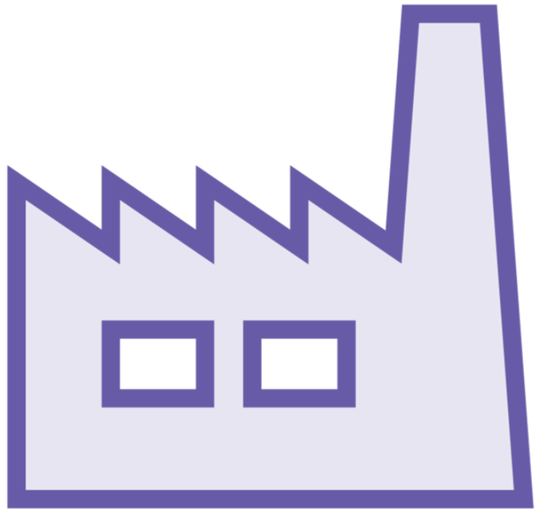
Advanced Creational Patterns

Zachary Bennett
Software Engineer

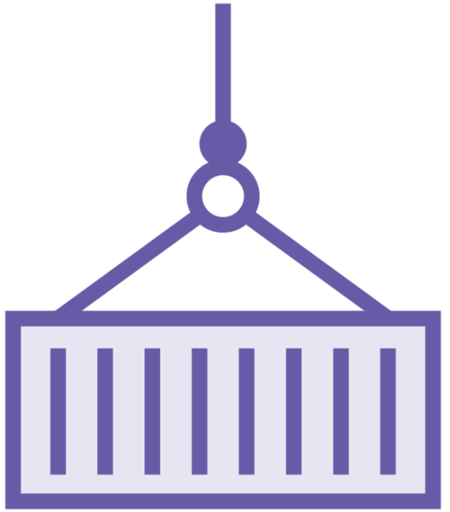
@z_bennett_ zachbennettcodes.com

Advanced Creational Patterns



Factory Method
Create objects dynamically by type

Abstract Factory
Create object families



Dependency Injection
Decouple dependencies

Benefits

More maintainable

More reusable

More flexible

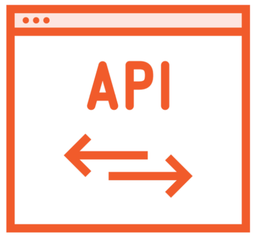
More testable

Create dynamic systems using best practices

Factory Method Pattern

Why the Factory Method Pattern?

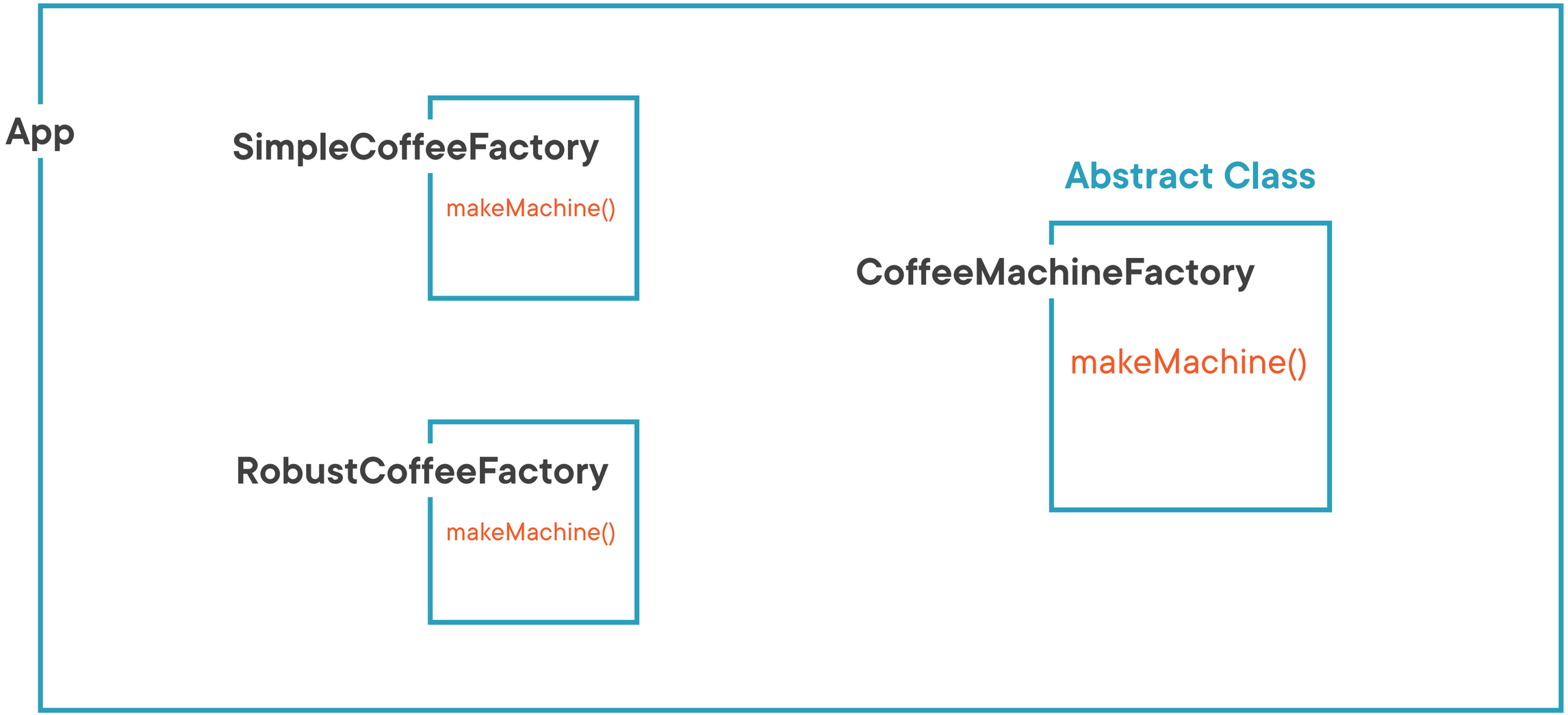
Flexibility



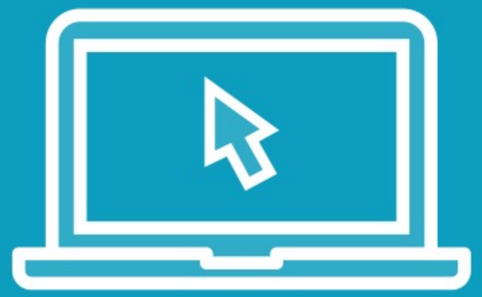
Generic code that is easily extensible

Decoupling

The Factory Method Pattern



Demo



Implement the Factory Method Pattern

CoffeeMachineFactory

Delegate object creation

Abstract Factory Pattern

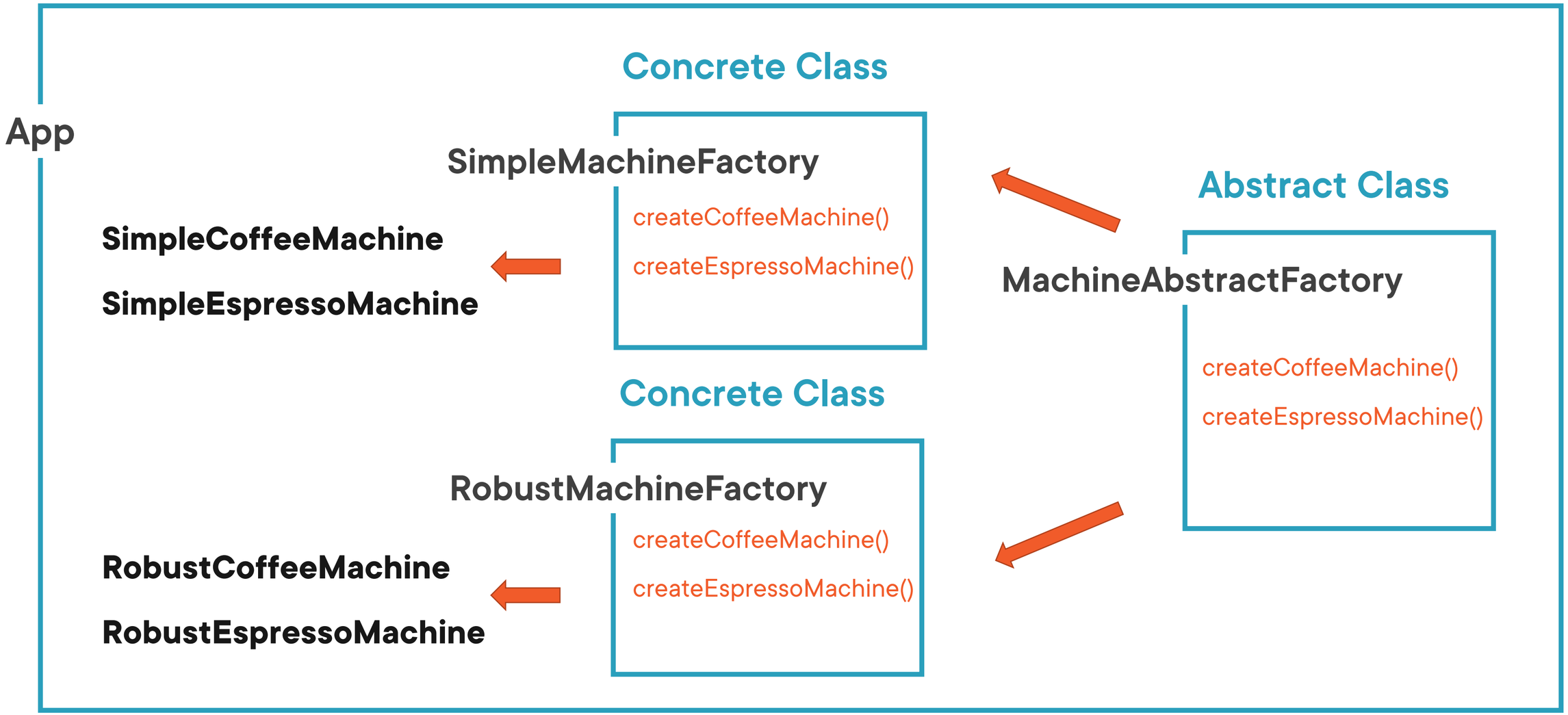
Why the Abstract Factory Pattern?

You want to create families of objects

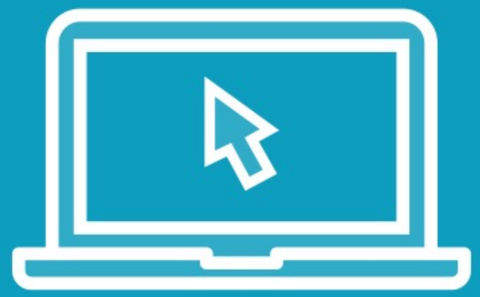
You want to hide concrete object creation from clients

You want to decouple object creation from dependent classes

The Abstract Factory Pattern



Demo



Implement the Abstract Factory Pattern

CoffeeFactory

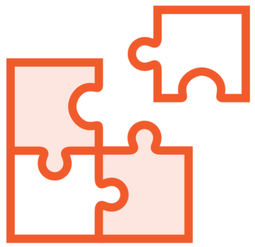
Concrete factories

Decouple object family creation

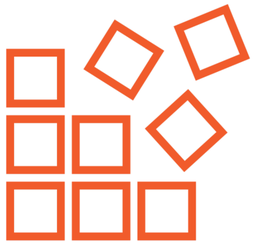
Dependency Injection Pattern

Why Dependency Injection?

Less complex testing

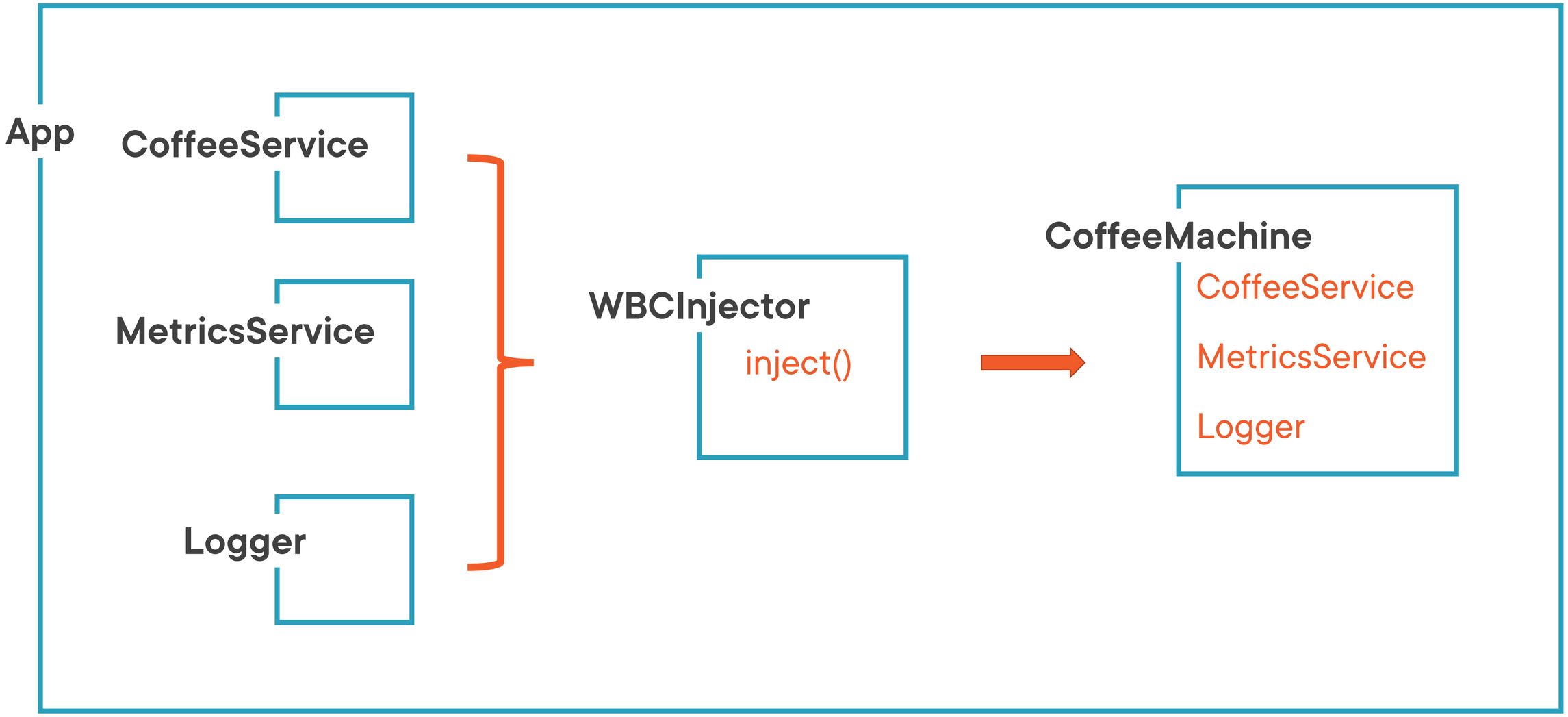


Decoupling of dependencies

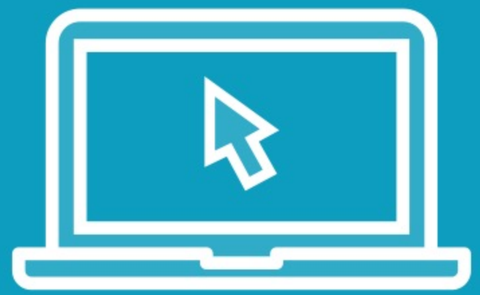


Better separation of concerns

Dependency Injection



Demo



Implement Dependency Injection

Simple constructor injection

Summary

SOLID Design Principles

Creational Patterns

- Singleton
- Builder
- Prototype
- Factory Method
- Abstract Factory
- Dependency Injection

More reusable, maintainable, and testable