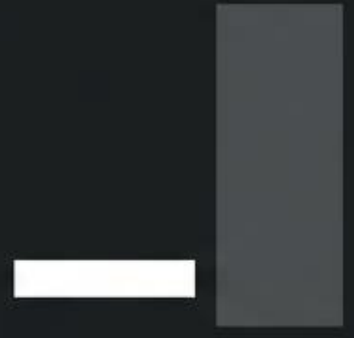
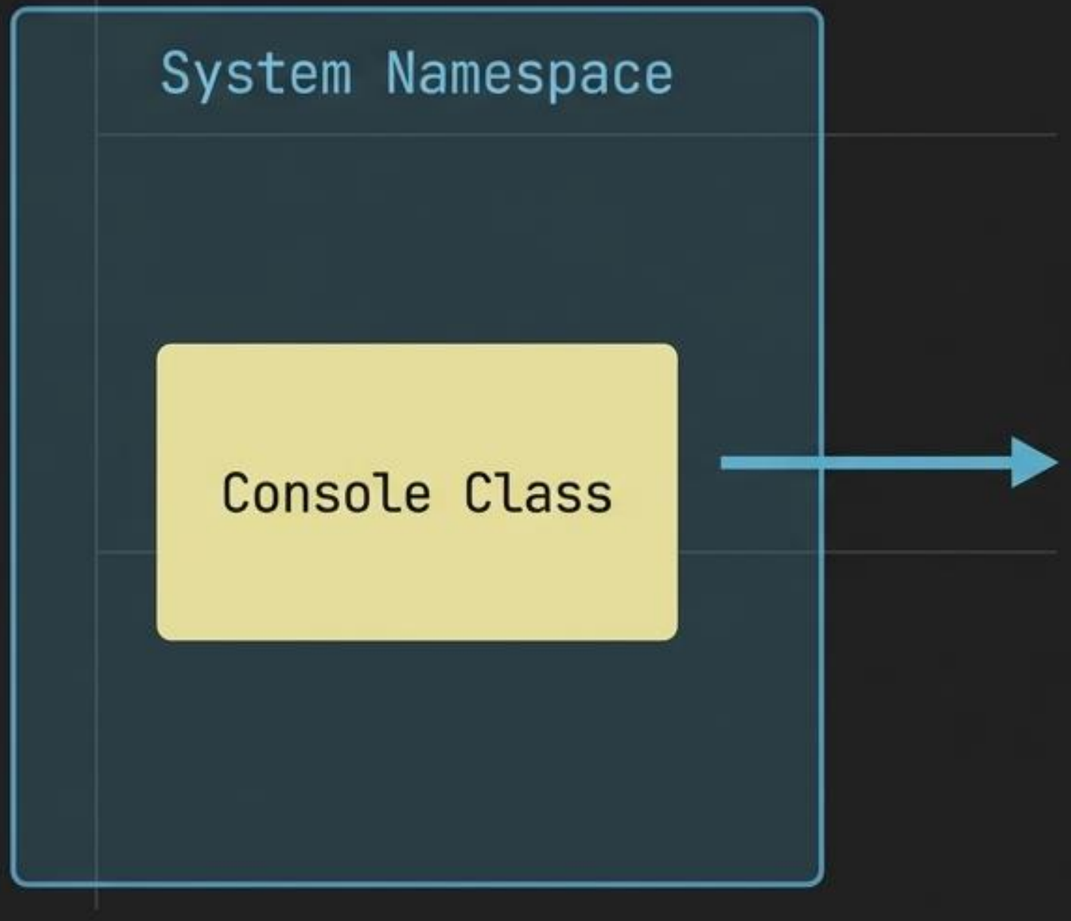
```
using System;
```

C# Konsol Uygulamaları: Dijital Sohbet

Bilgisayar ile İletişim Kurma Sanatı: System.Console



Araç Kutusu: System İsim Uzayı (Namespace)



.NET Framework içerisinde Console sınıfı, System kütüphanesinin (namespace) içinde yaşar.

Kodun en başına using System; ekleyerek, her seferinde System.Console yazma zahmetinden kurtuluruz.

```
using System; // Kütüphaneyi dahil et
namespace DersUygulamasi
{
    class Program
    {
        // ...
    }
}
```

System, C# dilinin temel yapı taşlarını barındıran konteynerdir.

Sessizliği Bozmak: Write vs. WriteLine

Console.Write

```
Console.Write("Merhaba");
Console.Write("Dünya");
```

```
MerhabaDünya
```

Console.WriteLine

```
Console.WriteLine("Merhaba");
Console.WriteLine("Dünya");
```

```
Merhaba ↵
Dünya ↵
```

Write: İmleç aynı satırda bekler. | WriteLine: Satır sonu karakteri (\n) ekler.

Esneklik Gücü: Metot Aşırı Yükleme (Overloading)

`Console.WriteLine` tek bir metot değildir; 19 farklı versiyonu vardır.

Derleyici (Compiler), gönderdiğiniz veri tipine en uygun versiyonu otomatik seçer.

Kod Üzerinde **Overloading**

```
int sayi = 42;
double ondalik = 3.14;
bool durum = true;

// Aynı metot, farklı veri tipleri
Console.WriteLine("Sadece Metin");  // string
Console.WriteLine(sayi);            // int
Console.WriteLine(ondalik);         // double
Console.WriteLine(durum);           // bool
```

C# derleyicisi veri tipini algılar ve doğru versiyonu çalıştırır.

Çıktı Düzenleme: Yer Tutucular (Placeholders)

Karmaşık (Concatenation)

```
Console.WriteLine("Öğrenci: "
+ ad + " Notu: " + not);
```

Zarif (Placeholders)

```
Console.WriteLine("Öğrenci:
{0}, Notu: {1}", ad, not);
```

Metinleri + ile birleştirmek yerine, {0} ve {1} gibi yer tutucular kullanarak formatlı çıktı oluşturmak daha okunaklıdır.

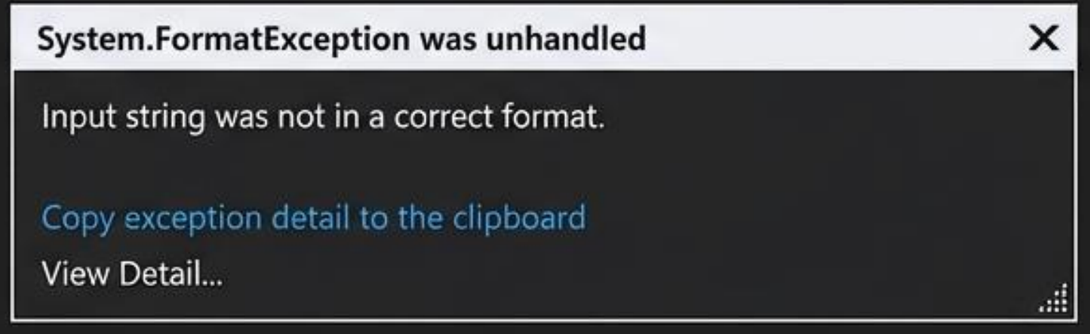
Makine Dinliyor: Console.ReadLine()

```
  Console.Write("Adınızı giriniz: ");
\nstring isim = Console.ReadLine(); // İmleç bekliyor...
```

Programın akışını durdurur ve kullanıcı Enter tuşuna basana kadar bekler.

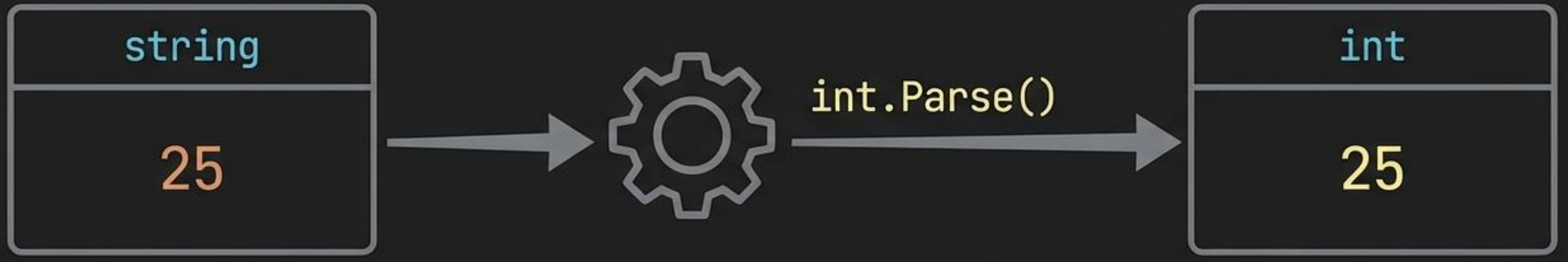
KRİTİK BİLGİ:
Console.ReadLine() sonucu HER ZAMAN string (metin) türünde geri döndürür.

Veri Tipi Çatışması: String vs. Int



Matematiksel işlem yapmak için dönüşüm şarttır.
'50' + 10 = 5010 (Birleştirme), 60 (Toplama) değildir.

Çevirmen: int.Parse() Metodu




```
Console.Write("Yaşınız: ");
string gelenVeri = Console.ReadLine(); // "25" (String)

// Dönüştürme işlemi
int yas = int.Parse(gelenVeri);         // 25 (Int)
int gelecekYas = yas + 1; // Artık matematiksel işlem yapılabilir
```

Gerçek Dünya Verileri: Ondalıklı Sayılar

int
42

double
42.50

```
Console.Write("Ürün fiyatı: ");
// double.Parse kullanımı
double fiyat = double.Parse(Console.ReadLine());
```

Tam sayılar için `int`, kesirli sayılar için `double` kullanılır.
Parse mantığı aynıdır.

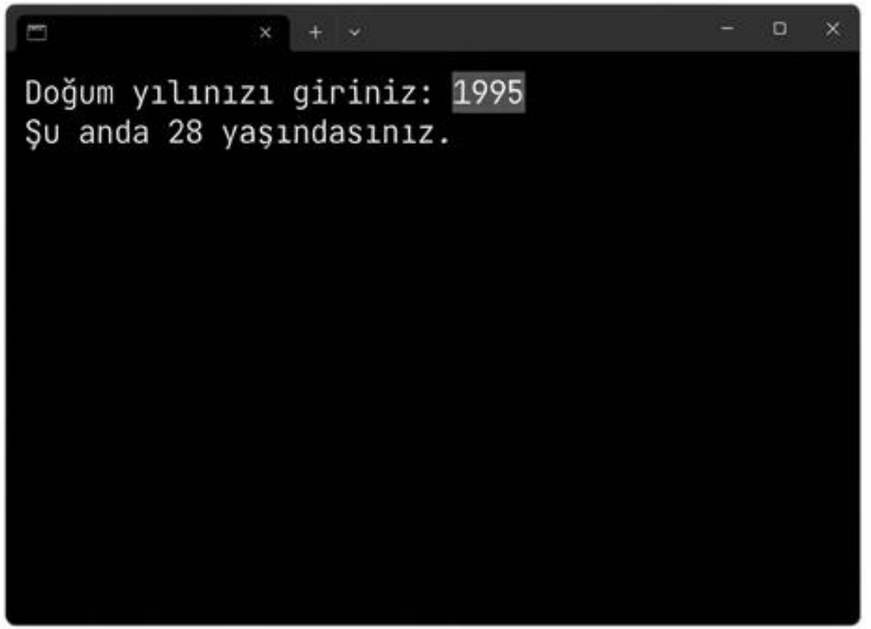
Hepsini Birleştirelim: Örnek Uygulama

```
using System;
class Program {
  static void Main() {
     // 1. Çıktı (Soru)
     Console.Write("Doğum yılınızı giriniz: ");

     // 2. Girdi ve Dönüştürme
     int dogumYili = int.Parse(Console.ReadLine());

     // 3. İşlem
     int yas = 2023 - dogumYili;

     // 4. Formatlı Çıktı
     Console.WriteLine("Şu anda {0} yaşındasınız.", yas);
  }
}
```



Özet ve En İyi Uygulamalar

```
{
    ✓ Kütüphane: Kodun başına using System; ekleyin.

    ✓ Çıktı: Satır sonu için WriteLine, devam etmek için Write kullanın.

    ✓ Format: Okunabilirlik için {0} yer tutucularını tercih edin.

    ✓ Girdi: ReadLine her zaman string döndürür, bunu unutmayın.

    ✓ Dönüşüm: Matematiksel işlemler için int.Parse veya
      double.Parse şarttır.
}
```