C# dilinde `record`, öncelikle veri saklama ve veri transferi amacıyla kullanılan immutable (değişmez) veri yapılarıdır. `record` türleri, sınıflara (classes) benzer şekilde çalışır, ancak özellikle veri odaklı uygulamalar için tasarlanmışlardır.

### Record'ların Temel Özellikleri

1. \*\*Immutable (Değişmez) Yapı\*\*: Varsayılan olarak, `record` türleri immutable'dır. Ancak, isterseniz mutable (değiştirilebilir) yapıda da tanımlanabilir.

2. \*\*Value Equality (Değer Eşitliği)\*\*: `record` türlerinde, nesneler referanslarına göre değil, değerlerine göre eşitlik kontrolü yapılır. Bu, iki `record` nesnesinin aynı verilere sahip olması durumunda eşit kabul edilmesini sağlar.

3. \*\*Kolay Veri Saklama ve Transferi\*\*: `record` türleri, veri taşıma (DTO) ve veri saklama işlemleri için idealdir. Özellikle, veri odaklı programlama için tasarlanmışlardır.

4. \*\*C# 9.0 ve Üzeri\*\*: `record` türleri, C# 9.0 ve üzeri sürümlerde desteklenir.

### Record Tanımlama ve Kullanımı

#### Temel Record Tanımı

```csharp

public record Person(string FirstName, string LastName);

```

Bu tanım, `Person` adında bir `record` oluşturur ve iki otomatik özellik (property) tanımlar: `FirstName` ve `LastName`.

#### Kapsamlı Örnek

```csharp

using System;

public record Person(string FirstName, string LastName)

{

public string FullName => $"{FirstName} {LastName}";

}

class Program

{

static void Main()

{

var person1 = new Person("John", "Doe");

var person2 = new Person("John", "Doe");

Console.WriteLine(person1.FullName); // John Doe

Console.WriteLine(person1 == person2); // True, değer eşitliği

}

}

```

### Immutable Records (Değişmez Record'lar)

Varsayılan olarak, `record` türleri immutable'dır. Bu, oluşturulduktan sonra özelliklerinin değiştirilemeyeceği anlamına gelir. Ancak, immutable yapıyı korurken nesneleri "with" ifadesiyle kopyalayarak yeni bir nesne oluşturabilirsiniz:

```csharp

var person1 = new Person("John", "Doe");

var person2 = person1 with { LastName = "Smith" };

Console.WriteLine(person1); // John Doe

Console.WriteLine(person2); // John Smith

```

### Mutable Records (Değiştirilebilir Record'lar)

İsterseniz, `record` türlerini değiştirilebilir (mutable) olarak da tanımlayabilirsiniz:

```csharp

public record Person

{

public string FirstName { get; init; }

public string LastName { get; init; }

}

// Kullanım

var person = new Person { FirstName = "John", LastName = "Doe" };

// İlk değer atandıktan sonra değiştirme yapılamaz

// person.FirstName = "Jane"; // Derleme hatası

```

### Inheritance (Kalıtım)

`record` türleri, sınıflar gibi kalıtım destekler:

```csharp

public record Person(string FirstName, string LastName);

public record Student(string FirstName, string LastName, int StudentId) : Person(FirstName, LastName);

```

### Özet

- \*\*Immutable Yapı\*\*: `record` türleri varsayılan olarak immutable'dır, ancak mutable olarak da tanımlanabilirler.

- \*\*Değer Eşitliği\*\*: `record` türlerinde değer eşitliği kullanılır.

- \*\*Veri Odaklı Uygulamalar\*\*: Özellikle veri saklama ve transferi için uygundur.

- \*\*Kalıtım ve Genişletilebilirlik\*\*: `record` türleri kalıtım destekler.

C# dilinde `record` türleri, veri odaklı programlama ve nesne eşitliği gerektiren senaryolar için güçlü ve kullanışlı bir araçtır.