

# Elements of a Domain Model: Value Objects & Services



**Steve Smith**

Force Multiplier for  
Dev Teams

@ardalis | ardalisc.com

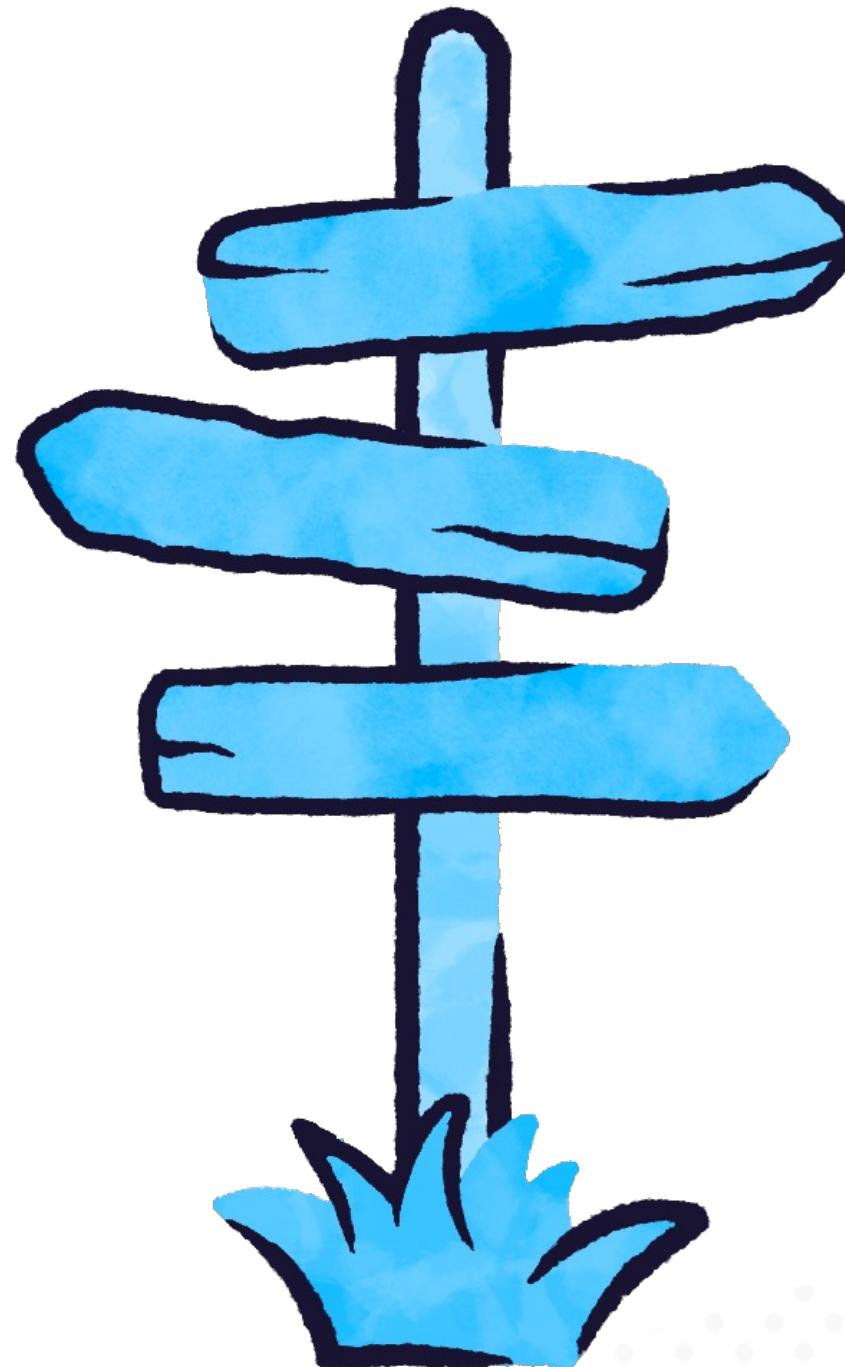


**Julie Lerman**

Software Coach,  
DDD Champion

@julielerman | thedatafarm.com

# Module Overview



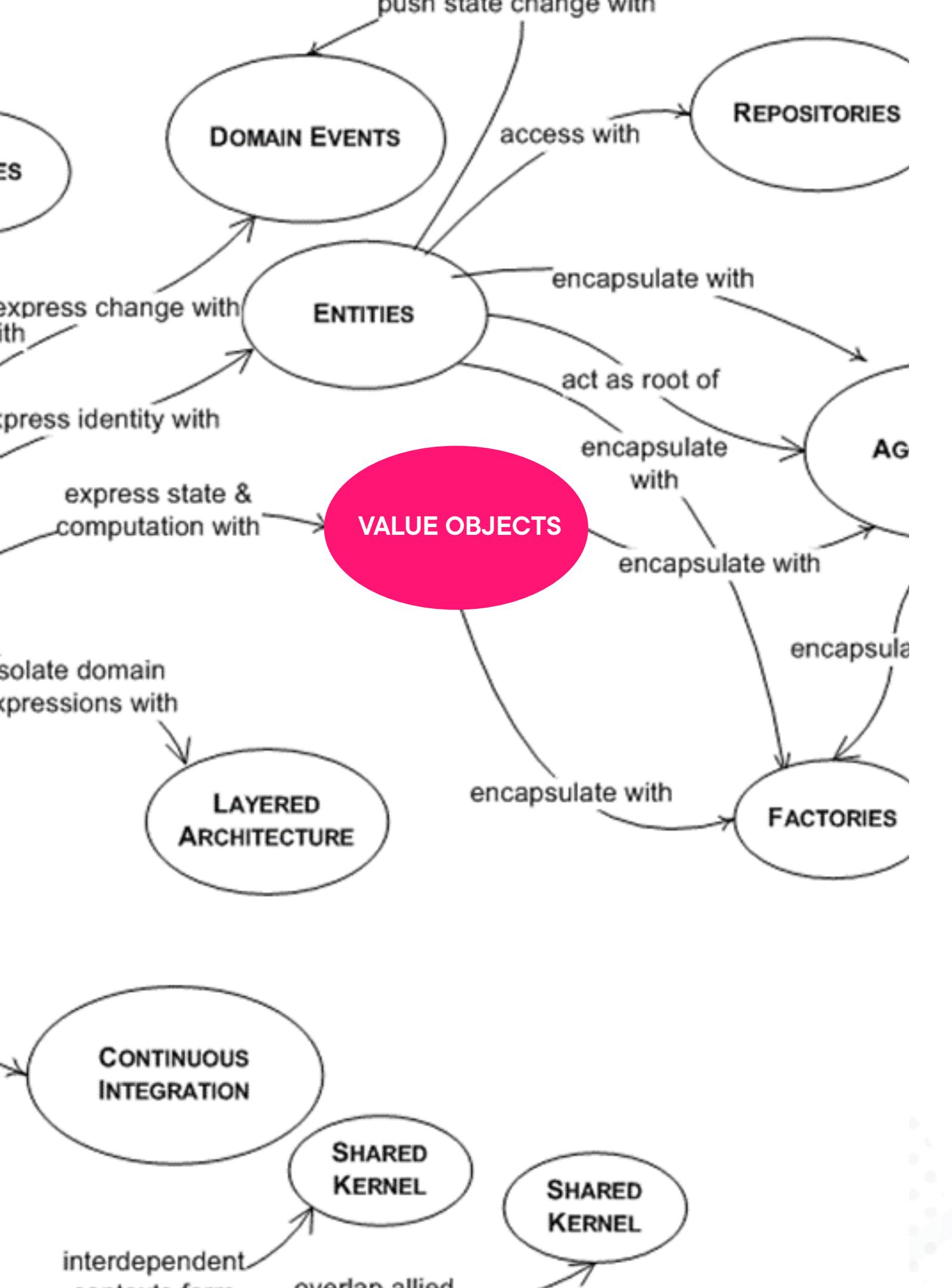
- Get acquainted with value objects**
- How entities embrace value objects**
- Advice from Eric Evans and Vaughn Vernon**
- Implement value objects in code**
- What are domain services?**
- Features & examples of domain services**

# Getting Acquainted with Value Objects

# Two Types of Objects in DDD

**Defined by an identity**

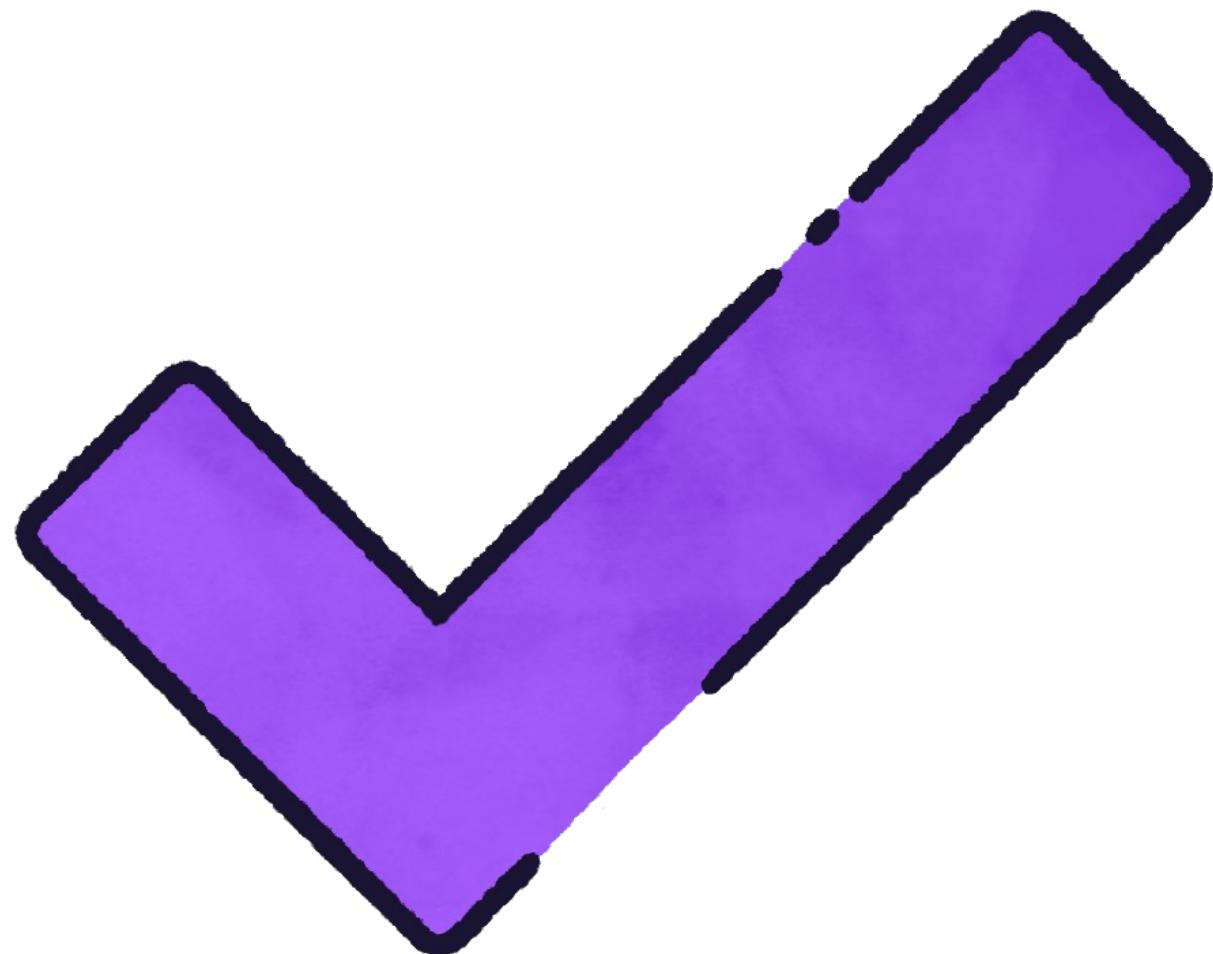
**Defined by its value**



# Value Objects in DDD

## Mind Map

# Value Object Attributes



**Measures, quantifies, or describes a thing  
in the domain**

**Identity is based on composition of values**

**Immutable**

**Compared using all values**

**No side effects**

# Putting Value Objects into Context for .NET Devs

**Value Types**

**Defined with structs**

**Reference Types**

**Defined with classes**

**DDD Entities &  
Value Objects**

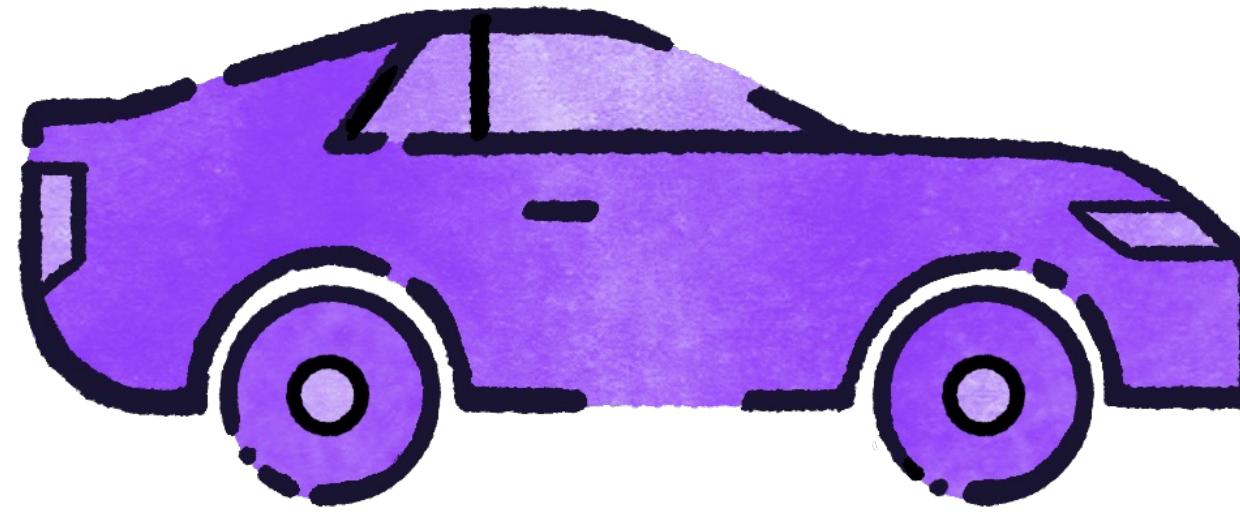
**Typically defined  
with classes**



# **Recognizing Commonly Used Value Objects**

# String is a value object.

# **String: Our Favorite Value Object**



**C A R**

# String: Our Favorite Value Object



C A T

Marlon lives with author, John Elliott

# String: Our Favorite Value Object



**S CAR**

# String: Our Favorite Value Object



**S C A R**



**D O G D E S S**

# String: Our Favorite Value Object



**S C A R**



**G O D D E S S**

# String: Our Favorite Value Object



**S C A R**



**G O D D E S S**

# String Methods Respect Immutability

## Replace (StringA, StringB)

Returns a new string in which all occurrences StringA in the current instance are replaced with StringB.

## ToUpper()

Returns a copy of this string converted to uppercase.

## ToLower()

Returns a copy of this string converted to lowercase.

# Money Is a Great Value Object

Company Worth

\$

50,000,000

# Money Is a Great Value Object

Company Worth

\$US

50,000,000

\$CN

\$AU

# Money Is a Great Value Object

Company Worth

\$US

50,000,000



# Money Is a Great Value Object

Company Worth

\$US

50,000,000

Company

- Id
- Worth Unit
- Worth Amount

# Money Is a Great Value Object

Company Worth

\$US

50,000,000

Company

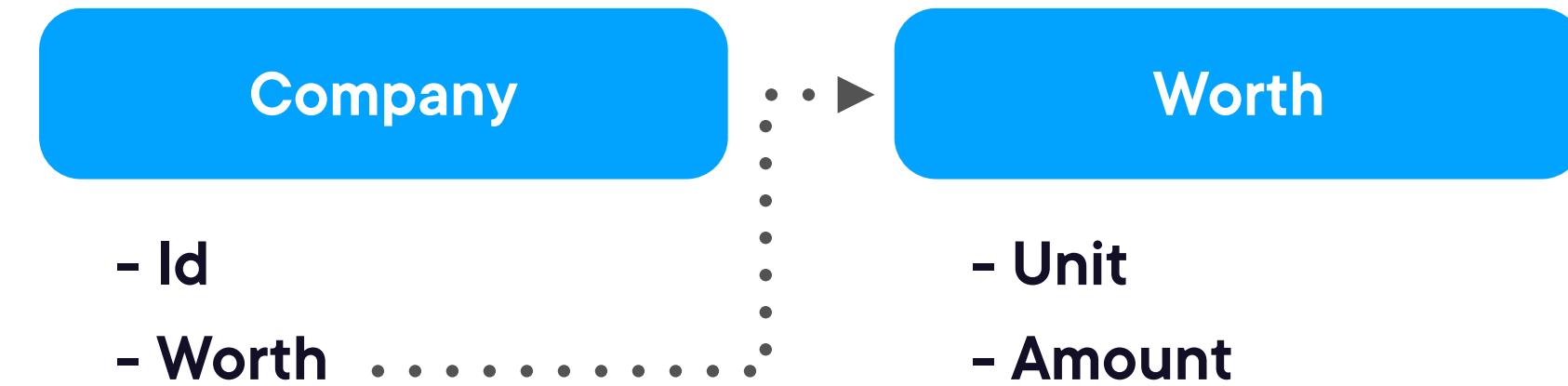
- Id
- Worth Amount

# Money Is a Great Value Object

Company Worth

\$US

50,000,000



Eric Evans

**“Dates are a classic value object and there’s all kinds of logic with them.”**

# **Date**TimeRange** as Value Object**

## **Patient Appointment**

**10:00 am Jan 4 – 11:00 am Jan 4**

## **Staff Meeting**

**2:00 pm Feb 1 – 3:15 pm Feb 1**

## **Date**TimeRange****

**Property**

**Start**

**Property**

**End**

**Constructor**

**Date**TimeRange** (start, end)**

# **DateTimeRange as Value Object**

## **Patient Appointment**

**10:00 am Jan 4 – 11:00 am Jan 4**

## **Staff Meeting**

**2:00 pm Feb 1 – 3:15 pm Feb 1**

**DateTimeRange**

**Start**

**End**

**DateTimeRange (start, end)**

**Appointment**

**- ClientId**

**- DoctorId**

**- PatientId**

**- DateTimeRange**

**Meeting**

**- RoomId**

**- StaffAttending**

**- DateTimeRange**



# **Getting More Insight from Eric Evans and Vaughn Vernon**

## Vaughn Vernon – Implementing Domain Driven Design

**It may surprise you to learn that we should strive to model using Value Objects instead of Entities wherever possible. Even when a domain concept must be modeled as an Entity, the Entity's design should be biased toward serving as a value container rather than a child Entity container.**

# When Considering Domain Objects

**Our Instinct:**

**Probably an entity**

**Maybe a value object**

**Vaughn Vernon's guidance:**

**Is this a value object?**

**Otherwise, an entity**

# Value Objects Can Be Used for Identifiers

## ClientIdValueObject.cs

```
public class ClientId
{
    public readonly Guid Id;
    public ClientId()
    {
        Id = Guid.NewGuid();
    }
    public ClientId(Guid id)
    {
        Id = id;
    }
}
```

[Equality and Hash override code]

}

# Explicit ID Value Objects Instead of Ints/GUIDs

Helps to Avoid Errors in Passed Parameters

Client.cs

```
public class Client :  
BaseEntity<ClientIdValueObject>  
{  
    . . .  
}
```

SomeService.cs

```
public class SomeService  
{  
    public void  
CreateAppointmentFor(ClientIdValueObj  
ect clientId, PatientIdValueObject  
patientId)  
    {  
        . . .  
    }  
}
```

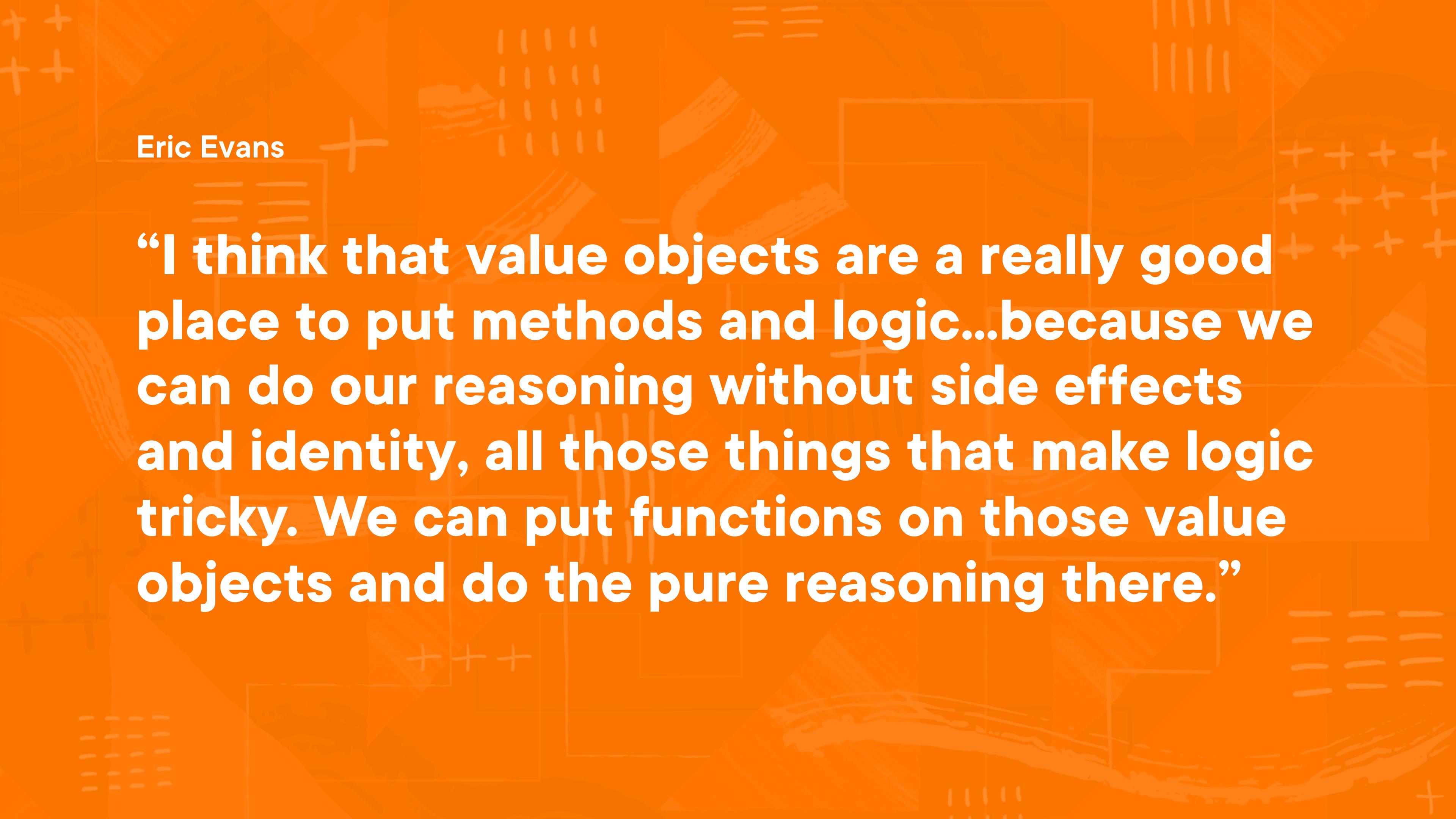
# Value Objects Can Be Used for Identifiers

## Client.cs

```
public class Client
{
    public ClientIdValueObject Id {get; set;}
}

// or

public class Client : BaseEntity<ClientIdValueObject>
{
    // Id property provided by base type
}
```



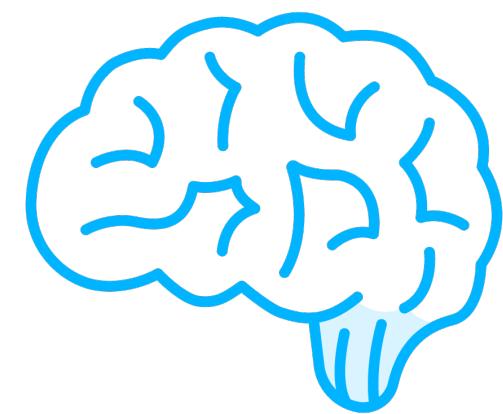
Eric Evans

**“I think that value objects are a really good place to put methods and logic...because we can do our reasoning without side effects and identity, all those things that make logic tricky. We can put functions on those value objects and do the pure reasoning there.”**

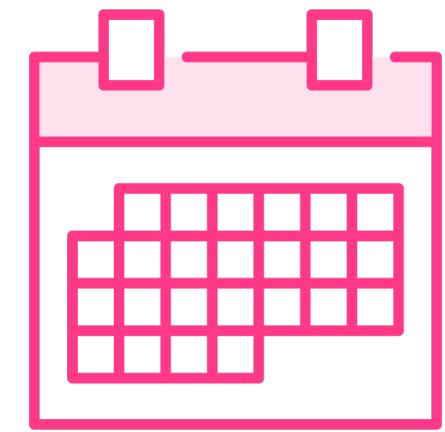
Eric Evans

**“Dates are a classic value object and there’s all kinds of logic with them.”**

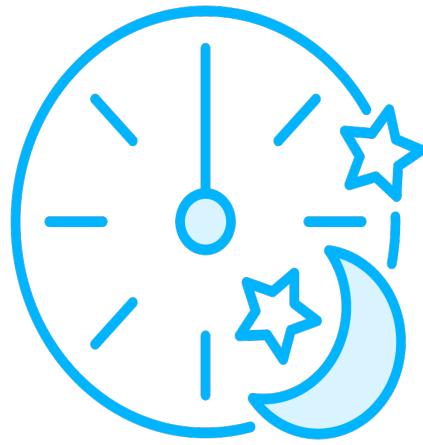
# Date Libraries as Value Object



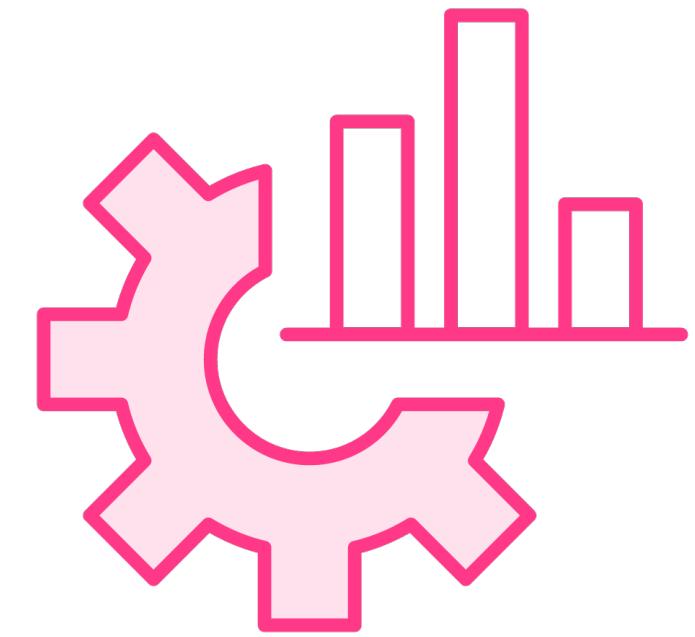
**Calculate age**



**Calculate  
date ranges**



**Perform complex  
time operations**



**... and more**

# Implementing Value Objects in Code

# Discovering Value Objects within Our Entities

## Appointment

- AppointmentId
- Start DateTime
- End DateTime
- PatientId
- ClientId
- DoctorId
- RoomId
- AppointmentTypeID

# Discovering Value Objects within Our Entities

## Appointment

- AppointmentId
- **DateTimeRange**
- PatientId
- ClientId
- DoctorId
- RoomId
- AppointmentTypeID

## DateTimeRange

- Start DateTime
- End DateTime

# Our DateTimeRange Value Object

```
public class DateTimeRange : ValueObject
{
    public DateTime Start { get; private set; }
    public DateTime End { get; private set; }

    public DateTimeRange(DateTime start, DateTime end)
    {
        // Ardalis.GuardClauses supports extensions with custom guards per project
        Guard.Against.OutOfRange(start, nameof(start), start, end);
        Start = start;
        End = end;
    }

    public DateTimeRange(DateTime start, TimeSpan duration)
        : this(start, start.Add(duration))
    {
    }

    // additional methods in next slide
}
```

**The state of a value object  
should not be changed  
once it has been created.**

# Methods in Our DateTimeRange Value Object

```
public class DateTimeRange : ValueObject
{
    // properties and constructors

    public DateTimeRange NewEnd(DateTime newEnd)
    {
        return new DateTimeRange(this.Start, newEnd);
    }

    public bool Overlaps(DateTimeRange dateTimeRange)
    {
        return this.Start < dateTimeRange.End && this.End > dateTimeRange.Start;
    }

    // used by base ValueObject type to implement equality
    protected override IEnumerable<object> GetEqualityComponents()
    {
        yield return Start;
        yield return End;
    }
}
```

# .NET DateTime Methods Work in a Similar Way

The screenshot shows a screenshot of the Microsoft Learn API browser. At the top, there's a navigation bar with links for .NET, Languages, Features, Workloads, APIs, Troubleshooting, and Resources. Below the navigation is a search bar with dropdowns for Version (.NET 8) and Search. The main content area has a breadcrumb trail: Learn / .NET / API browser / System / DateTime / Methods. On the left, there's a sidebar with a tree view of DateTime methods. A blue box highlights the `AddHours` method. The main content area has tabs for Reference and Definition. The Definition tab shows the namespace as System, assembly as System.Runtime.dll, and source as DateTime.cs. It describes the method as returning a new `DateTime` that adds the specified number of hours to the value of this instance. Below this is a C# code block with a copy button. The Parameters section defines `value` as a Double, describing it as a number of whole and fractional hours. The Returns section says it returns a `DateTime` object. The Exceptions section mentions `ArgumentOutOfRangeException` for values less than `DateTime.MinValue` or greater than `DateTime.MaxValue`.

Version  
.NET 8  
Search

Learn / .NET / API browser / System / DateTime / Methods /

C#

## DateTime.AddHours(Double) Method

Reference  Feedback

### Definition

Namespace: [System](#)  
Assembly: [System.Runtime.dll](#)  
Source: [DateTime.cs](#)

Returns a new [DateTime](#) that adds the specified number of hours to the value of this instance.

C#  Copy

```
public DateTime AddHours (double value);
```

### Parameters

**value** [Double](#)  
A number of whole and fractional hours. The `value` parameter can be negative or positive.

### Returns

[DateTime](#)  
An object whose value is the sum of the date and time represented by this instance and the number of hours represented by `value`.

### Exceptions

[ArgumentOutOfRangeException](#)  
The resulting `DateTime` is less than `DateTime.MinValue` or greater than `DateTime.MaxValue`.

Citation: <https://learn.microsoft.com/en-us/dotnet/api/system.datetime.addhours?view=net-8.0>

# Our Animal Type Value Object

```
public class AnimalType : ValueObject
{
    public string Species { get; private set; }
    public string Breed { get; private set; }

    public AnimalType(string species, string breed)
    {
        Species = species;
        Breed = breed;
    }

    // used by base ValueObject type to implement equality
    protected override IEnumerable<object> GetEqualityComponents()
    {
        yield return Breed;
        yield return Species;
    }
}
```



**Can you share advice about moving logic out  
of entities into value objects?**

**Generic logic  
makes sense in  
value objects**



**It's easier to test  
logic that's in a  
value object**

**Entity becomes  
an orchestrator**

**Higher level of  
abstraction in entities**



**More precise  
ubiquitous language**

# Working Towards a More Concise Language

**Extract primitives  
from an entity**



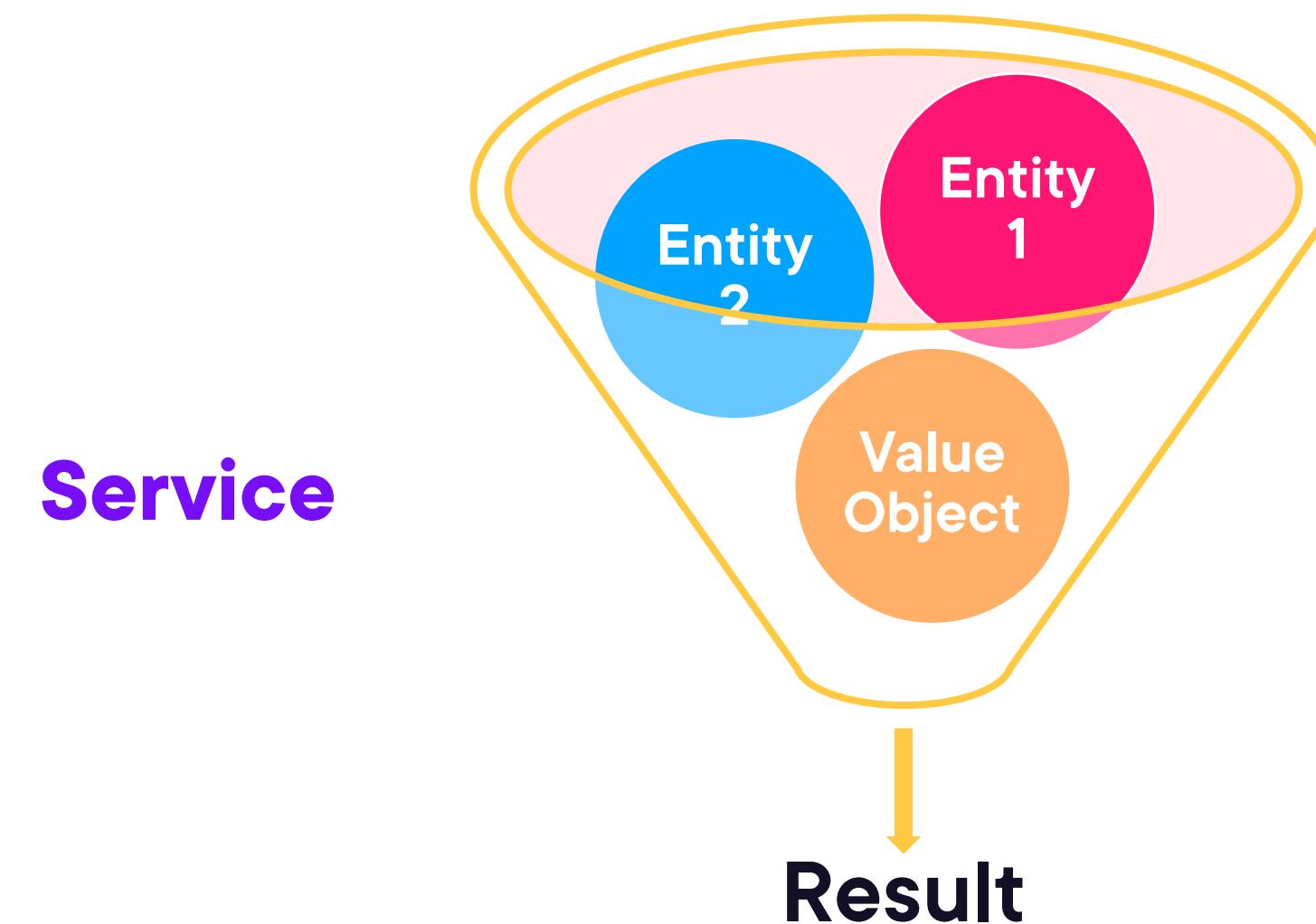
**into a value object**



# Understanding Domain Services

**Some operations make more  
sense in a domain service.**

# Domain Service Orchestrates Processes Across Objects



# Features of a Domain Service

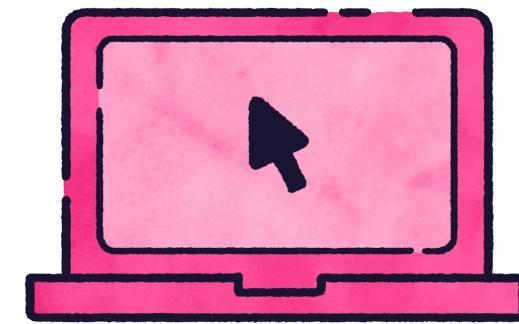
**Not a natural part of  
an entity or value object**

**Has an interface defined  
in terms of other domain  
model elements**

**Stateless, but may  
have side effects**

**Lives in the core of  
the application**

# Examples of Services in Different Layers



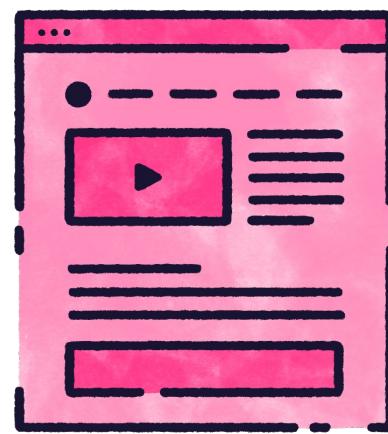
**UI and App**

**Message sending**

**Message processing**

**XML parsing**

**UI services**



**Domain**

**Orchestrating workflow**

**Transfer between accounts**

**Process order**



**Infrastructure**

**Send email**

**Log to a file**



# **Module Review and Resources**

# **Immutable**

**Refers to a type whose state cannot be changed once the object has been instantiated**

# **Value Object**

**An immutable class whose identity is dependent on the combination of its values**

---

# Domain Services

Provide a place in the model to hold behavior that doesn't belong elsewhere in the domain

# Side Effects

Changes in the state of the application or interaction with the outside world (e.g., infrastructure)

---

# Key Takeaways



- Value objects are used to measure, quantify, or describe**
- They are used as a property of an entity**
- They are identified by the composition of their values**
- Value objects are immutable and should have no side effects**
- Strings and dates are great examples of value objects**
- Domain services orchestrate across different parts of the domain model**
- Watch out for overuse of domain services!**

# **Build aggregates from entities and value objects**

# Resources Referenced in This Module

**Domain-Driven Design in C#9: Immutable Value Objects**  
Julie Lerman on Pluralsight blog - [bit.ly/PSBlogValueObjects](https://bit.ly/PSBlogValueObjects)

**Support for Value Objects in C#, Steve Smith blog post -**  
[ardalis.com/support-for-value-objects-in-csharp](https://ardalis.com/support-for-value-objects-in-csharp)

**Vaughn Vernon -** [vaughnvernon.com](https://vaughnvernon.com)

**Eric Evans -** [domainlanguage.com](https://domainlanguage.com)