Applying Design Patterns to Solve Everyday Problems

David Berry

@DavidCBerry13

http://buildingbettersoftware.blogspot.com

https://github.com/DavidCBerry13/DesignPatternsCode/

What is a Design Pattern?

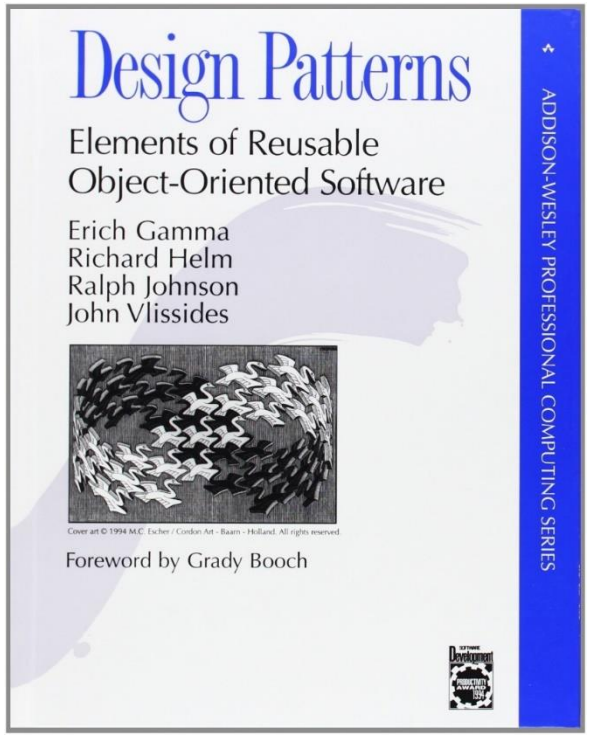
Wikipedia - https://en.wikipedia.org/wiki/Software_design_pattern

In software engineering, a software design pattern is a ***general reusable solution to a commonly occurring problem*** *within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a* ***description or template*** *for how to solve a problem that can be* ***used in many different situations****. Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.*

What is a Design Pattern? Really?

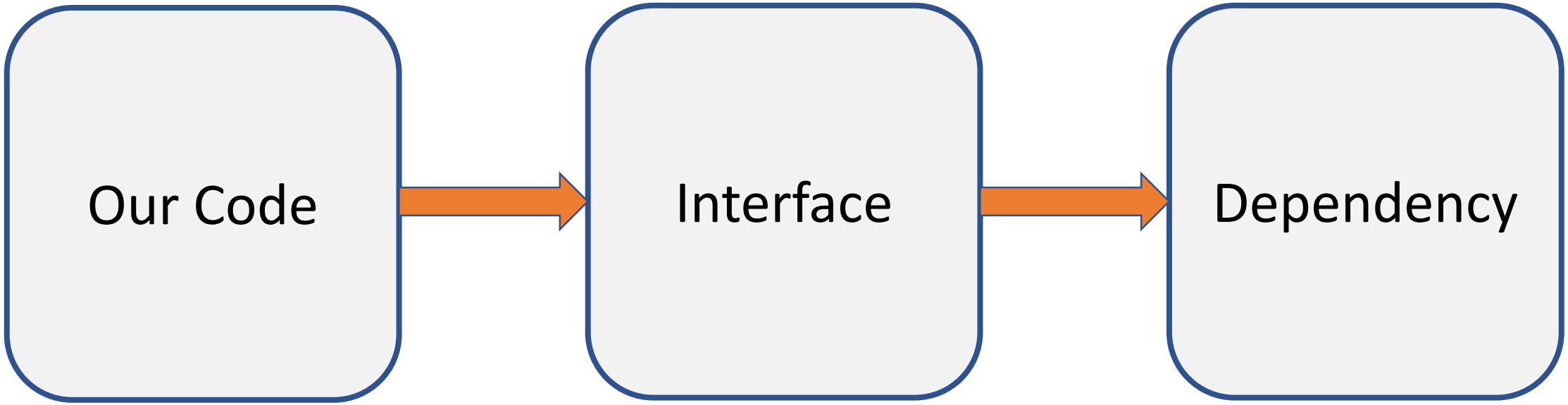
- Formalized, reusable solution to a common problem
- An elegant solution with a well thought out design
- Well understood by the software development community

How Do I Learn About These Things?



Books describe how to implement the different patterns, but it is often times difficult to know when to apply them

Isolation From Dependencies



Two Patterns Used Together

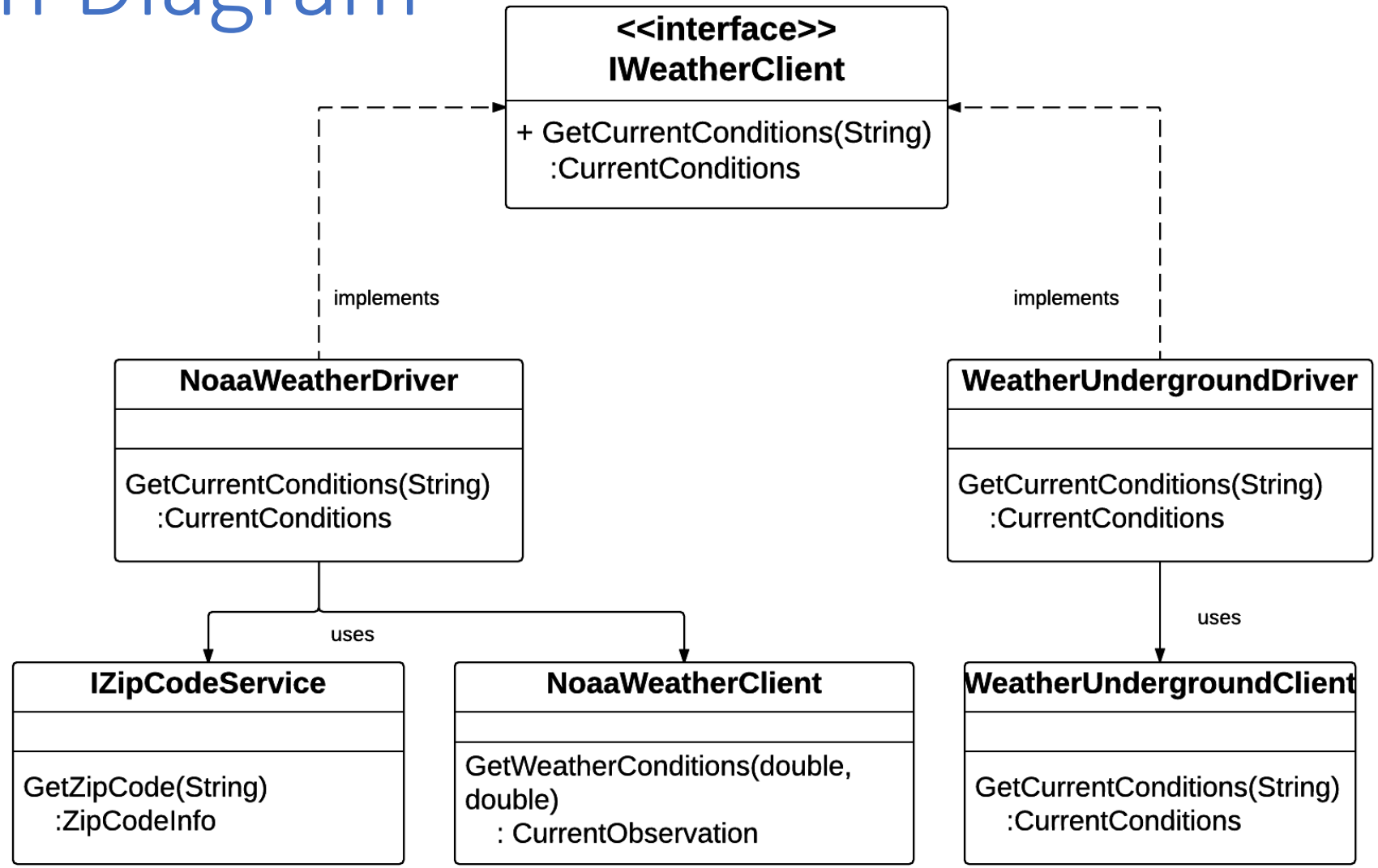
Bridge Pattern

- Uses an interface to isolate your code from different implementations
- Allows you to easily switch out implementations behind the bridge
- Allows code to be tested without the dependency
- Sets us up to use dependency injection

Adapter Pattern

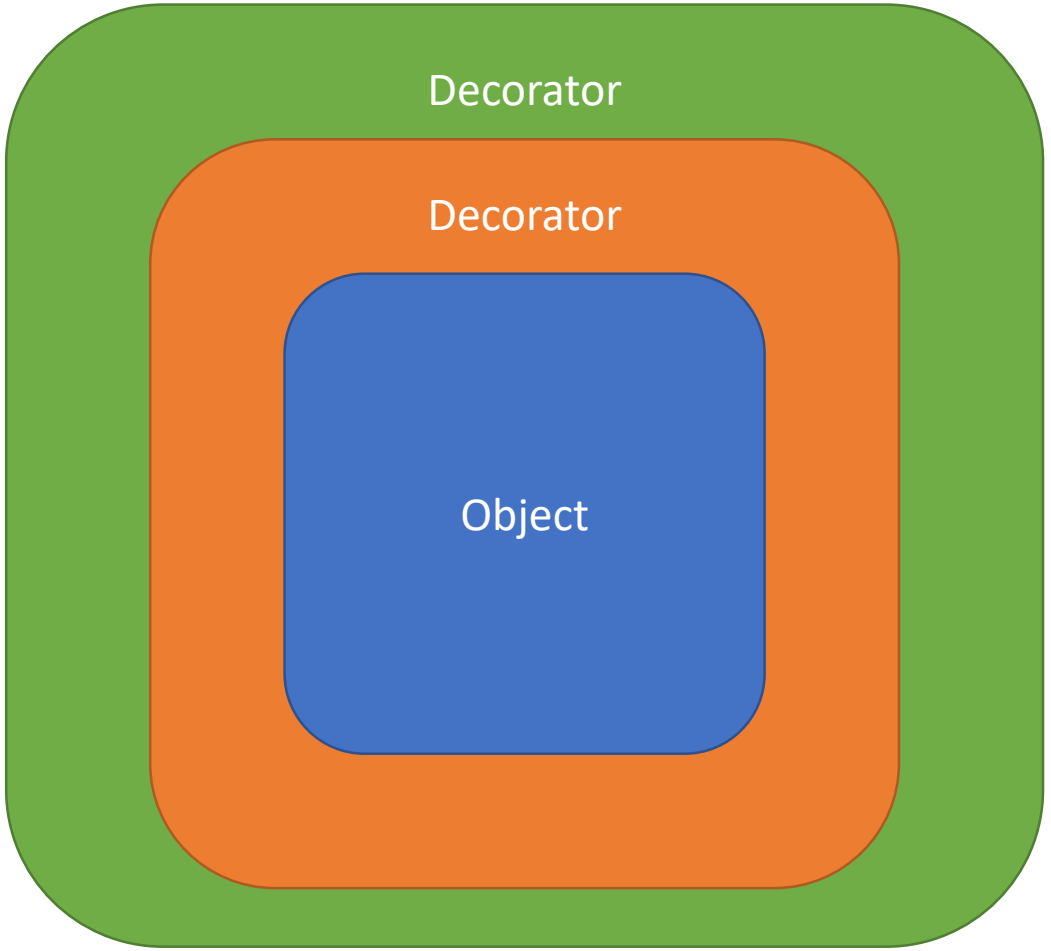
- Converts the interface of an existing class to another interface
- Used to make an existing class work with others without needing to change the source code

Solution Diagram

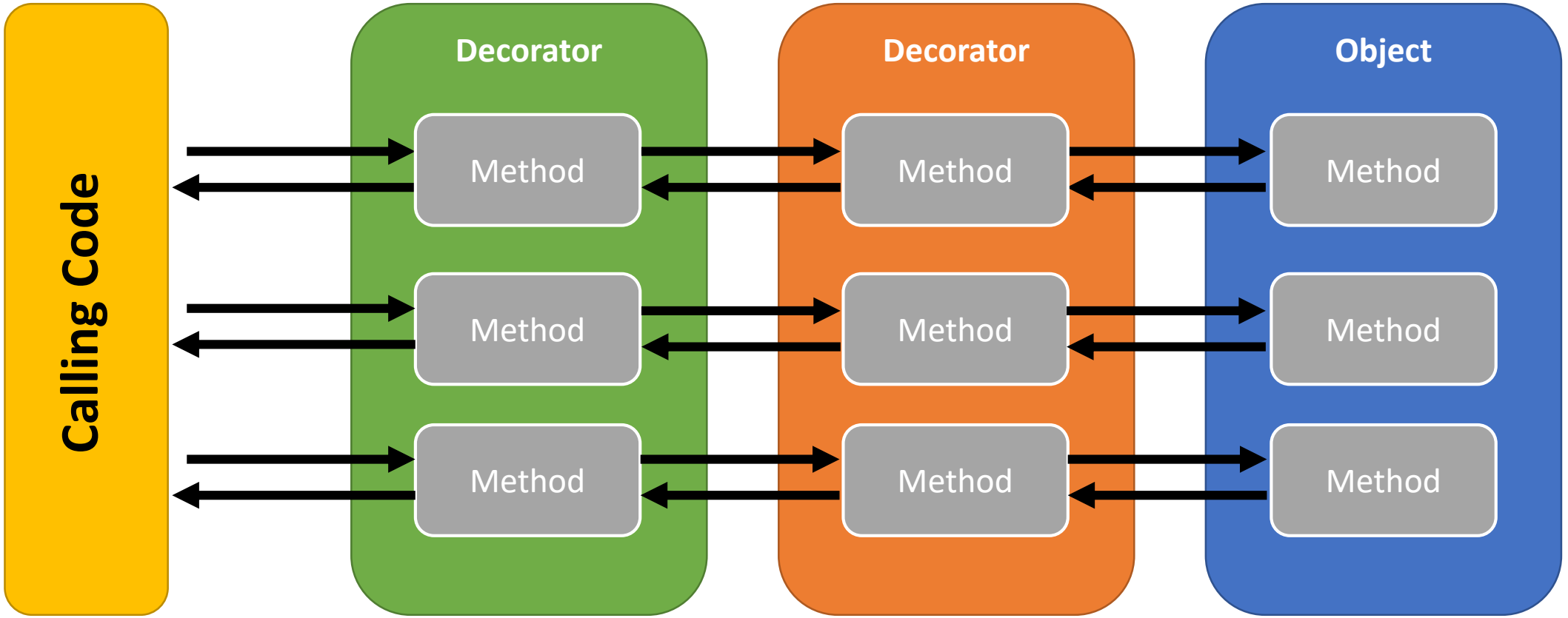


Decorator

- Effectively wraps one object inside of another
- Decorator objects implement the same interface as the object they are decorating
- Allows you to add behavior without modifying the original object
- Objects can be decorated multiple times
- Useful for cross cutting concerns



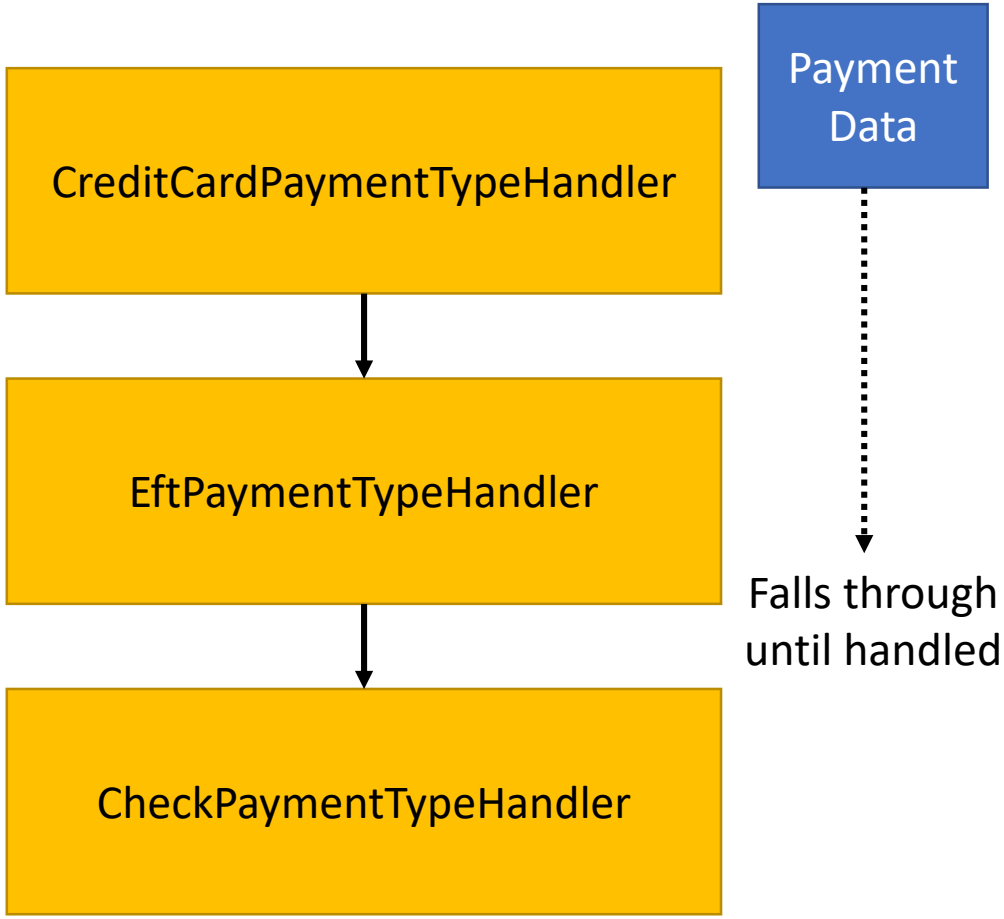
Decorator



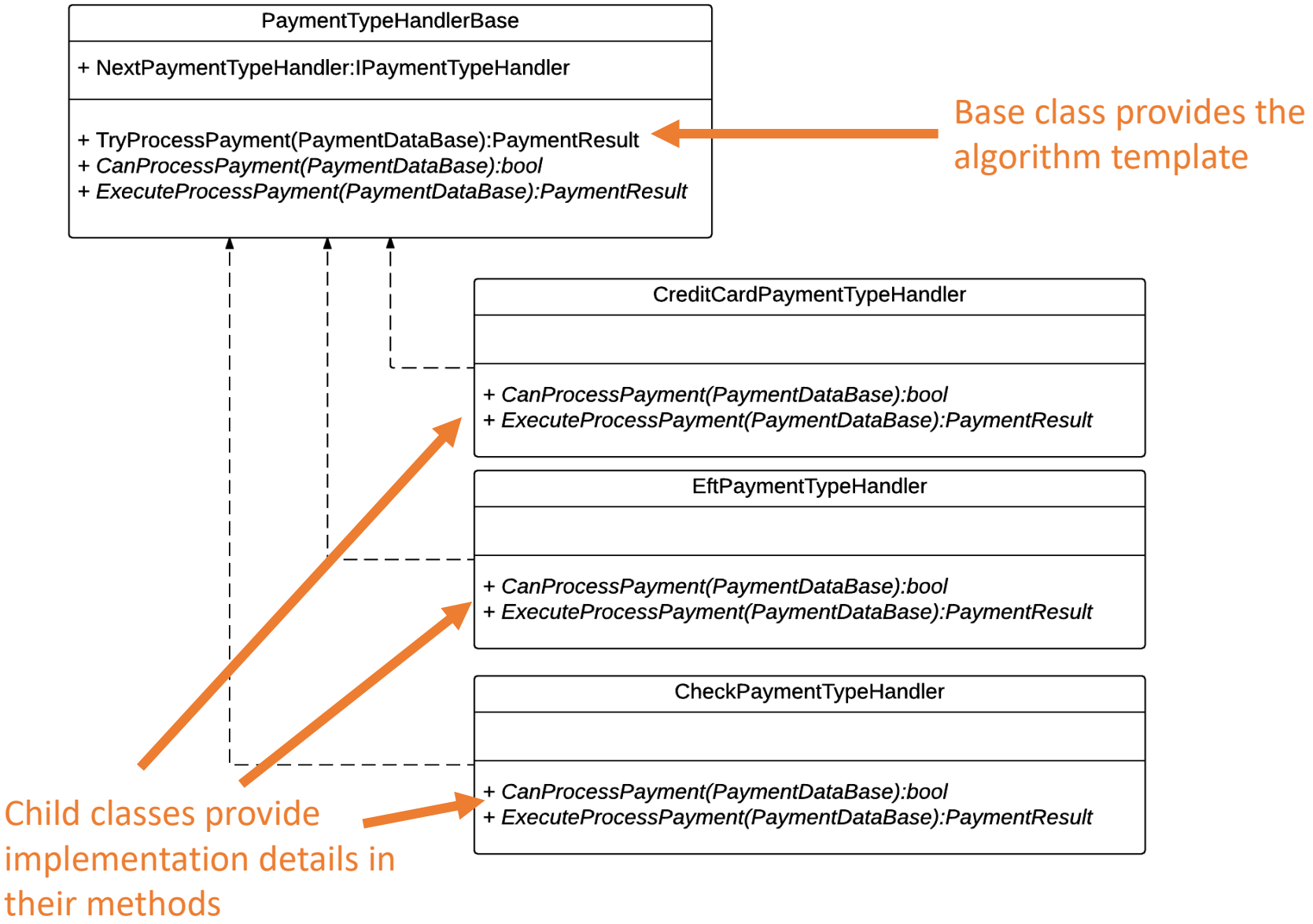
Allows you to intercept calls for the wrapped object (both inbound and outbound)

Chain of Responsibility

- Think of a series of multiple handlers that can each handle a request
- Each handler is focused on a single use case
- If a handler cannot process the request, it passes the request to the next handler in the chain
- Order can be important (depending on your use case)



Template Pattern



Resources

- Head First Design Patterns (Book)
  - http://shop.oreilly.com/product/9780596007126.do
- Design Patterns On-Ramp (Pluralsight Course)
  - https://www.pluralsight.com/courses/design-patterns-on-ramp
- C# Interfaces (Pluralsight Course)
  - https://www.pluralsight.com/courses/csharp-interfaces
- Design Patterns Library (Pluralsight Course)
  - https://www.pluralsight.com/courses/patterns-library