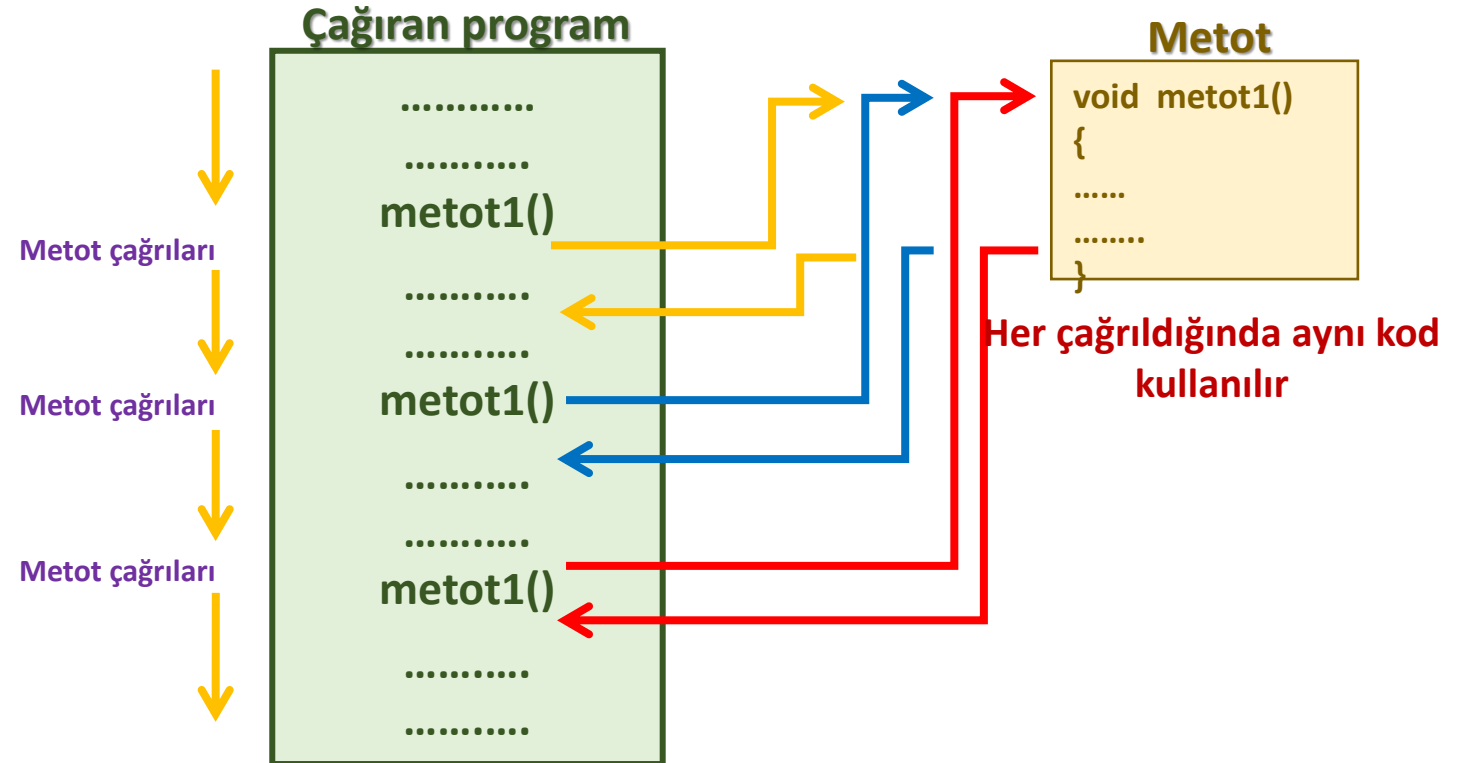
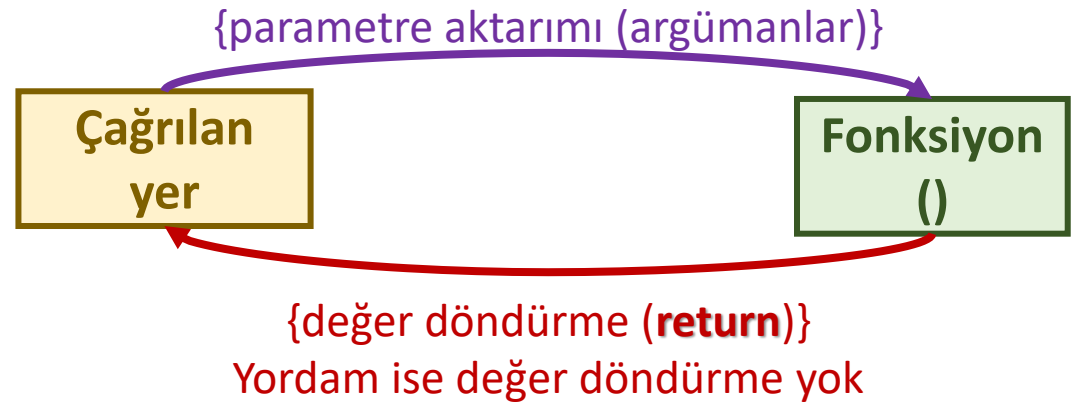
```
int kareAl(int x)
{
    return x*x;
}
```

```
void mesaj(int mesajNo)
{
    switch (mesajNo)
    {
        case 0 : cout << "Mesaj 0"; break;
        case 1 : cout << "Mesaj 1"; break;
        default: cout << "Tanımsız..."; break;
    }
}
```

Önemli Özellikler

- Metotlara mümkün olduğu kadar az görev verilmelidir.
- Bir metot az ve öz iş yapmalıdır.
 - Örneğin bir metot hem diziye sıralayıp hem de dizinin elemanlarını ekrana yazdırmamalıdır.
- Değişken isimlendirme kuralları aynen geçerlidir.
- Metot çağrılırken istenen parametreler eksiksiz girilmelidir.
- Geri dönüş değeri herhangi bir türden olabilir. Geri dönüş değeri olmayan metotlar **void** anahtar sözcüğü ile bildirilmelidir.
- Metotların geri dönüş değerini uyumlu bir türden nesneye atamak gerekir.
 - Örn. Geri dönüş değeri **double** ise **int** türünden bir nesneye dönüşüm yapılmadan atanamaz.
- **Void** metotlarda **return** anahtar sözcüğünü bir ifade ile birlikte kullanılamaz.
- Metot içerisinde başka bir metot bildirilemez.
- Metot içerisinde tanımlanan değişken metot dışında geçersizdir.

Metot kullanımı ve kontrol akışı



Metot kullanımı ve kontrol akışı

Dönüş tipi yok

```
void main()  
{  
.....  
.....  
metot1();  
.....  
}  
  
void metot1()  
{  
.....  
.....  
}
```

Metot çağırma

Noktalı virgül yok

Metot bildirim

Metot ifadeleri
(gövde)

Metot tanımı

Noktalı virgül yok

Metot-Örnek: Girilen 2 sayıdan büyük olanı bulma

Ekran

```
1. Sayıyı Giriniz
125
2. Sayıyı Giriniz
78
En büyük sayı :125
```

1 reference

```
static int EnBuyuk(int s1,int s2)
{
    if (s1 > s2)
        return s1;
    else
        return s2;
}
```

0 references

```
static void Main(string[] args)
{
    int sayi1, sayi2;
    //Sayıları kullanıcıdan alma
    Console.WriteLine("1. Sayıyı Giriniz");
    sayi1 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("2. Sayıyı Giriniz");
    sayi2 = Convert.ToInt32(Console.ReadLine());

    //Metot çağırma
    int buyukSayi = EnBuyuk(sayi1, sayi2);

    Console.WriteLine("En büyük sayı :{0}", buyukSayi);

    Console.ReadLine();
}
```

Çalışma Sorusu

Girilen 3 sayıdan en büyüğünü bulan programı bir önceki slayttaki EnBuyuk metodunu kullanan yeni bir metot ile yazınız.

Değer ve Referans Parametreleri

- Referans türleri metotlara aktarılırken bit bit kopyalanmazlar, yani metoda aktarılan bir referanstır.
- Metot içerisinde bu referans ile direkt işlem yapmak nesnenin değerini değiştirebilir. Bu nedenle dikkatli olmak gerekir.
- C#'da bütün değer tipleri metotlara bit bit kopyalanarak geçirilir.
 - Örn int türünden bir nesne metoda parametre olarak aktarıldığında metot içerisinde bu değer değiştirilse bile ana değişken değişmez.
- Bir referans türü olan örneğin dizileri metoda parametre olarak aktarır isek dizinin bütün elemanları metoda parametre olarak kopyalanmaz. Parametre olarak kopalanan sadece ilgili dizinin referansıdır. Metot içerisinde referansla ilgili değişiklik yaptığımızda orijinal nesneyi de değiştirmiş oluruz.
- C# varsayılan olarak bütün değer tipleri metotlara değer olarak; referans tipleri ise referans olarak aktarılır.
- Bazı durumlarda değer tipleri de referans olarak aktarılmak istenebilir.(Bknz: **ref** ve **out** anahtar sözcükleri)

```

1 reference
static void DegerTipiAktarma(int deger)
{
    deger = 54;
}

1 reference
static void ReferansTipiAktarma(int[] deger)
{
    deger[0] = 100;
}

0 references
static void Main(string[] args)
{
    //Değer tipi aktarım
    Console.WriteLine("Değer ile Aktarma");
    Console.WriteLine("-----");
    int not = 49;
    Console.WriteLine(not);
    DegerTipiAktarma(not);
    Console.WriteLine(not);

    //Referans tipi aktarım
    Console.WriteLine("\nReferans ile Aktarma");
    Console.WriteLine("-----");
    int[] notlar = { 15, 35, 55, 90 };
    Console.WriteLine(notlar[0]);
    ReferansTipiAktarma(notlar);
    Console.WriteLine(notlar[0]);

    Console.ReadLine();
}

```

Değer ile Aktarım

Referans ile Aktarım

```

Değer ile Aktarma
-----
49
49

Referans ile Aktarma
-----
15
100

```

Metot Aşırı Yükleme ve İmza Kavramı

- **Metot Aşırı Yükleme**

- Aynı isme sahip fonksiyonun aldığı parametrelere göre farklı şekilde çalışmasıdır.

```
float aylıkMaas(float mesaiSaati, float saatUcreti)
```

Metot imzası

```
float aylıkMaas(float mesai, float saatUcreti);  
float aylıkMaas(float maas);  
float aylıkMaas();
```

- Metotlar aşırı yüklenerek aynı isimde birden fazla metot oluşturulabilir.
- İsimler aynı olduğuna göre hangi metodu çağırdığımız nasıl anlaşılacaktır?
 - **Metot İmzası (method signature)**
- **Metot imzası;** bir metodun karakteristik özelliklerini içeren bir bilgidir. Metodun adı, parametrelerinin sayısı ve türleri ile alakalıdır. **Geri dönüş değeri metodun imzasına dahil değildir.**

Metot-Örnek: Aşırı Yükleme

```
1 reference
static void Metot1(int x, int y)
{
    Console.WriteLine("1. metot çağrıldı.");
}

1 reference
static void Metot1(float x, float y)
{
    Console.WriteLine("2. metot çağrıldı.");
}

1 reference
static void Metot1(string x, string y)
{
    Console.WriteLine("3. metot çağrıldı.");
}

0 references
static void Main()
{
    Metot1("deneme", "deneme");
    Metot1(5, 6);
    Metot1(10f, 56f);

    Console.ReadLine();
}
```

```
4 references
private static int EnBuyuk(int sayi1, int sayi2)
{
    if (sayi1 > sayi2)
        return sayi1;
    else
        return sayi2;
}

2 references
private static int EnBuyuk(int sayi1, int sayi2, int sayi3)
{
    int buyuk = EnBuyuk(sayi1, sayi2);
    return EnBuyuk(buyuk, sayi3);
}

1 reference
private static int EnBuyuk(int sayi1, int sayi2, int sayi3, int sayi4)
{
    int buyuk = EnBuyuk(sayi1, sayi2, sayi3);
    return EnBuyuk(buyuk, sayi4);
}
```

Değişken Sayıda Parametre Alan Metotlar

- Bazı durumlarda metotlara göndereceğimiz parametrelerin sayısı belli olmayabilir.
- ***params*** anahtar sözcüğü kullanılır.
- Params anahtar sözcüğü değişken sayıda eleman içerebilen bir veri yapısı tanımlar. Metoda gelen her bir parametre bu dizinin bir elemanı olacak şekilde ayarlanır.

Metot-Ornek: Değişken sayıda parametre alabilen Topla() metodu

Ekran

```
0  
10  
30  
60  
100  
150
```

6 references

```
static int Toplam(params int[] sayilar)  
{  
    if (sayilar.Length == 0)  
        return 0;  
    int toplam = 0;  
    foreach (int sayi in sayilar)  
        toplam += sayi;  
    return toplam;  
}
```

0 references

```
static void Main()  
{  
    Console.WriteLine(Toplam());  
    Console.WriteLine(Toplam(10));  
    Console.WriteLine(Toplam(10,20));  
    Console.WriteLine(Toplam(10,20,30));  
    Console.WriteLine(Toplam(10,20,30,40));  
    Console.WriteLine(Toplam(10, 20, 30, 40,50));  
  
    Console.ReadLine();  
}
```


Metot-Ornek:
Değişken sayıda
parametre
alabilen Ciz()
metodu.

Ekran

```
0 |  
1 |  
2 |  
3 |  
4 |  
kelime - 5 - 12,5 - * - 26,5 -
```

2 references

```
static void Ciz(int yon,params object[] nesne)  
{  
    if (nesne.Length == 0)  
        Console.WriteLine("Sembol girilmedi");  
    else  
    {  
        if (yon == 0)  
            foreach (object o in nesne)  
                Console.Write(o.ToString() + " - ");  
  
        else  
            foreach (object o in nesne)  
                Console.WriteLine(o.ToString()+" | ");  
    }  
}
```

0 references

```
static void Main()  
{  
    Ciz(1, 0, 1, 2, 3, 4);  
    Ciz(0, "kelime", 5, 12.5f, '*', 26.5);  
  
    Console.ReadLine();  
}
```

!

- 
- Console.WriteLine() metodunun da istenilen sayıda parametre alabildiğine dikkat ediniz.

```
Console.WriteLine("{0} - {1} - {2} - {3}", "C#", "Programlama", 2020, "Bahar");
```

```
C# - Programlama - 2020 - Bahar
```

Opsiyonel(Optional) Parametreler

- Metot bildirimi sırasında istenilen parametrelere varsayılan değerler verilerek ilgili parametrenin gönderilmediği durumlarda metot içinde alacağı değer belirtilmektedir.

```
2 references
static void FiyatYaz(int gerekliFiyat, string opsiyonelBirim= "TL")
{
    Console.WriteLine("Ödenmesi gereken: {0} {1} ", gerekliFiyat, opsiyonelBirim);
}
0 references
static void Main()
{
    FiyatYaz(20);
    FiyatYaz(5, "Dolar");

    Console.ReadLine();
}
```

```
Ödenmesi gereken: 20 TL
Ödenmesi gereken: 5 Dolar
```

İsimlendirilmiş(Named) Parametreler

- Metot çağrımında parametre sırasından bağımsız olarak parametre adı belirtilerek parametreler metoda gönderilebilir.

```
static void Main()
{
    //Metod parametrelerin tanımlandığı sırada normal olarak çağrılabilir.
    SiparisDetay("Kotan Cafe", 25, "Filtre Kahve");

    //Named argümanlar istenilen sırada parametrelere gönderilebilir.
    SiparisDetay(siparisNo:25,urunAdi:"Filtre Kahve",satıcı:"Kotan Cafe");
    SiparisDetay(urunAdi: "Filtre Kahve", satıcı: "Kotan Cafe",siparisNo: 25);

    //Doğru sıralamada kullanılırsa named argümanlar named olmayan argümanlar ile
    //birlikte kullanılabilir.
    SiparisDetay("Kotan Cafe", 25, urunAdi:"Filtre Kahve");
    SiparisDetay(satıcı:"Kotan Cafe", 25, urunAdi: "Filtre Kahve");

    //Sıralama dışında kullanılırsa karmaşık argümanlar geçersiz olur.
    //Aşağıdaki örnekler derleyici hatasına neden olacaktır.
    SiparisDetay(urunAdi:"Filtre Kahve",25,"Kotan Cafe");
    SiparisDetay(25,satıcı:"Kotan Cafe","Filtre Kahve");

    Console.ReadLine();
}

7 references
static void SiparisDetay(string satıcı, int siparisNo, string urunAdi)
{
    Boş veya boşluk, tab gibi boşluk içeren ifadeler
    if (string.IsNullOrEmpty(satıcı))
    {
        Console.WriteLine("Satıcı adı girilmelidir");
        return;
    }

    Console.WriteLine($"Satıcı: {satıcı}, Sipariş #: {siparisNo}, Ürün: {urunAdi}");
}
```

```
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
Satıcı: Kotan Cafe, Sipariş #: 25, Ürün: Filtre Kahve
```

Özyineli(Recursive) Metotlar

- Metodun kendi kendini çağırmasıdır.
- *Klasik bir örnek: Faktöriyel Hesaplama*

5 references

```
static int faktoriyel(int sayi)
{
    if (sayi == 0)
        return 1;

    return sayi * faktoriyel(sayi - 1);
}
```

0 references

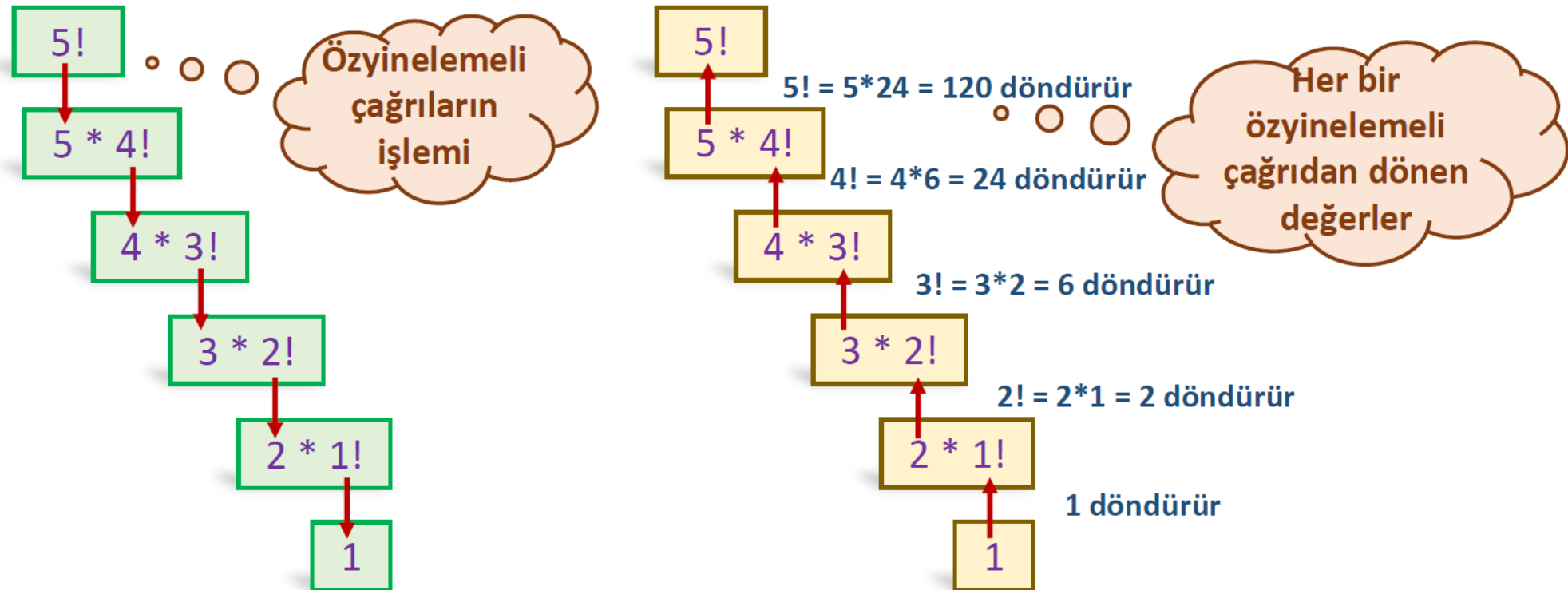
```
static void Main()
{
    Console.WriteLine(faktoriyel(0));
    Console.WriteLine(faktoriyel(1));
    Console.WriteLine(faktoriyel(5));
    Console.WriteLine(faktoriyel(8));

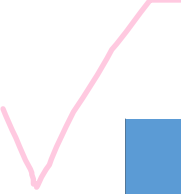
    Console.ReadLine();
}
```

```
1
1
120
40320
```

Özyineli Metotlar

➤ Faktöriyel hesabı $\Rightarrow n! = n * (n-1)!$





System.Math Sınıfı ve Metotları

- Belirli matematiksel işlemleri yapmak için Sistem isim alanında bulunan Math sınıfı kullanılabilir.
- Math sınıfı metotları statik olarak tanımlandığı için herhangi bir nesne tanımlaması yapmadan kullanabiliriz.

System.Math

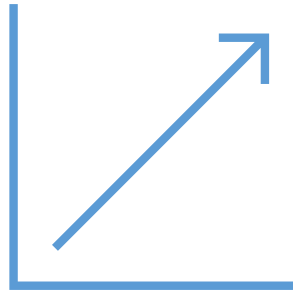
Kullanım	Açıklama
Math.PI	pi=3,14159265358979
Math.E	e=2,71828182845905
Abs(x)	Mutlak değer alma
Cos(x)	Kosinüs alma
Sin(x)	Sinüs alma
Tan(x)	Tanjant alma
Ceiling(x)	X sayısını x'ten büyük en küçük tamsayıya yuvarla
Floor(x)	X sayısını x'ten küçük en büyük tamsayıya yuvarla
Max(x,y)	X ve y sayısından en büyüğü
Min(x,y)	X ve y sayısından en küçüğü
Pow(x,y)	X üzeri y hesapla
Sqrt(x)	Karekök hesapla
Log(x)	X sayısının e tabanında logaritması
Exp(x)	E üzeri x hesapla
Log10(x)	X sayısının 10 tabanında logaritması



Çalışma Sorusu

Bir önceki slayttaki tüm Math sınıfı metotları için örnek yapalım.

Örnek Sorular



Örnek 1: Değişken sayıda parametre alan ve kendisine gönderilen bütün sayıların karesini hesaplayıp bir diziye aktararak bu dizinin referansını geri döndüren bir metod yazınız.



Örnek 2: Fibonacci sayı dizisini kendisine gönderilen değer için hesaplayan bir programı özyineli(recursive) metod kullanarak yazınız. 0'dan 10'a kadar olan sayıların Fibonacci değerlerini ekrana yazdırınız.

Fibonacci sayı dizisi: 0 1 1 2 3 5 8 13 21 34 ...

Referanslar

- Sefer Algan her yönüyle C#
- BTK Akademi C#-Engin Demiroğ
- <https://www.c-sharpcorner.com/blogs/data-types-in-c-sharp>
- Ü.Kocabıçak,C.Öz,N.Taşbaşı,S.İlyas Ders Notları
- Images:
- https://tr.wikipedia.org/wiki/C_Sharp