bir field (alan) ve property (özellik) kullanmanın farklı nedenleri vardır ve her birinin kendine özgü avantajları bulunmaktadır. İşte bu konuyu daha iyi anlamak için neden sadece public property kullanmak yerine field ve property kombinasyonunun tercih edilebileceğini açıklayan bazı sebepler:

1. \*\*Encapsulation (Kapsülleme)\*\*:

- Property kullanarak verilerinizi daha iyi koruyabilir ve kapsülleyebilirsiniz. `get` ve `set` metodları ile veri üzerinde kontrol sahibi olabilir ve gerekli doğrulama veya iş mantığını ekleyebilirsiniz.

- Örneğin, bir property'sine atanan değeri sınırlandırmak veya belirli bir aralıkta tutmak istiyorsanız, bunu `set` metodunda yapabilirsiniz.

2. \*\*Veri Doğrulama\*\*:

- Property'lerin `set` metodlarında belirli bir doğrulama mantığı yazabilirsiniz. Böylece, verinin geçersiz bir durumda olmasını engelleyebilirsiniz.

```csharp

private int age;

public int Age

{

get { return age; }

set

{

if (value < 0 || value > 120)

{

throw new ArgumentOutOfRangeException("Age must be between 0 and 120.");

}

age = value;

}

}

```

3. \*\*Lazy Initialization (Geç Başlatma)\*\*:

- Property'lerde `get` metodları ile değeri yalnızca ihtiyaç duyulduğunda başlatabilirsiniz. Bu, özellikle ağır nesnelerin oluşturulmasında performans kazandırabilir.

```csharp

private MyClass \_instance;

public MyClass Instance

{

get

{

if (\_instance == null)

{

\_instance = new MyClass();

}

return \_instance;

}

}

```

4. \*\*Bildirimler ve Olaylar\*\*:

- Property'ler, değer değişikliklerinde olay tetikleyebilir. Bu, özellikle UI (Kullanıcı Arayüzü) uygulamalarında veri bağlama işlemleri için kullanışlıdır.

```csharp

private string name;

public string Name

{

get { return name; }

set

{

if (name != value)

{

name = value;

OnPropertyChanged("Name");

}

}

}

```

5. \*\*Geriye Dönük Uyum\*\*:

- Field'ları property'lerle değiştirmek, ileride kodu değiştirirken geriye dönük uyumu korumanıza yardımcı olabilir. Başlangıçta field olarak tanımladığınız bir değişkeni property olarak değiştirmek, API'nizin geri kalanını etkilemeden yapılabilir.

Özetle, property'ler ve field'lar arasındaki farklar ve bu kombinasyonun kullanımı, verilerin daha iyi kontrol edilmesi, doğrulanması, performans optimizasyonu ve geriye dönük uyum gibi birçok avantaj sağlar. Bu nedenle, doğrudan public field kullanmak yerine property kullanmak genellikle daha iyi bir uygulama tasarımı sağlar.

fieldlar public de yapılabilir o zaman değil mi ?

Evet, field'lar public olarak yapılabilir, ancak bu genellikle önerilmez. Bunun sebebi, public field'ların kapsülleme ilkesini ihlal etmesi ve nesne yönelimli programlamanın (OOP) temel prensiplerinden biri olan veri gizliliğini sağlamada yetersiz kalmasıdır.