

C# Design Patterns: Decorator

THE DECORATOR PATTERN



David Berry

PLURALSIGHT AUTHOR

@davidcberry13

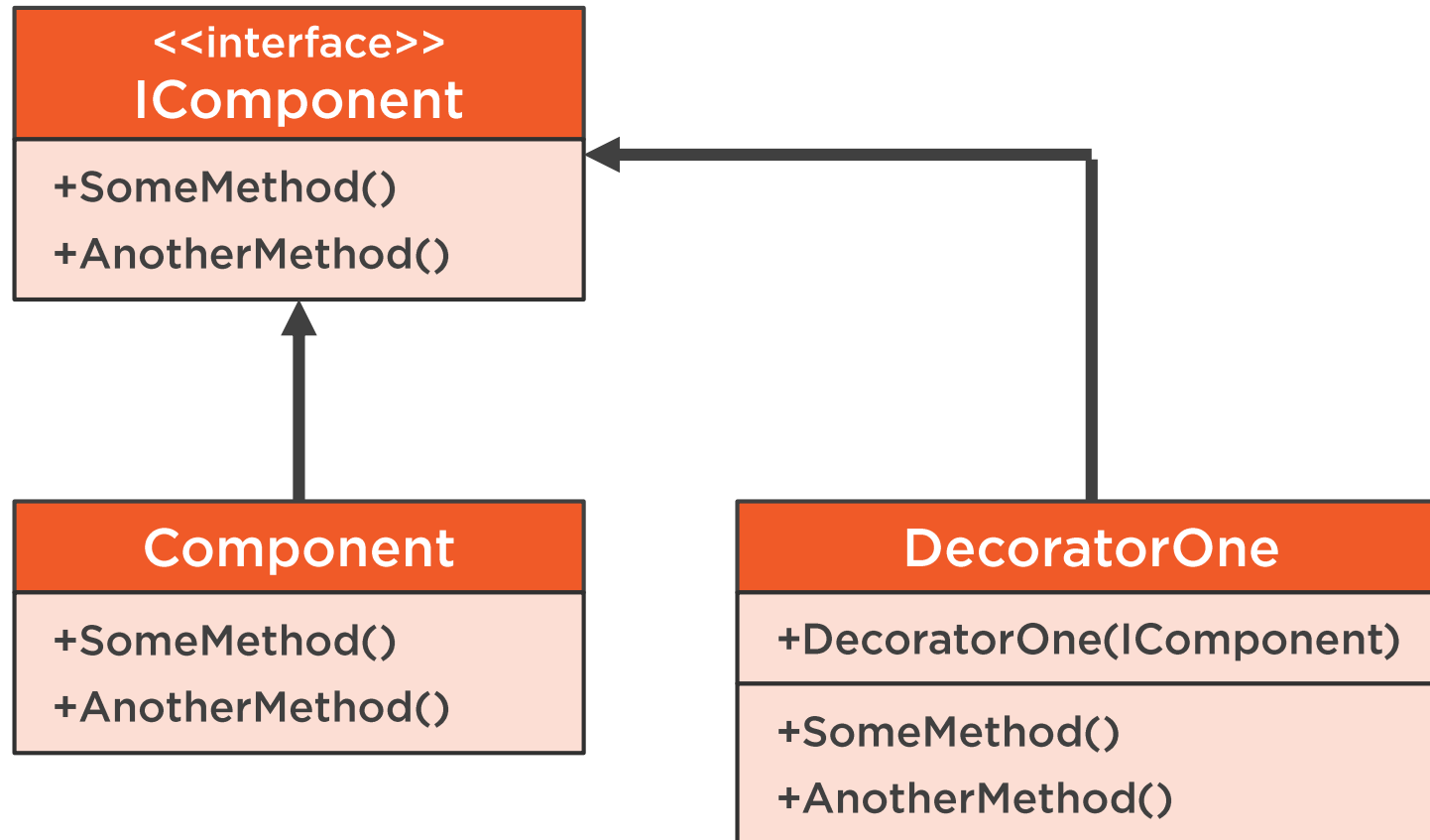


Decorator Pattern

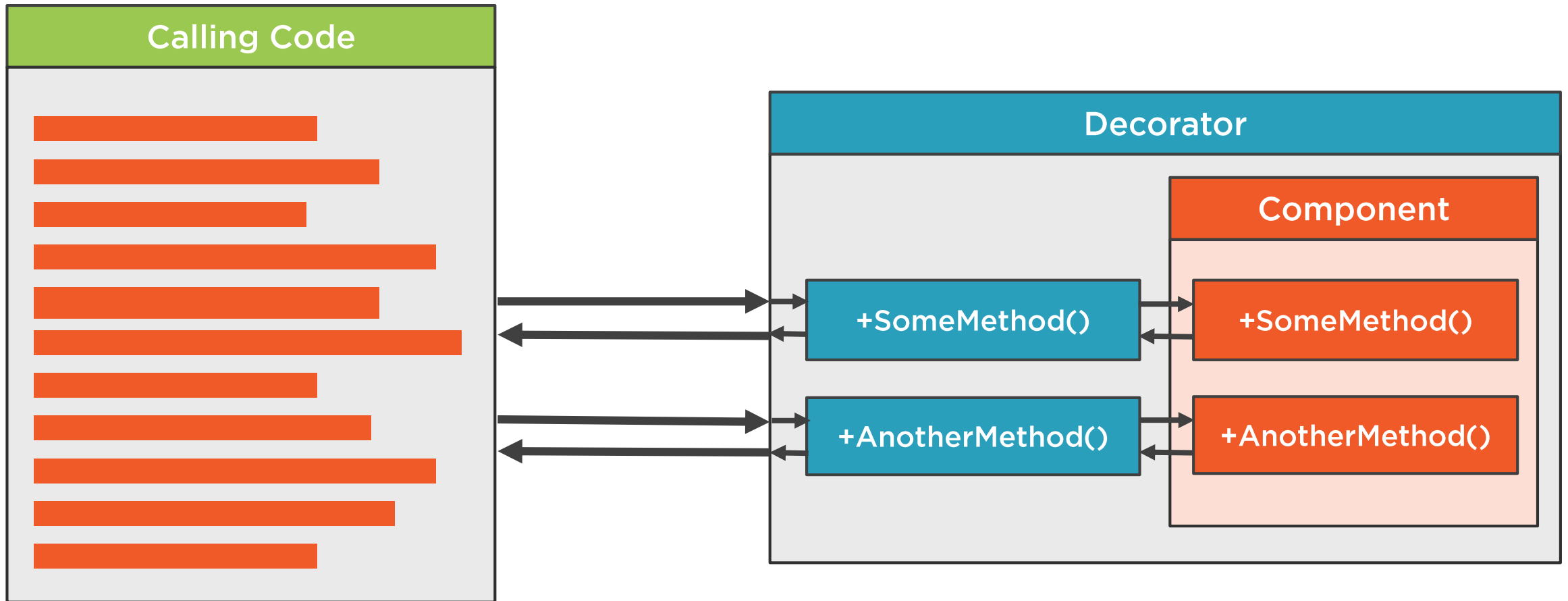
A structural design pattern used for dynamically adding behavior to a class



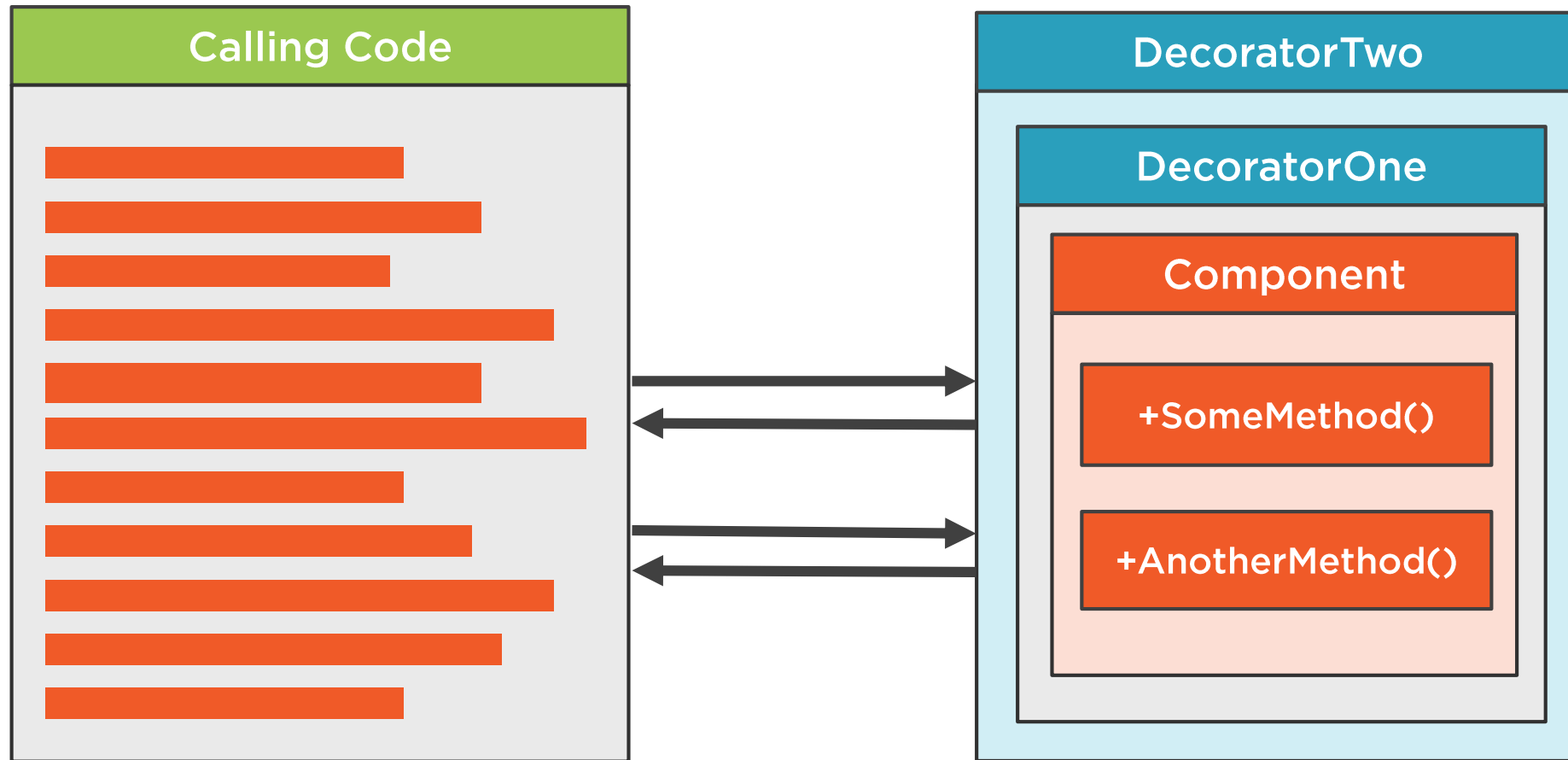
Decorator Pattern Structure



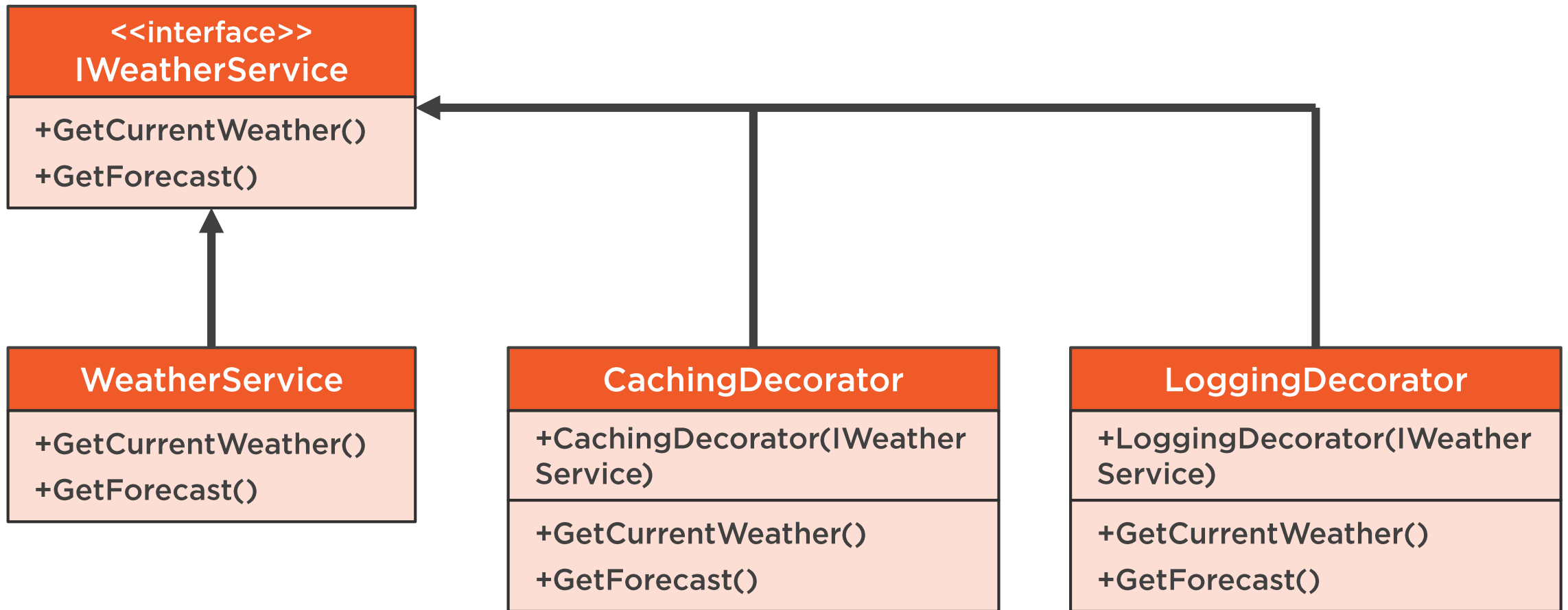
Onion Structure of the Decorator Pattern



Multiple Levels of Decoration



Decorator Pattern Example



Using Decorator Objects

```
// Standard component instantiation  
IWeatherService weatherService = new WeatherService();
```

```
// Instantiation with decorator objects  
IWeatherService weatherService =  
    new CachingDecorator  
        new LoggingDecorator(  
            new WeatherService())));
```



Logging Decorator

Log how often a method was called, how long it took, parameters and responses

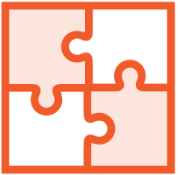


Caching Decorator

Cache weather conditions, forecasts for a city to reduce the number of external API calls



Decorator Demo Summary



Multiple decorators can be used in conjunction with one another

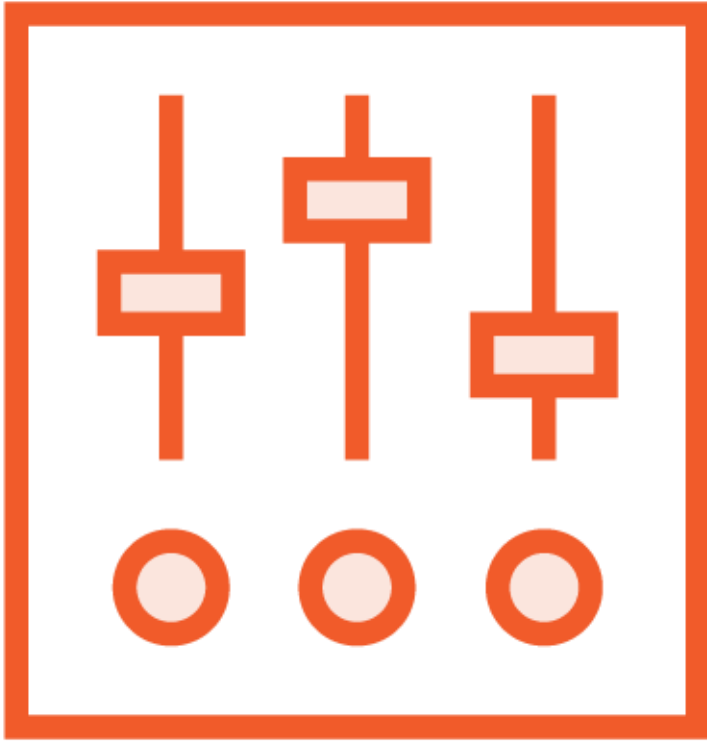


Each decorator can focus on a single task, promoting separation of concerns



Decorator classes allow functionality to be added dynamically





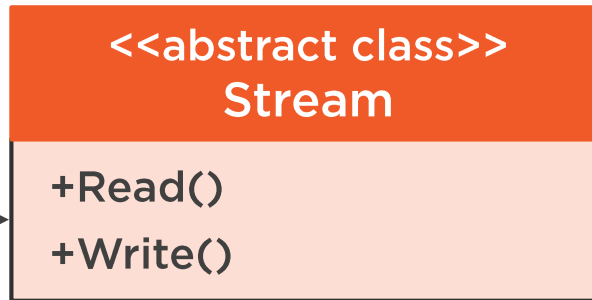
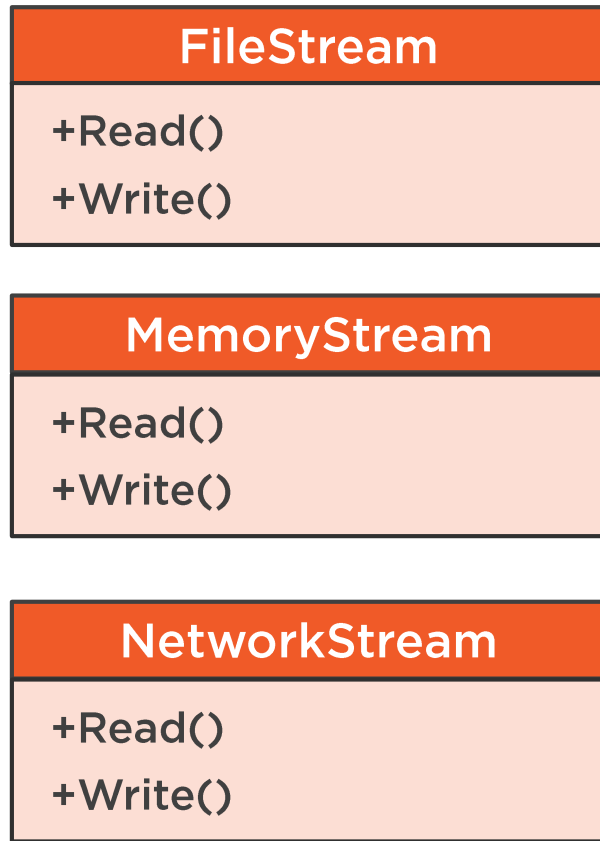
IoC container configuration varies between .NET Core and libraries in .NET Framework



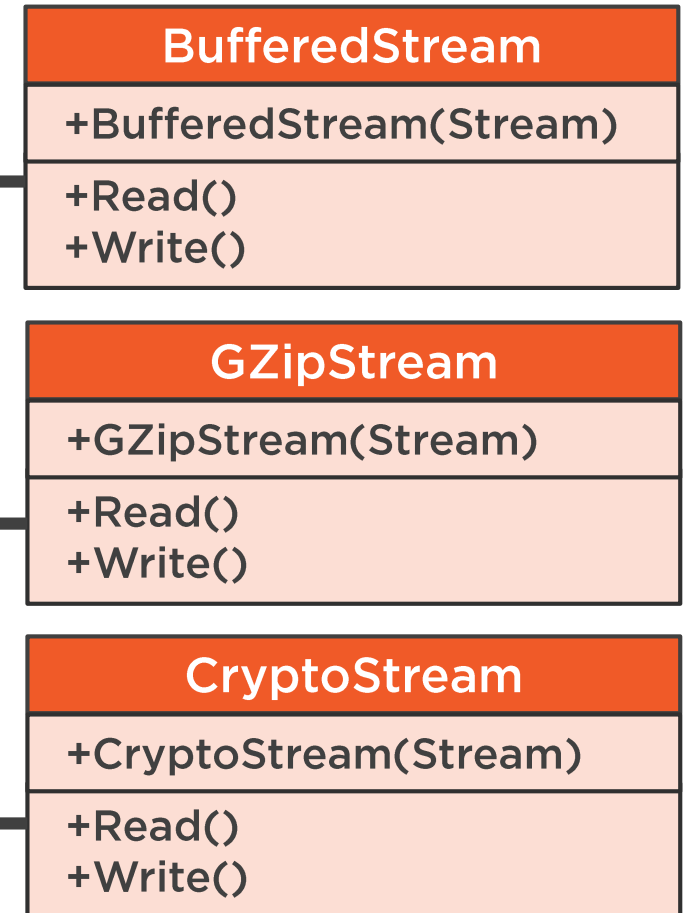
For other IoC libraries like Autofac and Ninject or others, check the documentation



Concrete Implementations



Decorators



You will frequently see the decorator pattern used in frameworks and libraries





Making Use of the Decorator Pattern in Your Projects



Decorator Pattern Use Cases



Cross cutting concerns

Manipulate data going to/from component

Question

What if your component does not have an interface/extend from a base class?

Answer

Extract an interface from the class



What if You Cannot Change the Component?

