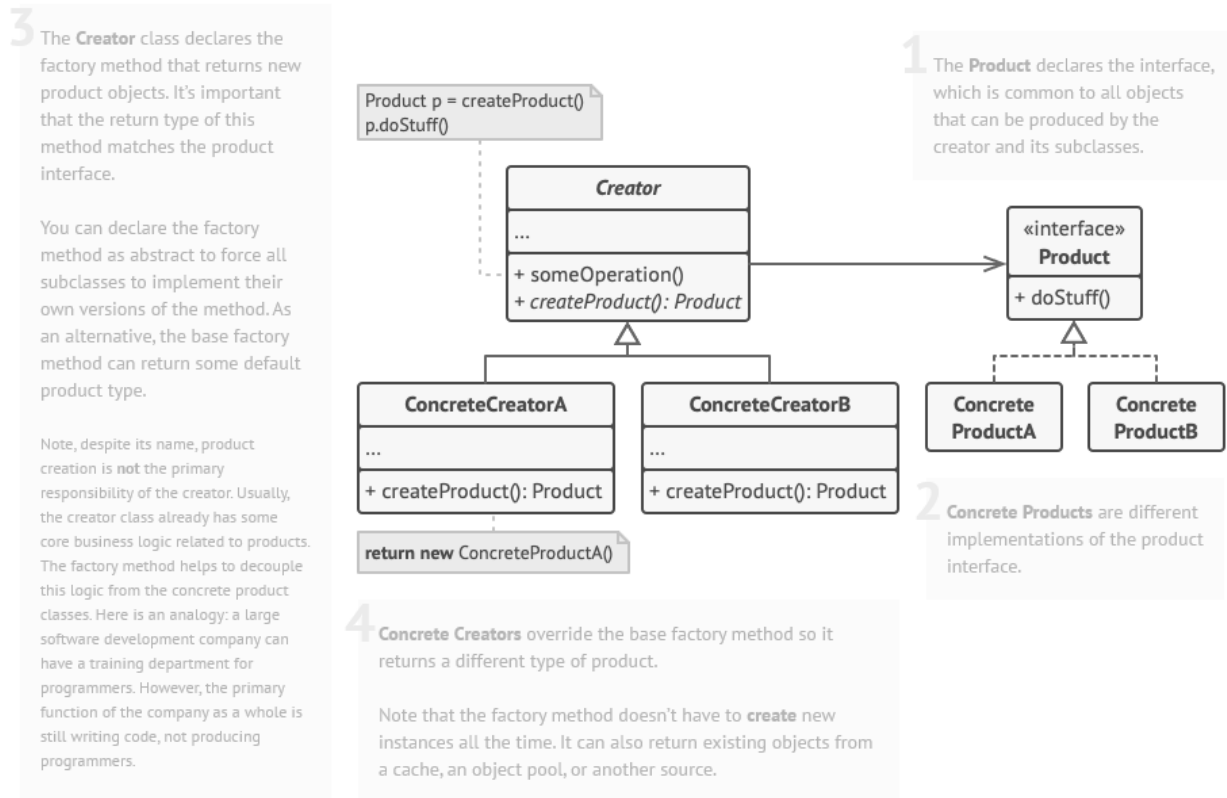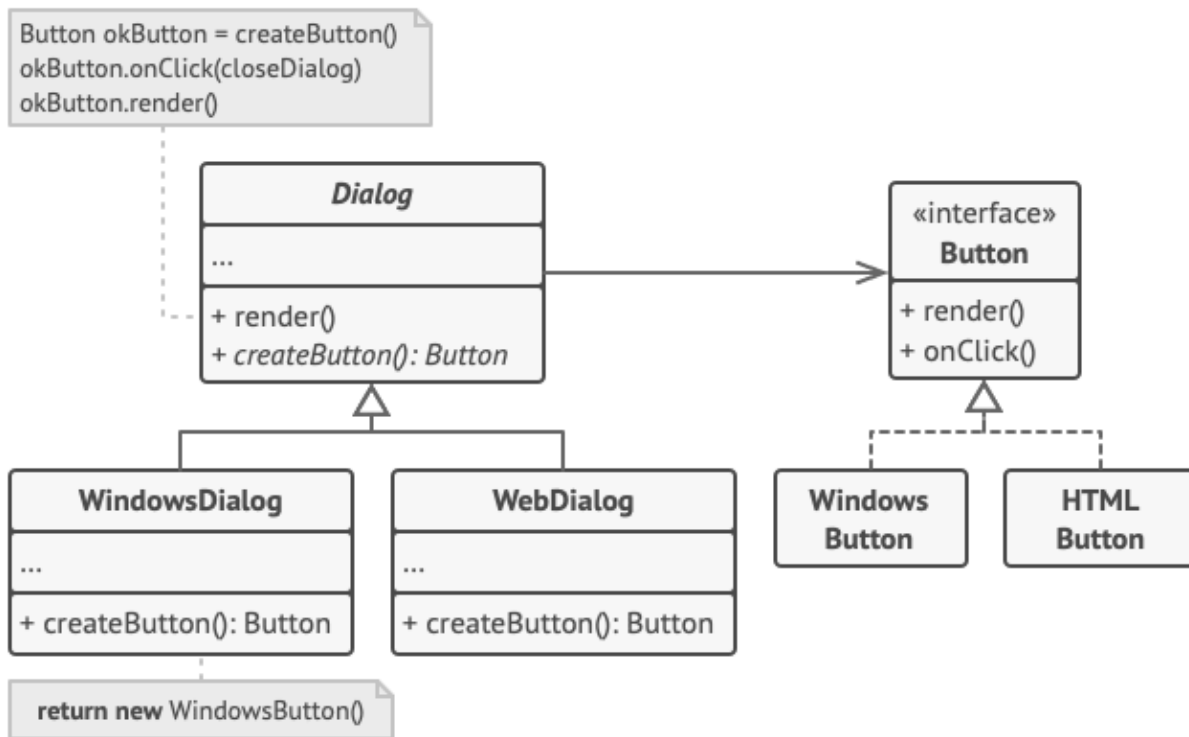# Intent

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

Structure

3 The **Creator** class declares the factory method that returns new product objects. It's important that the return type of this method matches the product interface.

You can declare the factory method as abstract to force all subclasses to implement their own versions of the method. As an alternative, the base factory method can return some default product type.

Note, despite its name, product creation is **not** the primary responsibility of the creator. Usually, the creator class already has some core business logic related to products. The factory method helps to decouple this logic from the concrete product classes. Here is an analogy: a large software development company can have a training department for programmers. However, the primary function of the company as a whole is still writing code, not producing programmers.

Product p = createProduct()
p.doStuff()

Creator

...

+ someOperation()
+ *createProduct(): Product*

ConcreteCreatorA

...

+ createProduct(): Product

return new ConcreteProductA()

ConcreteCreatorB

...

+ createProduct(): Product

«interface»
Product

+ doStuff()

**Concrete
ProductA**

**Concrete
ProductB**

1 The **Product** declares the interface, which is common to all objects that can be produced by the creator and its subclasses.

2 **Concrete Products** are different implementations of the product interface.

4 **Concrete Creators** override the base factory method so it returns a different type of product.

Note that the factory method doesn't have to **create** new instances all the time. It can also return existing objects from a cache, an object pool, or another source.

Example

```
Button okButton = createButton()
okButton.onClick(closeDialog)
okButton.render()
```

Dialog
...
+ render() + *createButton(): Button*

«interface» **Button**
+ render() + onClick()

WindowsDialog
...
+ createButton(): Button

WebDialog
...
+ createButton(): Button

Windows Button

HTML Button

```
return new WindowsButton()
```

The cross-platform dialog example.

Applicability

- Use the Factory Method when you don't know beforehand the exact types and dependencies of the objects your code should work with.

- Use the Factory Method when you want to provide users of your library or framework with a way to extend its internal components.

- Use the Factory Method when you want to save system resources by reusing existing objects instead of rebuilding them each time.