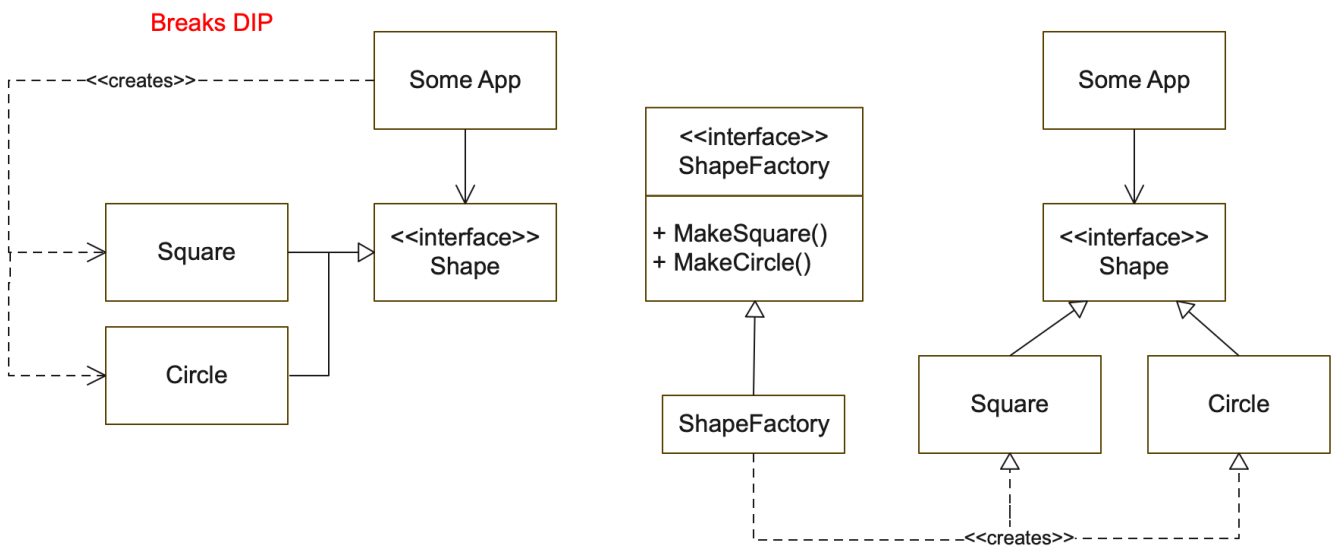
3. Factory Patterns(New)

Definition

Creational patterns.
Creating instances of concrete objects while depending only on abstract interfaces.
DIP.

Motivation



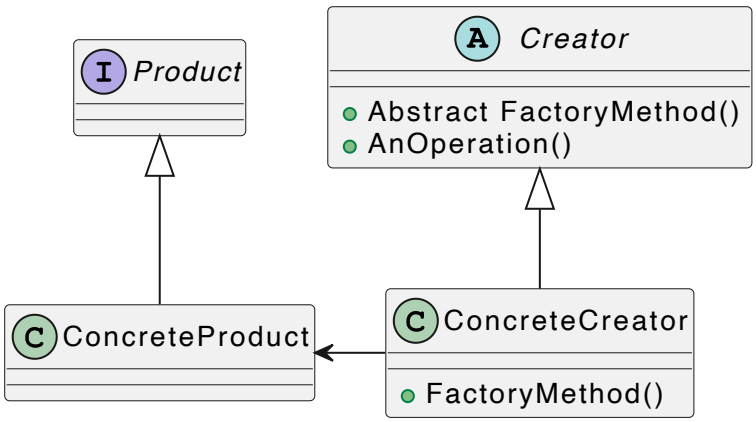
Factory Method

Definition: Defer instantiation to its subclasses.

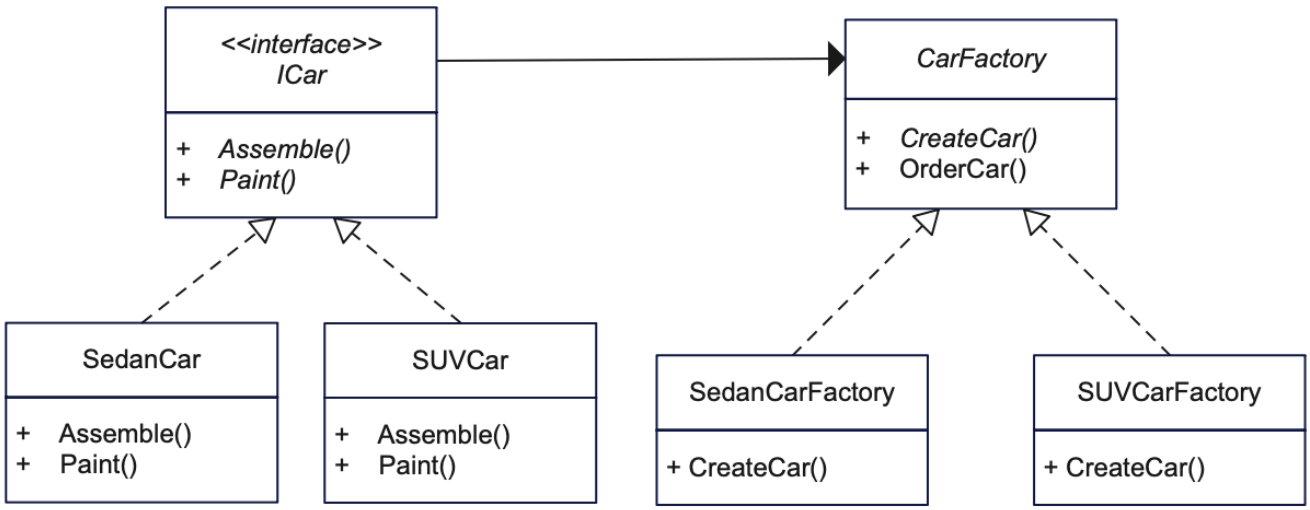
Inheritance. OCP.

Creating a single type of object an delegates which concrete class to instantiate to subclasses.

Concrete products refer to the interface, not class.



Example

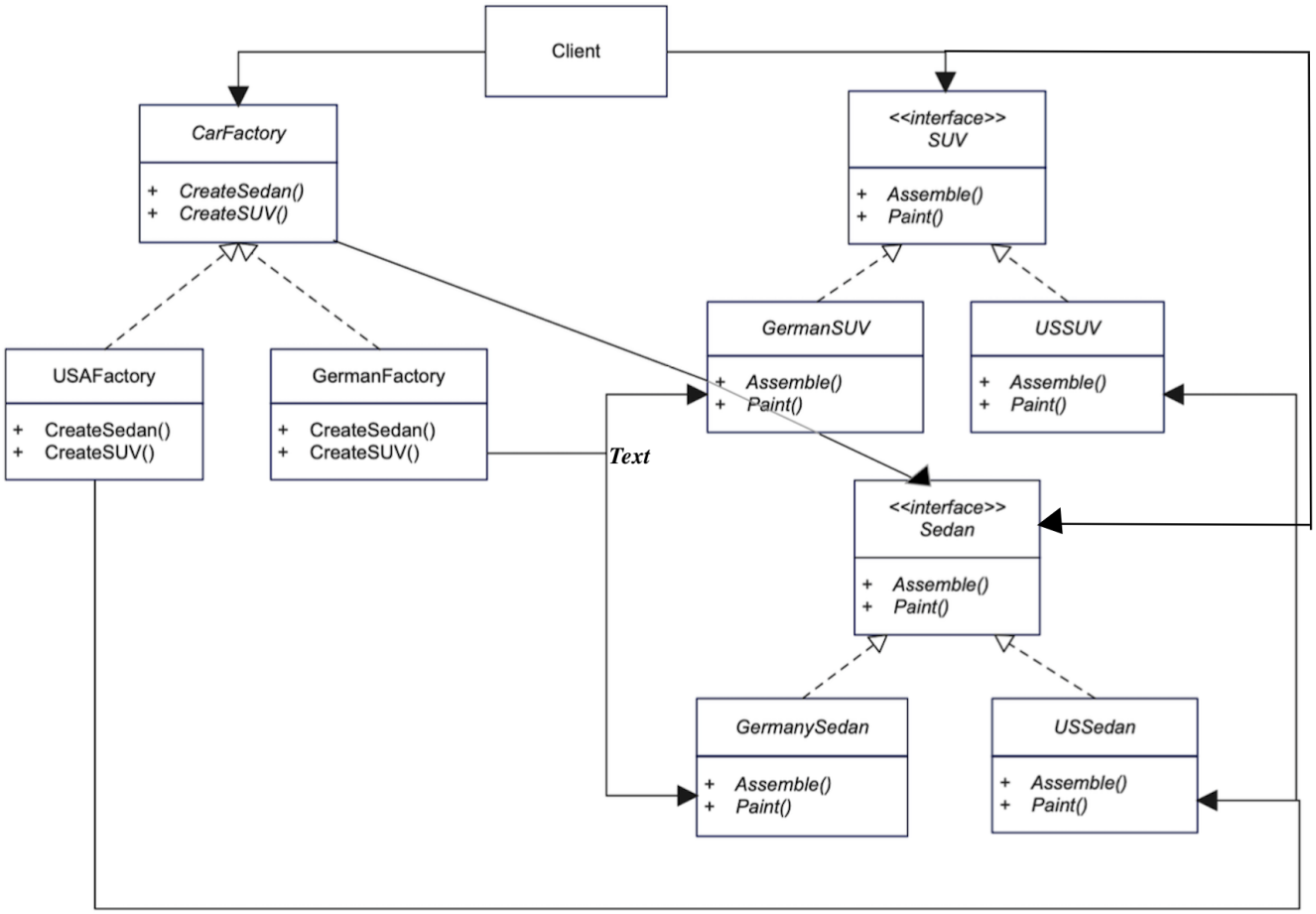


Abstract Factory

> **Definition:** Create families of related or dependent objects without specifying concrete classes

Composition. OCP.

Example



Pros / Cons

	Factory Method Pattern	Abstract Factory Pattern
Pros	- Loose coupling - Extensible - Customizable - Supports unit testing	- Creates families of related objects - Loose coupling - Extensible - High level of abstraction
Cons	- Requires subclassing - Limited to creating a single product type	- Can lead to complex class hierarchies - Increased initial setup effort - Less flexibility in product variations
Suitable Use Cases	- When creating a single type of object with varying implementations	- When working with families of related objects or collaborating object types

SOLID

S Factories make it possible to create a class with the single responsibilty to create objects.
O New product variations added through subclassing without modifying existing code.
L Supports LSP indirectly by enabling substitutability as the created objects adhere to a common interface/base class. Substitutability is ensured within the family of objects in the abstract factory.

I Indirectly supports ISP as it allows clients to only depend on necessary methods and interfaces, promoting loose coupling.
D Loose coupling. Reduce the dependency of your application on concrete classes.

Comparison

Singleton.
Decorator (during run time)
Template method (also compile time)