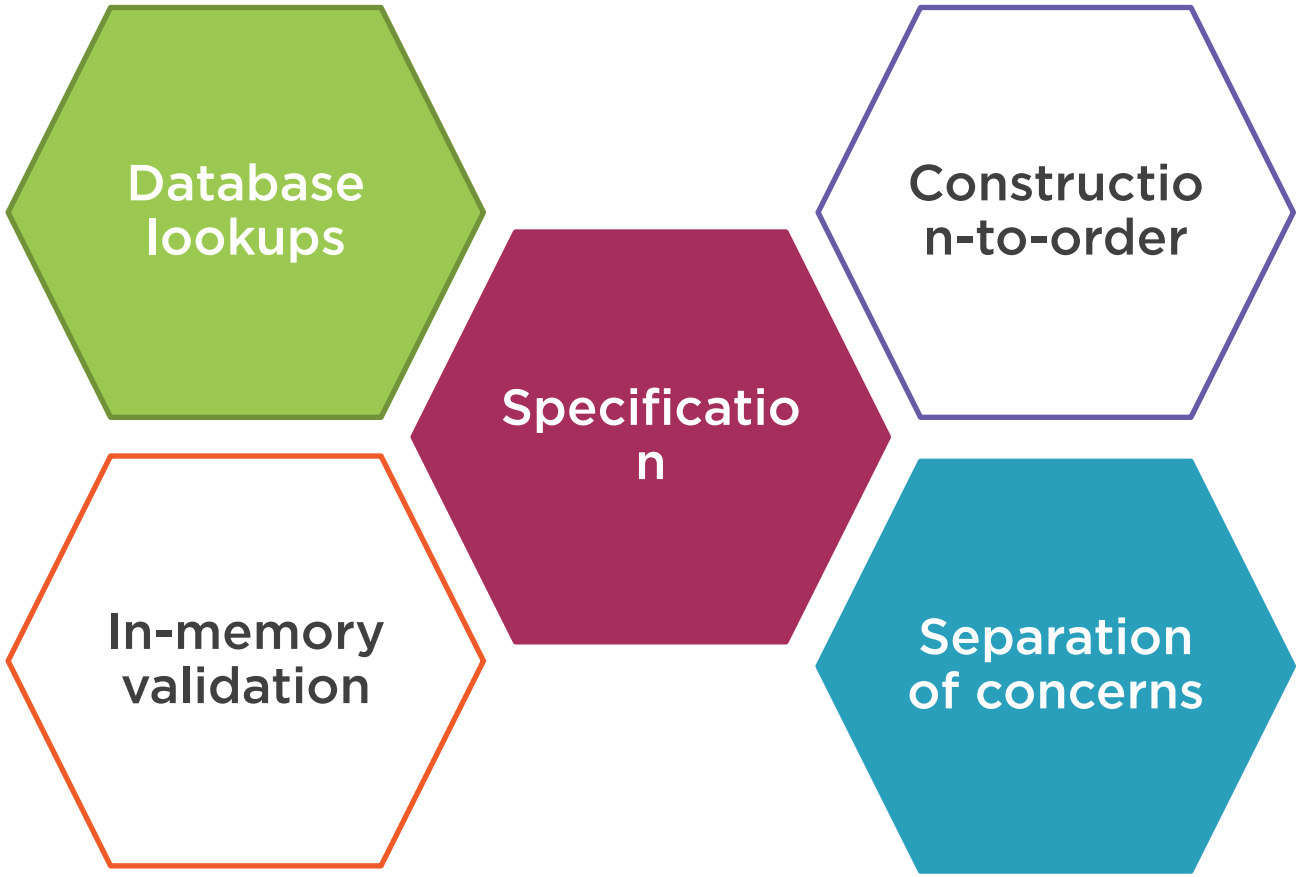
Specification Pattern in C#

INTRODUCTION

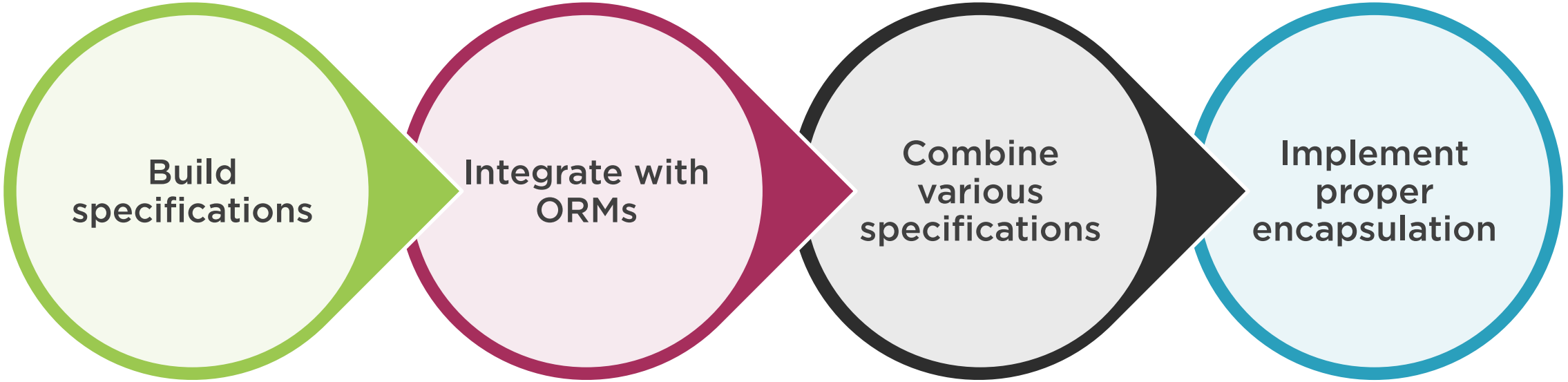
Vladimir Khorikov
PROGRAMMER

@vkhorikov www.enterprisecraftsmanship.com

Specification Pattern



The Purpose of This Course



Overview

Introduction

Implementing the Specification Pattern the Naive Way

Refactoring Towards Better Encapsulation

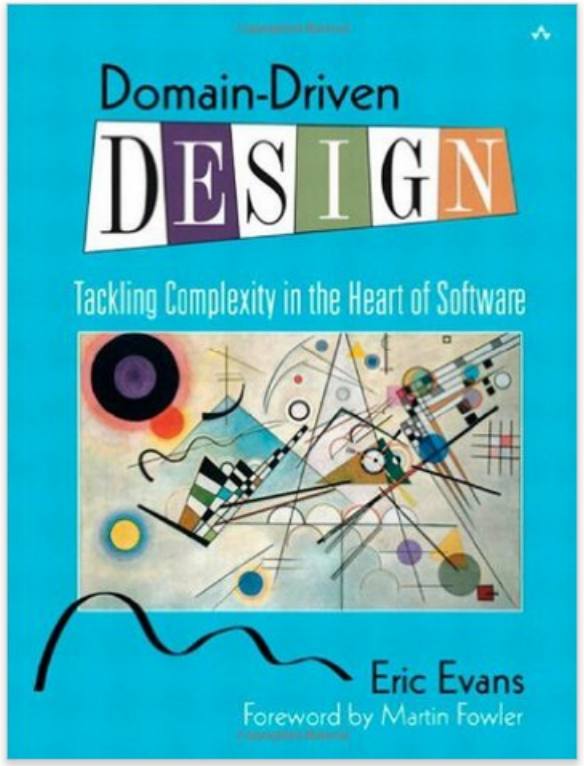
What is the Specification Pattern?

Avoiding domain knowledge duplication

Declarative approach

http://bit.ly/spec-pattern

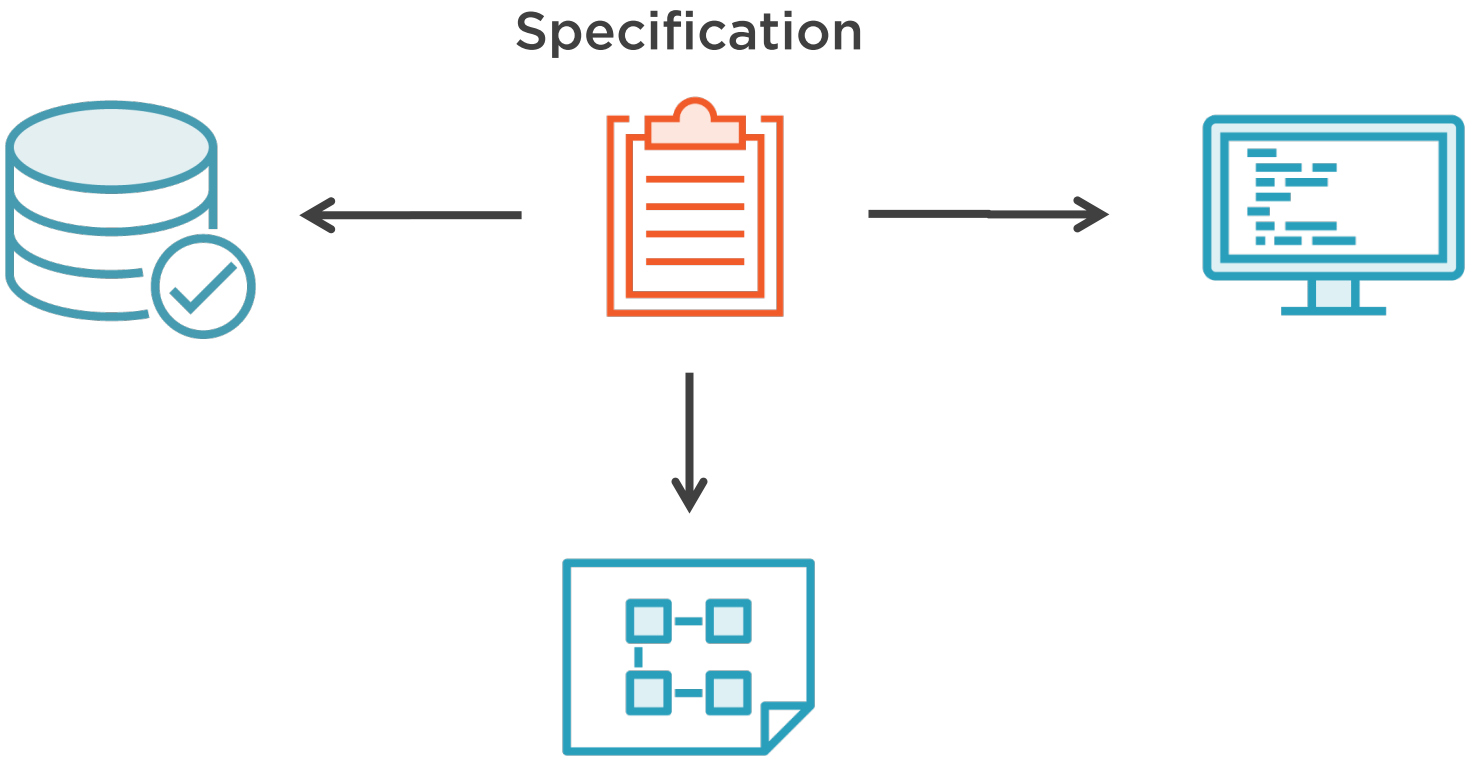
What is the Specification Pattern?



Domain-Driven Design: Tackling Complexity in the Heart of Software

By Eric Evans

What is the Specification Pattern?



Main Use Cases

In-memory validation

Retrieving data from the database

Construction-to-order

Sample Project Introduction

- Look at a sample project
- Implement in-memory validation and data retrieval the old fashioned way
- Discuss the drawbacks of this implementation
- Refactor using the specification pattern

Domain-Driven Design in Practice

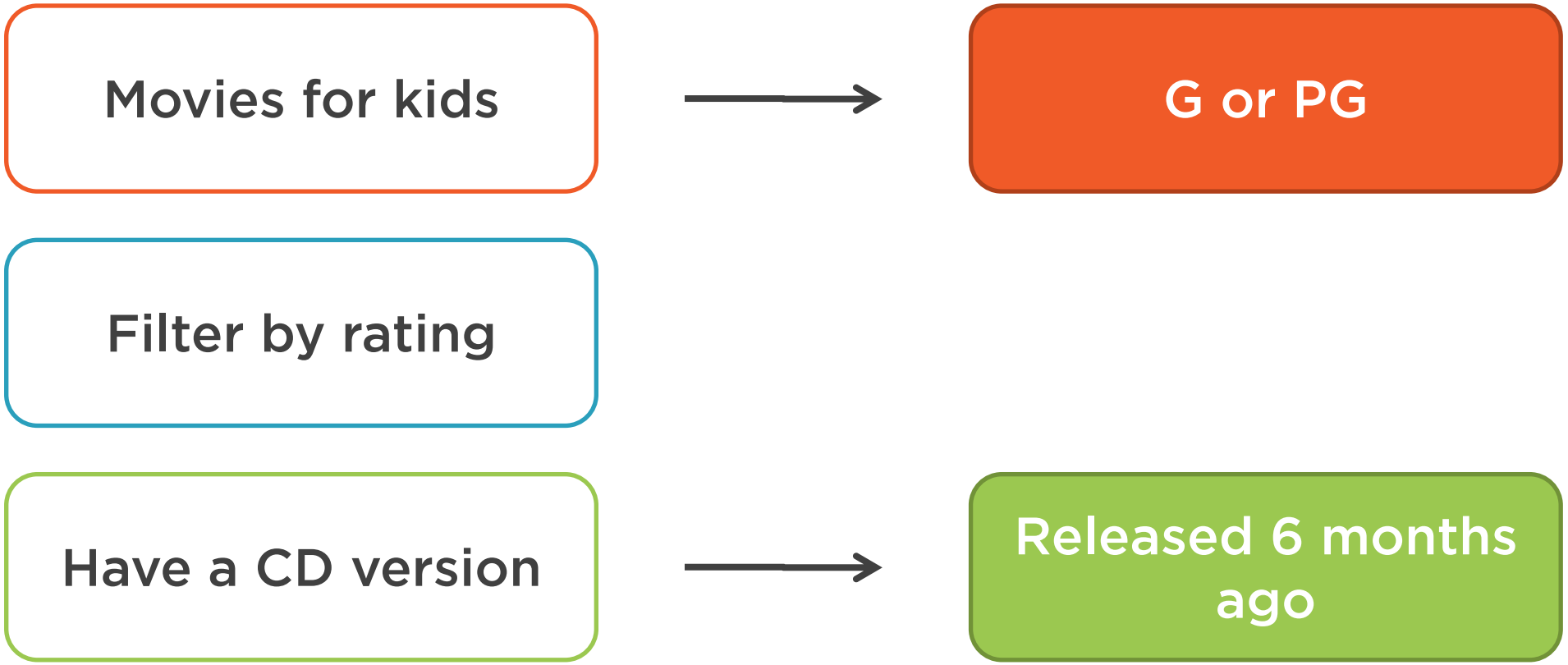
by Vladimir Khorikov

A descriptive, in-depth walk-through for applying Domain-Driven Design principles in practice.

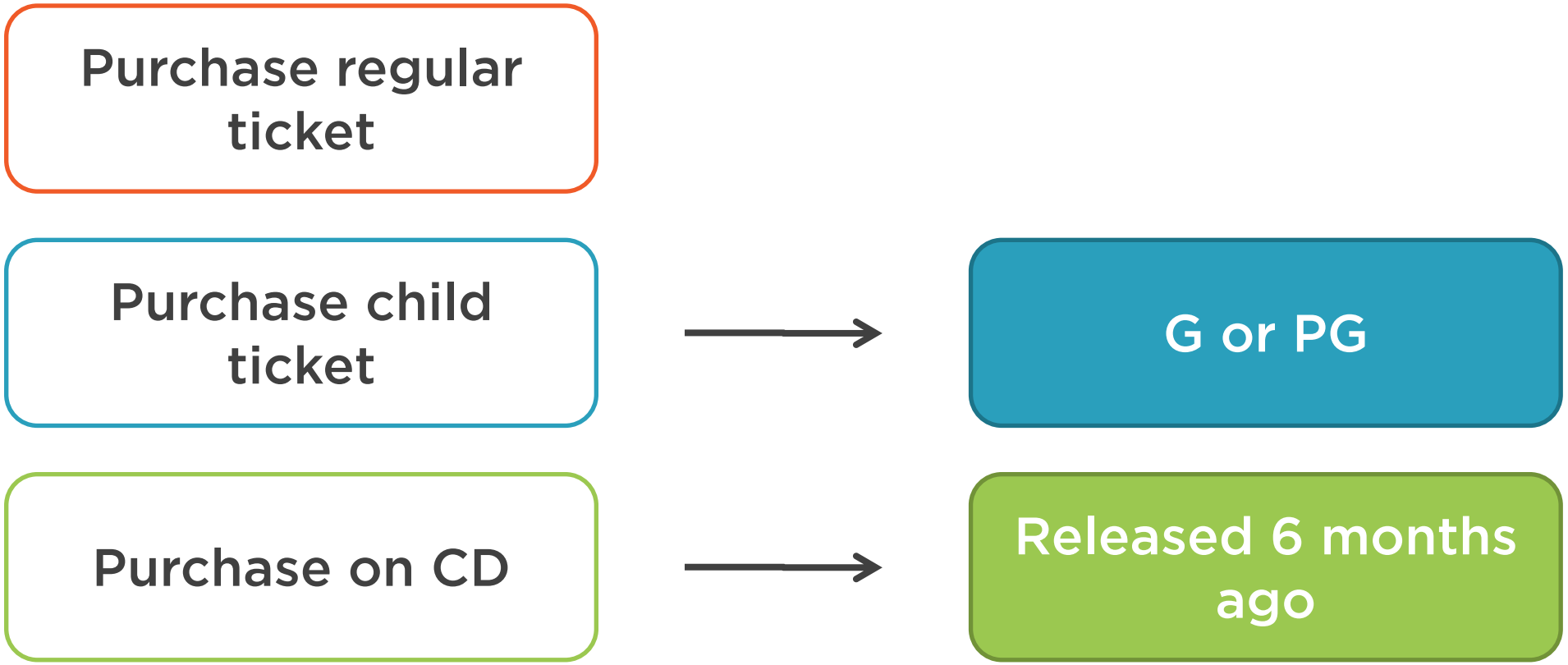
▶ Resume Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Recommended

Adding New Search Options



Adding New Purchase Options



Applying Functional Principles in C#

by Vladimir Khorikov

Functional programming in C# can give you insight into how your programs will behave. You'll learn the fundamental principles that lie at the foundation of functional programming, why they're important, and how to apply them.

▶ Resume Course

Table of contents | Description | Transcript | Exercise files | Discussion | Recommended

Ex

Code Overview

Data retrieval

A kids movie

Has CD version

Input validation

A kids movie

Has CD version

Code Overview

```
public IReadOnlyList<Movie> GetList(
    bool forKidsOnly,
    double minimumRating,
    bool availableOnCD)
{
    using (ISession session = SessionFactory.OpenSession()) {
        return session.Query<Movie>()
            .Where(x =>
                (x.MpaaRating <= MpaaRating.PG || !forKidsOnly) &&
                x.Rating >= minimumRating &&
                (x.ReleaseDate <= DateTime.Now.AddMonths(-6) || !availableOnCD))
            .ToList();
    }
}
```

```
if (movie.MpaaRating > MpaaRating.PG)
{
    MessageBox.Show("The movie is not suitable for children", "Error",
        MessageBoxButton.OK, MessageBoxImage.Error);
    return;
}
```

Code Overview

Duplicate the domain knowledge

Damage performance

Specification pattern

Summary

Purpose of the specification pattern:

- Encapsulate domain knowledge into a single unit
- Reuse in various scenarios

Use cases:

- In-memory validation
- Retrieving data from the database
- Creation of new objects

Worked on a sample project

- Search functionality
- New purchase options
- Couldn't reuse domain knowledge efficiently
- Had to either introduce duplication, or rely on inefficient SQL queries

In the Next Module

Implementing the Specification Pattern the Naive Way