Making Your C# Code More Functional

Zoran Horvat
CEO AT CODING HELMET

@zoranh75 http://csharpmentor.com

Functional Programming

How Hard Can It Be?

Higher-order function – Receives a function, returns a function, or both

Pure function – No side effects, same result for same argument values

Lazy evaluation – Expression not evaluated before result is required

Pattern matching – Match object against patterns as flow control

Value type – Object never changes and behaves like a plain value

Referential transparency – Expression can be replaced with its value

Functional Programming

How Hard Can It Be?

Lazy evaluation

Pattern matching

Pure functions

Value types

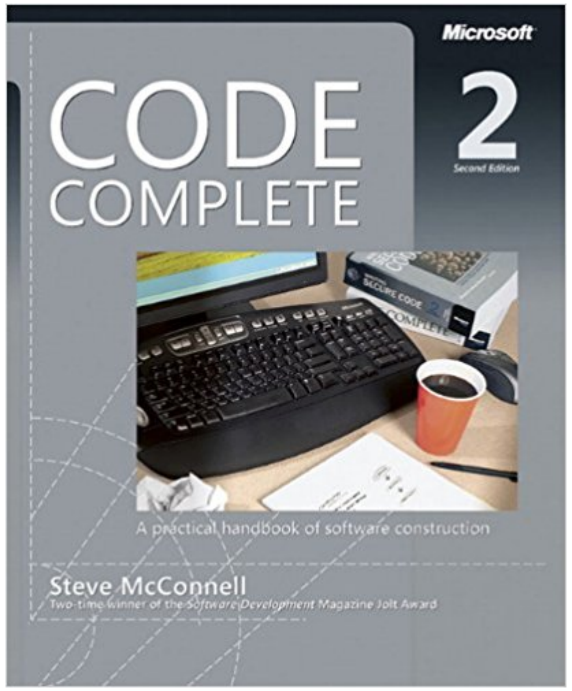
Higher-order functions

Referential transparency

C#

Primarily object-oriented language

Programming into a Language, Not in It

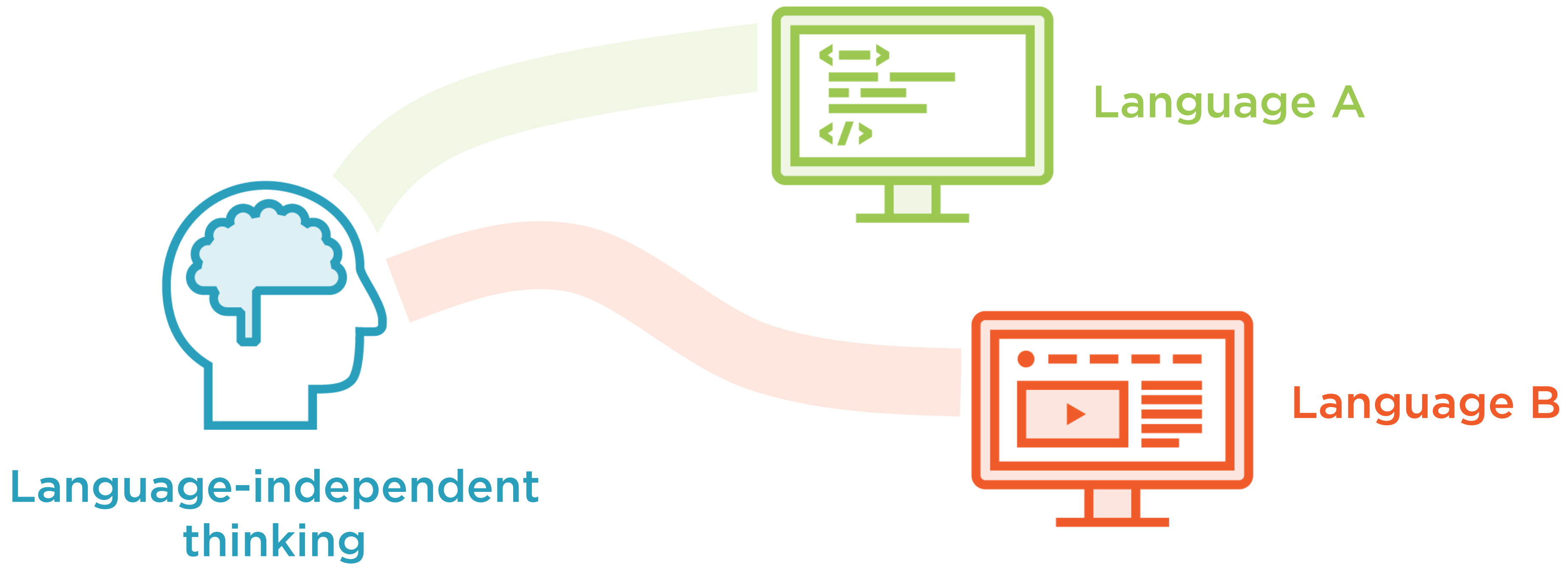

Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award

Steve McConnell
Code Complete

Programmers who program ***"into" a language*** first decide what thoughts they want to express, and then they determine how to express those thoughts using the tools provided by their specific language.

Google for: "programming into a language"

Programming into a Language, Not in It



General ideas vs. language idioms

When Idioms Are Useful

Guarded assignment

```
var x = obj ?? throw new NullReferenceException();
```

✓ Short and simple

Alternative

```
var x = obj == null
  ? throw new NullReferenceException()
  : obj;
```

✓ Functional

✗ Overengineered

✗ Goes against the language

When Idioms Are Obtrusive

Returning a function delegate

```
using System;
```
✓ No dependencies

```
Func<BankCard> CreateCardFactory(Date expires) { ... }
```
✓ Short and simple

✓ Functional

Alternative

```
using Domain.Interfaces;
```
✗ Additional dependencies

```
IBankCardFactory CreateCardFactory(Date expires) { ... }
```
✓ Object-oriented

```
interface IBankCardFactory
{
  BankCard Create();
}
```
✗ Additional types

Programming in vs. into a Language

Programming in a language

```
var x = obj ?? throw new NullReferenceException();
```

✓ Short and simple

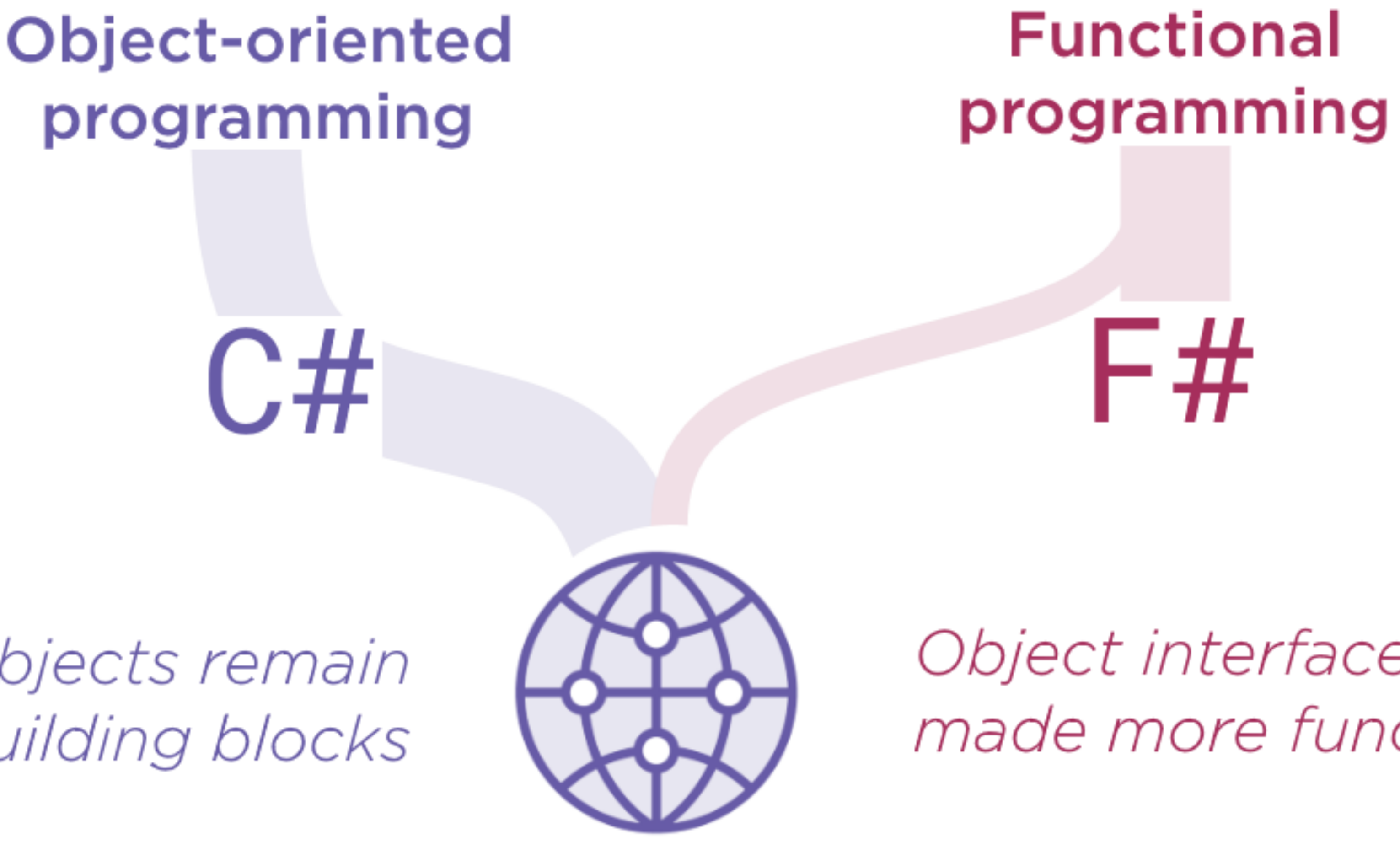
Programming into a language

```
Func<BankCard> CreateCardFactory(Date expires) { ... }
```

✓ Short and simple

There is the way to include functional programming concepts ***into*** C# ***without damaging the design***

Mixing Paradigms in C#



Objects remain primary building blocks

Object interfaces made more functional

What Follows in This Course

Challenging the Object-oriented Mindset
Identifies common shortcomings of pure object orientation

Adding Functional-style Filters to Object Model
Cautiously adds bits of functional design into the object model

What Follows in This Course

Introducing Pure Functions to Object Design
Showcases pure functions as foundation for the rest of the course

Memoization with Pure Functions
Improves run time performance of pure functions

Working with Pure Member Functions
Makes object member functions pure

What Follows in This Course

Pattern Matching with C# 7
Uses pattern matching as primary means of flow control

Metaprogramming with Extension Methods
Uses extension methods as primary means of attaching behavior to types

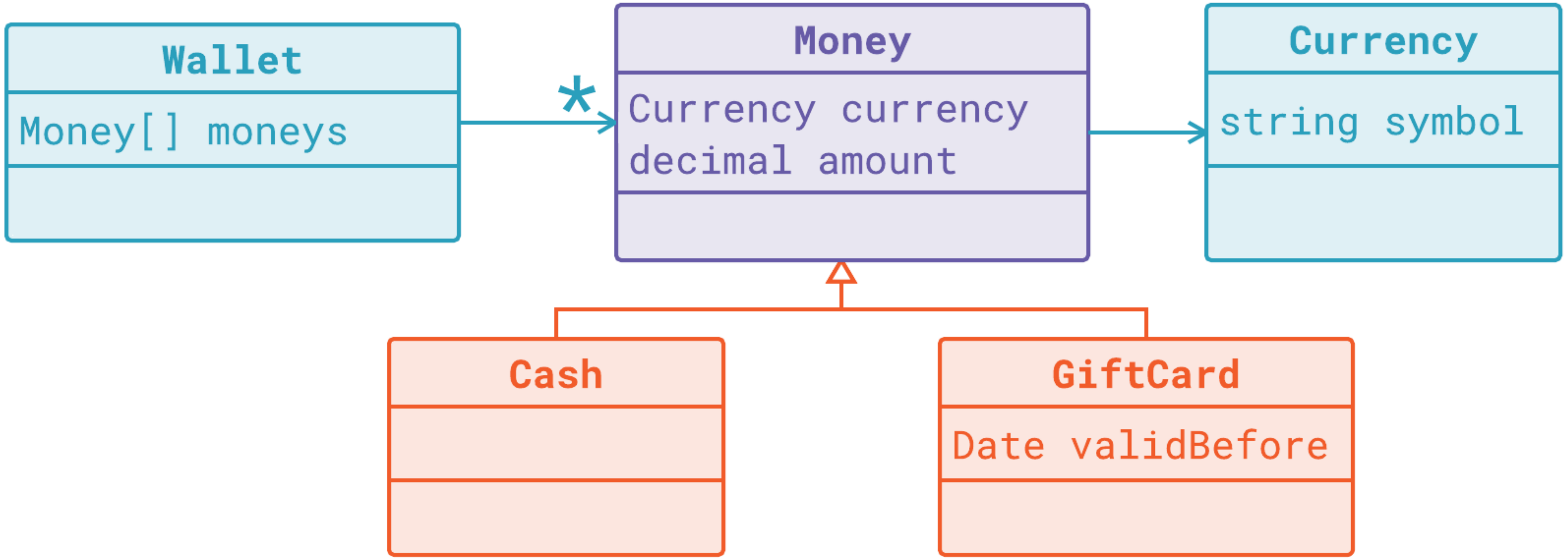
Function Composition with Object Model
Makes functions composable in the object model

What Follows in This Course

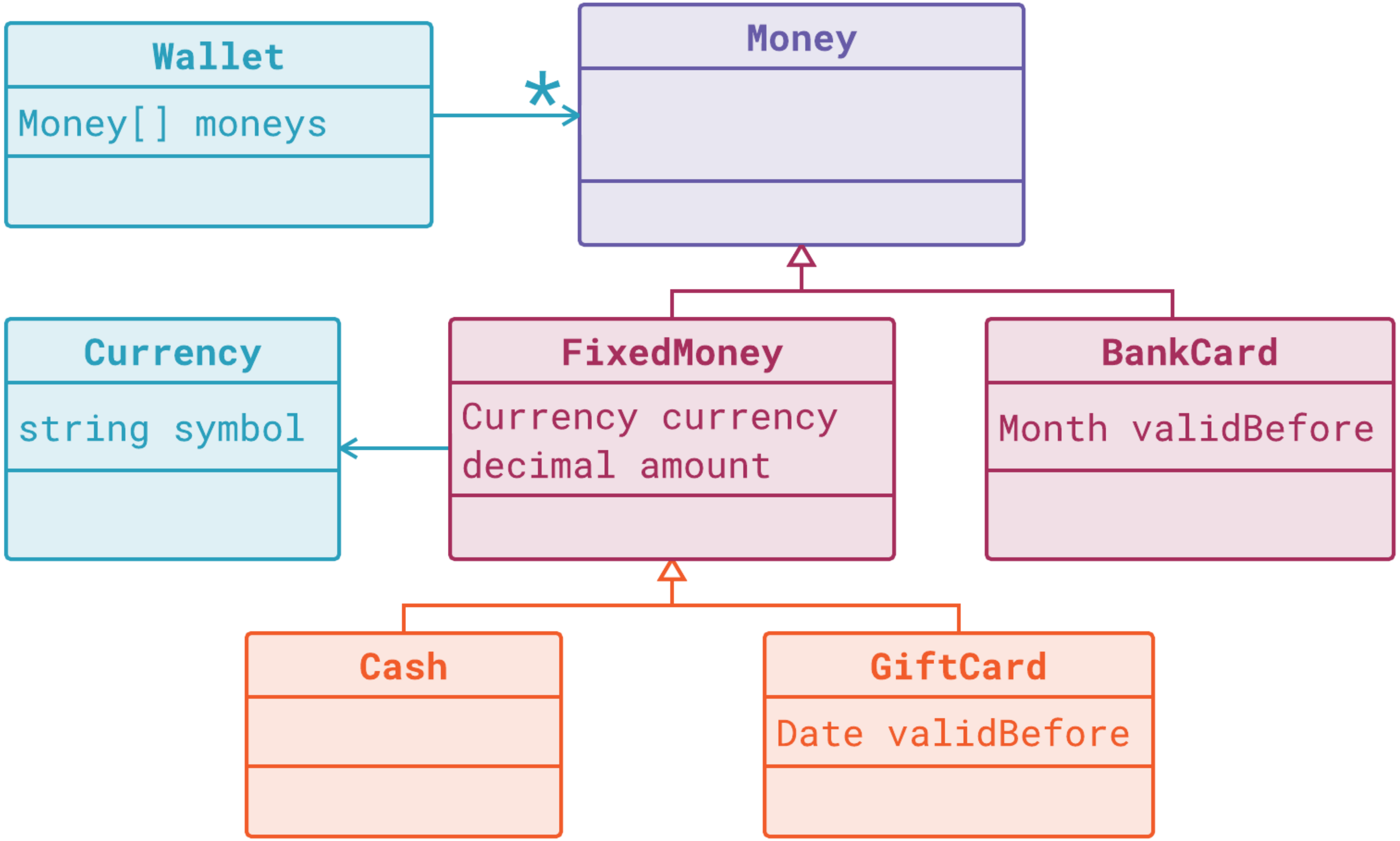
Understanding Railway-oriented Programming
Relates railway-oriented programming to optional objects

Handling Errors in Functional Style
Applies railway-oriented programming to error handling

Developing an Object Model



Developing an Object Model



Tackling the Drawbacks of Object Design

Methods know too much
One method is typically responsible for an entire complex operation

Hard to manage in large projects
Complex dependencies are an obstacle in large code bases

Include functional concepts
Model remains object oriented, with more functional design of classes

Summary

Polymorphism as a burden

- Increases complexity of objects
- Polymorphism is a liability
- Makes it hard to develop rich domain models

Summary

In the rest of the course

- Leverage functional design to untangle objects
- Reduce complexity of the object model

Functional interface

- Classes remain
- Operations chopped into slices
- Combine simple functions
- Build larger features out of them

Next module:
Adding Functional-style Filters to Object Model