

Bir önceki bölümde bir metodu nasıl oluşturabileceğimizi görmüştük. Şimdi ise bir argüman ya da parametre alan ve geriye değer döndüren metotları inceleyeceğiz. Bu bölüme özellikle özen göstermeniz gerekmektedir. Bu bölüm sizlerin kaliteli bir yazılımcı olmanız konusunda size oldukça büyük önem katacaktır.

METODLAR

2. BÖLÜM

Argüman Alan ve Geriye Değer
Döndüren Metodlar

Oğuzhan Karagüzel

İçindekiler

ŞEKİLLER DİZİNİ.....	2
TABLOLAR DİZİNİ.....	3
GİRİŞ	4
2. BÖLÜM PARAMETRELER VE GERİ DÖNÜŞLER	4
PARAMETRESİZ METODLARA KISA BİR BAKIŞ	4
Metot Özetleri (Method Summaries)	5
PARAMETRE ALAN METODLAR.....	6
Metotlara Parametre (Argüman) Yollama	8
GERİYE DEĞER DÖNDÜREN METOTLAR.....	12
Local ve Global Değişkenler	13
Geriye Döndürülecek Değeri Belirleme	15
ÖDEV.....	16

ŞEKİLLER DİZİNİ

Şekil 1 İki sayıyı toplayan program.....	4
Şekil 2 ToplamaProgramı metodu.....	5
Şekil 3 Yapay zeka ile metot adlandırma ve özet yazma.....	6
Şekil 4 Metot özeti görüntüleme kısa yolu	6
Şekil 5 Parametre alan metot	7
Şekil 6 AddTwoNumber metodu düzenlenmiş hali.	8
Şekil 7 Argümanların sıra ile gönderilmesi	9
Şekil 8 Hatalı Metod Kullanımı.....	9
Şekil 9 Parametre isimleri kullanılarak metoda argüman yollama	10
Şekil 10 Parametre sırası değiştirerek argüman yollama.....	10
Şekil 11 Konumsal ve isimlendirilmiş parametrelerin karmaşık kullanımı.....	10
Şekil 12 Hatalı Karmaşık Kullanım.....	11
Şekil 13 Varsayılan değerli parametreler	11
Şekil 14 Varsayılan değer karışık kullanım.	11
Şekil 15 Local ve Globa Değişkenler.....	13
Şekil 16 Main içerisinde metot yazımı	13
Şekil 17 Local ve global değişken hatası.	13
Şekil 18 Geriye değer döndüren metot	15

TABLolar DİZİNİ

Tablo 1 Metot Argüman Özeti	12
-----------------------------------	----

Öğuzhan KARAGÜZEL

GİRİŞ

Metotlar ya da (fonksiyonlar) programlama dünyasının olmazsa olmaz elemanlarıdır. Tekrar tekrar kod yazmaktan bizi kurtarır. Bir kod bloğunu daha anlaşılabilir kılar. Hatta bir kod bloğuna resmen ad vermiş oluruz ve bu işi yapar diye tanımlamış oluruz. Bundan dolayı metotlar konusu oldukça önemlidir.

Ancak metotlar konusu oldukça geniş kapsamlıdır. Bundan dolayı bu konu size 4 ana bölümde anlatılacaktır. Bu konuya pür dikkat yaklaşmanızı tavsiye ediyorum. Bu konuyu iyi öğrenmek yazılım dünyasına atılacak en önemli adımlardan birisidir.

2. BÖLÜM PARAMETRELER VE GERİ DÖNÜŞLER

1. bölümde bir metodu nasıl tanımlayabileceğimizi öğrendik. Şimdi ise parametrelili ve geriye değer döndürebilen metotları inceleyeceğiz.

Şunu unutmayın. Metotların gövdeleri her zaman ki kod yazdığını alandır. Farklı hiçbir şey yoktur. Ancak metot içerisinde ki kodların metot dışındaki kodlardan bağımsız olduğunu unutmayın. Ne diyoruz. Hadi bakalım;

PARAMETRESİZ METODLARA KISA BİR BAKIŞ

İlk olarak iki sayıyı toplayan bir program yazalım.

```
static void Main()
{
    Console.WriteLine("İlk sayıyı giriniz");
    string ilkRakam = Console.ReadLine();

    Console.WriteLine("İkinci sayıyı giriniz");
    string ikinciRakam = Console.ReadLine();

    int ilkSayi = Convert.ToInt32(ilkRakam);
    int ikinciSayi = Convert.ToInt32(ikinciRakam);

    int toplam = ilkSayi + ikinciSayi;
    Console.WriteLine($"İki sayının toplamı: {toplam}");

    Console.ReadLine();
}
```

Şekil 1 İki sayıyı toplayan program

Şimdi bu kod bloğu gördüğünüz üzere “main” metodunun içerisinde yazılmış durumda. İstersek benzer şekilde başka bir metodun içerisine taşıyıp main içerisinde metodumuzu çalıştırabiliriz. Hadi buna “ToplamaProgramı” diye yeni bir metot açalım ve orada çağıralım.

```

static void Main()
{
    ToplamaProgrami();
}
static void ToplamaProgrami()
{
    Console.WriteLine("İlk sayıyı giriniz");
    string ilkRakam = Console.ReadLine();

    Console.WriteLine("İkinci sayıyı giriniz");
    string ikinciRakam = Console.ReadLine();

    int ilkSayi = Convert.ToInt32(ilkRakam);
    int ikinciSayi = Convert.ToInt32(ikinciRakam);

    int toplam = ilkSayi + ikinciSayi;
    Console.WriteLine($"İki sayının toplamı: {toplam}");

    Console.ReadLine();
}

```

Şekil 2 ToplamaProgrami metodu

Gördüğünüz üzere programı çalıştırdığınızda aynı işlemi yapar. Sadece kodlarımızı başka bir metoda taşımış olduk. Metoda işlevine uygun bir ad vererek metodumuzun ne iş yaptığını diğer yazılımcılara anlatmış olduk. Hazır bu konuya değinmişken metot özetlerine bir bakış atalım.

Metot Özetleri (Method Summaries)

Yorum satırlarının anlatıldığı derste size xml yorum satırlarından “///” bahsetmiştim. O zaman henüz metotları bilmiyordunuz. Şimdi ise öğrenmenin tam zamanı. Yazılımda ilerledikçe şuna ihtiyaç duyacaksınız. “Bu projeye doküman lazım”. İşte bu dokümanı oluşturmanın en güzel yollarından biriside metod özetleridir.

Metodunuzun üzerine “///” yazdığınıza visual studio size otomatik olarak özet şablonunu çıkaracaktır.

Çıkan yorum satırlarına metodunuzun ne yaptığını kısaca yazmanız beklenir.

Peki ne yazacağınızı bilmiyor ya da kafanızda toplayamıyor musunuz? Ya da metodunuzu düzgün isimlendirdiğinizi düşünmüyor musunuz? Bu durumda metodunuzu yapay zekaya verin. Ondan bir özet yazmasını ve düzgün adlandırmasını isteyin. Şekil 4’teki metodu ben chatgpt’ye verdim. Ondan özet yazmasını ve metodu düzgün isimlendirmesini istedim.

```

static void Main()
{
    AddTwoNumbers();
}
/// <summary>
/// Prompts the user to enter two numbers, calculates their sum, and displays the result.
/// </summary>
static void AddTwoNumbers()
{
    Console.WriteLine("Enter the first number:");
    string firstInput = Console.ReadLine();

    Console.WriteLine("Enter the second number:");
    string secondInput = Console.ReadLine();

    int firstNumber = Convert.ToInt32(firstInput);
    int secondNumber = Convert.ToInt32(secondInput);

    int sum = firstNumber + secondNumber;
    Console.WriteLine($"The sum of the two numbers is: {sum}");

    Console.ReadLine();
}

```

Şekil 3 Yapay zeka ile metod adlandırma ve özet yazma.

Gördüğünüz üzere chatgpt bizim için metodu güzelce adlandırdı ve özet yazdı. Sizde aynı şekilde metodlarınızı adlandırabilir ve özet yazabilirsiniz.

Peki bu ne işimize yarayacak? Sizden başka bir yazılımcı eğer sizinle aynı projede çalışıyorsa metodunuzun ne iş yaptığını çok daha hızlı ve kolay anlayacak. Ayrıca eğer metodunuzun ne işe yaradığını unutmak yazılımcıların yaşadığı sık bir durumdur. Bu durumun önüne geçmek için metodlarınıza özet yazmak faydanıza olacaktır.

Ayrıca

```

static void Main()
{
    AddTwoNumbers();
}
/// <summary>
/// Prompts the
/// </summary>
static void AddTwoNumbers()

```

void Program.AddTwoNumbers()
Prompts the user to enter two numbers, calculates their sum, and displays the result.
Describe with Copilot

Şekil 4 Metod özeti görüntüleme kısa yolu

Şekil 4'teki gibi metodunuzu çağırdığınız yerde Mouse imleci ile üzerine geldiğinizde, açılan küçük pencerede metodunuzun özetini görebilirsiniz.

Şimdi asıl konumuza dönelim.

PARAMETRE ALAN METODLAR

SOLID kavramını ileride öğreneceksiniz. İsterseniz şimdiden araştırabilirsiniz. Ancak ne olduğunu pek anlamayacaksınız. Bunun için nesne tabanlı programlamayı iyi derecede bilmeniz ve uygulamanız gerekmektedir.

Nesne tabanlı programlamanın uyulması gereken ya da uyulduğunda kalite bir program yazılmasını sağlayan 5 temel kuralı vardır. Bu kuralların baş harfleri birleştirilerek SOLID kavramı oluşturulmuştur.

Biz bu kuralların ilki olan “S” **Single Responsibility Principle (SRP)** yani **Tek Sorumluluk Prensipli** olarak adlandırılır. Bu kural bir metodun sadece bir iş yapması gerektiğini söyler. Eğer bir metod birden fazla iş yapıyor ise bunu parçalamamız gerekmektedir.

Şu an için metodlarınızı nasıl parçalamanız gerektiği konusunda bir fikrinizin olmadığını biliyorum. Ondan dolayı basitçe başlayalım. “AddTwoNumbers” adlı metodumuzu inceleyelim. Bu metod;

1. Kullanıcıdan sayıları istiyor.
2. Aldığı sayıları “int” değişken türüne çeviriyor.
3. Ardından sayıları topluyor.
4. Toplamı ekrana yazdırıyor.

Gördüğünüz üzere metodumuzu okumaya çalıştığımızda 4 işlemi birden yapmaya çalıştığını görebilirsiniz. Sizde bu şekilde metodlarınızın ne iş yaptığını anlamlı ifadeler ile sıralayarak anlayabilirsiniz. Zaten birkaç temel anlama yönteminden birisidir.

Peki şu anda ki bilgi seviyesi ile bu metodu nasıl parçalarız. Kullanıcıdan sayıları isteme işlemi eğer metodun dışında yaparsak metod bu sayıyı nasıl okuyacak? İşte burada devreye metod parametreleri devreye giriyor.

“()” ifadesi yapımızın bir metod olduğunu söyler. Bu parantez içerisine değişken adı ve türü vererek her dışarıdan bu değişkenleri isteriz hem de bu değişkenleri metod içerisinde kullanabiliriz.

Kısaca gösterirsek.

```
static void Main()  
{  
    string firstMessage = "Hello World!"; // metoda yollanan string tipindeki değişken dışarıda oluşturulabilir.  
    Metot(firstMessage);  
    Metot("Merhaba Dünya!"); // metoda yollanan string tipindeki değişken doğrudan metod içerisinde oluşturulabilir.  
}  
static void Metot(string message) // metod içerisine alınan string tipindeki message değişkeni ile metod içerisinde işlem yapılabilir.  
{  
    message += " \nBu mesaj metod aracılığı ile yazdırılmıştır"; // message değişkenine yeni bir değer atanabilir.  
    Console.WriteLine(message); // message değişkeni ekrana yazdırılabilir.  
}
```

Şekil 5 Parametre alan metod

Gördüğünüz üzere “Metot” adlı metodumuz dışarıdan değişken olarak mesajı ekrana yazdırır. Bu programı kendiniz de yazarak test edebilirsiniz.

Şekil 5'te ki yorum satırlarını dikkatlice okuyun. Metodunuza göndereceğiniz parametreyi oluşturup sonra da yollayabilirsiniz. Doğrudan metodu çağırdığınız “()” içerisinde de oluşturabilirsiniz. Şimdi bu bilgi ile asıl metodumuzu parçalamayı deneyelim.

Metodumuz toplayacağı iki sayıyı dışarıdan alsın. Bunları toplansın. Ardından kullanıcıya mesaj versin. Aklınızdaki soruyu duyabiliyorum. Bu sayıyı neden metodun içerisinde ekrana yazdırıyoruz? Buna geri dönüş tiplerinde değineceğiz ve bu sorunu da orada çözeceğiz.

Hadi metodumuzu düzenleyelim.


```

static void Main()
{
    Console.WriteLine("Enter the first number:");
    string firstInput = Console.ReadLine();

    Console.WriteLine("Enter the second number:");
    string secondInput = Console.ReadLine();

    int firstNumber = Convert.ToInt32(firstInput);
    int secondNumber = Convert.ToInt32(secondInput);

    AddTwoNumbers(firstNumber, secondNumber);

    Console.ReadLine();
}

/// <summary>
/// Prompts the user to enter two numbers, calculates their sum, and displays the result.
/// </summary>
static void AddTwoNumbers(int firstNum, int secondNum)
{
    int sum = firstNum + secondNum;
    Console.WriteLine($"The sum of the two numbers is: {sum}");
}

```

Şekil 6 AddTwoNumber metodu düzenlenmiş hali.

Gördüğünüz üzere metodumuz artık neredeyse tek bir iş yapar hale geldi. Hâlâ daha fazlalık olsa da büyük bir kısmını temizlemiş bulunuyoruz. Programınızı çalıştırdığınızda ilk seferki gibi sonucu elde edebildiğinizi görebilirsiniz.

Parametre alan metotları öğrendiğinizi düşünüyorum. Bu metodumuzda hala daha ekrana yazı yazdırdığımız için tamamen tek göreve düşürebilmiş değiliz. Bunu yapabilmek için şimdi geri dönüşü öğrenmemiz gerekmektedir. Ancak bundan önce metotlara parametre yollamayı inceleyelim.

Metotlara Parametre (Argüman) Yollama

Metotlar, belirli bir işlemi gerçekleştirmek için çağrıldığında, **dışarıdan argüman (argument)** alabilir. Argümanlar, metot içerisinde kullanılacak **değerleri dışarıdan iletmek** için kullanılır. C#'ta metotlara argüman göndermenin birkaç yolu vardır. Bunlardan en yaygın olanları **konumsal (positional) argümanlar** ve **isimlendirilmiş (named) argümanlardır**.

1. Konumsal Argümanlar (Positional Arguments)

Konumsal argümanlar, **metot çağrılırken** parametrelerin **tanımlandıkları sıraya göre** argümanların verilmesini gerektirir. Yani, metot **hangi sırayla parametre aldıysa, argümanlar da aynı sırayla gönderilmelidir**.

Örnek:

```
static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
}
```

Şekil 7 Argümanların sıra ile gönderilmesi

Çıktı:

İsim: Ahmet, Yaş: 25, Şehir: İstanbul

Burada isim → Ahmet, yaş → 25, şehir → İstanbul olacak şekilde argümanlar tanımlandıkları sırayla metoda gönderildi.

Konumsal Argümanlarda Dikkat Edilmesi Gerekenler

- Argümanlar, parametre sırasına uygun olarak verilmelidir.
- Veri türleri uyumlu olmalıdır. Örneğin, yas yerine yanlışlıkla "Ankara" gibi bir string değer verilirse derleme hatası alınır.

Yanlış Kullanım:

```
static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
    KullaniciBilgisi(25, "Ahmet", "İstanbul"); // Hata! int yerine string bekleniyor.
}
```

Şekil 8 Hatalı Metod Kullanımı

Bu kod hata verecektir çünkü metot (string, int, string) formatında parametre beklerken, burada (int, string, string) verilmiştir.

2. İsimlendirilmiş Argümanlar (Named Arguments)

İsimlendirilmiş argümanlar, metot çağrılırken hangi parametreye hangi değer gideceğini açıkça belirlememizi sağlar. Sıra önemli değildir çünkü argümanlar, parametre adları ile eşleştirilerek gönderilir.

Örnek:

```

static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
    KullaniciBilgisi(25, "Ahmet", "İstanbul"); // Hata! int yerine string bekleniyor.
    KullaniciBilgisi(isim: "Ahmet", yas: 25, sehir: "İstanbul"); //Parametre isimleri belirtilerek gönderildi.
}

```

Şekil 9 Parametre isimleri kullanılarak metoda argüman yollama

Çıktı:

İsim: Ahmet, Yaş: 25, Şehir: İstanbul

Burada, parametrelerin sırası **değiştirilse bile** hata alınmaz. Çünkü her argüman, ilgili parametre adıyla eşleştirilmiştir.

Örneğin, sırayı **değiştirebiliriz**:

```

static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
    KullaniciBilgisi(25, "Ahmet", "İstanbul"); // Hata! int yerine string bekleniyor.
    KullaniciBilgisi(isim: "Ahmet", yas: 25, sehir: "İstanbul"); //Parametre isimleri belirtilerek gönderildi.
    KullaniciBilgisi(sehir: "İstanbul", yas: 25, isim: "Ahmet"); //Parametre isimleri belirtilerek gönderildi. Sıra değişti.
}

```

Şekil 10 Parametre sırası değiştirerek argüman yollama

Bu **doğru çalışacaktır** çünkü parametrelerin isimleri açıkça belirtilmiştir.

İsmlendirilmiş Argümanların Avantajları

- ✓ Daha okunaklı kod yazmayı sağlar.
- ✓ Uzun parametre listelerinde daha anlaşılır bir çağrı yapılabilir.
- ✓ Konumsal argümanların sırasını koruma zorunluluğunu ortadan kaldırır.

3. Konumsal ve İsmlendirilmiş Argümanları Birlikte Kullanma

Konumsal ve ismlendirilmiş argümanlar **birlikte** kullanılabilir. Ancak **önce konumsal argümanlar, sonra ismlendirilmiş argümanlar** gelmelidir.

Doğru Kullanım:

```

static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
    KullaniciBilgisi(25, "Ahmet", "İstanbul"); // Hata! int yerine string bekleniyor.
    KullaniciBilgisi(isim: "Ahmet", yas: 25, sehir: "İstanbul"); //Parametre isimleri belirtilerek gönderildi.
    KullaniciBilgisi(sehir: "İstanbul", yas: 25, isim: "Ahmet"); //Parametre isimleri belirtilerek gönderildi. Sıra değişti.
    KullaniciBilgisi("Ahmet", yas: 25, sehir: "İstanbul"); //Sadece isim parametresi konumsal olarak gönderildi.
}

```

Şekil 11 Konumsal ve ismlendirilmiş parametrelerin karmaşık kullanımı

Burada ilk parametre konumsal, diğerleri isimlendirilmiş olarak verilmiştir.

Yanlış Kullanım:

```
static void Main()
{
    void KullaniciBilgisi(string isim, int yas, string sehir)
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    KullaniciBilgisi("Ahmet", 25, "İstanbul"); // Argümanlar sırayla gönderildi.
    KullaniciBilgisi(25, "Ahmet", "İstanbul"); // Hata! int yerine string bekleniyor.
    KullaniciBilgisi(isim: "Ahmet", yas: 25, sehir: "İstanbul"); //Parametre isimleri belirtilerek gönderildi.
    KullaniciBilgisi(sehir: "İstanbul", yas: 25, isim: "Ahmet"); //Parametre isimleri belirtilerek gönderildi. Sıra değişti.
    KullaniciBilgisi("Ahmet", yas: 25, sehir: "İstanbul"); //Sadece isim parametresi konumsal olarak gönderildi.
    KullaniciBilgisi(yas: 25, "Ahmet", "İstanbul"); // HATA!
```

Şekil 12 Hatalı Karmaşık Kullanım

Bu kullanım geçerli değildir çünkü konumsal argümanlar isimlendirilmiş argümanlardan sonra kullanılamaz.

4. Varsayılan Parametre Değerleri (Default Parameter Values)

Bazı parametreler için varsayılan değerler belirlenebilir. Eğer metot çağrılırken bu parametreler için bir değer verilmezse, varsayılan değerler kullanılır.

Örnek:

```
static void Main()
{
    void KullaniciBilgisi(string isim, int yas = 18, string sehir = "Bilinmiyor")
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    // Metot çağrılırken eksik parametreler varsa varsayılan değerler kullanılır.
    KullaniciBilgisi("Mehmet");
```

Şekil 13 Varsayılan değerli parametreler

Çıktı:

İsim: Mehmet, Yaş: 18, Şehir: Bilinmiyor

Burada yas ve sehir parametreleri için varsayılan değerler atanmış olduğu için, çağrıda sadece isim verilmesine rağmen hata alınmadı.

İsimlendirilmiş Argümanlarla Kullanımı

```
static void Main()
{
    void KullaniciBilgisi(string isim, int yas = 18, string sehir = "Bilinmiyor")
    {
        Console.WriteLine($"İsim: {isim}, Yaş: {yas}, Şehir: {sehir}");
    }

    // Metot çağrılırken eksik parametreler varsa varsayılan değerler kullanılır.
    KullaniciBilgisi("Mehmet");
    KullaniciBilgisi("Mehmet", sehir: "İzmir"); // isimlendirilmiş argüman ile karışık kullanım!
```

Şekil 14 Varsayılan değer karışık kullanım.

Çıktı:

İsim: Mehmet, Yaş: 18, Şehir: İzmir

Burada yas değeri verilmediği için varsayılan olan **18** kullanıldı, ancak şehir açıkça belirtildiği için "İzmir" olarak atandı.

Tablo 1 Metot Argüman Özeti

Argüman Türü	Açıklama	Avantajları
Konumsal Argümanlar (Positional Arguments)	Argümanlar, parametrelerin tanımlandığı sırayla iletilmelidir.	Hızlı ve basittir.
İsimlendirilmiş Argümanlar (Named Arguments)	Argümanlar, parametre isimleri belirtilerek iletilir.	Daha okunaklı ve esnektir.
Konumsal + İsimlendirilmiş Kullanım	Önce konumsal , sonra isimlendirilmiş argümanlar kullanılabilir.	Daha okunaklı hale getirilir.
Varsayılan Parametreler (Default Parameter Values)	Belirli parametreler için varsayılan bir değer tanımlanabilir.	Opsiyonel parametreler tanımlamak mümkündür.

Sonuç

- **Kısa ve basit metot çağrıları için konumsal argümanlar** kullanabilirsiniz.
- **Daha okunaklı ve anlaşılır bir yapı için isimlendirilmiş argümanlar** tercih edebilirsiniz.
- **Bazı parametrelerin opsiyonel olmasını istiyorsanız varsayılan değerler** kullanabilirsiniz.
- **Konumsal ve isimlendirilmiş argümanları birleştirmek, hem hızlı hem de esnek bir yapı sunar.**

Bu konular, **kod okunabilirliğini artırma** ve **hata ihtimalini azaltma** açısından oldukça önemlidir.

Bu konuyu anladığınızı düşünüyorum. Şimdi asıl konumuza dönelim ve değer geriye döndüren metotlara bakalım.

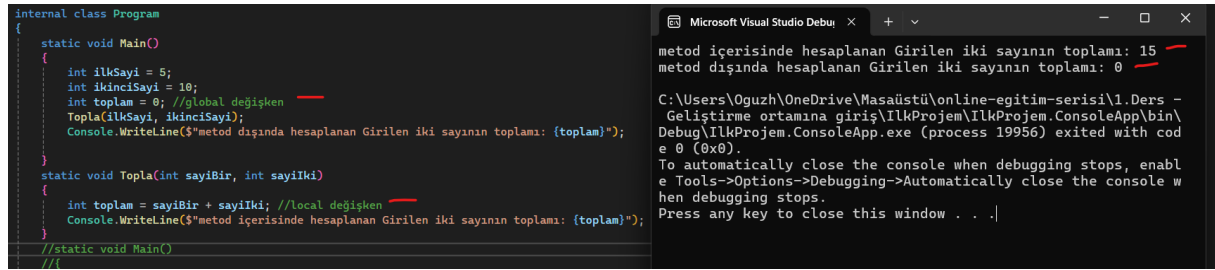
GERİYE DEĞER DÖNDÜREN METOTLAR

Gördüğünüz üzere metodumuz hala daha iki iş yapmaktadır. Bunda bir sakınca yoktur. Ayağınıza dolanmadığı sürece. Ancak ne kadar yalın hale getirirsek. Ya da sorumluluğunu tanımlayıp tek sorumlu hale getirirsek o kadar iyi olur.

Metodumuz iki sayıyı toplamakta ve ekrana yazdırmaktadır. Ekrana yazdırma işini biz yapsak ve metodumuz sadece iki sayıyı toplasa nasıl olur? Harika öyle değil mi? Ancak bu topladığı sayıya nasıl ulaşacağız? İşte sorun burada başlıyor.

Bu konuya başlamadan önce size local ve global değişkenlerden kısaca bahsetmem gerekiyor.

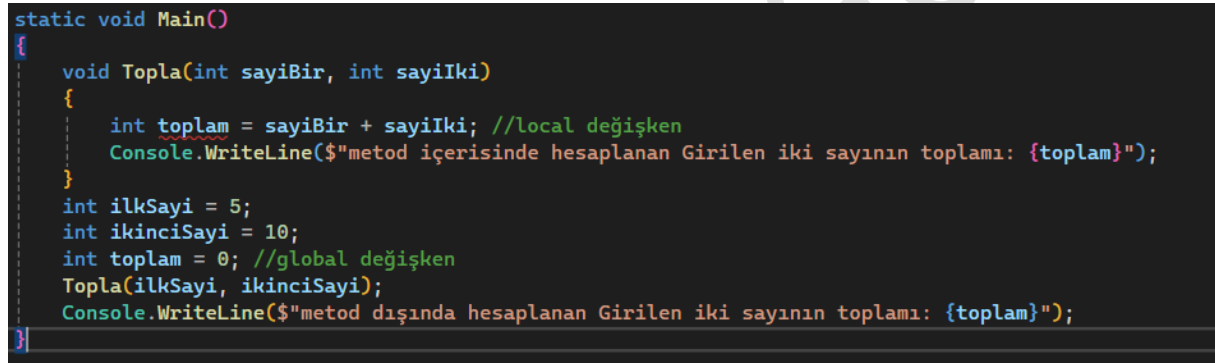
Local ve Global Değişkenler



Şekil 15 Local ve Globa Değişkenler

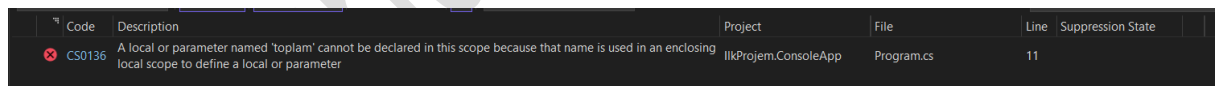
Şekil 15'te göreceğiniz üzere hem ana metodumuz olan main içerisinde bir toplam değeri var. Ayrıca Topla metodunun içerisinde de toplam değeri var. Farkındaysanız her süslü parantez içerisindeki değişkenler o parantez içerisinde kalır. Dışarıyı etkilemez ya da dışarıdan etkilenmez. Yani her metod kendi içerisinde işlem yapar.

Ancak istersek topla metodunu main içerisinde de tanımlayabiliriz. Bu durumda nasıl bir davranış göreceğiz?



Şekil 16 Main içerisinde metod yazımı

Şekil 16'da da göreceğiniz üzere topla metodunun içerisindeki toplam değişkeni hata vermektedir.



Şekil 17 Local ve global değişken hatası.

Hatayı doğrudan chatgpt'ye soralım. Bakalım ne açıklama yapacak.

ChatGpt'nin cevabı

Bu hatanın sebebi, toplam isimli değişkenin hem Main metodu içinde hem de Topla metodu içinde tekrar tanımlanmasıdır.

Hatanın Sebebi:

- Main metodu içinde toplam isimli bir değişken tanımlıyorsunuz:

```
int toplam = 0; // Global değişken
```

- Topla metodu içinde de **aynı isimde** bir değişken tanımlıyorsunuz:

int toplam = sayiBir + sayilki; // Local değişken

- Ancak Topla metodu, Main metodu içinde **yerel (nested) bir fonksiyon** olduğu için **dıştaki toplam değişkenini zaten görüyor**.
- C# derleyicisi, iç içe geçmiş (nested) bir metotta **dıştaki bir değişkenle aynı isimde bir değişken tanımlamaya izin vermez**. Çünkü **hangi değişkenin kullanılacağı konusunda belirsizlik oluşabilir**.

Çözüm Önerileri:

1. **Ya Topla metodu içinde toplam değişkeninin ismini değiştir**

```
static void Main()
{
    void Topla(int sayiBir, int sayilki)
    {
        int sonuc = sayiBir + sayilki; // 'toplam' yerine 'sonuc' kullandık
        Console.WriteLine($"metod içerisinde hesaplanan Girilen iki sayının toplamı: {sonuc}");
    }
    int ilkSayi = 5;
    int ikinciSayi = 10;
    int toplam = 0; // Global değişken
    Topla(ilkSayi, ikinciSayi);
    Console.WriteLine($"metod dışında hesaplanan Girilen iki sayının toplamı: {toplam}");
}
```

2. **Ya da toplam değişkenini Main içinde tanımlamayıp, Topla metodunun sonucu döndürmesini sağla:**

```
static void Main()
{
    int Topla(int sayiBir, int sayilki)
    {
        return sayiBir + sayilki;
    }
    int ilkSayi = 5;
    int ikinciSayi = 10;
```

```
int toplam = Topla(ilkSayi, ikinciSayi); // 'toplam' burada atanıyor

Console.WriteLine($"metod dışında hesaplanan Girilen iki sayının toplamı: {toplam}");

}
```

Sonuç:

C#'ta aynı isimde bir değişkeni iç içe geçmiş bloklarda (nested functions) tanımlayamazsın. Çözüm olarak **değişken ismini değiştirebilir** ya da **metotlardan değer döndürerek dışarıda saklayabilirsin**.

Gördüğünüz üzere yapay zekadan gayet güzel bir cevap aldık. İçe gömülmüş metodlar. Yani bir süslü parantez içerisine yazılmış metodlar o süslü parantezdeki değişkenlere ulaşabilir. Ancak o süslü parantez seviyesinden metodun içerisine ulaşamazsınız. Bunun için ya farklı değişken adı kullanacaksınız ya da metodunuz bambaşka bir değişkeni geriye döndürecek. En güzeli geriye değişken döndürmek. Hadi bakalım neler yapabiliyoruz?

Geriye Döndürülecek Değeri Belirleme

İlk olarak metodumuzun geriye hangi değeri geriye döndüreceğini belirlemeliyiz. Asıl ilgilendiğimiz metod iki adet int değerini toplamaktadır. Bu durumda geriye de bir int değeri döndürmek mantıklıdır.

O zaman kurallarımız şu şekilde.

1. Artık void yerine geriye döndüreğimiz değeri yazacağız.
2. Geriye değeri döndürebilmek için return anahtar kelimesini kullanacağız.
3. Return anahtar kelimesi kullanıldığında metoddan çıkıldığı unutulmamalı. Bir metoddan birden fazla return olabilir. Bunu ileride daha detaylıca inceleyeceğiz.

Bu kurallar altında metodumuzu düzenlersek

```
static void Main()
{
    Console.WriteLine("enter the first number:");
    string firstinput = Console.ReadLine();

    Console.WriteLine("enter the second number:");
    string secondinput = Console.ReadLine();

    int firstnumber = Convert.ToInt32(firstinput);
    int secondnumber = Convert.ToInt32(secondinput);

    int sum = AddTwoNumbers(firstnumber, secondnumber);

    Console.WriteLine($"Answer: {sum}");
    Console.ReadLine();
}

/// <summary>
/// prompts the user to enter two numbers, calculates their sum, and displays the result.
/// </summary>
static int AddTwoNumbers(int firstNum, int secondNum)
{
    int sum = firstNum + secondNum;
    return sum;
}
```

Şekil 18 Geriye değer döndüren metod

Gördüğünüz üzere “AddTwoNumber” metodu lokalinde iki sayıyı topluyor ve sonucunu geriye döndürüyor. Bu sayede metodumuzu her dışarıdan bağımsız hem de tek görev ile sorumlu hale getirmeyi başardık.

Bu bölümü tamamlamış bulunuyoruz. Bir sonraki bölüme hazırlık için “ref, out, params” anahtar kelimelerini araştırmayı unutmayınız.

ÖDEV

1. “void, static, scope {}” kavramlarını araştırıp hakkında kısa bir doküman oluşturunuz.
2. Bir string parametre alan ve aldığı değerin sol tarafına 20 adet boşluk ekleyip geri döndüren bir metod oluşturunuz.
3. İki adet int tipinde parametre alan ve bunları çarpıp sonucu long tipinde geriye döndüren bir metod yazınız.