</> C# · İleri



C# — İleri Seviye

Generics, async/await, dosya işlemleri ve temiz kodla C#'ta ustalaşma.

SowaCodeHK — Ders Notları
sowacodehk.com.tr · Hazar Kanat

Bu notta ne öğreneceksin? C#'ın ileri konuları: generics (genel tipler), async/await ile asenkron programlama, dosya işlemleri, properties ve modern C# özellikleri, ve temiz kod prensipleri. "C# — Orta" notunu bitirdiysen buraya hazırsın.

Generics (Genel Tipler)

Aynı kodu farklı tiplerle çalıştırmak için generics kullanılır. `List<T>` zaten bir generic'tir. Kendin de yazabilirsin:

```
// T, herhangi bir tip olabilir
class Kutu<T> {
    public T Deger;
    public void Goster() {
        Console.WriteLine("Değer: " + Deger);
    }
}

Kutu<int> sayiKutusu = new Kutu<int> { Deger = 42 };
Kutu<string> metinKutusu = new Kutu<string> { Deger = "Merhaba" };
```

> **İpucu:** Generics sayesinde tip güvenliğini korurken kod tekrarından kurtulursun. "Her tip için ayrı sınıf yazma" derdi biter.

Async / Await — Asenkron Programlama

Uzun süren işlemler (veritabanı, dosya, internet) programı kilitlemesin diye asenkron çalıştırılır. Modern uygulamaların temelidir:

```
// Asenkron metot: async ile işaretlenir, Task döndürür
async Task<string> VeriGetir() {
    // await: sonucu bekle ama programı kilitleme
    await Task.Delay(2000);   // 2 saniye bekliyormuş gibi
    return "Veri hazır!";
}

// Kullanımı
string sonuc = await VeriGetir();
Console.WriteLine(sonuc);
```

- `async` — Metodun asenkron olduğunu belirtir.
- `await` — İşlemin bitmesini bekler, ama arayüzü dondurmaz.

- `Task` / `Task<T>` — Asenkron işlemin sonucunu temsil eder.

ASP.NET Core'da veritabanı çağrıları hep `await` ile yapılır — bu yüzden bu kavram web geliştirmede kritiktir.

Dosya İşlemleri

```
using System.IO;

// Dosyaya yaz
File.WriteAllText("not.txt", "Merhaba dosya!");

// Dosyadan oku
string icerik = File.ReadAllText("not.txt");
Console.WriteLine(icerik);

// Satır satır oku
string[] satirlar = File.ReadAllLines("not.txt");

// Dosya var mı?
if (File.Exists("not.txt")) {
    Console.WriteLine("Dosya mevcut.");
}
```

Properties (Özellikler)

Alanlara kontrollü erişim için property kullanılır — orta seviyedeki `private` + getter/setter'ın kısa ve modern hâli:

```
class Kisi {
    // Otomatik property
    public string Ad { get; set; }

    // Kontrollü property
    private int _yas;
    public int Yas {
        get { return _yas; }
        set { if (value >= 0) _yas = value; }   // negatif engellenir
    }
}

Kisi k = new Kisi { Ad = "Hazar", Yas = 25 };
```

Modern C# Özellikleri

```
// String interpolation ($ ile)
string ad = "Hazar";
Console.WriteLine($"Merhaba {ad}, hoş geldin!");

// Null kontrolü (?? ve ?.)
string isim = gelenDeger ?? "Bilinmiyor";   // null ise varsayılan
int? uzunluk = metin?.Length;               // metin null'sa hata verme

// Ternary (kısa if)
string durum = yas >= 18 ? "Yetişkin" : "Çocuk";
```

Temiz Kod Prensipleri

- Anlamlı isimler ver: `HesaplaToplamTutar()`, `x1()` değil.
- Her metot tek bir iş yapsın (tek sorumluluk).
- Kendini tekrar etme (DRY): tekrarlayan kodu metoda al.
- Sihirli sayılar yerine sabit (const) kullan.
- Yorumla "ne yaptığını" değil, "neden yaptığını" açıkla.

Özet

- Generics ile tip güvenli, tekrarsız kod yazılır.
- async/await, uzun işlemleri arayüzü kilitlemeden çalıştırır — web'de şarttır.
- Dosya işlemleri `System.IO` ile yapılır.
- Properties kontrollü veri erişimi sağlar; modern C# özellikleri kodu kısaltır.
- Temiz kod, projeni sürdürülebilir kılar.
- Sıradaki adım: **ASP.NET Core** ile gerçek web uygulaması geliştirmek.