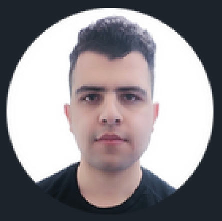
Method Chaining

Let’s make a Pizza by chaining methods.

```
IPizzaBuilder pizzaBuilder = new PizzaBuilder();

Pizza pizza = pizzaBuilder
        .SetSize("Large")
        .SetCrust("Thin Crust")
        .SetSauce("Tomato")
        .AddTopping("Pepperoni")
        .AddTopping("Mushrooms")
        .Build();

pizza.Display();
```

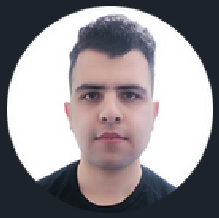


Elliot One

IPizzaBuilder Interface

The secret is that each method in IPizzaBuilder returns the same type as IPizzaBuilder, thus it enables calling its method on the returned type! However, the Build() method will return the final Pizza object. IPizzaBuilder is a fluent interface.

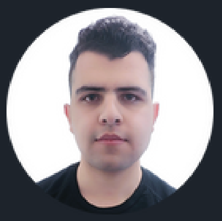
```
public interface IPizzaBuilder
{
    IPizzaBuilder SetSize(string? size);
    IPizzaBuilder AddTopping(string topping);
    IPizzaBuilder SetCrust(string? crust);
    IPizzaBuilder SetSauce(string sauce);
    Pizza Build();
}
```



Elliot One

Implementation of PizzaBuilder

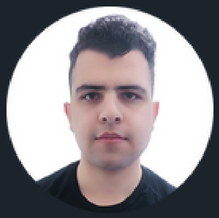
```csharp
public class PizzaBuilder : IPizzaBuilder
{
    private readonly Pizza _pizza = new();

    public IPizzaBuilder SetSize(string? size)
    {
        _pizza.Size = size;
        return this; // Return the builder instance
    }

    public IPizzaBuilder AddTopping(string topping)
    {
        _pizza.Toppings ??= new List<string>();
        _pizza.Toppings.Add(topping);
        return this; // Return the builder instance
    }

    public IPizzaBuilder SetCrust(string? crust)
    {
        _pizza.Crust = crust;
        return this; // Return the builder instance
    }

    public IPizzaBuilder SetSauce(string sauce)
    {
        _pizza.Sauce = sauce;
        return this; // Return the builder instance
    }

    public Pizza Build()
    {
        if (string.IsNullOrEmpty(_pizza.Size))
        {
            throw new InvalidOperationException(
                "Pizza size is not set.");
        }

        return _pizza;
    }
}
```



Elliot One

Pizza Class

```csharp
public class Pizza
{
    public string? Size { get; set; }
    public List<string> Toppings { get; set; } = new();
    public string? Crust { get; set; }
    public string? Sauce { get; set; }

    public void Display()
    {
        Console.WriteLine($"Size: {Size}");
        Console.WriteLine($"Crust: {Crust}");
        Console.WriteLine($"Sauce: {Sauce}");
        Console.WriteLine("Toppings:");

        foreach (var topping in Toppings)
        {
            Console.WriteLine($" - {topping}");
        }
    }
}
```



Elliot One

Elliot One

Enjoyed Reading This?

Reshare and Spread Knowledge.