Switch Expressions

Switch Expressions

```
string mesaj = "";
switch (DateTime.Now.DayOfWeek)
{
    case DayOfWeek.Monday:
        mesaj = "Bu gün Pazartesi";
        break;
    case DayOfWeek.Tuesday:
        mesaj = "Bu gün Salı";
        break;
    case DayOfWeek.Wednesday:
        mesaj = "Bu gün Çarşamba";
        break;
    case DayOfWeek.Thursday:
        mesaj = "Bu gün Perşembe";
        break;
    case DayOfWeek.Friday:
        mesaj = "Bu gün Cuma";
        break;
    case DayOfWeek.Saturday:
        mesaj = "Bu gün Cumartesi";
        break;
    case DayOfWeek.Sunday:
        mesaj = "Bu gün Pazar";
        break;
}
Console.WriteLine(mesaj);
```

- Tek satırlık işlemler için maliyet düşürücü ve kullanışlı semantiklerdir.

```
static void Main(string[] args)
{
    string mesaj = DateTime.Now.DayOfWeek switch
    {
        DayOfWeek.Monday => "Bu gün Pazartesi",
        DayOfWeek.Tuesday => "Bu gün Salı",
        DayOfWeek.Wednesday => "Bu gün Çarşamba",
        DayOfWeek.Thursday => "Bu gün Perşembe",
        DayOfWeek.Friday => "Bu gün Cuma",
        DayOfWeek.Saturday => "Bu gün Cumartesi",
        DayOfWeek.Sunday => "Bu gün Pazar"
    };
    Console.WriteLine(mesaj);
}
```

When Şartı İle **Switch Expressions**

```
string mesaj = DateTime.Now.DayOfWeek switch
{
    var x when x == DayOfWeek.Monday => "Bu gün Pazartesi",
    var x when x == DayOfWeek.Tuesday => "Bu gün Salı",
    var x when x == DayOfWeek.Wednesday => "Bu gün Çarşamba",
    DayOfWeek.Thursday when DateTime.Now.Month > 5 => "Bu gün Perşembe",
    var x when x == DayOfWeek.Friday => "Bu gün Cuma",
    var x when x == DayOfWeek.Saturday => "Bu gün Cumartesi",
    var x when x == DayOfWeek.Sunday => "Bu gün Pazar",
    var x => "Hiçbiri" //ya da _ => "Hiçbiri"
};
```

Tuple Patterns

- Tuple patterns ise switch yapılanmasını Tuple nesnelerini kontrol edebilecek şekilde hem standart hemde yeni yapılanmayla bizlere sunmaktadır.

```
string adi = "Nevin";
int yasi = 27;

string mesaj = "";
switch (adi, yasi)
{
    case ("Hüseyin", 20):
        mesaj = "Hoşgeldin Hüseyin";
        break;
    case ("Cansu", 37):
        mesaj = "Hoşgeldin Cansu";
        break;
    case ("Nalan", 21):
        mesaj = "Hoşgeldin Nalan";
        break;
    case ("Nevin", 27):
        mesaj = "Hoşgeldin Nevin";
        break;
    case ("Hilmi", 35):
        mesaj = "Hoşgeldin Hilmi";
        break;
}
Console.WriteLine(mesaj);
```

```
static void Main(string[] args)
{
    string adi = "Nevin";
    int yasi = 27;

    string mesaj = (adi, yasi) switch
    {
        ("Hüseyin", 20) => mesaj = "Hoşgeldin Hüseyin",
        ("Cansu", 37) => mesaj = "Hoşgeldin Cansu",
        ("Nalan", 21) => mesaj = "Hoşgeldin Nalan",
        ("Nevin", 27) => mesaj = "Hoşgeldin Nevin",
        ("Hilmi", 35) => mesaj = "Hoşgeldin Hilmi",
        (_, _) => "Hoşgeldin"
    };
    Console.WriteLine(mesaj);
}
```

When Şartı İle **Tuple Patterns**

```
string adi = "Nevin";
int yasi = 27;

string mesaj = (adi, yasi) switch
{
    ("Hüseyin", 20) => "Hoşgeldin Hüseyin",
    ("Cansu", 37) => "Hoşgeldin Cansu",
    var x when x.adi == "Nalan" && x.yasi == 21 => "Hoşgeldin Nalan",
    var x when x.adi == "Nevin" && x.yasi == 27 => "Hoşgeldin Nevin",
    ("Hilmi", 35) when 3 > 5 => "Hoşgeldin Hilmi",
    _ when false => "Hoşgeldin" // ya da (_, _) => "Hoşgeldin"
};
```

Positional Patterns

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Ogrenci ogrenci = new Ogrenci
        {
            Adi = "Gençay",
            Soyadi = "Yıldız",
            Meslek = "Yazılım Uzmanı"
        };

        var isim = ogrenci switch
        {
            ("Ahmet", "Yıldırım") => "Ahmet Yıldırım",
            ("Hilmi", "Celayir") => "Hilmi Celayir",
            ("Nevin", "Yıldız") => "Nevin Yıldız",
            ("Gençay", "Yıldız") => "Gençay Yıldız",
            (_, _) => "Hiçbiri",
            _ => "Hiçbiri"
        };
    }

    2 references
    class Ogrenci
    {
        2 references
        public string Adi { get; set; }
        2 references
        public string Soyadi { get; set; }
        1 reference
        public string Meslek { get; set; }
        0 references
        public void Deconstruct(out string adi, out string soyadi)
        {
            adi = Adi;
            soyadi = Soyadi;
        }
    }
}
```

- Positional patterns ise Deconstruct özelliği olan nesneleri kıyaslamak yahut değersel karşılaştırmak için kullanılan bir gelişimdir.

When Şartı İle **Positional Patterns**

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Ogrenci ogrenci = new Ogrenci
        {
            Adi = "Gençay",
            Soyadi = "Yıldız",
            Meslek = "Yazılım Uzmanı"
        };

        var isim = ogrenci switch
        {
            ("Ahmet", "Yıldırım") => "Ahmet Yıldırım",
            var x when x.Adi == "Hilmi" && x.Soyadi == "Celayir" => "Hilmi Celayir",
            ("Nevin", "Yıldız") when 3 == 5 => "Nevin Yıldız",
            var x when x.Adi == "Gençay" && x.Soyadi == "Yıldız" => "Gençay Yıldız",
            (_, _) => "Hiçbiri",
            _ => "Hiçbiri"
        };
        Console.WriteLine(isim);
    }

    2 references
    class Ogrenci
    {
        4 references
        public string Adi { get; set; }
        4 references
        public string Soyadi { get; set; }
        1 reference
        public string Meslek { get; set; }
        0 references
        public void Deconstruct(out string adi, out string soyadi)
        {
            adi = Adi;
            soyadi = Soyadi;
        }
    }
}
```

Property Patterns

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Ogrenci ogrenci = new Ogrenci
        {
            Adi = "Gençay",
            Soyadi = "Yıldız",
            Meslek = "Yazılım Uzmanı"
        };
        double maas = 0;
        switch (ogrenci.Meslek)
        {
            case "Kasap":
                maas = 50;
                break;
            case "Tesisat Ustası":
                maas = 45;
                break;
            case "Otobüs Şöförü":
                maas = 55;
                break;
            case "Yazılım Uzmanı":
                maas = 52;
                break;
        }
        Console.WriteLine(maas);
    }
}

2 references
class Ogrenci
{
    1 reference
    public string Adi { get; set; }
    1 reference
    public string Soyadi { get; set; }
    2 references
    public string Meslek { get; set; }
}
```

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Ogrenci ogrenci = new Ogrenci
        {
            Adi = "Gençay",
            Soyadi = "Yıldız",
            Meslek = "Yazılım Uzmanı"
        };
        double maas = ogrenci switch
        {
            { Meslek: "Kasap" } => 50,
            { Meslek: "Tesisat Ustası" } => 45,
            { Meslek: "Otobüs Şöförü" } => 55,
            { Meslek: "Yazılım Uzmanı" } => 52
        };
        Console.WriteLine(maas);
    }
}

2 references
class Ogrenci
{
    1 reference
    public string Adi { get; set; }
    1 reference
    public string Soyadi { get; set; }
    5 references
    public string Meslek { get; set; }
}
```

- Property patterns, nesnenin propertylerine girerek belirli durumları hızlı bir şekilde kontrol etmemizi gerçekleştiren ve bunu farklı değerler için birden fazla kez tekrarlı bir şekilde yapmamıza olanak sağlayan güzel bir gelişimdir.

When Şartı İle **Property Patterns**

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Ogrenci ogrenci = new Ogrenci
        {
            Adi = "Gençay",
            Soyadi = "Yıldız",
            Meslek = "Yazılım Uzmanı"
        };
        double maas = ogrenci switch
        {
            { Meslek: "Kasap" } => 50,
            var x when x.Meslek == "Tesisat Ustası" => 45,
            { Meslek: "Otobüs Şöförü" } when !true => 55,
            var x when x.Meslek == "Yazılım Uzmanı" => 52,
            var x => 0 //ya da _ => 0
        };
        Console.WriteLine(maas);
    }

    2 references
    class Ogrenci
    {
        2 references
        public string Adi { get; set; }
        2 references
        public string Soyadi { get; set; }
        5 references
        public string Meslek { get; set; }
        0 references
        public void Deconstruct(out string adi, out string soyadi)
        {
            adi = Adi;
            soyadi = Soyadi;
        }
    }

}
```