Class vs Record

Class vs Record: Difference between class and record type in C# - Josip Miskovic

The main difference between class and record type in C# is that a record has the main purpose of storing data, while a class defines responsibility. Records are immutable, while classes are not. Simply put, a class is an OOP concept that wraps data with functionality,

https://josipmisko.com/posts/c-sharp-class-vs-record

```
rd Window

int Size { get; init; }
string Color { get; init; }
```

```
public class Window : IEquatable<Window>
{
    [CompilerGenerated]
    [DebuggerBrowsable(DebuggerBrowsableState.Ne
    private readonly int <Size>k__BackingField;

    [CompilerGenerated]
    [DebuggerBrowsable(DebuggerBrowsableState.Ne
    private readonly string <Color>k__BackingFie

    [System.Runtime.CompilerServices.Nullable(1)
    [CompilerGenerated]
    protected virtual Type EqualityContract
    {
        [System.Runtime.CompilerServices.Nullabl
        [CompilerGenerated]
        get
        {
            return typeof(Window);
        }
    }

    public int Size
    {
        [CompilerGenerated]
        get
        {
            return <Size>k__BackingField;
        }
        [CompilerGenerated]
        init
        {
            <Size>k__BackingField = value;
        }
    }

    public string Color
    {
```

The main difference between class and record type in C# is that a record has the main purpose of storing data, while class define responsibility. **Records are immutable, while classes are not.**

records are designed for the common case of "data only" types. They more closely resemble structs than classes. A class is an OOP concept that wraps data with functionality. A record is just a set of data.

```
record Point
    {
        public int X { get; init; }
        public int Y { get; init; }
    }
    class Rectangle
    {
       public Point TopLeft { get; set; }
       public Point BottomRight { get; set; }
    }
```

Immutability isn't something that you "prove". It is enforced by the language. But note that records do not have to be immutable (just like structs and classes don't have to be). But they are primarily designed for immutability purposes (hence just a set of related data). Examples might include points, coordinates or the parts of a complex number. By themselves the data points mean nothing, but combined together you have something useful. That is what a record represents.

```
var topLeft = new Point() { X = 10, Y = 20 };
topLeft.X = 20;  //Compiler error, X cannot be written

var rect = new Rectangle() { TopLeft = topLeft, BottomRight = new Point() { X = 20, Y = 40 } };
rect.TopLeft = new Point() { X = 50, Y = 50 };  //Writable
```