# Introducing C# 9: Improved pattern matching

2020-06-23 by anthonygiretti

#### Introduction

Pattern matching has been introduced in C# 6 and has well evolved since. The latest improvement was pretty interesting on C# 8 C# 8Miguel Bernard (): . If you have missed all new features in C# 8.

In this article I will show you all the great new features of pattern matching.

#### Relational patterns C# 9 allows you to use relational pattern which enables the use of <, >, <= and >= in patterns like this:

using CSharp9Demo.Models;

```
using System;
     namespace CSharp9Demo
4
         class Program
8
             static void Main(string[] args)
9
10
                 var product = new Product { Name = "Food", CategoryId = 4 };
                 GetTax(product); // Returns 5
11
12
13
14
             private static int GetTax (Product p) => p.CategoryId switch
15
16
                 1 => 0,
                 < 5 => 5,
18
                 > 20 => 15,
19
20
23
         public class Product
24
             public string Name { get; set; }
26
             public int CategoryId { get; set; }
27
28
29
30
         public class Book : Product
31
32
             public string ISBN { get; get; }
         public class ElectronicProduct : Product
36
37
             public bool HasBluetooth { get; set; }
38
39
Program.cs hosted with W by GitHub
                                                                                                      view raw
```

### using CSharp9Demo.Models;

this:

Logical patterns

using System;

C# 9 lets you use logical operators like 'and', 'or' and 'not', they can even be combined with relational patterns like

```
namespace CSharp9Demo
           class Program
               static void Main(string[] args)
  8
  9
                   var product = new Product { Name = "Food", CategoryId = 4 };
 10
                   GetTax(product); // Returns 5
 11
 12
 13
 14
               private static int GetTax(Product p) => p.CategoryId switch
 15
 16
 17
                   0 \text{ or } 1 \Rightarrow 0,
                   > 1 and < 5 \Rightarrow 5,
 18
                   > 20 => 15,
 19
                   _ => 10
 20
 23
           public class Product
 24
               public string Name { get; set; }
 26
               public int CategoryId { get; set; }
 27
 28
 29
 30
           public class Book : Product
 31
               public string ISBN { get; get; }
 34
 35
           public class ElectronicProduct : Product
 36
 37
               public bool HasBluetooth { get; set; }
 38
 39
 Program.cs hosted with 🛡 by GitHub
                                                                                                          view raw
Not patterns
'not' logical operator can also be used in a if statement (it works also with a ternary statement), like this:
```

## class Program

namespace CSharp9Demo

using System;

using CSharp9Demo.Models;

8 static void Main(string[] args) 9 var product = new Product { Name = "Food", CategoryId = 4 }; 10

```
GetDiscount(product); // Returns 25
                  GetDiscount2(product); // Returns 25
 12
 13
 14
              private static int GetDiscount (Product p)
 16
 17
 18
 19
                      return 25;
 20
 21
                    return 0;
 22
 23
 24
              private static int GetDiscount2 (Product p) => p is not ElectronicProduct ? 25 : 0;
 25
 26
 27
          public class Product
 28
 29
              public string Name { get; set; }
 30
              public int CategoryId { get; set; }
 32
          public class Book : Product
 34
              public string ISBN { get; get; }
 35
 36
 37
          public class ElectronicProduct : Product
 38
 39
 40
              public bool HasBluetooth { get; set; }
 41
 42
 Program.cs hosted with W by GitHub
                                                                                                    view raw
Simple type pattern
Another nice improvement of C# 9: Simple type pattern. When a type matches, the underscore symbol _ (commonly
named discard parameter) can be omitted, thats makes the syntax lighter:
      using CSharp9Demo.Models;
  1
      using System;
      namespace CSharp9Demo
          class Program
  6
```

10 var product = new Product { Name = "Food", CategoryId = 4 }; GetDiscount(product); // Returns 25

```
static void Main(string[] args)
12
13
14
             private static int GetDiscount(Product p) => p switch
16
17
                 ElectronicProduct => 0, // ElectronicProduct _ => 0 before C# 9
                 Book b => 75, // Book b _ => 75 before C# 9
18
19
20
22
         public class Product
23
24
             public string Name { get; set; }
26
             public int CategoryId { get; set; }
27
28
         public class Book : Product
29
30
             public string ISBN { get; get; }
31
33
34
         public class ElectronicProduct : Product
35
36
             public bool HasBluetooth { get; set; }
37
38
Program.cs hosted with W by GitHub
                                                                                                      view raw
```