## Pushing Logic Down from Services to Domain Classes



Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com



## Recap: Refactoring Customer's Constructor and Collection

Introduced a constructor

**Encapsulated the collection** 

Reduced the number of mutating APIs



### Recap: Refactoring Customer Status



### Identified the concept of Customer Status

```
public CustomerStatusType Type;
public ExpirationDate;
Value Object
```



### Build up abstractions on top of other abstractions

```
public bool IsAdvanced =>
    Type == CustomerStatusType.Advanced && !ExpirationDate.IsExpired;
```



### Pushing Logic Down: Promote Customer

Refactoring towards rich domain model

**Proper encapsulation** 

Pushing domain logic to entities and value objects



### Recap: Pushing Logic Down from Services to Entities

Use cases for domain services

Referring to external resources

Logic that doesn't naturally fit any entity or value object



### Summary



### Transformed the anemic domain model into a rich, fully encapsulated one

- No domain class should be able to reside in an invalid state
- Move all operations upon domain classes to those classes themselves

Used constructors to ensure all new instances are valid

Encapsulated the work with collections

Introduced a new abstraction: customer status

Pushed domain logic from services to entities

- Reduced the number of mutating APIs

Refactored the Movie class into a hierarchy of classes

#### In the Next Module

# Organizing the Application Services Layer

