Applying Design Patterns to Solve Everyday Problems

Milwaukee Code Camp

October 15, 2016

David Berry

Thank You – Principal Sponsor

http://www.SkylineTechnologies.com

Thank our Principal Sponsor by tweeting and following
@SkylineTweets

Thank You – Silver Sponsors

August 7th – 9th 2017
Kalahari Resort, Wisconsin Dells
http://www.ThatConference.com

About Me

Solution Architect for
Robert W Baird in Milwaukee

Pluralsight Author

Occasional social media contributor

- Twitter - @DavidCBerry13
- Blog – http://buildingbettersoftware.blogspot.com

Code on Github

- https://github.com/DavidCBerry13/DesignPatterns-MilwaukeeCodeCamp

What is a Design Pattern?

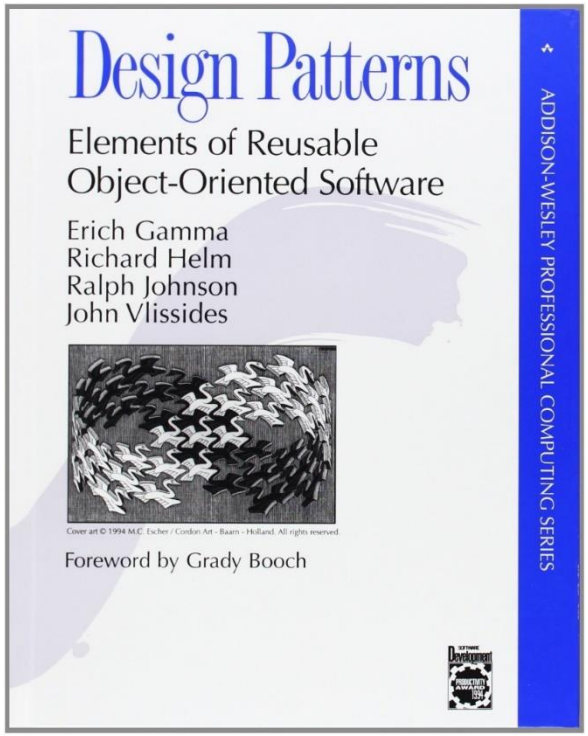
Wikipedia - https://en.wikipedia.org/wiki/Software_design_pattern

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?

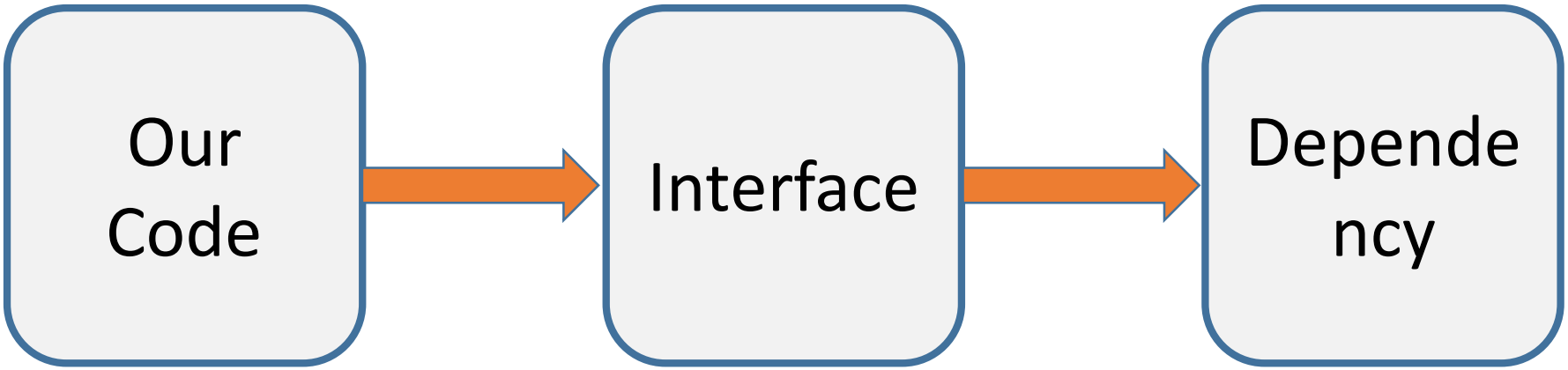
- Formalized, reusable solution to a common problem
- An elegant solution with a well thought out design
- Well understood by the software development community

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

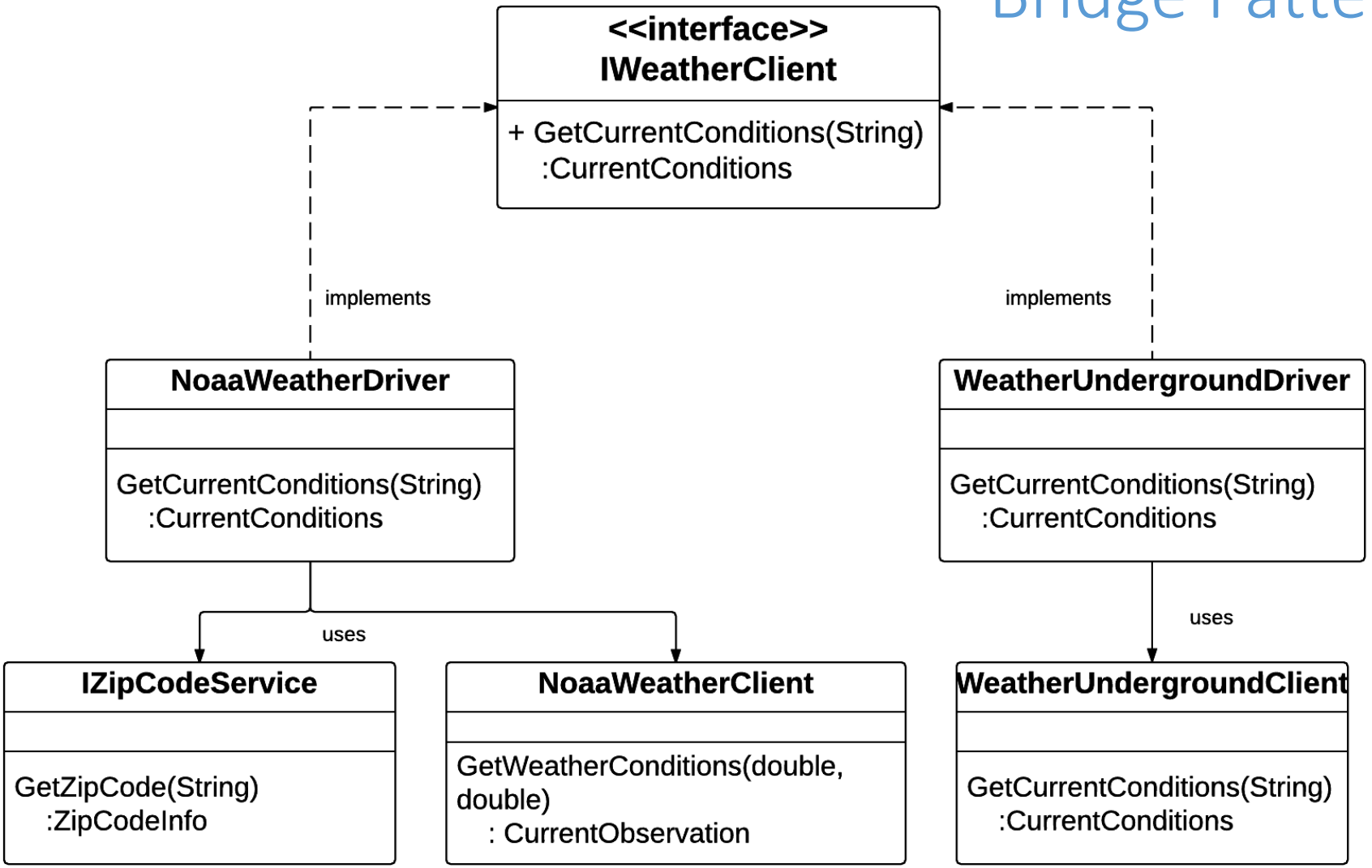
Isolation From Dependencies



Bridge Pattern

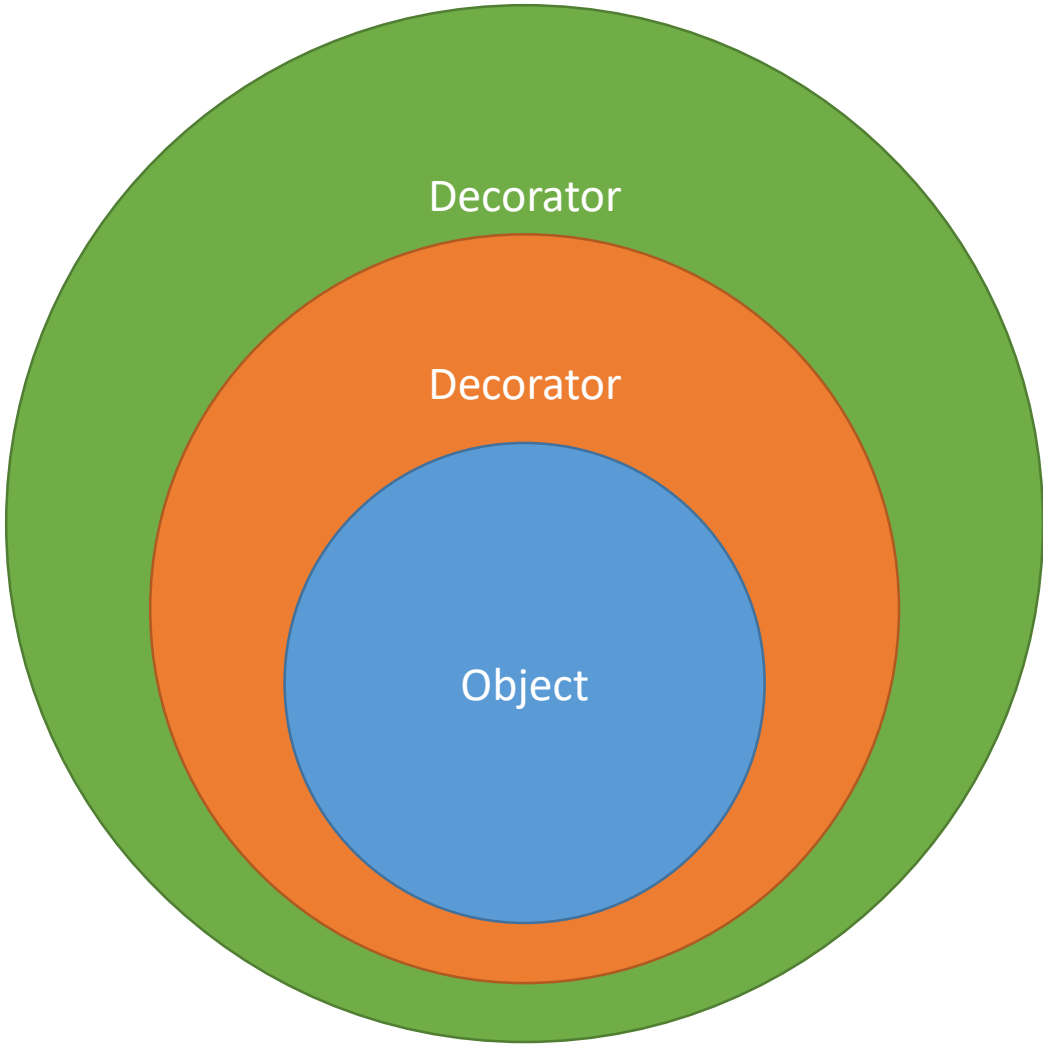
- Uses an interface to isolate your code from different implementations
- Allows you to easily switch out implementations behind the bridge
- Allows code to be tested without the dependency
- Useful whenever you have external dependencies

Bridge Pattern



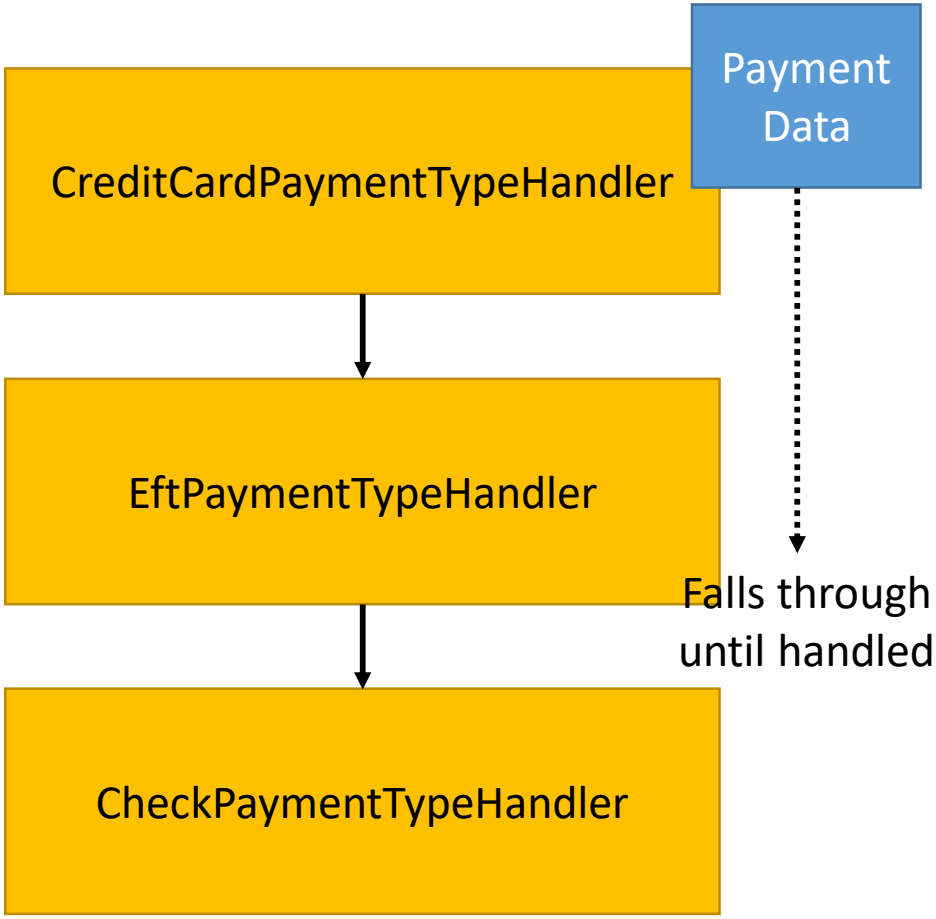
Decorator

- Effectively wraps one object inside of another
- Decorator objects implement the same interface as the object they are decorating
- Objects can be decorated multiple times
- Useful for cross cutting concerns

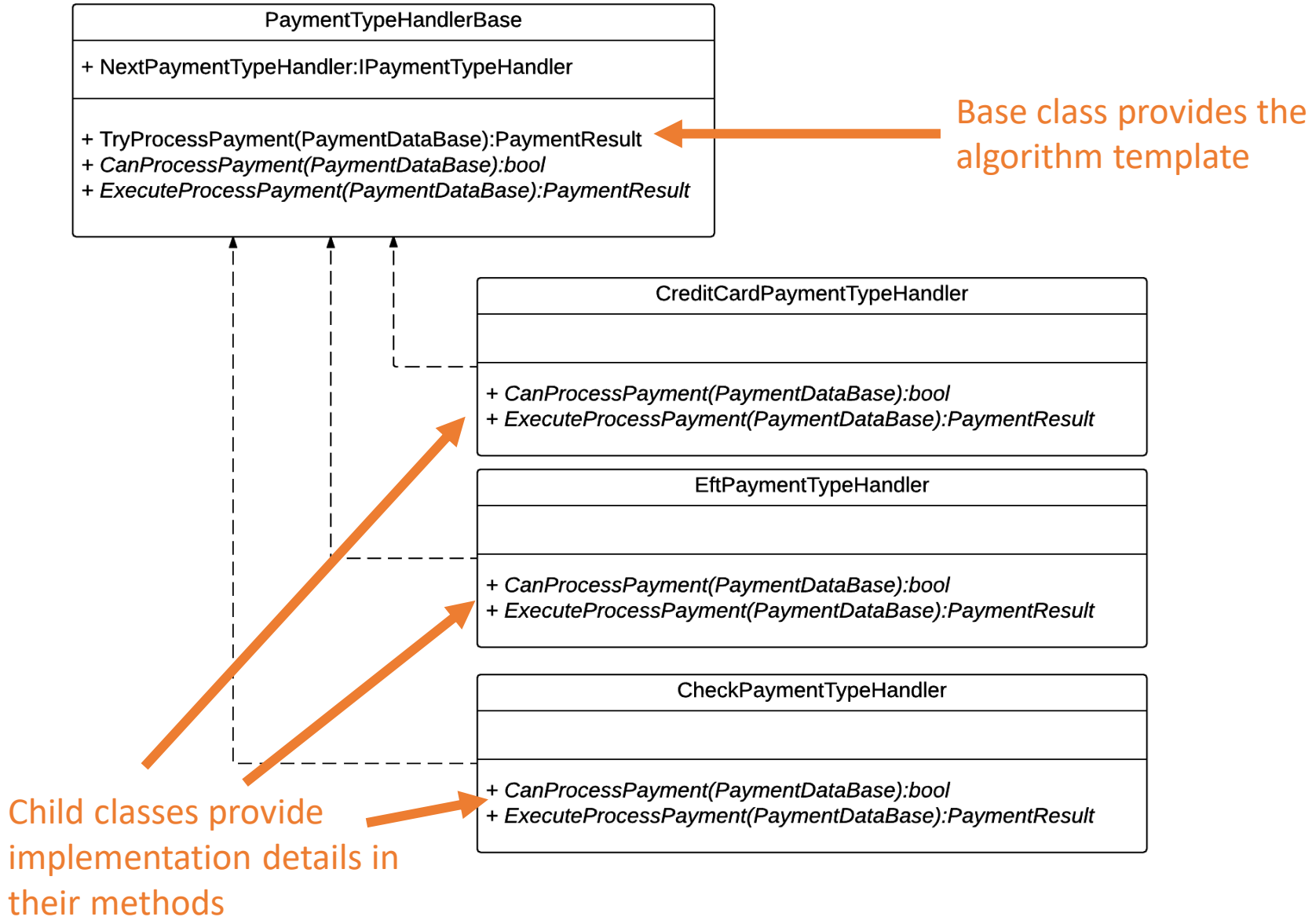


Chain of Responsibility

- 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