Introducing Functional Types and Functions

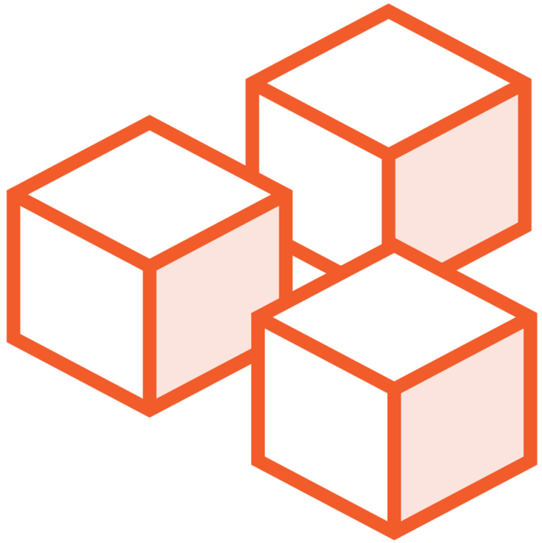
Zoran Horvat

CEO at Coding Helmet

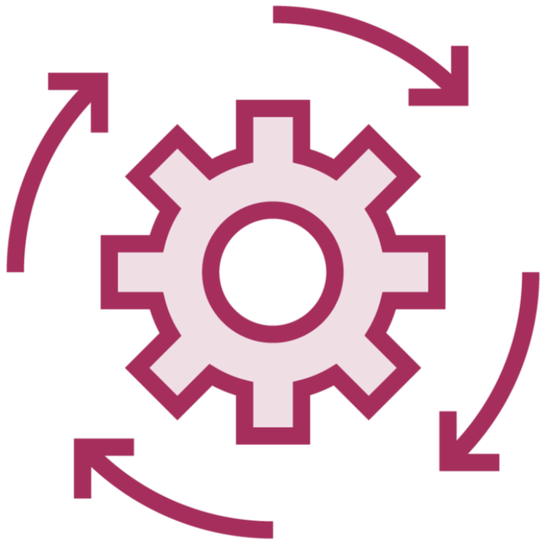
@zoranh75 https://codinghelmet.com

The Functional Designing Process

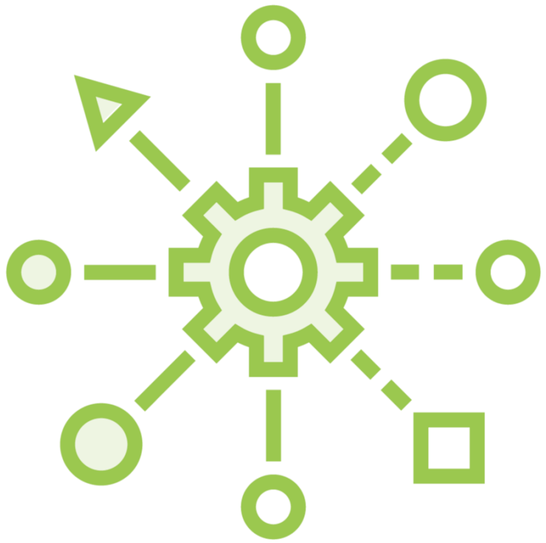
Identify *names* in the domain specification



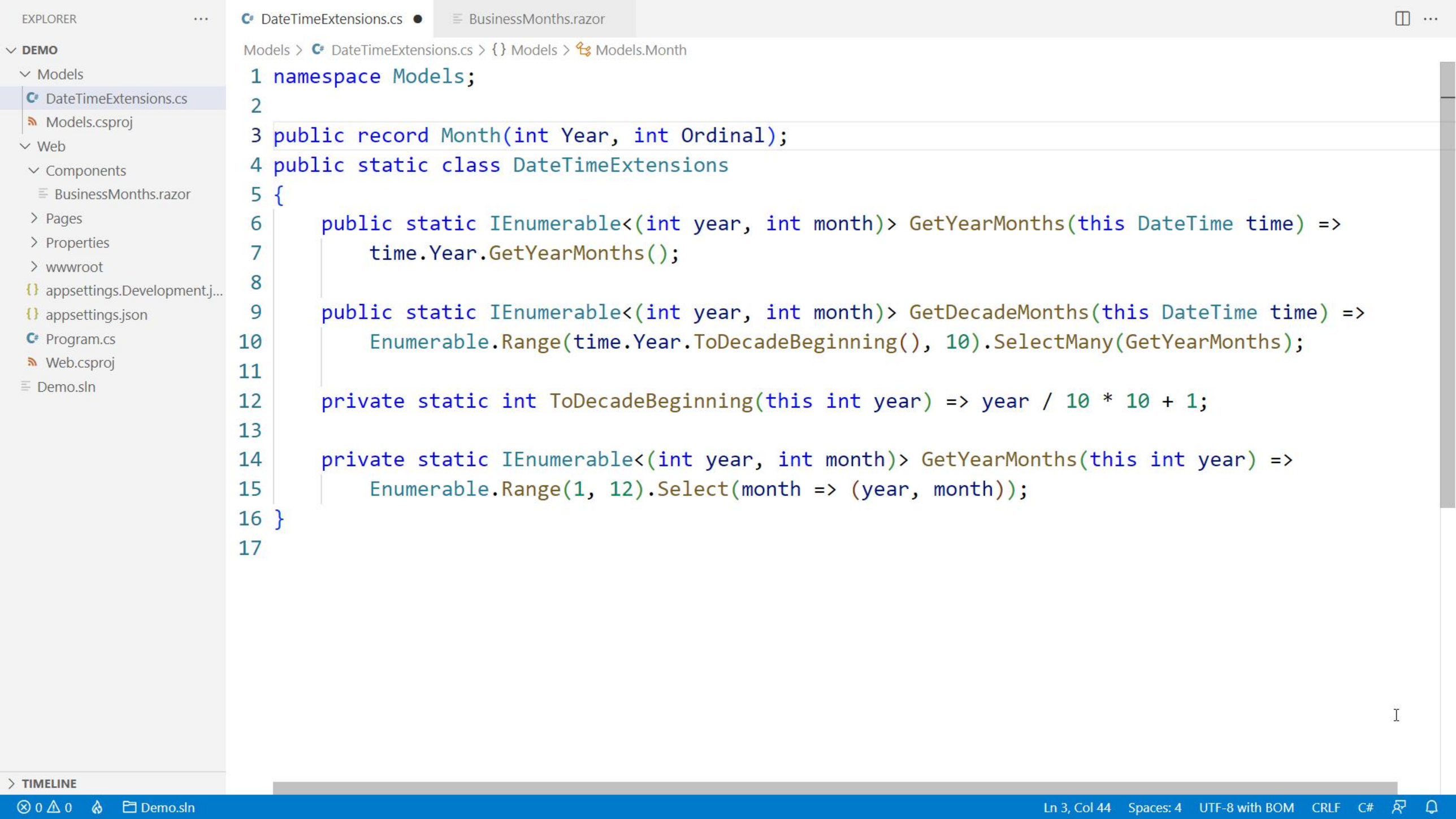
Represent each with a *type*

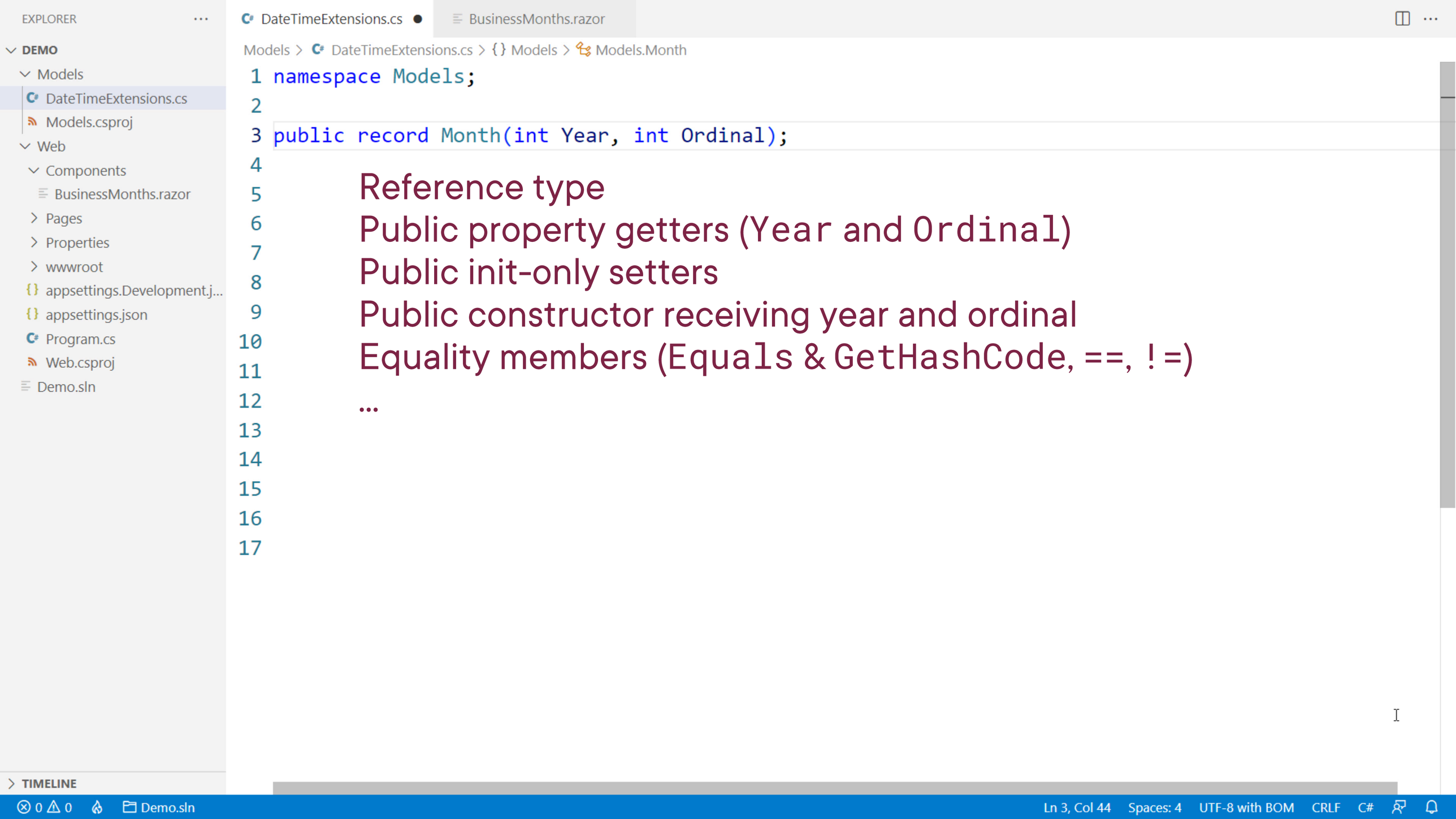


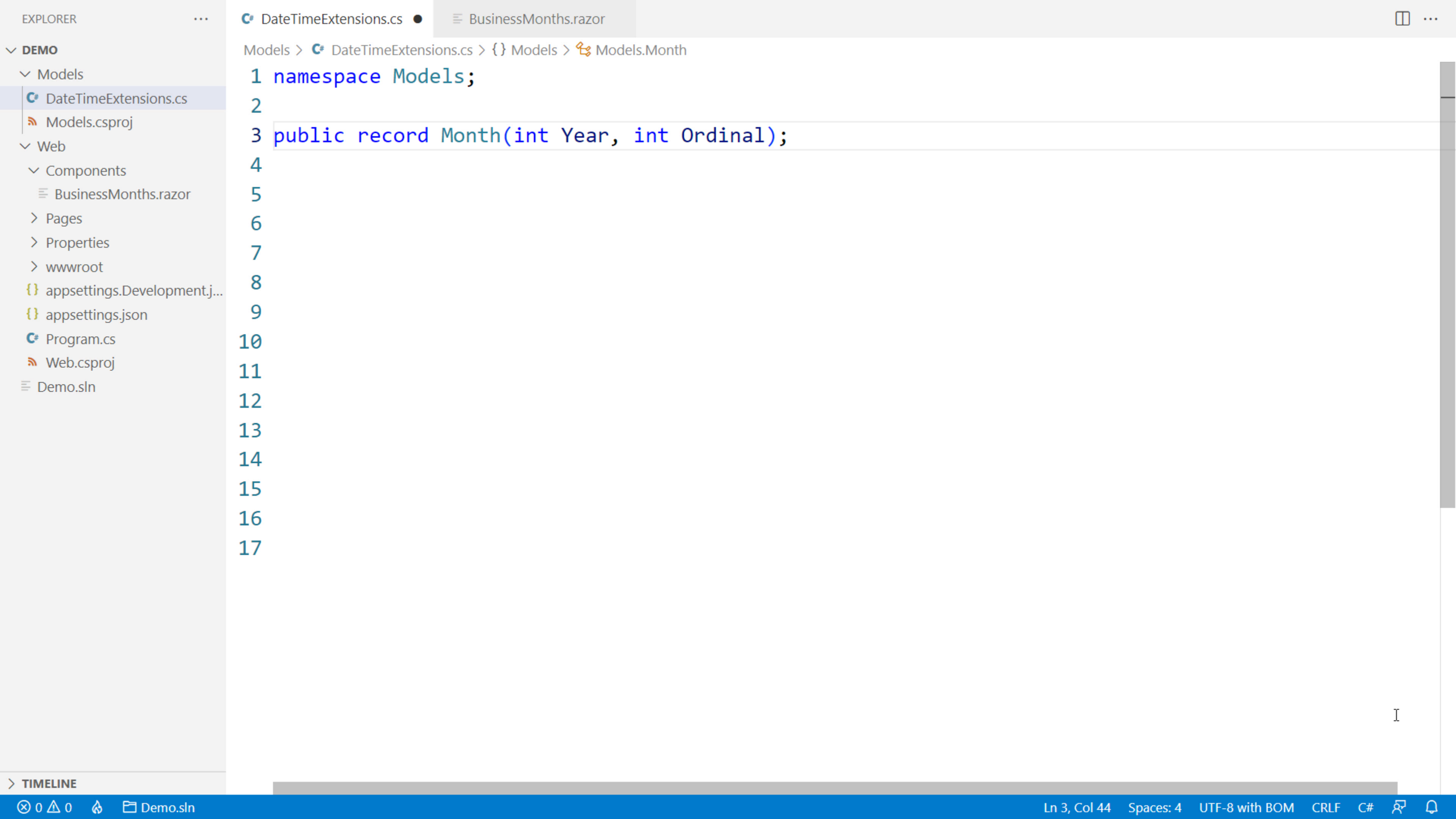
Represent processes with *functions*



Return a *result* from every function


EXPLORER
DEMO
Models
DateTimeExtensions.cs
Models.csproj
Web
Components
BusinessMonths.razor
Pages
Properties
wwwroot
appsettings.Development.j...
appsettings.json
Program.cs
Web.csproj
Demo.sln
TIMELINE
DateTimeExtensions.cs
BusinessMonths.razor
Models > DateTimeExtensions.cs > { } Models > Models.Month
1 namespace Models;
2
3 public record Month(int Year, int Ordinal);
4 public static class DateTimeExtensions
5 {
6 public static IEnumerable<(int year, int month)> GetYearMonths(this DateTime time) =>
7 time.Year.GetYearMonths();
8
9 public static IEnumerable<(int year, int month)> GetDecadeMonths(this DateTime time) =>
10 Enumerable.Range(time.Year.ToDecadeBeginning(), 10).SelectMany(GetYearMonths);
11
12 private static int ToDecadeBeginning(this int year) => year / 10 * 10 + 1;
13
14 private static IEnumerable<(int year, int month)> GetYearMonths(this int year) =>
15 Enumerable.Range(1, 12).Select(month => (year, month));
16 }
17
0 0
Demo.sln
Ln 3, Col 44
Spaces: 4
UTF-8 with BOM
CRLF
C#


EXPLORER
DEMO
Models
DateTimeExtensions.cs
Models.csproj
Web
Components
BusinessMonths.razor
Pages
Properties
wwwroot
appsettings.Development.j...
appsettings.json
Program.cs
Web.csproj
Demo.sln
TIMELINE
DateTimeExtensions.cs
BusinessMonths.razor
Models > DateTimeExtensions.cs > { } Models > Models.Month
1 namespace Models;
2
3 public record Month(int Year, int Ordinal);
4
5 Reference type
6 Public property getters (Year and Ordinal)
7 Public init-only setters
8 Public constructor receiving year and ordinal
9 Equality members (Equals & GetHashCode, ==, !=)
10 ...
11
12
13
14
15
16
17
0 0
Demo.sln
Ln 3, Col 44
Spaces: 4
UTF-8 with BOM
CRLF
C#


EXPLORER
DEMO
Models
DateTimeExtensions.cs
Models.csproj
Web
Components
BusinessMonths.razor
Pages
Properties
wwwroot
appsettings.Development.j...
appsettings.json
Program.cs
Web.csproj
Demo.sln
TIMELINE
DateTimeExtensions.cs
BusinessMonths.razor
Models > DateTimeExtensions.cs > { } Models > Models.Month
1 namespace Models;
2
3 public record Month(int Year, int Ordinal);
4
5
6
7
8
9
10
11
12
13
14
15
16
17
0 0
Demo.sln
Ln 3, Col 44
Spaces: 4
UTF-8 with BOM
CRLF
C#

```
1 namespace Models;
2
3 public record Year(int Number);
4
```

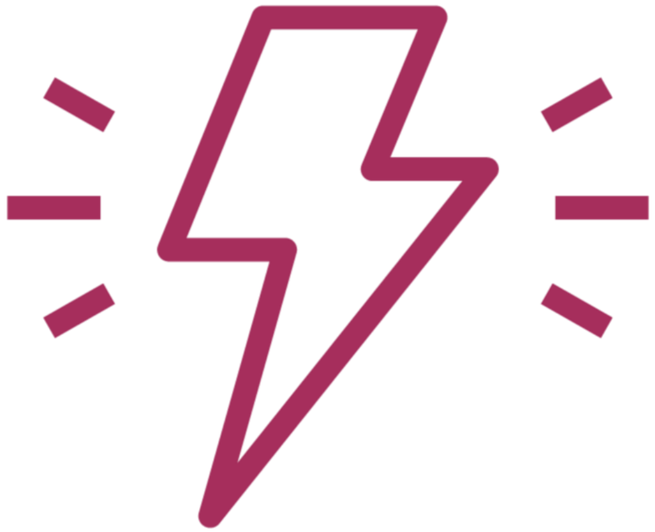
Validation in Records

Generated constructor has no validation

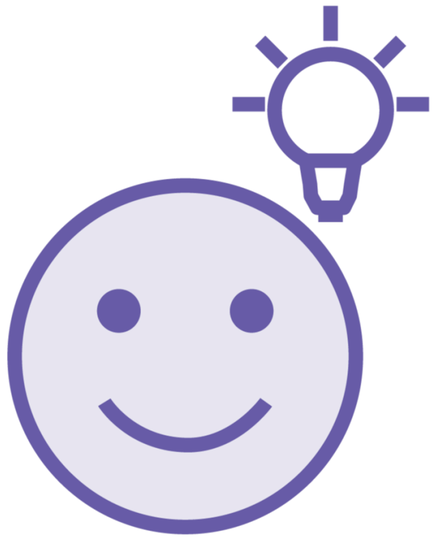
It only assigns values to components

Any input values are fine

Validation in Records



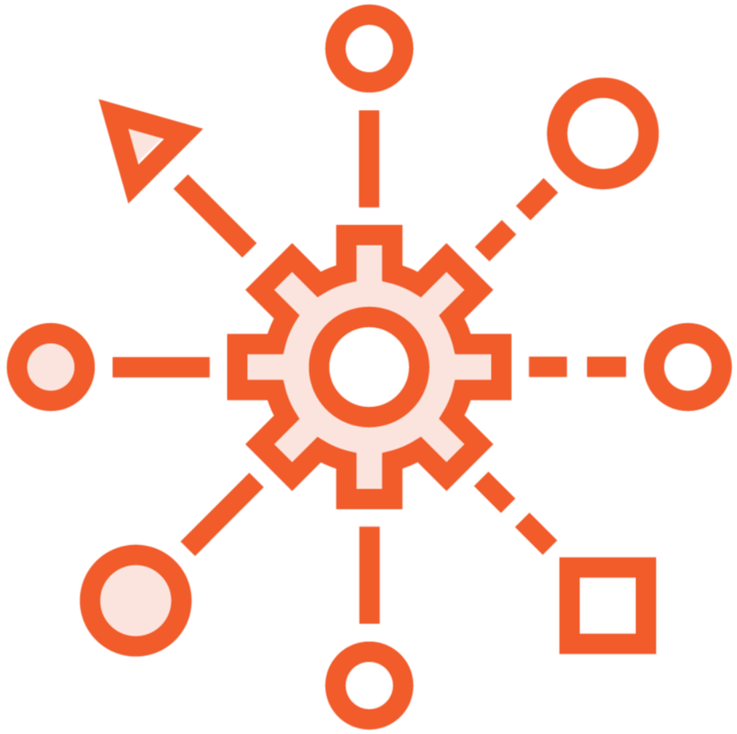
Measure 1:
Do not call bare constructor



Measure 2:
Think of a record as a value

It attains a meaning from the context in which it is used

Functional Programming Fundamentals



Separate types from functions

- Use static and extension methods in C#

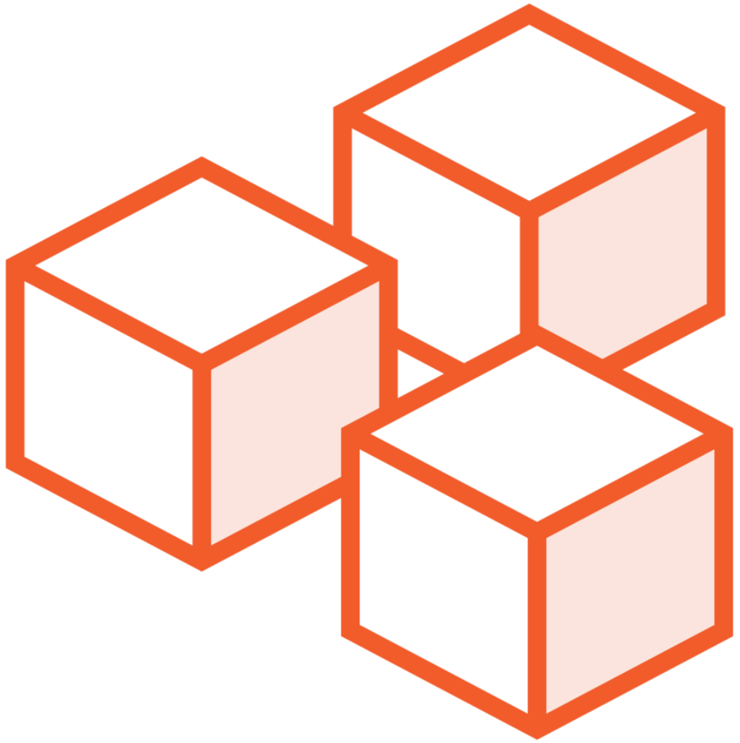
Do not allow redefining of functions

- Forbid method overriding

Function applies to a type and its subtypes

- Method's location becomes less important

Using Objects in a Functional Design



Everything is an object in C#

- Even a lambda is an object

Do not try to avoid using objects

- Instance-level methods can be functional!

Object-oriented vs. functional design

- Object-oriented design composes objects
- Functional design composes functions

Define small, non-virtual,
self-contained methods
that return results that can be used
in subsequent function composition

Feel free to define primitives
on the class to which they apply

Move specialized transforms
to a separate static class

Summary

Composition

- Type composition, function composition

Types consist of other types

Small functions apply to types

- Open for function composition

Turns code into queries on objects

- Improves code reuse

Up Next:
Modeling the Domain with Types