

# Module 01:

## "What are Design Patterns?"



# Agenda

- ▶ Introducing Design Patterns
- ▶ Design Patterns Background
- ▶ Our Approach
- ▶ General OO Principles



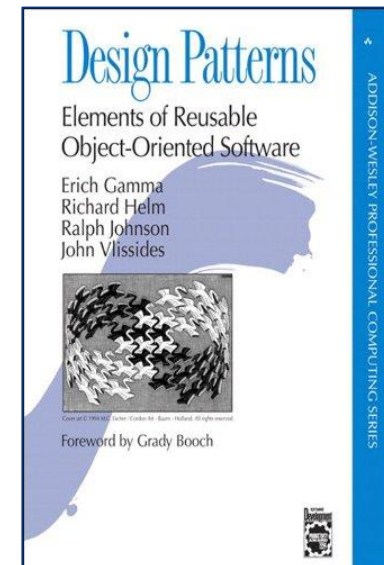
# Introducing Design Patterns

- ▶ Reusable techniques for commonly occurring software design problems
- ▶ A common “OO design language” for developers
- ▶ Template solutions are starting points for development
  - Application design
  - Relationship between components and/or classes
  - Presented at an abstract level
  - Avoid re-inventing new solutions every day
  - Provide a uniform design quality in system design
- ▶ Design Patterns describe structure – not algorithms!



# Design Patterns Background

- ▶ 1977
  - Christopher Alexander
  - Architect proposing a pattern language for buildings
- ▶ 1987:
  - Kent Beck and Ward Cunningham
  - Conference paper presenting similar ideas for software
- ▶ 1994:
  - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (a.k.a. "Gang of Four ☺")
  - *Design Patterns: Elements of Reusable Object-Oriented Software*
  - ISBN-13: 978-0-201-63361-0
  - <https://www.amazon.co.uk/Design-Patterns-Object-Oriented-Addison-Wesley-Professional-ebook/dp/B000SEIBB8>



# 23 Gang of Four Design Patterns

- ▶ Considered the cornerstone of design patterns
- ▶ Classifies patterns into categories such as
  - Creational
  - Structural
  - Behavioral
- ▶ Presented using in general terms using UML
- ▶ Presented in a language-agnostic way



# Beautiful! Classic! But...

- ▶ ... 23-25 years is a long time in IT!
- ▶ Amount of design patterns theory and practice continually evolving and being refined
  - Evolves with research and practice
  - Evolves with new architectural paradigms
  - Refined by programming languages such as C#
- ▶ Consequently,
  - New design patterns emerge
  - Additional categories of patterns seem natural
    - User interface
    - Concurrency
    - Distributed
    - ...



# Our Approach

- ▶ Goal: *Present an up-to-date account of modern design patterns specialized for current C# and .NET*
- ▶ Include all original 23 design patterns from GoF in newer and more modern versions
- ▶ Include additional newer design patterns not included in GoF
- ▶ Present C#-optimized versions of all patterns (preferably in .NET Core)
- ▶ Use practical examples instead of general terms and UML



# General OO Principles

- ▶ Foundational principles ensuring adaptable and maintainable code
  - Program to an interface – not an implementation
  - Favor object composition over inheritance
  - Favor loose coupling between classes
    - Sometimes even at the expense of duplication
  - Single Responsibility Principle
    - Every class should have only one reason to change
  - Open/Closed Principle
    - Classes should be open for extension, but closed for modification





