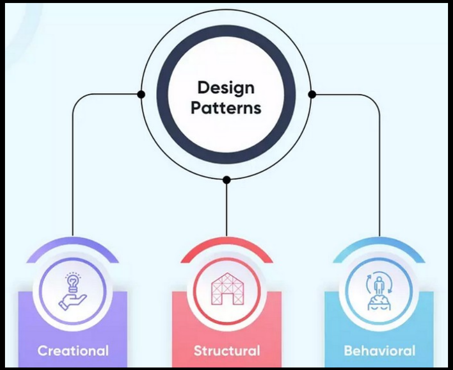


Creational Design Pattern << Builder >>

Ajith de Silva @epita.fr | 2025

Creational Design Patterns :: About

Concerned with Object-Oriented concept

Offer how to create instances form objects in:

… Flexible

… Efficient

… Maintainable

Provide INTERFACE for creating Instances in SUPPER CLASS

Allows SUBCLASSES to alter TYPE of instance to create

Great Decoupling with other code. HIGH Flexibility

Centralize creational logic

Creational Design Patterns

1. **Singleton**

   … One INSTANCE.

   … Global access

2. **Factory**

   … Interface to create instances in supper class.

   … Subclass decides type of Instance`

3. **Abstract Factory**

   … Group similar Factory based on Instances types

4. **Builder**

   … Create complex instance step by step

5. **Prototype**

   … Clone Existing instance

Builder Design Pattern

1 **In Brief….**

… construct complex instance step by step

… helps to create different types and representation using same construction code

2 **HOW ??**

Define Abstract/Interface class with required methods/properties

… make a constructor with parameters that you want to use while creating instances

… these parameters will determine how to construct the instances

Implement concrete class by implementing the Define Abstract/Interface

Builder Design Pattern :: MORE

1 **Constructor overloading**

… helps to make faster implementation by using matching constructor

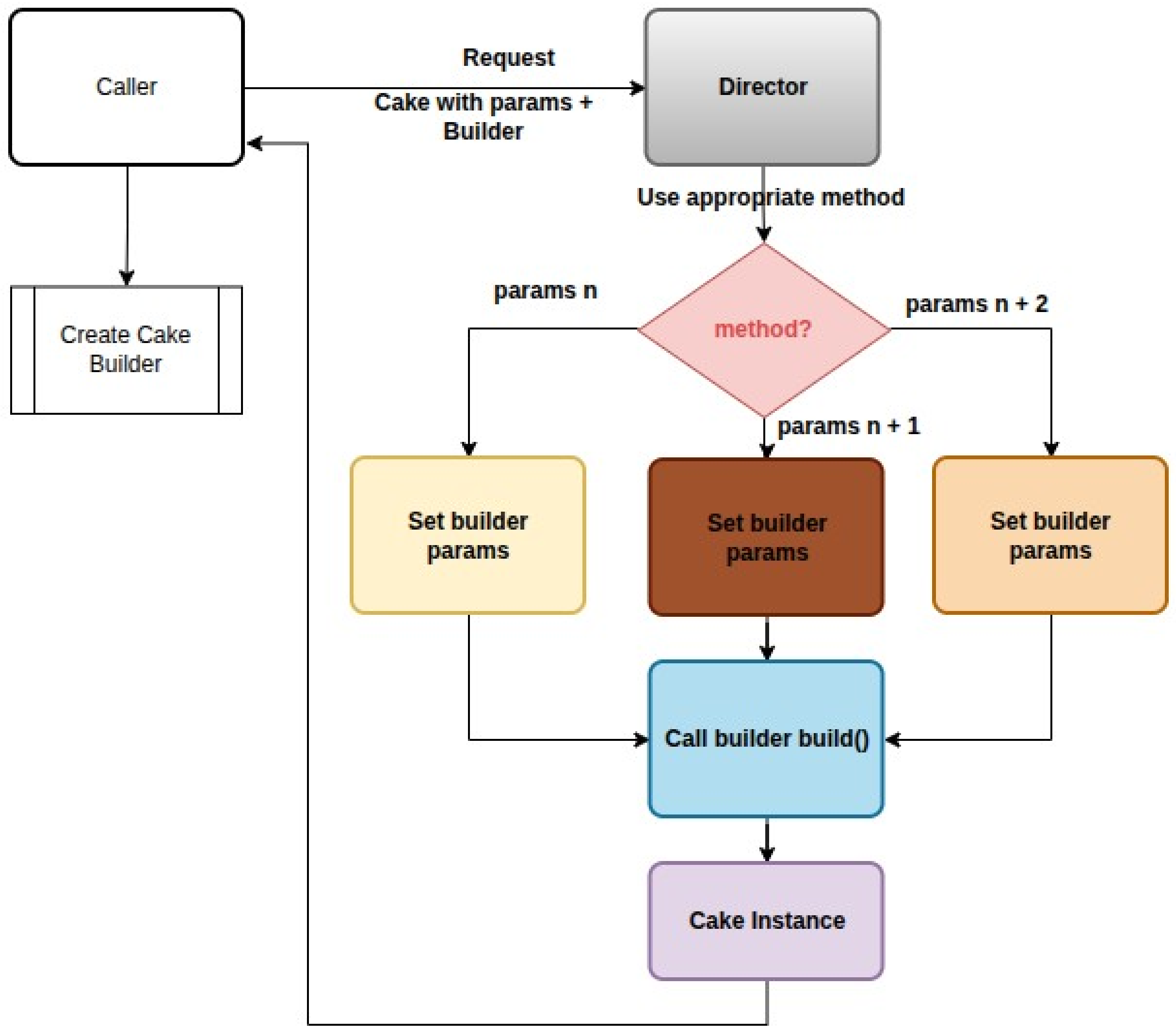
… Reduces concrete classes

… single responsibility principle

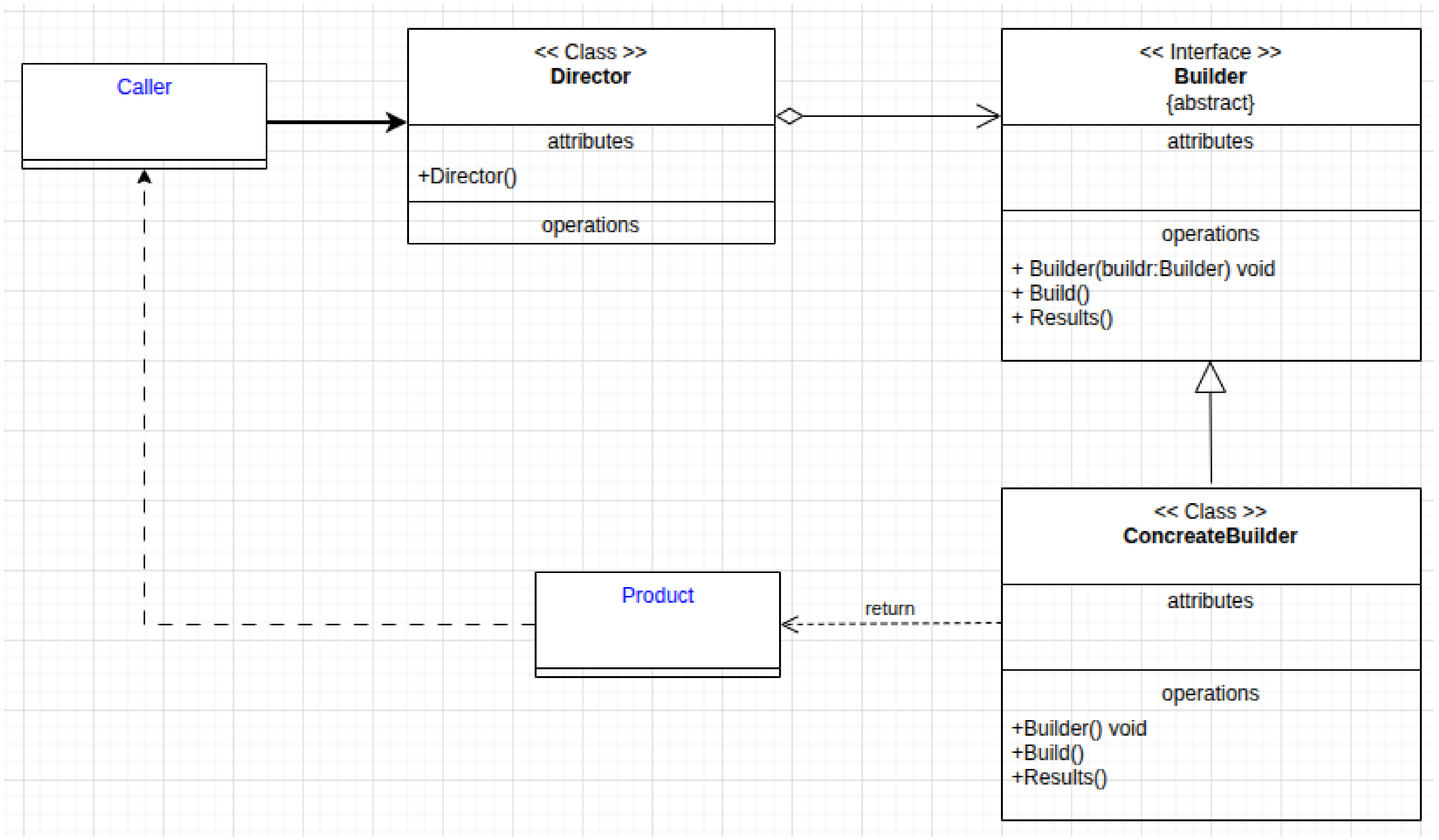
2 **Complexity??**

… Will be more complex when you have more parameters

Builder Design Pattern :: Flowchart



Builder Design Pattern :: UML



Builder Design Pattern :: How to Implement

1. **Define the interface with required methods & properties for the Builder**
2. **Implement the interface in concrete class for each implementation**
3. **Create a Director class**

   … must have a multiple methods to accept all possible parameters

   … (method overrides)

4. **Client/caller will use the Director to create required instance by passing builder instance & required parameters**

   … Director will use the given Builder to build requested instance by calling the concrete builder

5. **Director returns the created instance to the caller**

Builder Design Pattern :: Practical

1. **Check/study the builder sample**
2. **Run/Debug and check how it works**
3. **Extend concrete implementation**