

C# Design Patterns: Facade

INTRODUCTION OVERVIEW



David Starr

@elegantcoder | elegantcode.com



Overview



The problems to solve

Introducing the Façade pattern

Using Façades in practice



The Problem to Solve

ClassA

- + Method1
- + Method2
- + Method3
- + Method4
- + Method5
- + Method6
- + Method7
- + Method8
- + Method9
- + Method10

Program

- ClassA.Method2
- ClassA.Method1
- ClassA.Method4
- ClassA.Method3



Applying Façade

ClassA

- + Method1
- + Method2
- + Method3
- + Method4
- + Method5
- + Method6
- + Method7
- + Method8
- + Method9
- + Method10

FacadeClass

- + Method1
- + Method2
- + Method3
- + Method4

Program

- FacadeClass.Method2
- FacadeClass.Method1
- FacadeClass.Method4
- FacadeClass.Method3



```
static void Main(string[] args)
{
    var servicesFacade = new ServicesFacade();
    servicesFacade.DoSomethingExplicitA();
    servicesFacade.DoSomethingExplicitB();
}
```

Demo

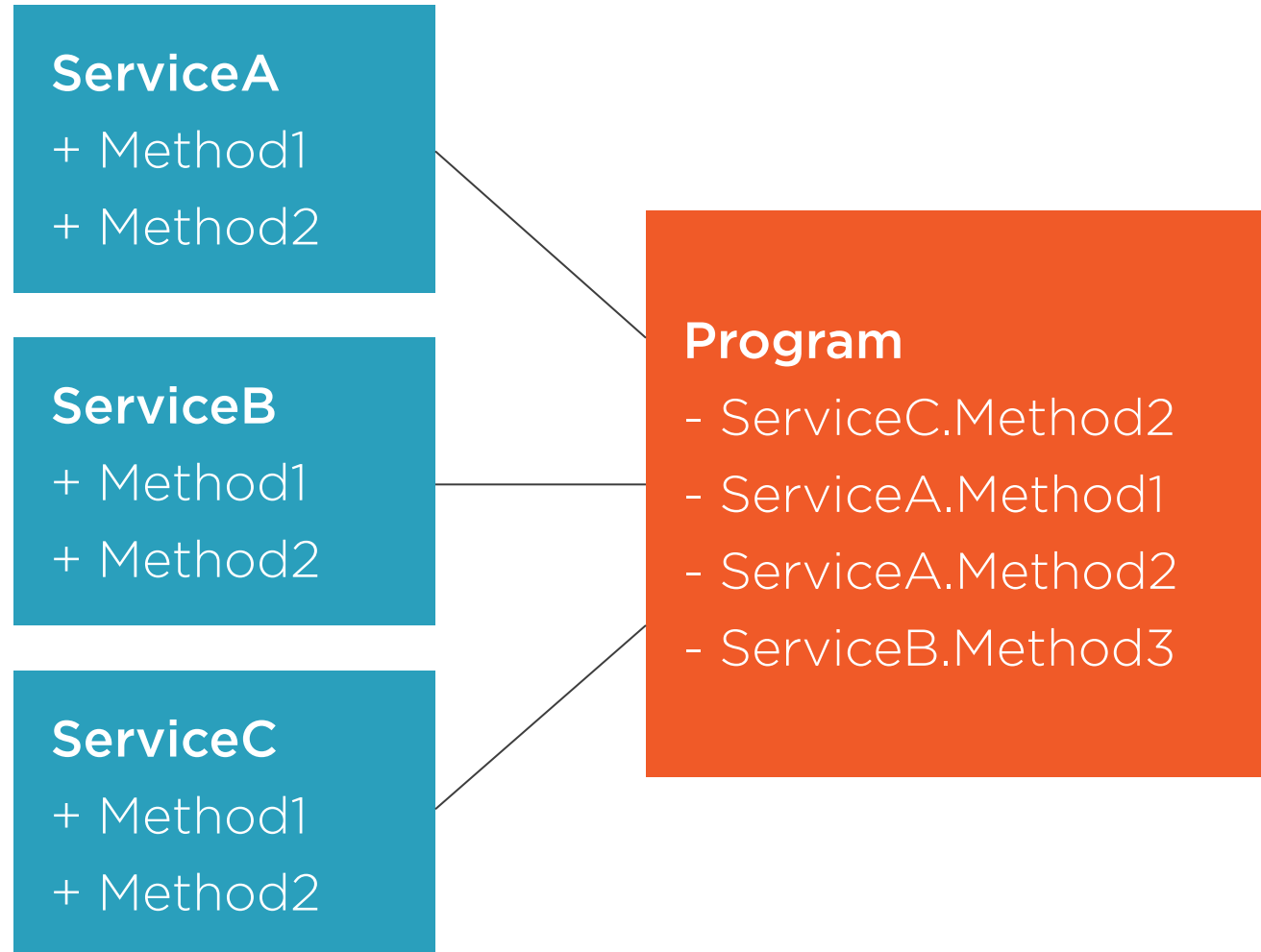


Making it tangible

See working code



The Problem to Solve



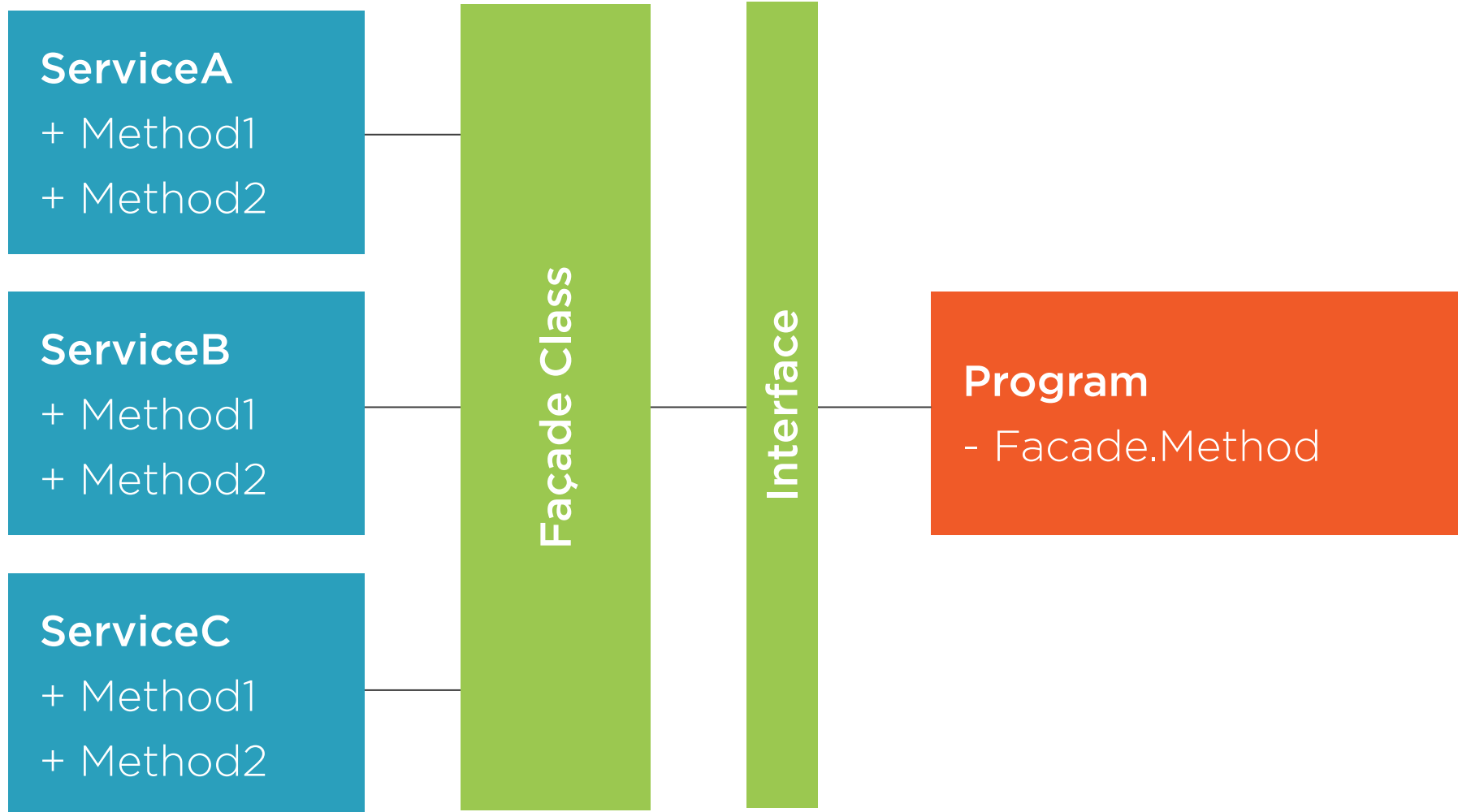
```
static void Main(string[] args)
{
    | var serviceA = new ServiceA();
    | serviceA.Method1();

    | var serviceB = new ServiceB();
    | string serviceBString = serviceB.Method2();

    | var serviceC = new ServiceC();
    | string serviceCString = serviceC.Method2();

    Console.WriteLine($"{serviceBString} - {serviceCString}");
}
```


The Façade Pattern



Demo



Making it tangible

See it in real code



Summary



Use Façade to provide a single interface to multiple worker classes

Using a Façade interface is a best practice

Can be used to mask giant classes with lots of methods

