Юрин Михаил

ПИ-15-1

Паттерн «Factory Method»

***1. Назначение и цель применение паттерна***

**Фабричный метод** — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов.

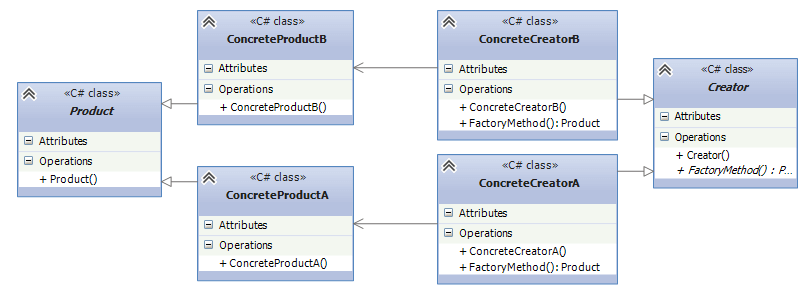
***Когда надо применять паттерн:***

• Когда заранее неизвестно, объекты каких типов необходимо создавать

• Когда система должна быть независимой от процесса создания новых объектов и расширяемой: в неё можно легко вводить новые классы, объекты которых система должна создавать.

• Когда создание новых объектов необходимо делегировать из базового класса классам наследникам

***2. UML представление паттерна***



• Абстрактный класс **Product** определяет интерфейс класса, объекты которого надо создавать.

• Конкретные классы **ConcreteProductA** и **ConcreteProductB** представляют реализацию класса Product. Таких классов может быть множество

• Абстрактный класс **Creator** определяет абстрактный фабричный метод FactoryMethod(), который возвращает объект Product.

• Конкретные классы **ConcreteCreatorA** и **ConcreteCreatorB** - наследники класса Creator, определяющие свою реализацию метода FactoryMethod(). Причем метод FactoryMethod() каждого отдельного класса-создателя возвращает определенный конкретный тип продукта. Для каждого конкретного класса продукта определяется свой конкретный класс создателя.

Таким образом, класс Creator делегирует создание объекта Product своим наследникам. А классы ConcreteCreatorA и ConcreteCreatorB могут самостоятельно выбирать какой конкретный тип продукта им создавать.

***3. Области применения***

• 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 создания новых.

***4. Особенности паттерна***



***5. Пример реализации паттерна C#***

