Factory Method

> Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

- AKA Virtual Constructor.
- Factory method pattern is **class scope**.
- The main technique used in factory method is inheritance.
- In factory method, a class lets its subclass decide what it wants to associate with.
- One way to know if the pattern is factory method or abstract factory is to look for the context of the client.
    - If the client is an object outside of the hierarchy using association → Abstract Factory.
    - If the client is the class itself, which postpones the decision to a derived class → Factory Method.
- Can use Factory Method very effectively to create base classes with common functionality and let some of the derived classes change a few of the details around.
    - This is OCP compliant, because you don't have to modify the base class if you want to change what each of the base class should have.

When to use Factory Method?

- A class can't anticipate the class of objects it must create. A class wants its subclasses to specify the objects it creates.
- Classes delegate responsibility to one of the several helper subclasses and you want to localise the knowledge of which helper subclass is the delegate.

Consequences of using Factory Method

- Provides hooks for subclasses.
- Connects parallel class hierarchies.

Factory Method vs. Other Patterns

- Abstract Factory often implemented with Factory Method.
    - The Abstract Factory itself has derived classes inheriting from it.
    - The client uses association to talk to the Factory.
    - But the Factory itself uses inheritance.
    - If you focus only on the Factory and its derived classes, that’s Factory Method.
- Factory Methods usually called within Template Methods.
- Prototypes don’t require subclassing the creator. However, they often require initialising operation on the product class. Factory Method doesn’t require such an operation.
- Disadvantage → proliferation of subclasses. You may end up creating many different classes to have different combinations.