Treat Primitive Obsession with Value Objects

Steve Smith

@ardalis

steve@nimblepros.com | NimblePros.com

NimblePros

Better software development. Delivered.

What is Primitive Obsession

- What's a primitive?
  - Built in language types like string, int, etc.

- What's an obsession?

obsession noun

ob·ses·sion äb-ˈse-shən əb-

Synonyms of *obsession* ›

1 : a persistent disturbing preoccupation with an often unreasonable idea or feeling

broadly : compelling motivation

an *obsession* with profits

has an *obsession* with gambling

Is it *unreasonable* to use primitives for everything?

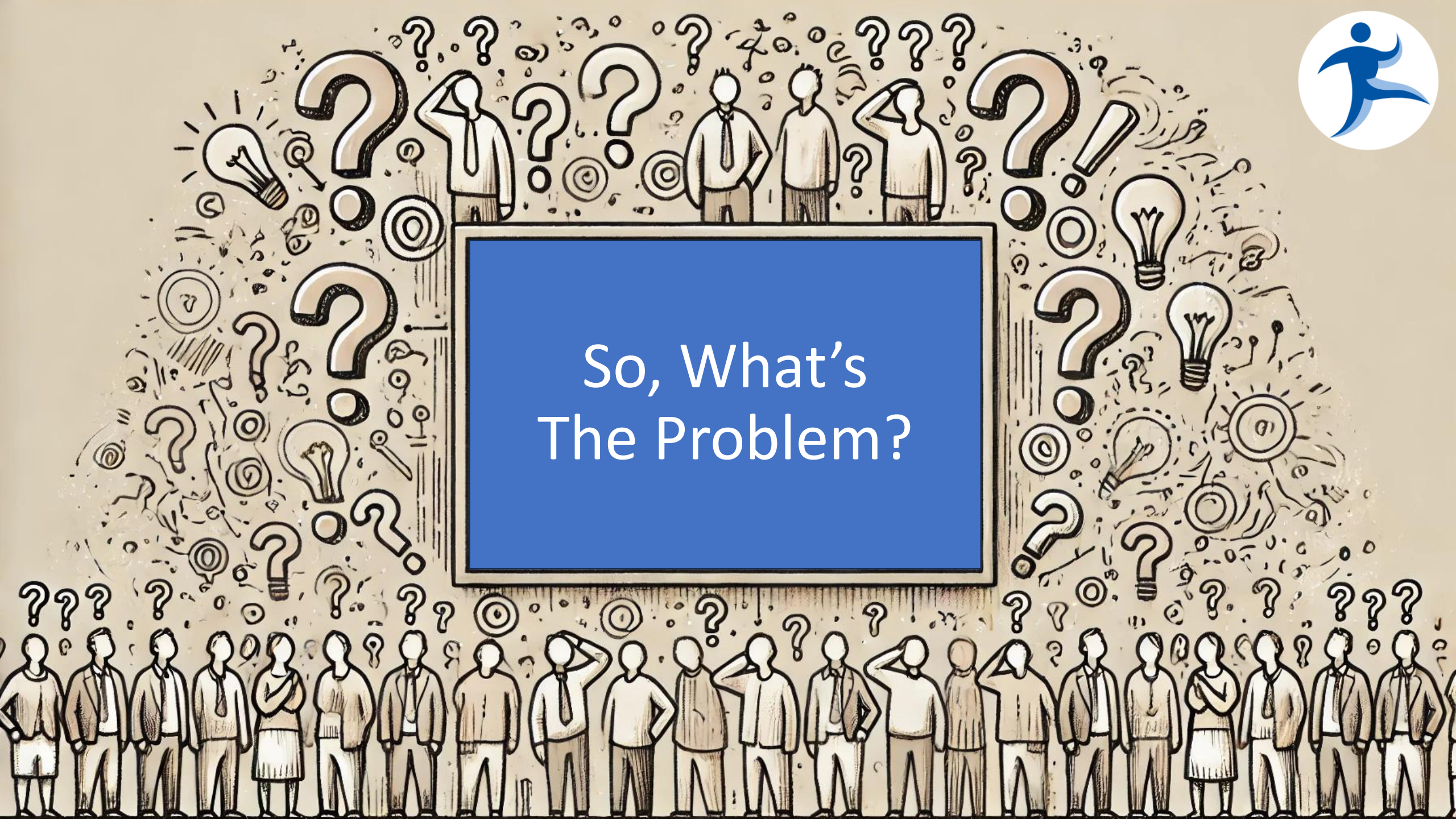
- I mean, you'll spend a lot less time refactoring interfaces if all of your methods just look like this:

```
public string DoSomething(string input)
{
  // do something and return something
}
```

But what about this?

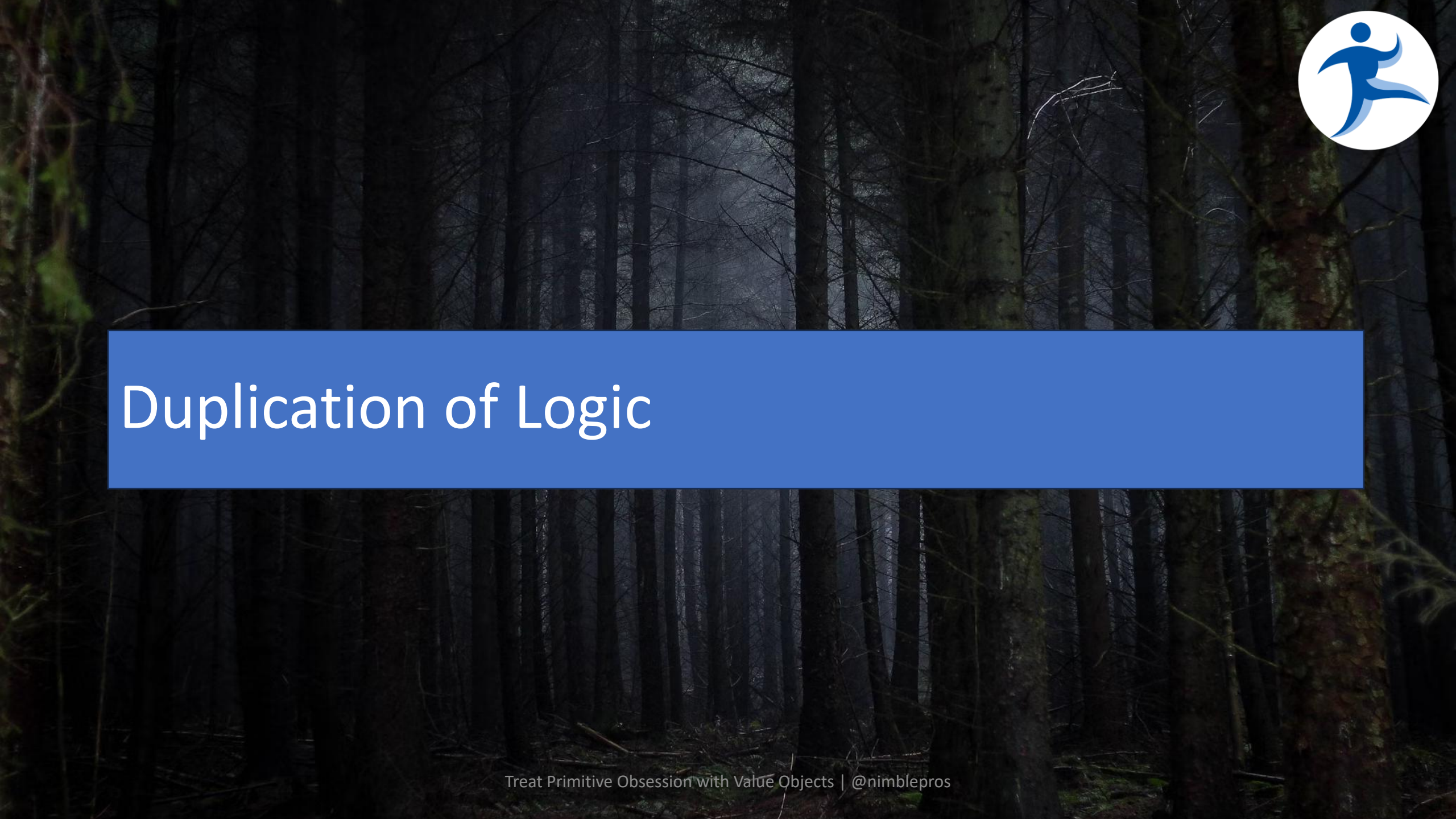
```
public class Customer
{
  public int Id { get; set; }
  public string FirstName { get; set; }
  public string LastName { get; set; }
  public string Street1 { get; set; }
  public string Street2 { get; set; }
  public string City { get; set; }
  public string State { get; set; }
  public string PostalCode { get; set; }
  public string Country { get; set; }
}
```

Raise your hand if your code at work looks something like this

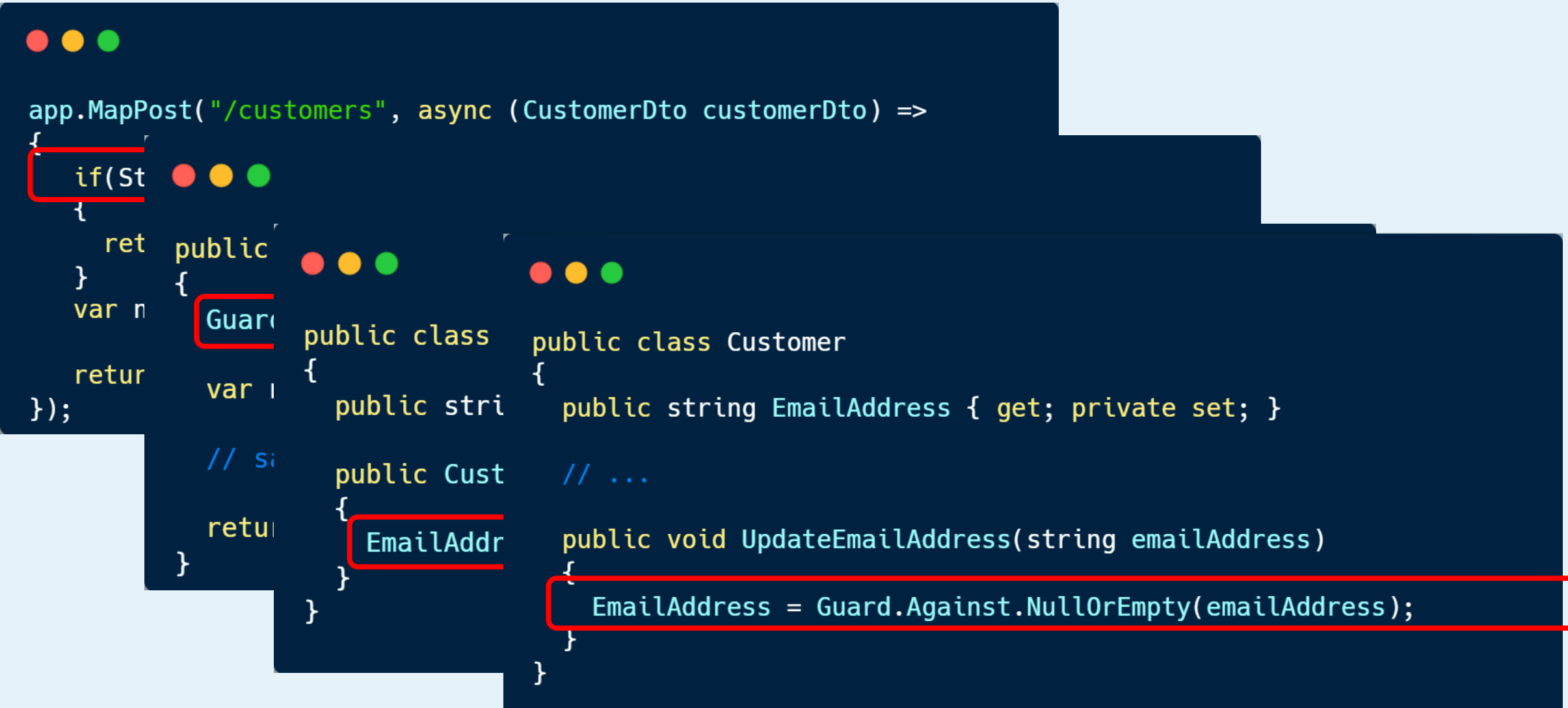

So, What's
The Problem?

Increased Risk of Errors

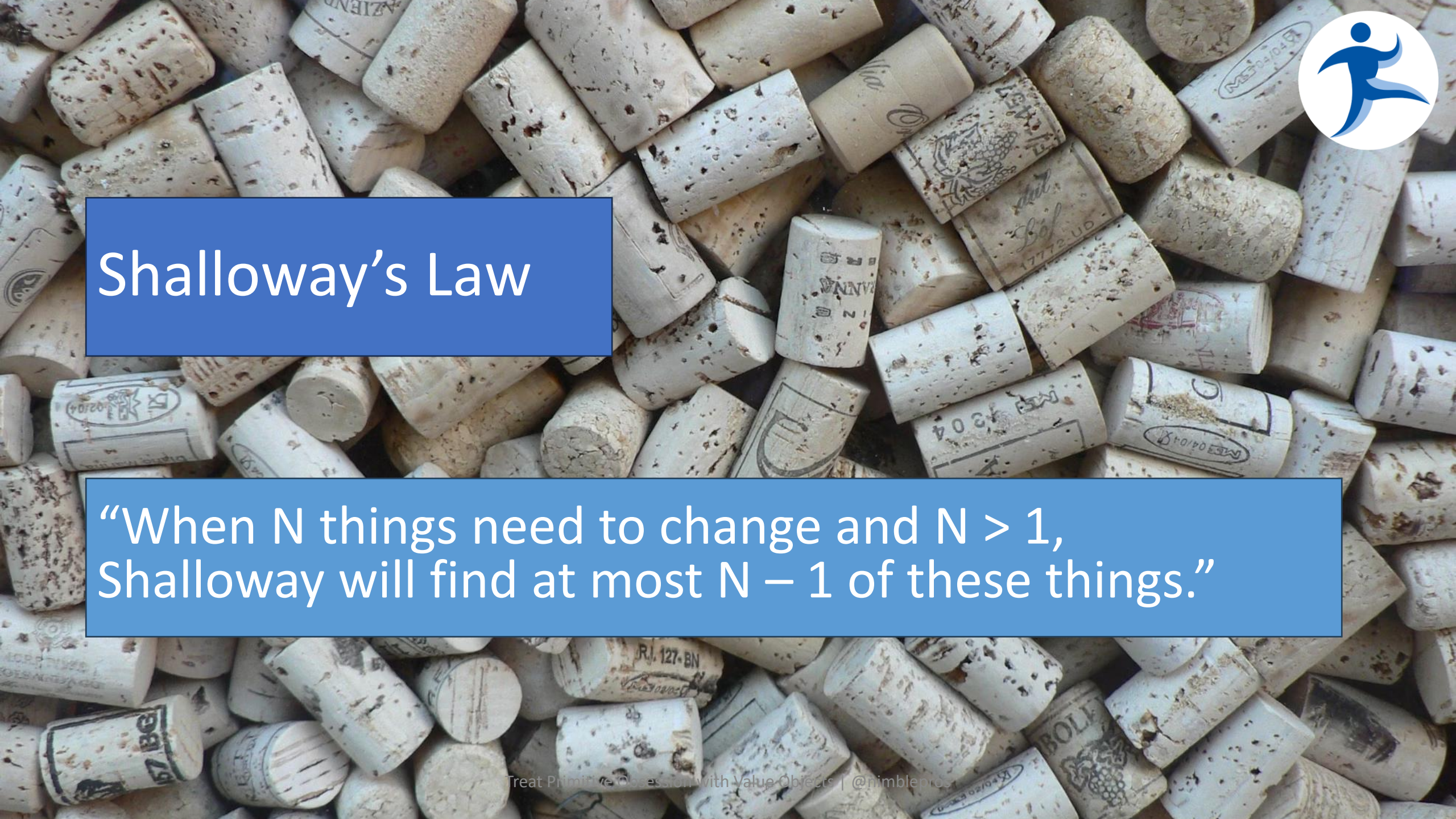
This compiles and runs fine



Duplication of Logic


app.MapPost("/customers", async (CustomerDto customerDto) =>
public class
public class Customer
public string EmailAddress { get; private set; }
public void UpdateEmailAddress(string emailAddress)
EmailAddress = Guard.Against.NullOrEmpty(emailAddress);

What if the logic to validate the value changes?



Shalloway's Law

"When N things need to change and N > 1,
Shalloway will find at most N – 1 of these things."

“I’ll just check everything in the entity and extract a common method”

```
public class Contributor : EntityBase, IAggregateRoot
{
    9 references | AminSharifi, 192 days ago | 1 author, 1 change
    public string Name { get; private set; } = default!;

    7 references | AminSharifi, 192 days ago | 1 author, 1 change
    public Contributor(string name)
    {
        SetName(name);
    }

    1 reference | AminSharifi, 192 days ago | 1 author, 1 change
    public void UpdateName(string newName)
    {
        SetName(newName);
    }

    2 references | AminSharifi, 192 days ago | 1 author, 1 change
    public Contributor SetName(string newName)
    {
        this.Name = Guard.Against.NullOrEmpty(newName, nameof(newName));
        return this;
    }
```

```
public class Contributor : EntityBase, IAggregateRoot
{
    9 references | AminSharifi, 192 days ago | 1 author, 1 change
    public string Name { get; private set; } = default!;

    7 references | AminSharifi, 192 days ago | 1 author, 1 change
    public Contributor(string name)
    {
        SetName(name);
    }

    1 reference | AminSharifi, 192 days ago | 1 author, 1 change
    public void UpdateName(string newName)
    {
        SetName(newName);
    }

    2 references | AminSharifi, 192 days ago | 1 author, 1 change
    public Contributor SetName(string newName)
    {
        this.Name = Guard.Against.NullOrEmpty(newName, nameof(newName));
        RegisterDomainEvent(new ContributorUpdatedEvent(this.Id));
        return this;
    }
}
```

What happens when EF creates this type via its constructor?

(come to my talk tomorrow for more on how to reduce duplication of things like validation and other cross cutting concerns)

Reduced Type Safety

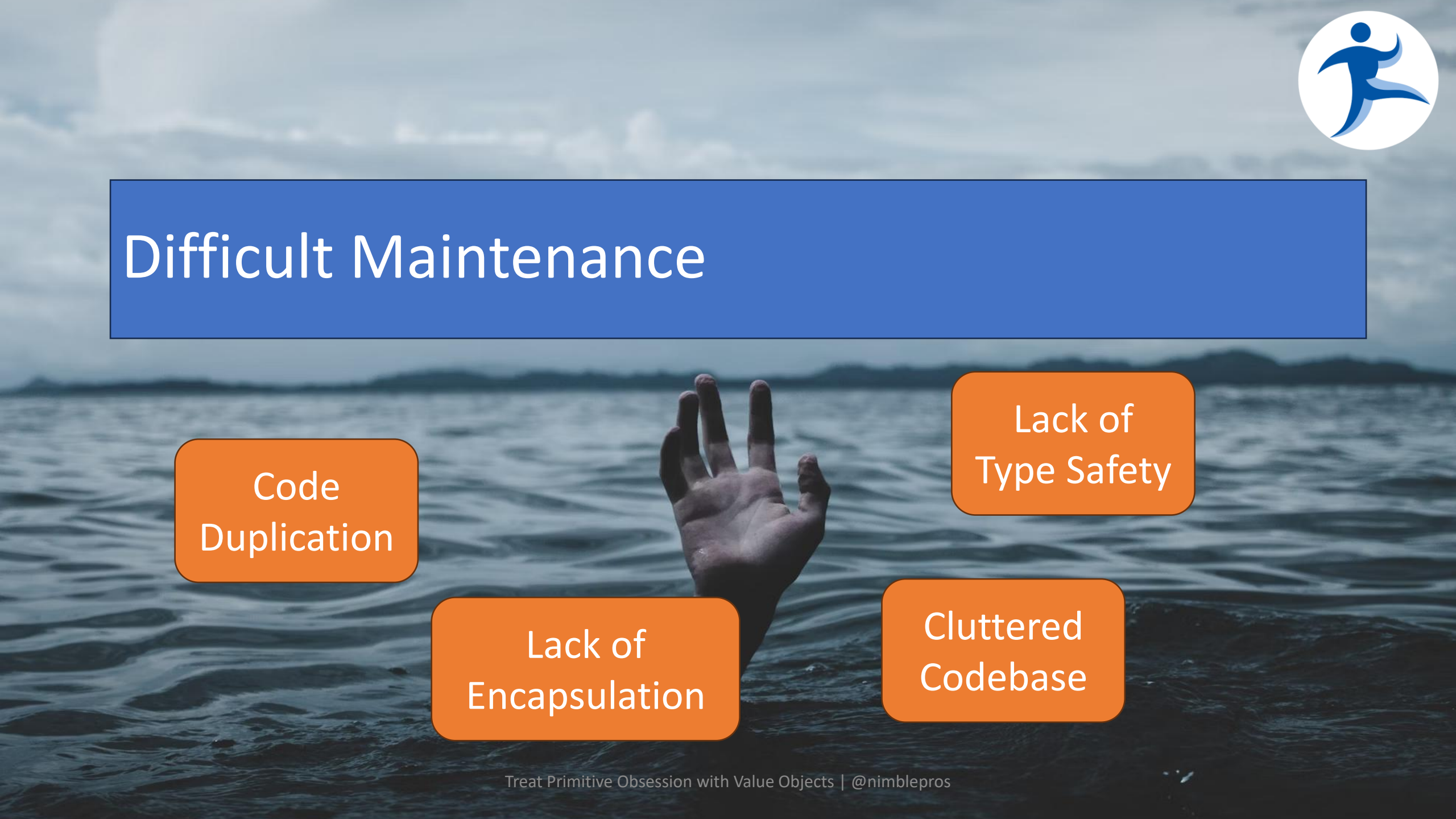
In Strongly-Typed Languages, the Compiler is Your Friend

Common Pitfalls

```
public void AddOrderToCustomer(int orderId, int customerId);

// somewhere else

AddOrderToCustomer(dto.CustomerId, dto.OrderId); // compiles fine!
```



Difficult Maintenance

Code Duplication

Lack of Type Safety

Lack of Encapsulation

Cluttered Codebase

https://deviq.com/practices/single-point-of-enforcement

Primitives Carry No History or Context

Any knowledge about the validity of a primitive is lost as soon as it’s passed to another method

Parse, Don’t Validate

- Validating doesn’t change the type
- Passing the type on to another method/object requires either **duplicate validation** or **accepting the risk** of using the values as-is
- **Parsing** one type into another, **more restrictive** type, is the key
  - A type that **can only represent valid states**
- Passing it into another method retains the information gained

https://lexi-lambda.github.io/blog/2019/11/05/parse-don-t-validate/

Make Invalid States Unrepresentable

Demo:
A World Without DateTime

Imagine if we didn't have a built-in DateTime type and everyone just used primitives for them.

Thankfully, DateTime exists

So we don’t have to check if

- Month > 12
- Day > 31
- Hour > 23
- Minute > 59

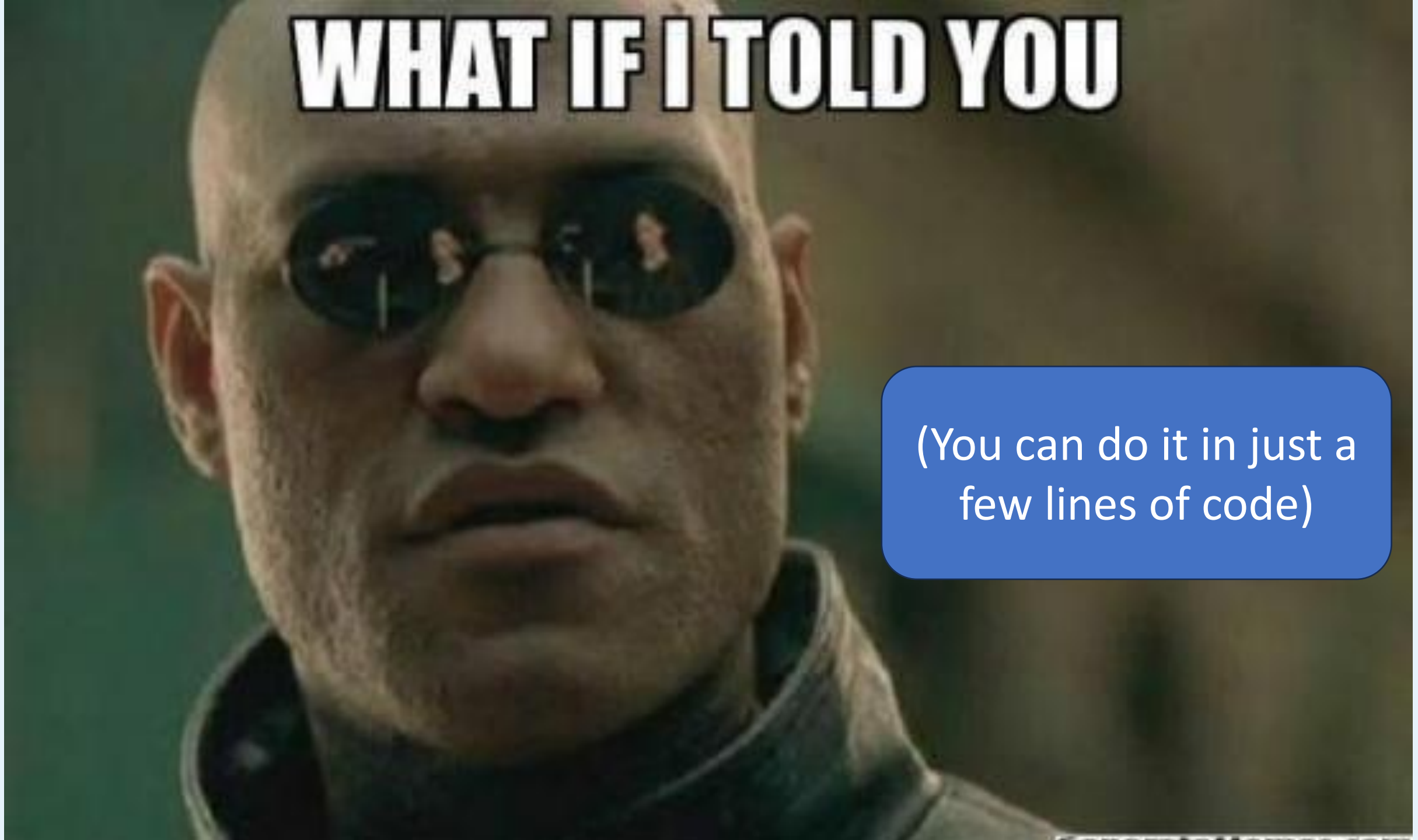
All over our applications.

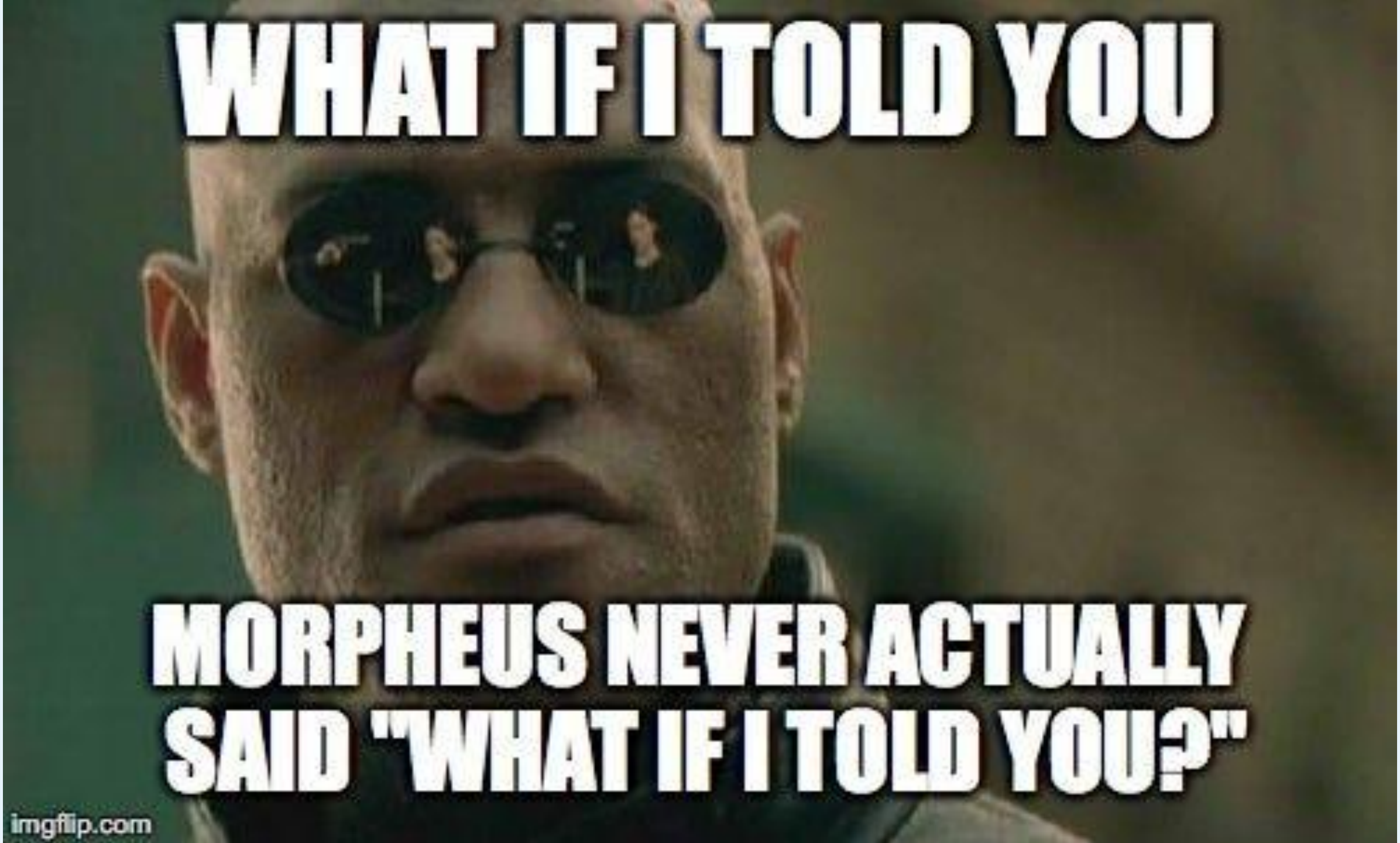
So, why don’t we do that with our custom types?

Code Blame 1700 lines (1494 loc) · 76.6 KB Raw

```
6     namespace System {
57        public struct DateTime : IComparable, IFormattable, IConvertible, ISerializable, IComparable<DateTime>,IEquatable<Date
1656          internal static Boolean TryCreate(int year, int month, int day, int hour, int minute, int second, int millisecond,
1680                        // if it is not valid time, we'll eventually throw.
1681                        // although this is unspecified datetime kind, we'll assume the passed time is UTC to check t
1682                        second = 59;
1683                    }
1684                    else
1685                    {
1686                        [illegible]
1687                    }
1688
1689
1690                long ticks = DateToTicks(year, month, day) + TimeToTicks(hour, minute, second);
1691
1692                ticks += millisecond * TicksPerMillisecond;
1693                if (ticks < MinTicks || ticks > MaxTicks) {
1694                    return false;
1695                }
1696                result = new DateTime(ticks, DateTimeKind.Unspecified);
1697                return true;
1698            }
1699        }
```

BECAUSE IT'S 1700 LINES OF CODE AND NOBODY'S GOT TIME FOR THAT!


WHAT IF I TOLD YOU
(You can do it in just a few lines of code)


WHAT IF I TOLD YOU
MORPHEUS NEVER ACTUALLY SAID "WHAT IF I TOLD YOU?"
imgflip.com

Let’s Talk About Solutions

That means types.

Sometimes these are specifically referred to as Value Objects

What is a Value Object?

A custom type that…

- Composed of Primitives and/or Value Objects
- Immutable
- Has No Identity
- Persisted as Part of an Entity
- Represents a Business Concept
- Compared by Properties

Value *Type* or Value *Object*?

- In .NET / C# / CLR there are Value Types (like **int**) and Reference Types (like **string**)

- This distinction has to do with whether pointers are involved and how type instances are passed between methods

- Value *Objects* are an unrelated concept, and may sometimes be implemented as either a reference type (class) or a value type (struct)

Option 1: Use a Base Type

https://www.nuget.org/packages/CSharpFunctionalExtensions

```
10  namespace CSharpFunctionalExtensions;
11
12  [Serializable]
13  public abstract class ValueObject
14  {
15      private int? _cachedHashCode;
16
17      protected abstract IEnumerable<object> GetEqualityComponents();
18
19      public override bool Equals(object obj)...
35      public override int GetHashCode()...
44
45      public static bool operator ==(ValueObject a, ValueObject b)...
60      public static bool operator !=(ValueObject a, ValueObject b)...
64
65      internal static Type GetUnproxiedType(object obj)...
76  }
```

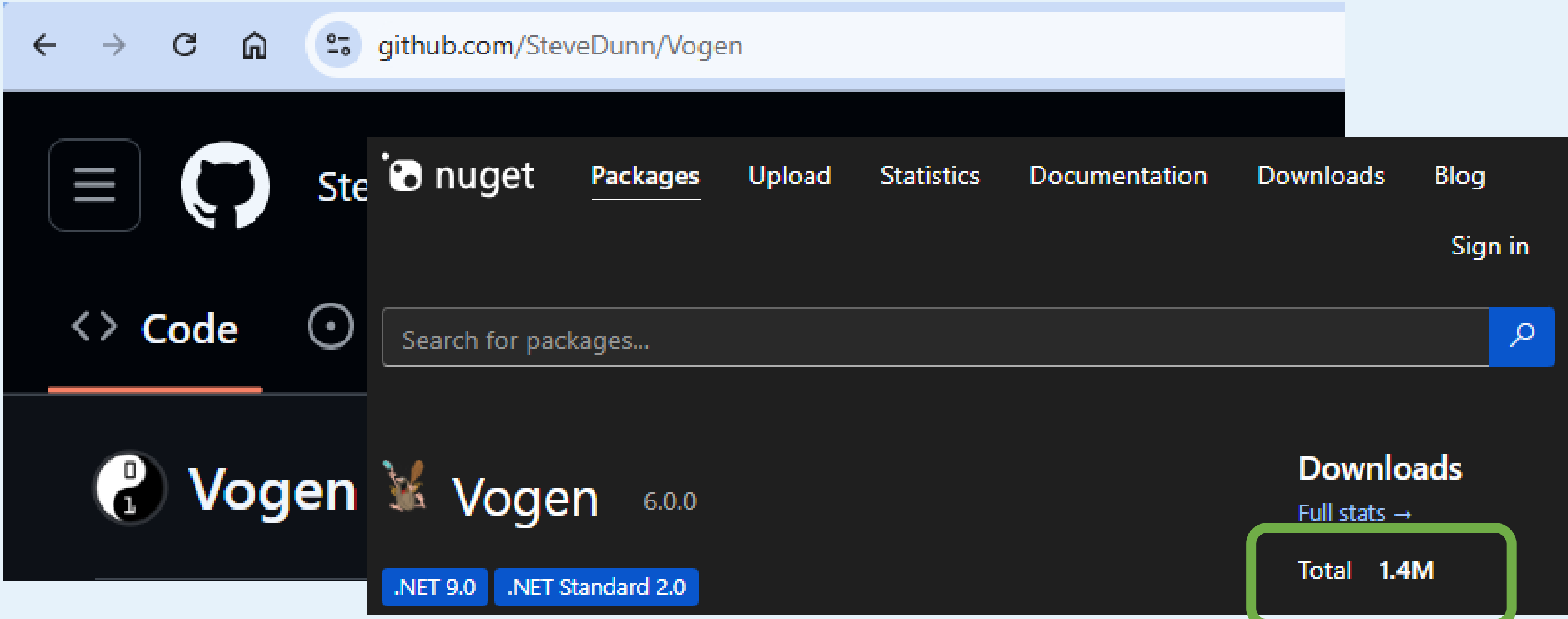
Also includes
ComparableValueObject which adds IComparable

Example Implementation

```
4 references | 0 changes | 0 authors, 0 changes
public class Address : ValueObject
{
    1 reference | 0 changes | 0 authors, 0 changes
    public Address(string address1, string city, string state,
        string postalCode, string country, string? address2 = null)...

    0 references | 0 changes | 0 authors, 0 changes
    protected override IEnumerable<object> GetEqualityComponents()
    {
        yield return Address1;
        yield return Address2 ?? "";
        yield return City;
        yield return State;
        yield return PostalCode;
        yield return Country;
    }
```

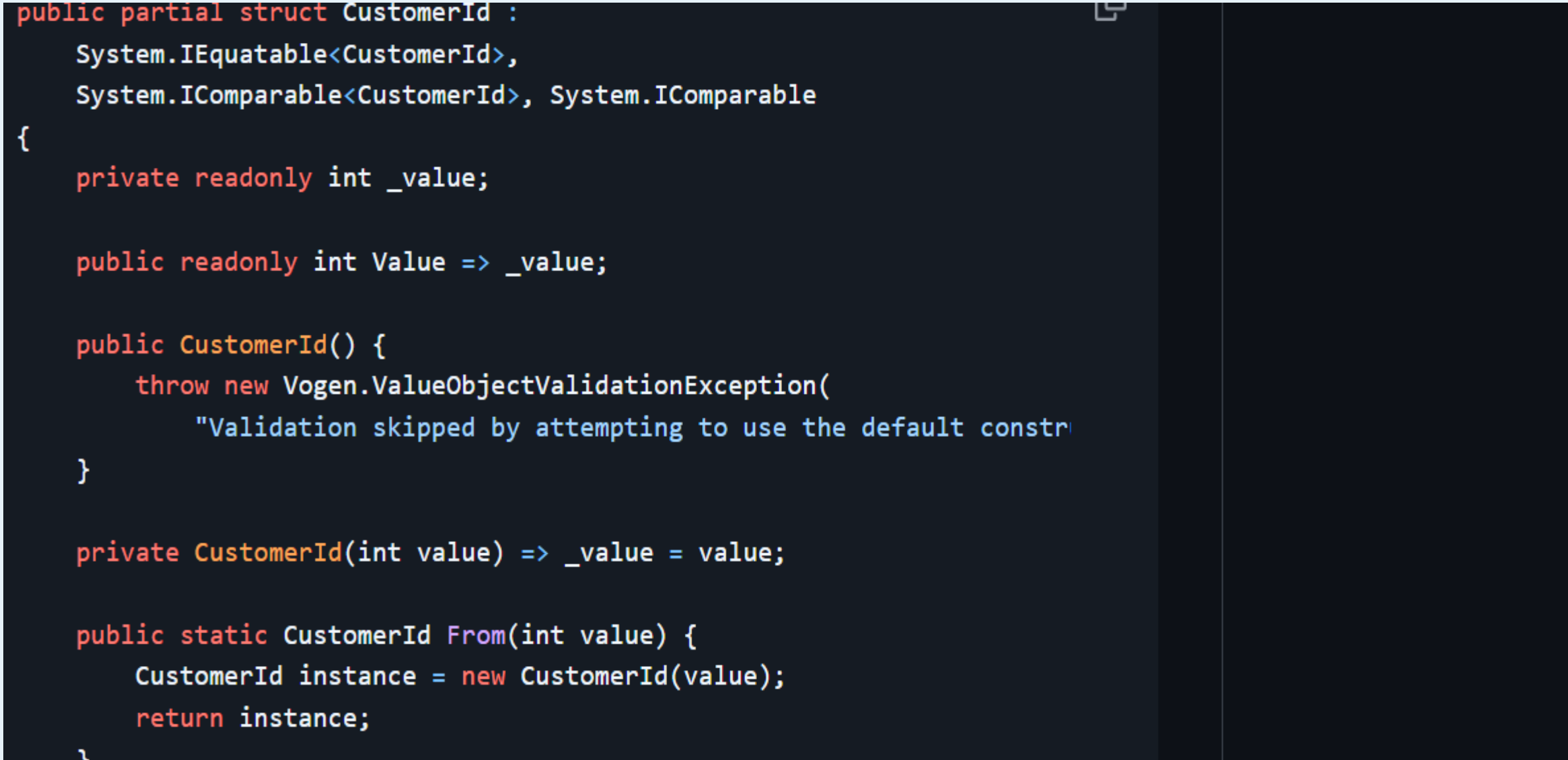
Option 2: Use Source Generation via Vogen



Add an Attribute to Your Value Object

```
[ValueObject<int>]
public partial struct CustomerId;
```

Vogen Generates Source Like This



```
public partial struct CustomerId :
    System.IEquatable<CustomerId>,
    System.IComparable<CustomerId>, System.IComparable
{
    private readonly int _value;

    public readonly int Value => _value;

    public CustomerId() {
        throw new Vogen.ValueObjectValidationException(
            "Validation skipped by attempting to use the default constr
    }

    private CustomerId(int value) => _value = value;

    public static CustomerId From(int value) {
        CustomerId instance = new CustomerId(value);
        return instance;
    }
```


WHY NOT BOTH?

```
public class Name(LegalName firstName, LegalName lastName) : ValueObject
{
    3 references | Steve Smith, Less than 5 minutes ago | 1 author, 2 changes
    public LegalName FirstName { get; private set; } = firstName;
[ValueObject<string>]
78 references | Steve Smith, Less than 5 minutes ago | 1 author, 1 change
public partial class LegalName
{
    5 references | Steve Smith, Less than 5 minutes ago | 1 author, 1 change
    private static Validation Validate(in string name) =>
        String.IsNullOrEmpty(name) ?
          Validation.Invalid("Name cannot be empty") :
          Validation.Ok;

    5 references | Steve Smith, Less than 5 minutes ago | 1 author, 1 change
    private static string NormalizeInput(string input) => input.Trim();
}
        yield return LastName;
    }
}
```

Ok but what about persistence?

Demo: Working with and Saving Value Objects

Using EF Core and Vogen

Summary

- Avoid Primitive Obsession
- No Single Point of Enforcement
- Parse, Don’t Validate
- Make Invalid States Unrepresentable (in your model)
- Think about DateTime as a great example of a Value Object
- Create composite Value Objects with a Base Type
- Create single value Value Objects with Vogen

Learn More

- Find me:
  - GitHub/ardalis
  - BlueSky: ardalis.com
  - YouTube/ardalis
  - Ardalis.com
- Training/Consulting/Help
  - NimblePros.com
- Courses on
  - Pluralsight
  - Dometrain
  - Academy.NimblePros.com
- Free SOLID Principles Email Course
  - https://nimblepros.com/email-courses
- Primitive Obsession
  - https://deviq.com/code-smells/primitive-obsession-code-smell
- Vogen
  - https://Github.com/SteveDunn/Vogen
- Demo Repo
  - https://github.com/ardalis/ValueObjectsDemo
- Clean Architecture Template/Sample
  - https://GitHub.com/ardalis/CleanArchitecture

I May Have Some Stickers/Card Decks/Swag

Thanks!

Don’t Forget To Fill Out Your Evals!