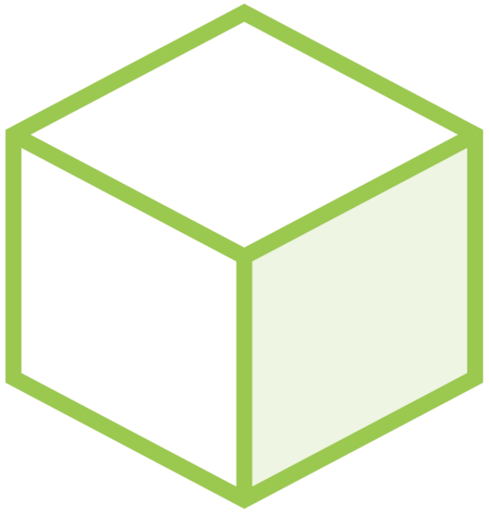
Modeling Missing Objects

Zoran Horvat
CEO at Coding Helmet

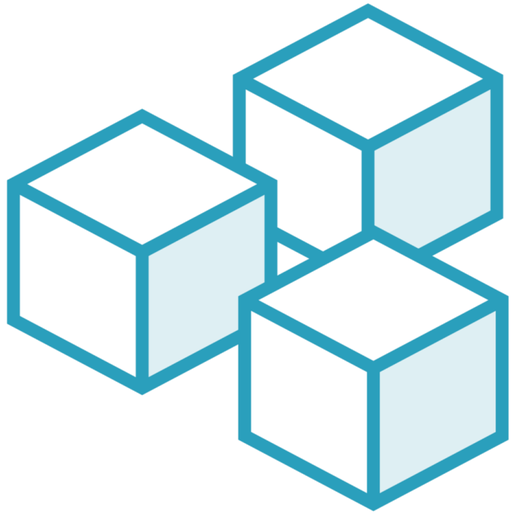
@zoranh75 https://codinghelmet.com

Type Instances at Run Time



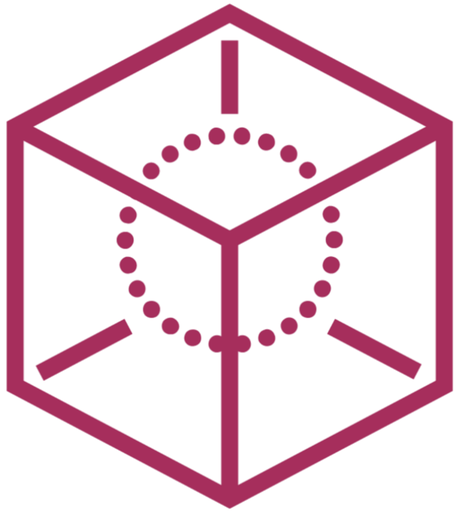
A single instance

This is the common use case of any type



Multiple instances

Usually in collections or sequences



No instance

Indicated as a missing object

```
5  namespace TestPersistence;
6
7  public class Inventory : IReadOnlyRepository<(Part part, DiscreteMeasure quantity)>
8  {
9      private PartsReadRepository Parts { get; } = new();
10
11     public IEnumerable<(Part part, DiscreteMeasure quantity)> TryFind(Guid id) =>
12         this.Parts.TryFind(id).Where(Exists).Select(part => (part, QuantityFor(part)));
13
14     private static DiscreteMeasure QuantityFor(Part part) =>
15         new("Piece", Exists(part) ? 1U : 0);
16
17     public IEnumerable<(Part part, DiscreteMeasure quantity)> GetAll() =>
18         this.Parts.GetAll().Where(Exists).Select(part => (part, QuantityFor(part)));
19
20     private static bool Exists(Part part) => part.Sku.Value[part.Sku.Value.Length / 2] % 5 == 2;
21 }
22
```

A method operating on optional objects

```
public IEnumerable<(Part part, DiscreteMeasure quantity)> TryFind(Guid id) =>
    this.Parts.TryFind(id).Where(Exists).Select(part => (part, QuantityFor(part)));
```

An optional part
from the parts repository

An optional part
from the inventory

An optional quantity
of a part in the inventory

Summary

Optional objects in C#

- Not another null
- Functional modeling tool
- Often used as a return value

Summary

Primitive functions defined on Option<T>

- `Map` applies a common function to an optional object
- `Filter` possibly filters out an optional object
- `Reduce` substitutes a default for a missing object
- Correspond to LINQ `Select`, `Where`, and `SingleOrDefault`

Summary

Using optional objects

- You will never reach out for `null` in domain-related code
- That also stands in object-oriented code
- There are no current plans to incorporate it in C#

Accept the optional mindset!

Up Next:
Modeling Complex Domain Objects