

# Module 16: "Template Method"



# Agenda

- ▶ Introductory Example: Pretty Printing Objects
- ▶ Challenges
- ▶ Implementing the Template Method Pattern
- ▶ Pattern: Template Method
- ▶ Overview of Template Method Pattern
- ▶ Pros and Cons



# Introductory Example: Pretty-printing Objects

```
class XmlPrettyPrinter
{
    public void PrintPreamble() =>
        Console.WriteLine(@"<?xml version=""1.0"" encoding=""UTF-8""?>");
    public void PrintBegin( string className ) =>
        Console.WriteLine($"<{className}>");
    public void PrintEnd( string className ) =>
        Console.WriteLine($"</{className}>");
    public void PrintProperty( string name, object value ) =>
        Console.WriteLine($"    <{name}>{value}</{name}>");
}
```

```
XmlPrettyPrinter pp = new XmlPrettyPrinter();
pp.PrintPreamble();
pp.PrintBegin(nameof(person));
foreach( ... ){ pp.PrintProperty(...); }
pp.PrintEnd(nameof(person));
```



# Challenges

- ▶ Do we have to repeat everything?
- ▶ What if we need to create, say, a JSON pretty-printer?

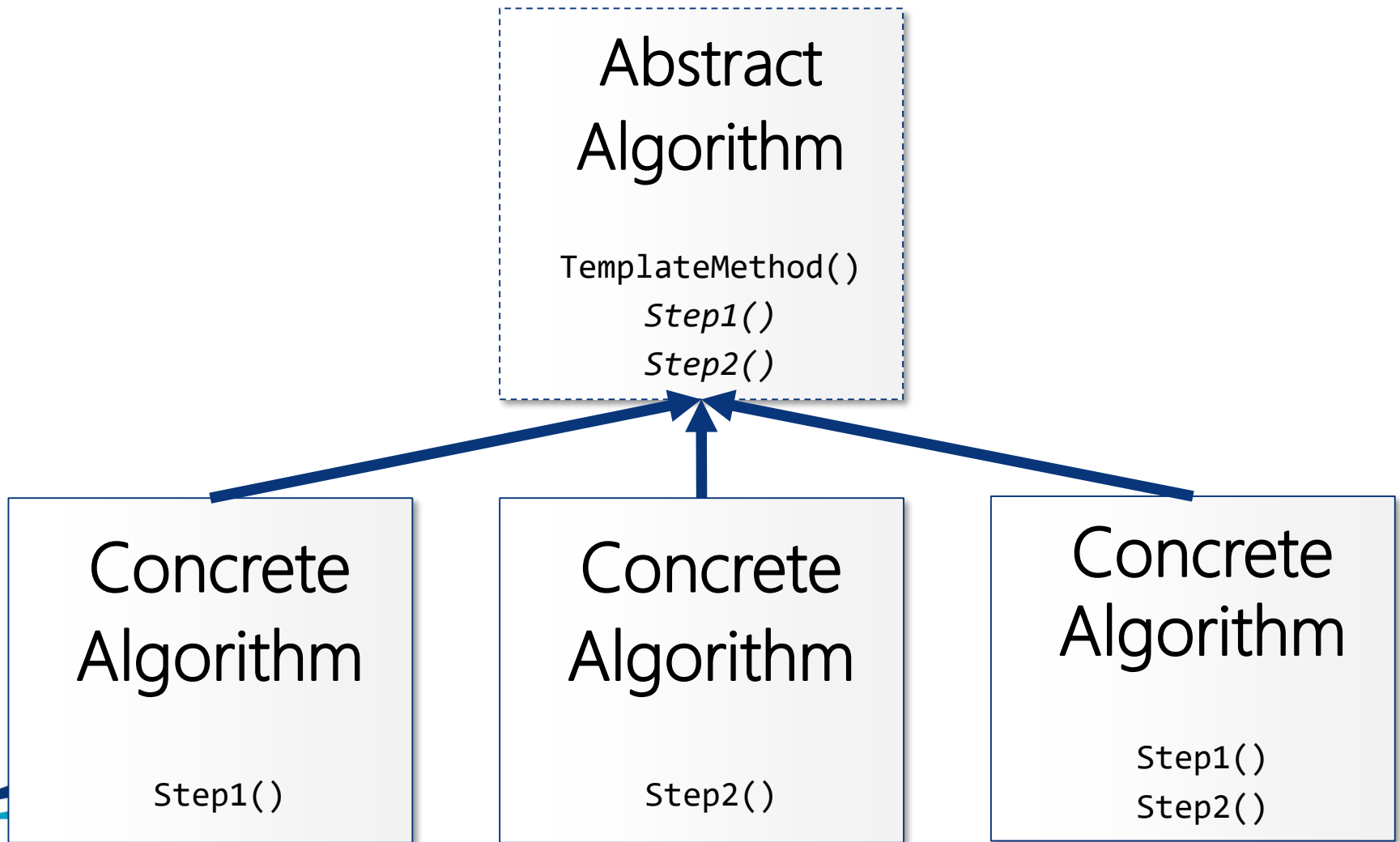


# Pattern: Template Method

- ▶ *Define the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.*
- ▶ Outline
  - Encapsulate general algorithm process in a template method
  - Use template method for multiply variations of same algorithm
  - Subclasses customize details of the individual steps
  - Base class template method always calls subclass methods
- ▶ Origin: Gang of Four



# Overview of Template Method Pattern



# Overview of Template Method Pattern

- ▶ Abstract Algorithm
  - Abstract base class for algorithm
  - Defines general algorithm flow in **TemplateMethod()**
  - Individual steps of the algorithm are available to be (re)defined in abstract or virtual **StepX()** methods
  
- ▶ Concrete Algorithm
  - Concrete subclass of Abstract Algorithm
  - (Re)defines any number of steps of the algorithm



# Pros and Cons

## ► Pros

- Simple way to achieve adaptation of algorithm steps
- Satisfies the Open/Closed Principle nicely

## ► Cons

- Algorithm steps are somewhat rigidly fixed
- Use Inheritance instead of Composition
  - See Strategy Pattern





