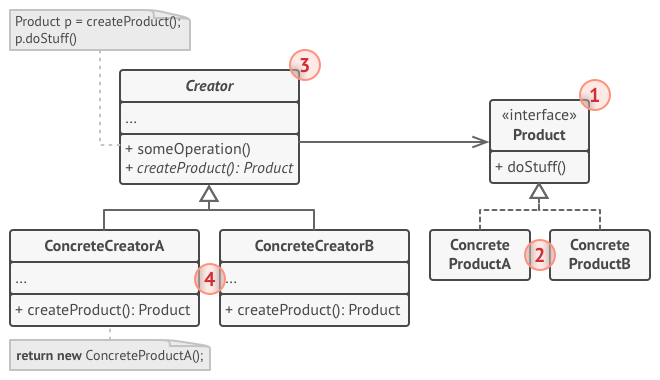
**ФАБРИЧНЫЙ МЕТОД -** определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов. Основан на наследовании: создание объектов делегируется субклассам, реализующим фабричный метод, где и указывается конкретный механизм создания нужных объектов. Не единственный, но один из наиболее мощных приемов обеспечения инверсии зависимостей - "создатель" не зависит от конкретных реализаций продукта, а работает с его абстракцией.

ПРИМЕНЕНИЕ:

- когда заранее неизвестны типы и зависимости объектов, с которыми должен работать ваш код

- когда вы хотите дать возможность пользователям расширять части вашего фреймворка или библиотеки

- когда вы хотите экономить системные ресурсы, повторно используя уже созданные объекты, вместо порождения новых

РЕАЛИЗАЦИЯ:

**1. ПРОДУКТ**: определяет общий интерфейс объектов, которые может произвести создатель и его подклассы. Например, "пицца".

**2. КОНКРЕТНЫЕ ПРОДУКТЫ**: содержат код различных продуктов. Будут отличаться реализацией, но интерфейс у них будет общий. Например, "нью-йоркская вегетарианская пицца" или "чикагская вегетарианская пицца"

**3. СОЗДАТЕЛЬ**: объявляет фабричный метод, который должен возвращать новые объекты продуктов. Тип результата метода должен совпадать с общим интерфейсом продуктов. Метод может объявляться абстрактным, чтобы все подклассы реализовали его по-своему, но также может иметь и дефолтную реализацию, возвращающую некий стандартный продукт. Сам класс, как правило, не ограничивается ФМ, и имеет код, который работает с продуктом, который возвращает ФМ. Например, "пиццерия".

**4. КОНКРЕТНЫЕ СОЗДАТЕЛИ:** по-своему реализуют ФМ, производя те или иные конкретные продукты, но возвращая их абстракции. Каждая создает свои версии, например, вегетарианских пицц. ФМ не обязан только создавать новые объекты: например, может возвращать существующие объекты из какого-то хранилища или кэша. Например, "нью-йоркская пиццерия" или "чикагская пиццерия".

ПЛЮСЫ:

- избавляет класс от привязки к конкретным классам продуктов

- выделяет код производства продуктов в одно место, упрощая поддержку кода

- упрощает добавление новых продуктов в программу

- реализует принцип открытости/закрытости

МИНУСЫ:

- может привести к созданию больших параллельных иерархий классов, так как для каждого класса продукта надо создать свой подкласс создателя