Design Patterns for the Working Programmer

David Berry
@DavidCBerry13
https://github.com/DavidCBerry13/DesignPatternsForTheWorkingProgrammer

What is a Design Pattern?

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

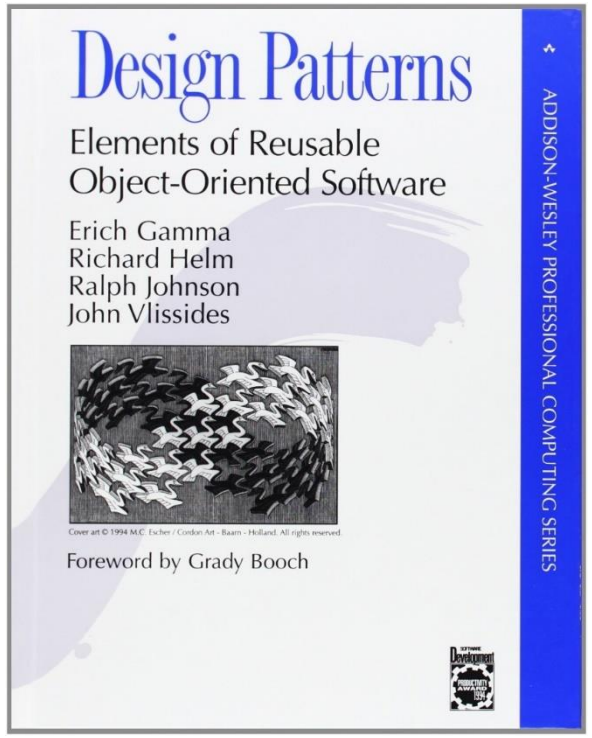
In software engineering, a software design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem 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?

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.

How Do I Learn These Things?



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

Why do we learn about design patterns?

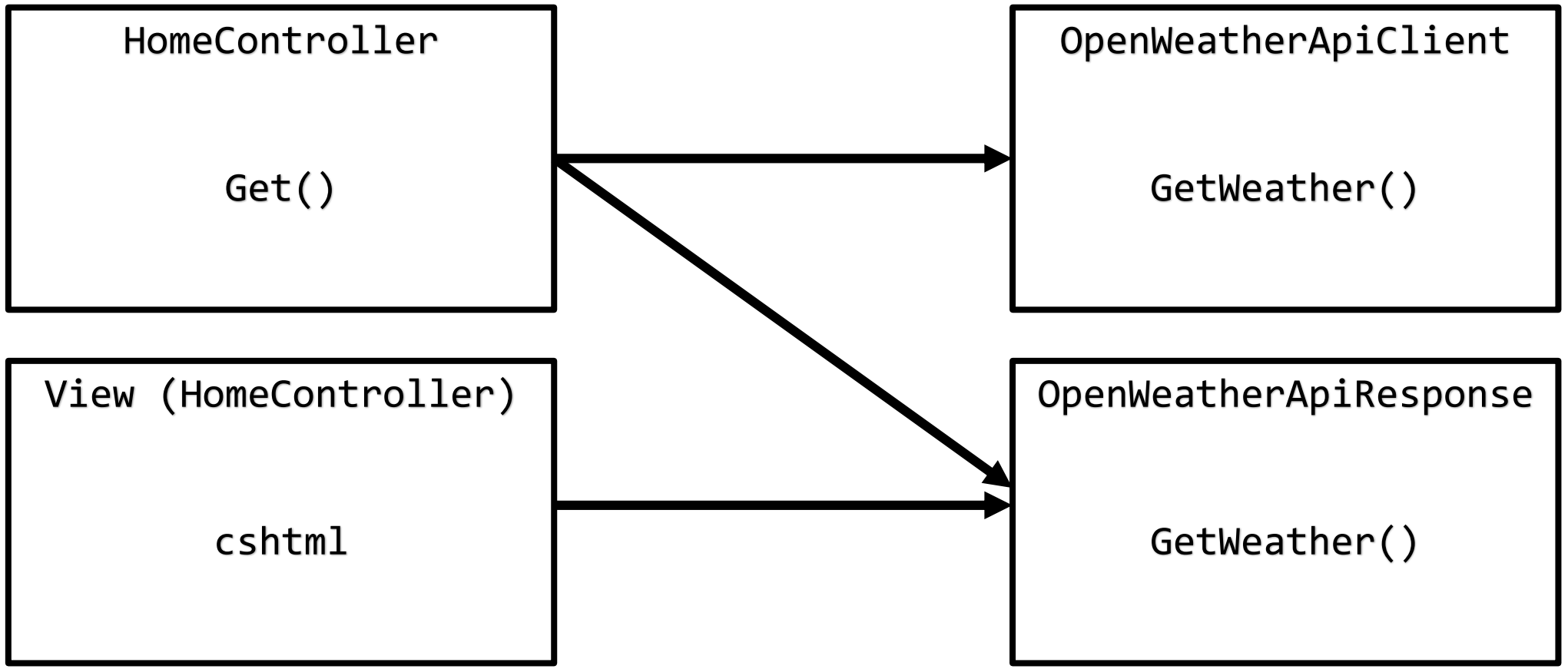
And how do we do it?

A Different Take on Learning Design Patterns

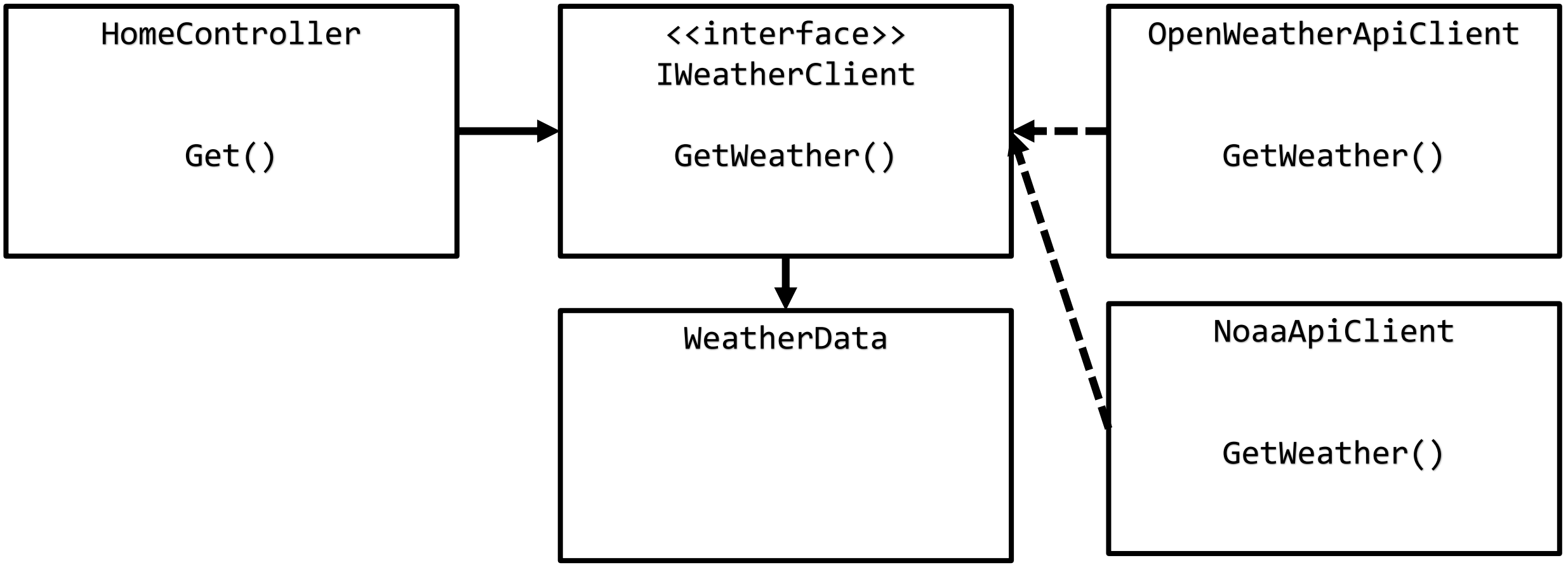
What techniques is the pattern trying to teach us

When should we be looking to apply a pattern

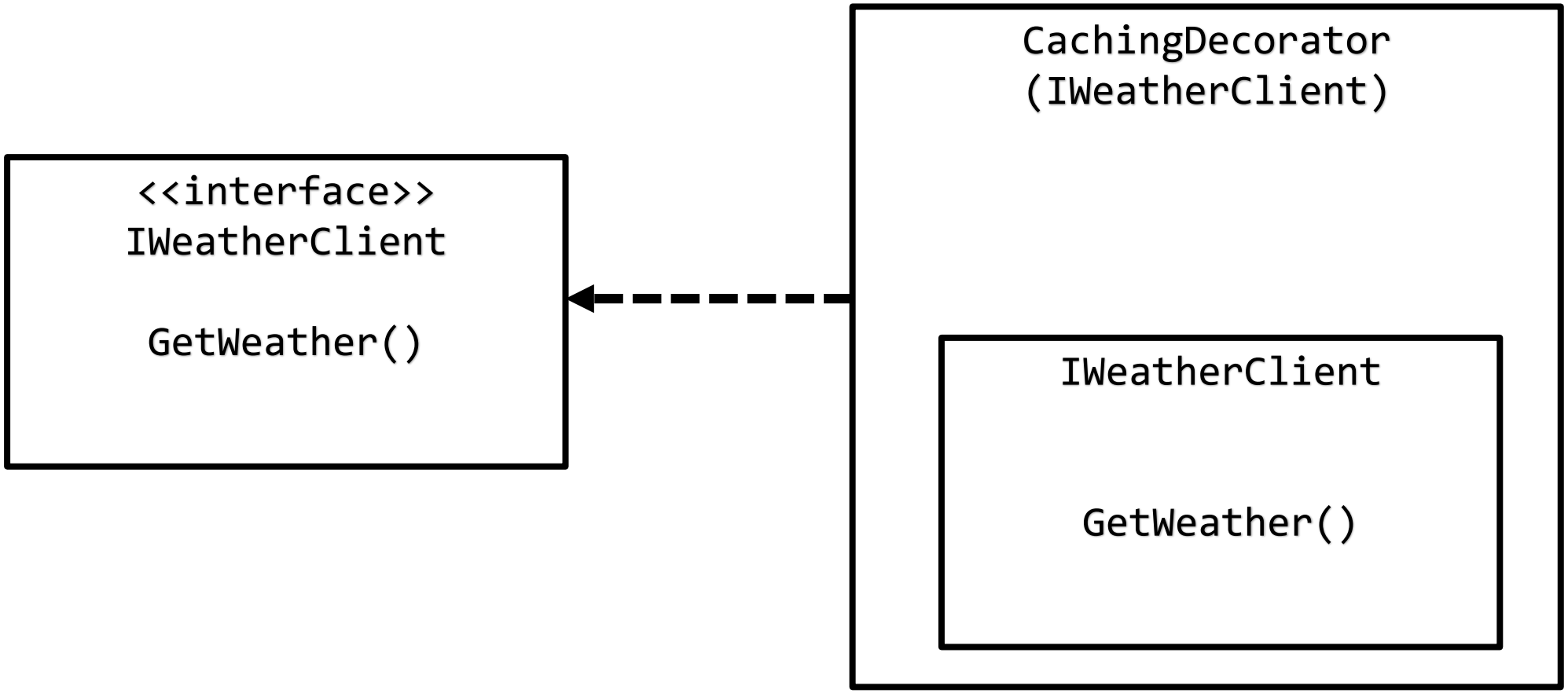
A Simple System



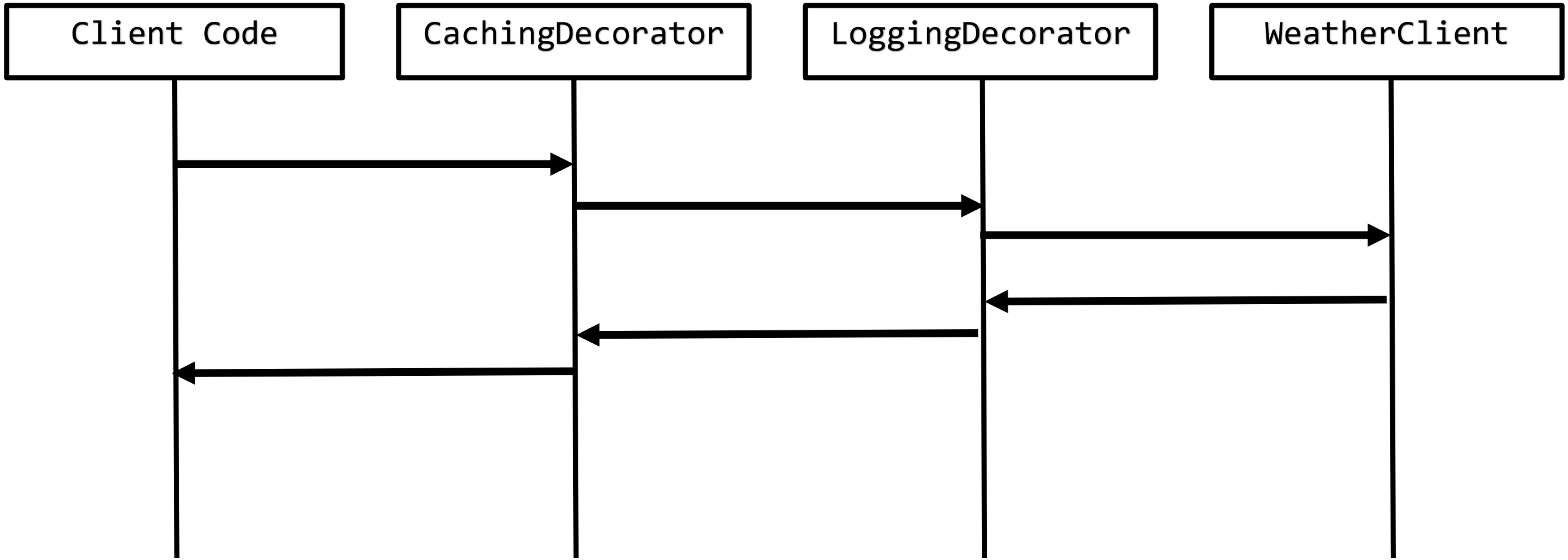
A Better Design



Decorator


<<interface>>
IWeatherClient
GetWeather()
CachingDecorator
(IWeatherClient)
IWeatherClient
GetWeather()

Decorator Call Sequence



Interfaces Summary

Wherever you have a system or module boundary, use an interface

- Loosely couples you to that system
- Creates a seam between components
- Allows you to plugin different implementations
- Facilitates Testing
- Sets you up to use other patterns like decorator

Patterns Summary

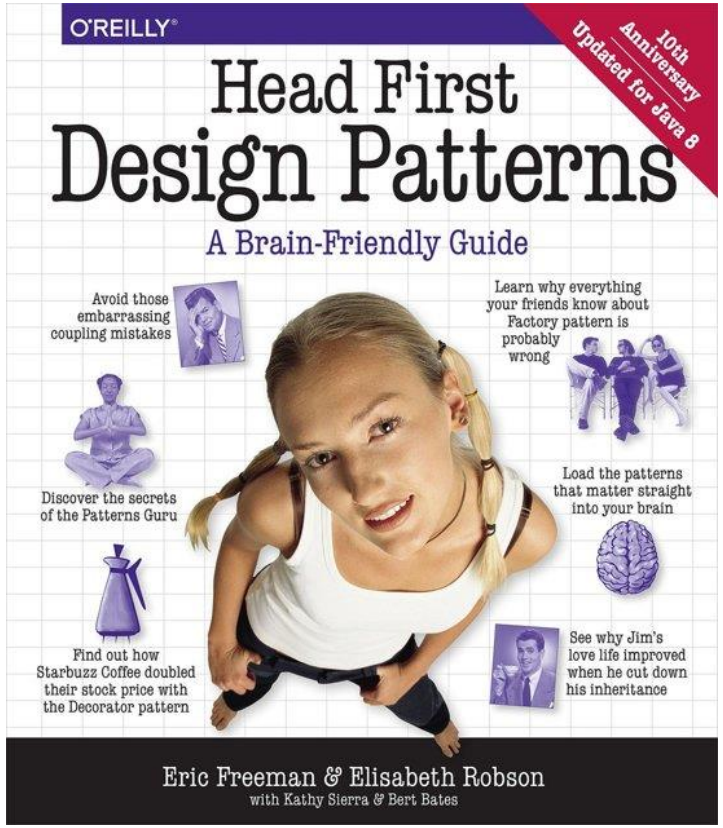
Think about what is the common interface and how you can separate different implementations into different classes

Allows you to easily plugin new rules or conditions as they emerge

Allows you to easily change the sequence logic is run in

Look to separate the logic that controls the process from the logic that is implemented in each step

Learning Resources



Head First Design Patterns

http://bit.ly/HeadFirstDesignPatternsBook

Design Patterns Library

http://bit.ly/DesignPatternsCourse

Source Code and Slides

Design Patterns for the Working Programmer

David Berry
@DavidCBerry13
https://github.com/DavidCBerry13/DesignPatternsForTheWorkingProgrammer