Making Your C# Code More Object-oriented

ATTAINING EXTENSIBILITY WITH OBJECT-ORIENTED CODE

Zoran Horvat
OWNER AT CODING HELMET CONSULTANCY

@zoranh75 codinghelmet.com

Principal Motives for Object Orientation

Why write OO code?

Because of the benefits
we receive back.

Then, what are the benefits?

That we will talk about
in this course...

Fundamental Principles of OO Programming

Principles of OO programming

Encapsulation,

Abstraction,

Inheritance,

Polymorphism

Foundations of OO programming

this pointer

Dynamic dispatch

Understanding this Pointer

Silently passed with a call to an instance-level function

Function then operates on an *object*

Bringing operations close to data

No more global functions

Operations have become the quality of data structures

Understanding Dynamic Dispatch

Object's duty

To carry data about its type

Type's duty

To keep track of its virtual functions table

To override some functions from its base type

Runtime's duty

Jump in when call is placed on an object

Find V-table of its generating type

Find concrete function address

From Structured to OO Languages

It is possible to write OO code in a structured language (like C)

It is possible to write assembler macros to make it OO

It is not only the programming language

Object orientation is the shift in thinking

From Structured to OO Languages

C++ and Objective C

Mixing paradigms

Structured and OO
at the same time

Java and C#

Object-oriented straight-up

Still support structured
programming practices

Shifting Your Mind to Become Object-oriented

Code

We are still surrounded by structured code

Code often looks like plain C

Textbooks

Still explain concepts through structured examples

Programmers

Becoming object-oriented requires a change

Start Thinking in Terms of OO Principles

Define objects

Which data goes with which operations?

Use polymorphism

Which operations require dynamic dispatch?

Bring life to objects

Let operations be determined dynamically

Substitute objects to modify behavior

Summary

Basic motivation to write OO code

- Improved chances for applications to survive in the real world

What follows in this course

- Practical advices on improving design

Summary

Avoiding branching instructions

- Replace branching with objects

Avoiding loops

- Build loops into sequential data structures

Turning data structures into objects

- Encapsulate operations in the data structure itself

Turning algorithms into strategy objects

- Replace strategies to modulate the algorithm

Summary

Treating some objects as simple values

- Treat them as you treat numbers
- This will simplify code and reduce bugs

Avoiding null references

- Null is not an object

Avoiding multiway branching

- Model dynamic rules with objects

Separating infrastructure from domain logic

- Figuring which instructions are infrastructure, and which are the domain