C# Design Patterns: Builder

IMPLEMENTING THE BUILDER PATTERN

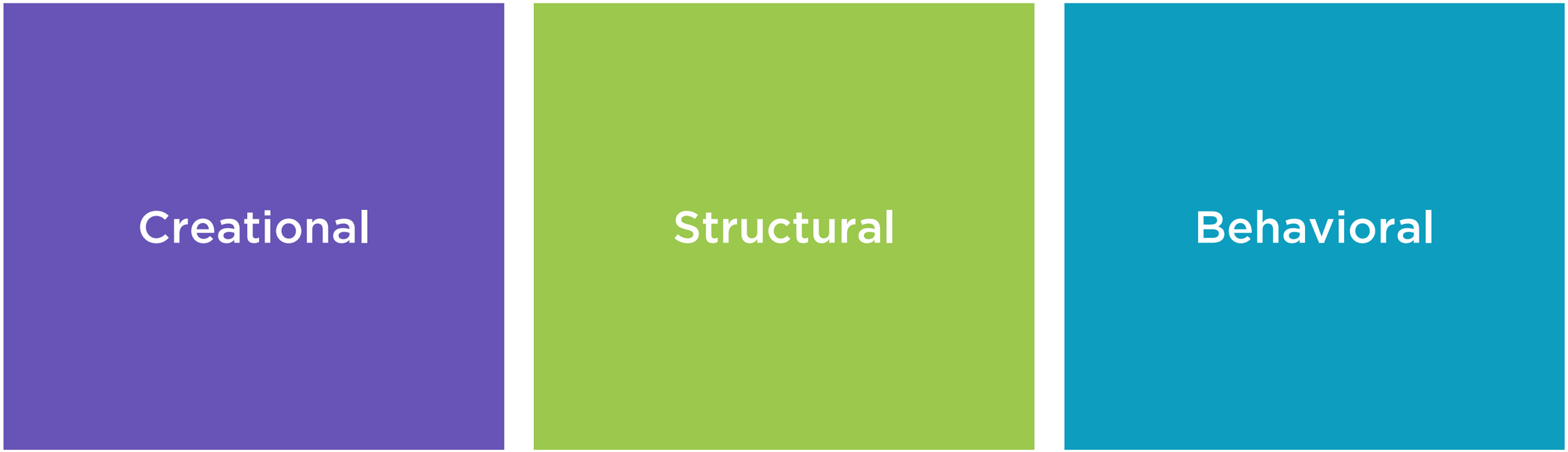
Harrison Ferrone
SOFTWARE DEVELOPER
@journeyman_programmer www.paradigmshiftdev.com

Overview

The Builder Pattern in Practice:

- Defining an object class
- Adding a builder interface
- Creating a concrete builder class
- Implement a director class
- Update to a Fluent Builder variation
- Review use cases and applications

Design Pattern Categories



"Separate the construction of a complex object from its representation so that the same construction process can create different representations."

- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*

Abstracts out initialization code

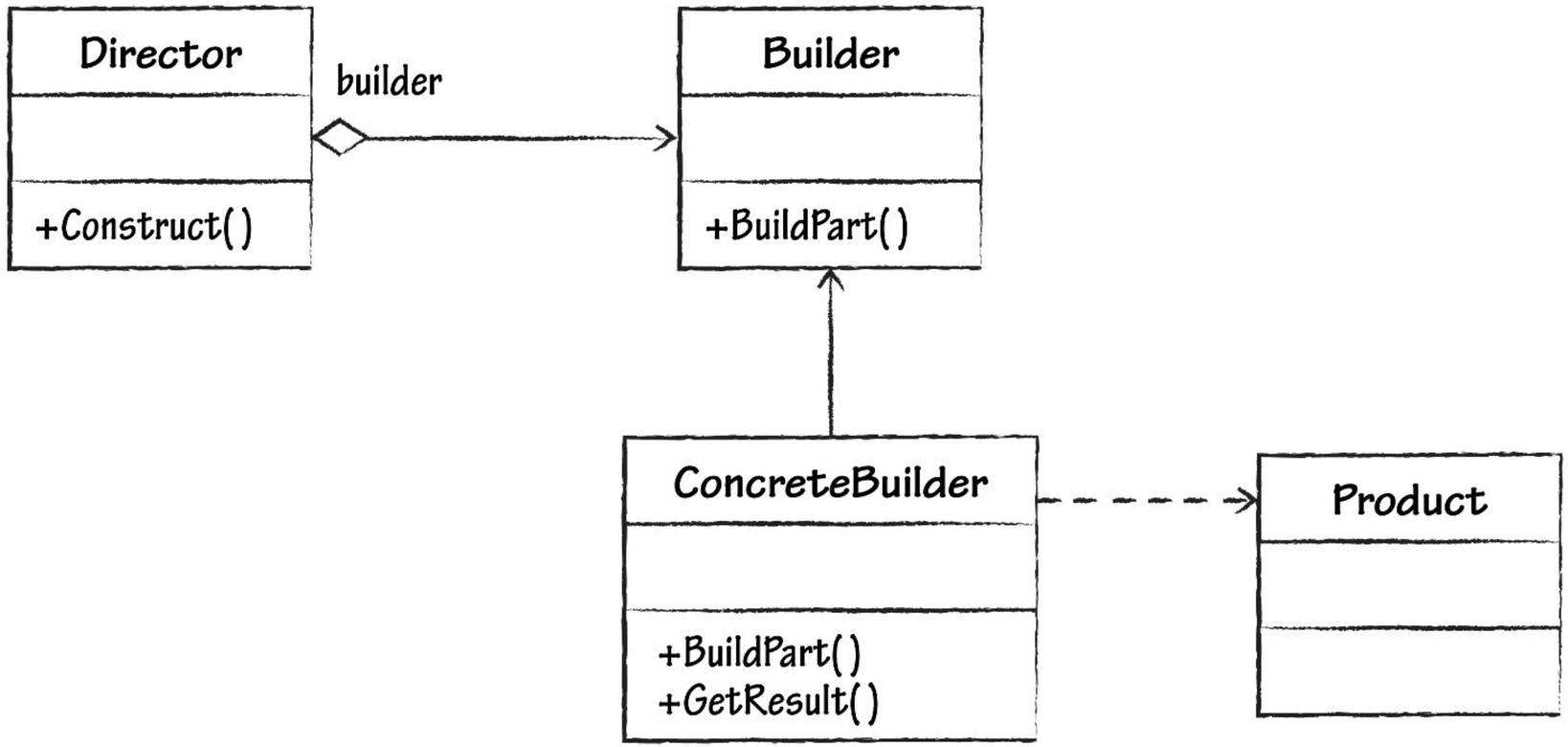
Representations created as concrete classes

- Constructed from interface blueprint

Director class handles actual object creation

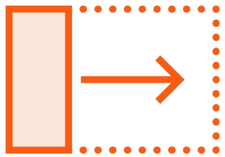
- Controls where & what data is used

Builder Pattern Diagram



The Builder design pattern is not a silver bullet

Keep in Mind



Builder is useful when creating complex objects

When object creation needs to be separate from its assembly

When different representations need to be created with finer control

Creating a Builder Interface

Why Not Abstract Classes?

Overkill in certain situations

Concrete classes will be varied

- Abstract parent class is not suitable

Implementing Concrete Builders

Adding a Director Class

Using a Fluent Builder

Use Cases and Implications

Builder Use Cases

Not for everything - it's overkill for most classes where subclassing, refactoring, or abstracted interfaces would be a better solution

Bloated class constructors are a dead giveaway

Lots of computation logic in class constructors needed to set class field values

Finite number of related classes that perform similar functions with different representations

Design Pattern Implications

Lets you vary a products internal representation

It isolates code for construction and representation

It gives you finer control over the construction process

Shares similarities with the Factory pattern

Summary

Where the Builder pattern fits

Creating an object and Builder interface

Creating Concrete Builder classes

The Director class in action

Pattern variations

Common use cases and implications

Happy Coding!