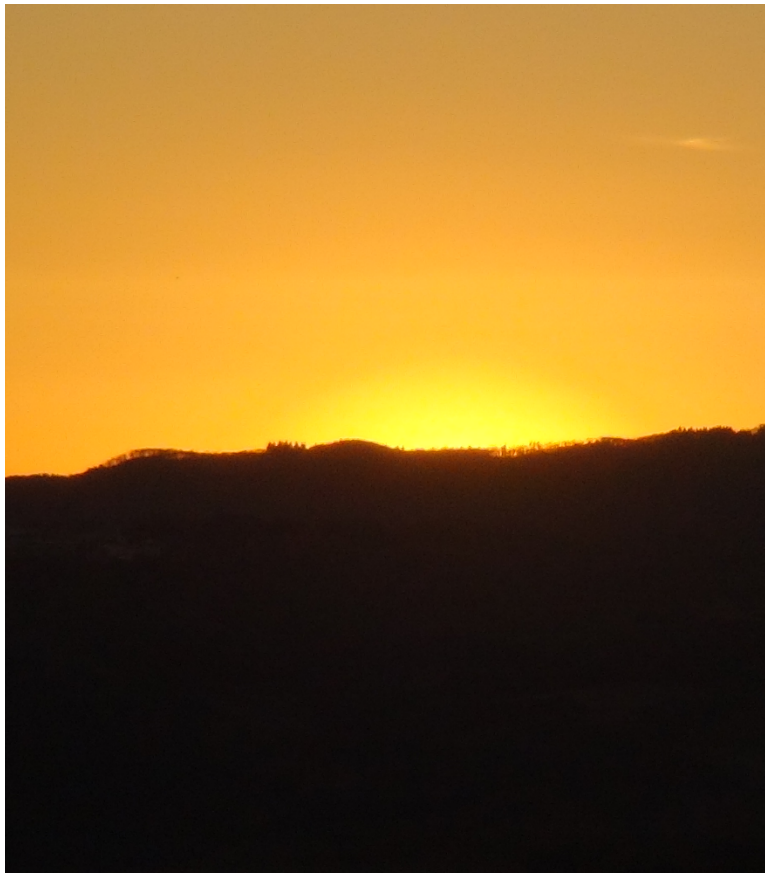
The Principle of Working with Objects, Not with Nulls

Zoran Horvat

PRINCIPAL CONSULTANT AT CODING HELMET

@zoranh75 csharpmentor.com

The Dawn of Null



Null used to help:

- Make languages straight-forward
- Simplify compilers
- Reduce model complexity in code

Null Today

Like null has lived longer than necessary…

Null in modern application development

- Test-driven Development
- Domain-driven Design
- Object-oriented programming
- Functional programming
- It’s all null-driven development in the end

`NullReferenceException`-driven Development

“Application is well-tested when there are no more `NullReferenceException`s.”

Anonymous

“Your code is bad, I’ve been testing it and I saw a `NullReferenceException`.”

Anonymous

“Something’s going on, we’re reading high `NullReferenceException`s/hour.”

Anonymous

Programming Without Null

Lots of benefits

- No null checks – shorter code
- No null exceptions – more stability

Disciplined coding without null

- But null still exists in the language

Richard Cobbe

Much Ado About Nothing: Putting Java's Null in Its Place

Linguistically, this null value serves two major purposes.

First, it serves as a value for uninitialized fields, guaranteeing that references to those fields are type-safe.

Second, programmers use it to encode a "missing" value or result [...]

`http://www.ccs.neu.edu/racket/pubs/dissertation-cobbe.pdf`

Anders Hejlsberg interview, October 2008

The A-Z of Programming Languages: C#

Fifty percent of the bugs that people run into today, coding with C# in our platform, and the same is true of Java for that matter, are probably null reference exceptions.

https://www.computerworld.com.au/article/261958/a-z_programming_languages_c_/?pp=3

Two Natures of the Null

Garbage-collected languages

Code which never sets a reference to null can still produce a null reference

Null can appear inside a partially initialized object

Used by the garbage collector to figure which references to follow

Common programmers

Use null to indicate missing objects

All references must be guarded if they can ever be set to null

Forgetting to do so equals a bug

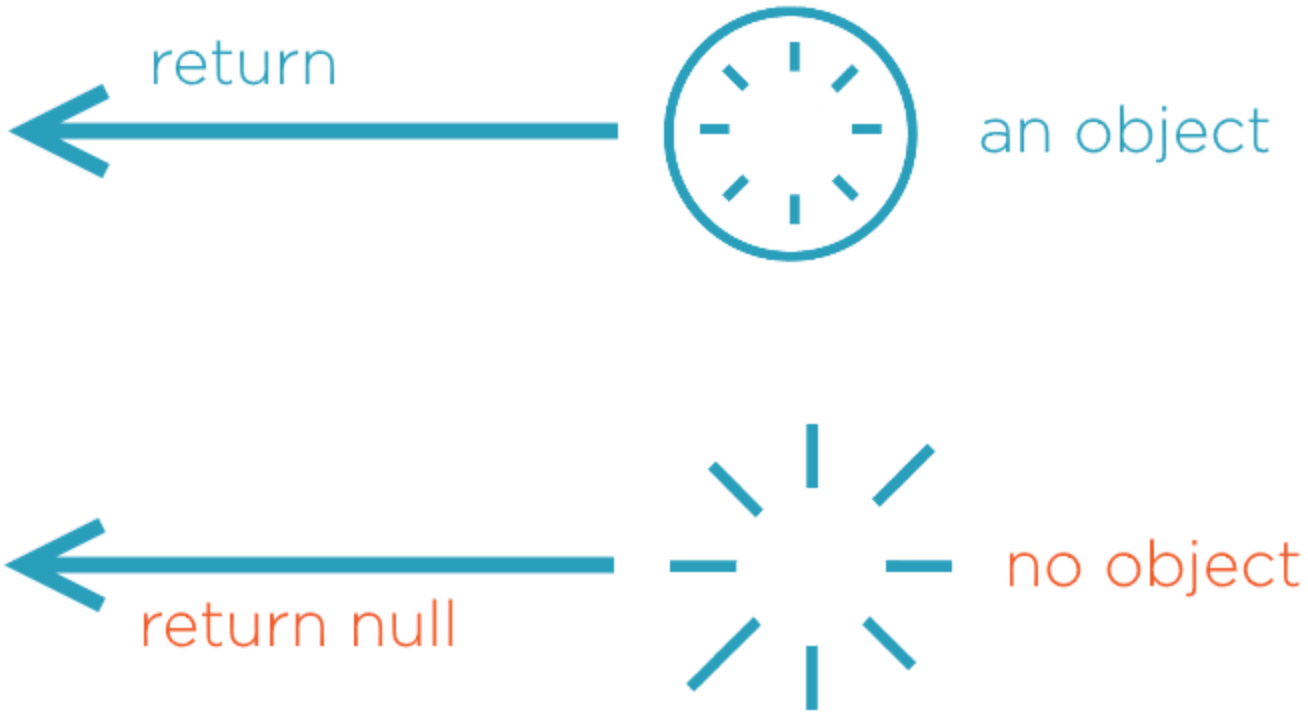
Null is not an object.

The principle of working with objects, not with nulls

Dealing with Missing Objects

An alternative

Callback pattern



Dealing with Missing Objects

An object is never
just missing

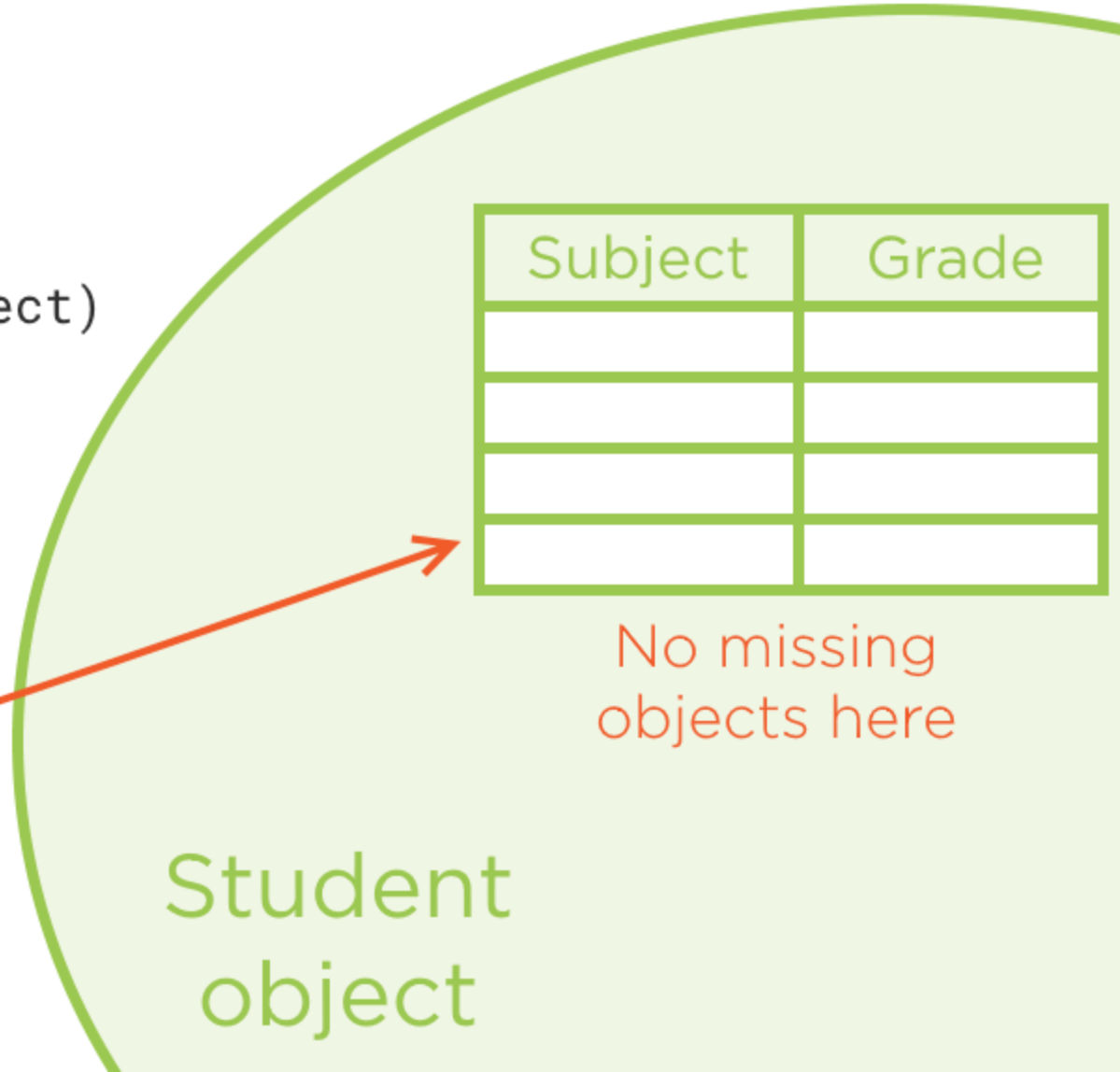
Containing
object

Dealing with Missing Objects

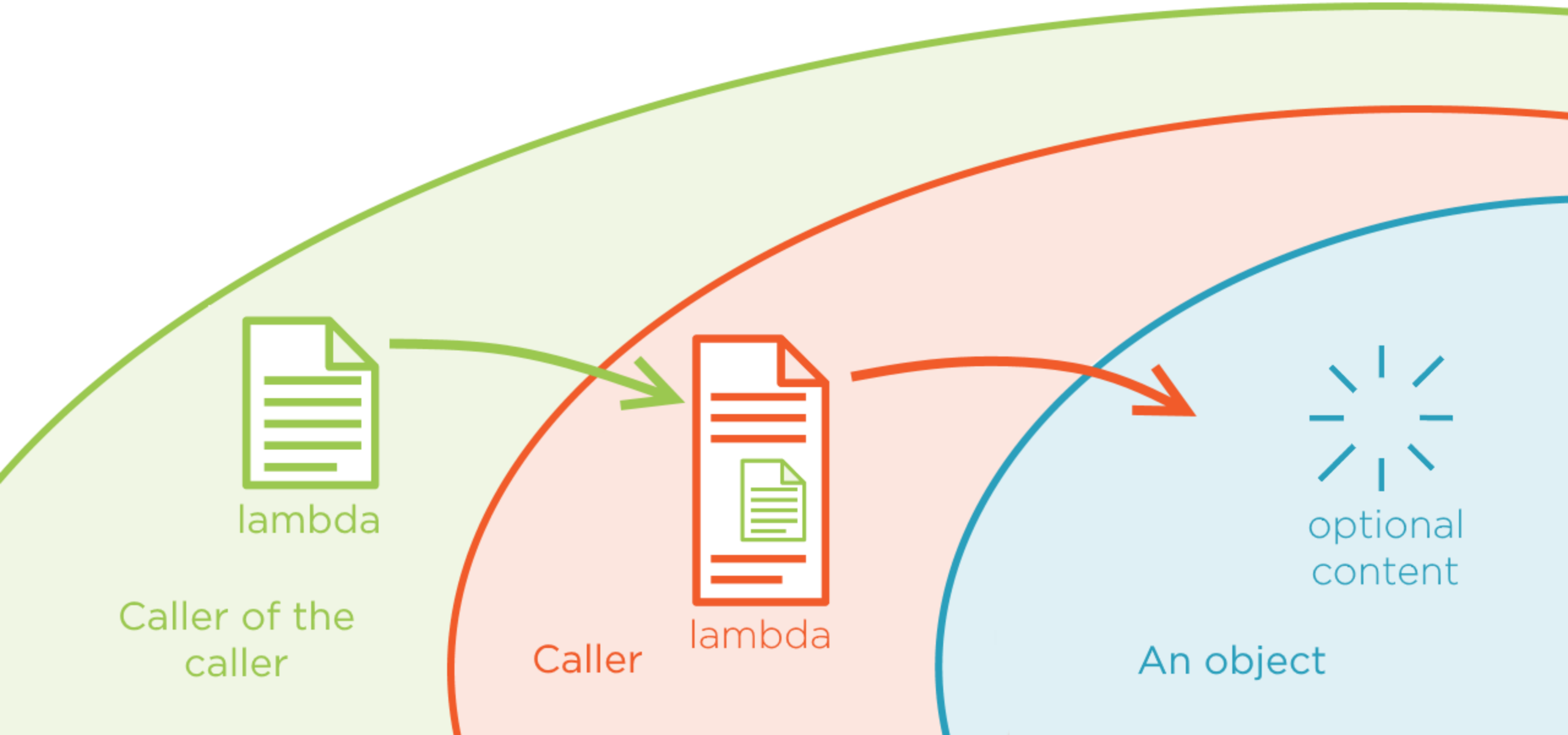
```
public class Student
{
   ...
   public Grade GetGrade(Subject subject)
   {
      ...
   }
}
```

Promises to return a grade ...

... from a dictionary which doesn't contain it



The Downside of the Callback Pattern



The Downside of the Callback Pattern

Good in small portions

Doesn't scale well
across classes and layers

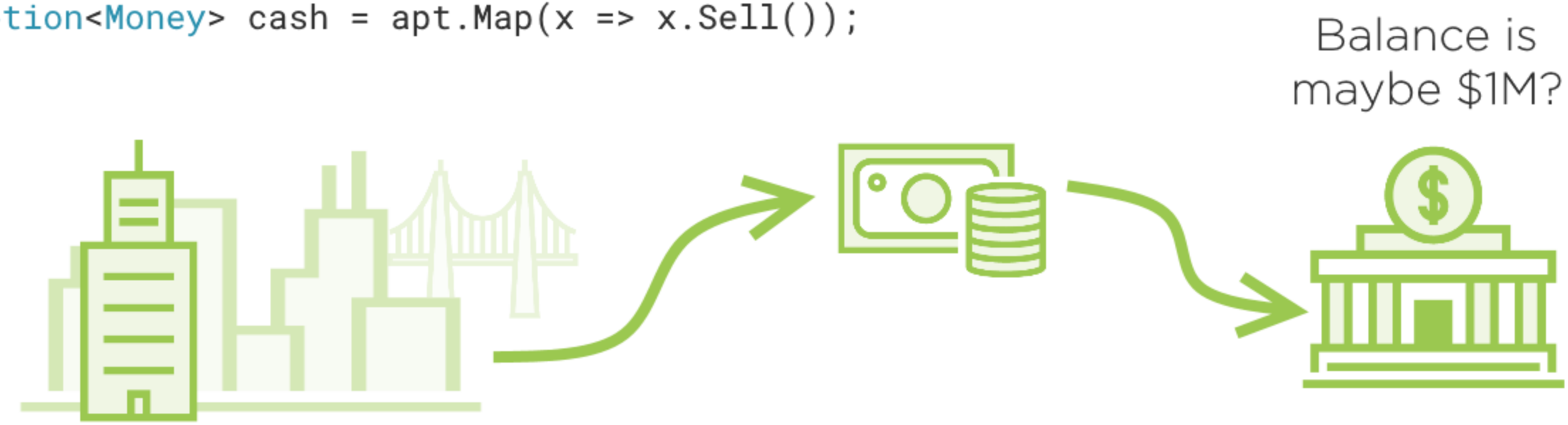
Which is fine,
as small problems
are so common

We need better
representation of
optional objects

Merits of Working with Optional Objects

```
class Seller
{
  Option<Apartment> GetOne(string location) { ... }
}

Option<Apartment> apt = seller.GetOne("Manhattan");
Option<Money> cash = apt.Map(x => x.Sell());
```



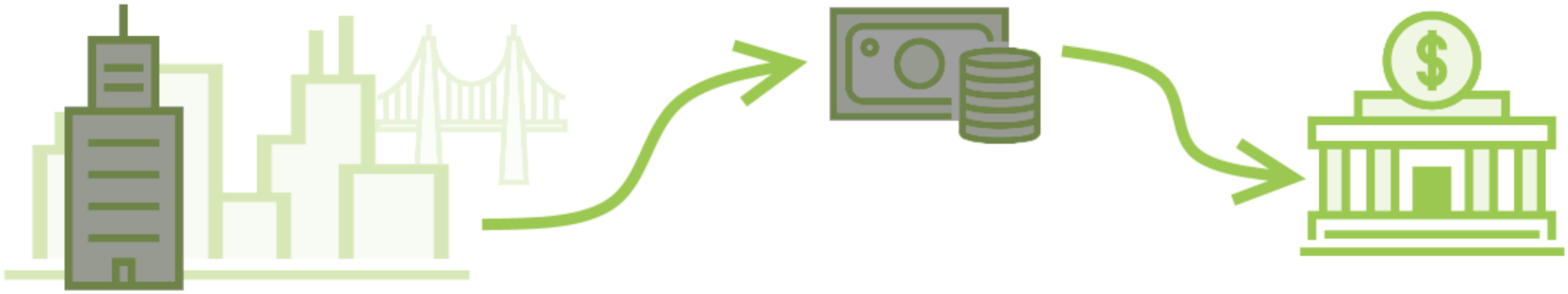
Merits of Working with Optional Objects

```
class Seller
{
  Option<Apartment> GetOne(string location) { ... }
}

Option<Apartment> apt = Option.None<Apartment>();
Option<Money> cash = Option.None<Money>();
```

Balance is zero
(not None,
but true zero)

~~Balance is maybe $1M?~~



Summary

Why the null?

- Internal concept in garbage-collected programming languages
- It can be avoided in day-to-day programming

How to understand missing objects?

- Object is always missing inside another object

Summary

Callback pattern

- Executes only when object is there
- Desperately lacks readability

Option type

- Instance is either some or none
- Method can return an option
- Return type communicates intention

Next module

Building Rich Domain Model