Factory Method

Фабричный метод

Назначение паттерна

Когда надо применять паттерн:

- Заранее неизвестно, объекты каких типов необходимо создавать
- Система должна быть независимой от процесса создания новых объектов, а также быть расширяемой: можно легко вводить новые классы, объекты которых надо создавать
- Когда создание новых объектов необходимо делегировать из базового класса классам наследникам

Делегирование — когда часть работы класс "перекладывает" на другие классы.

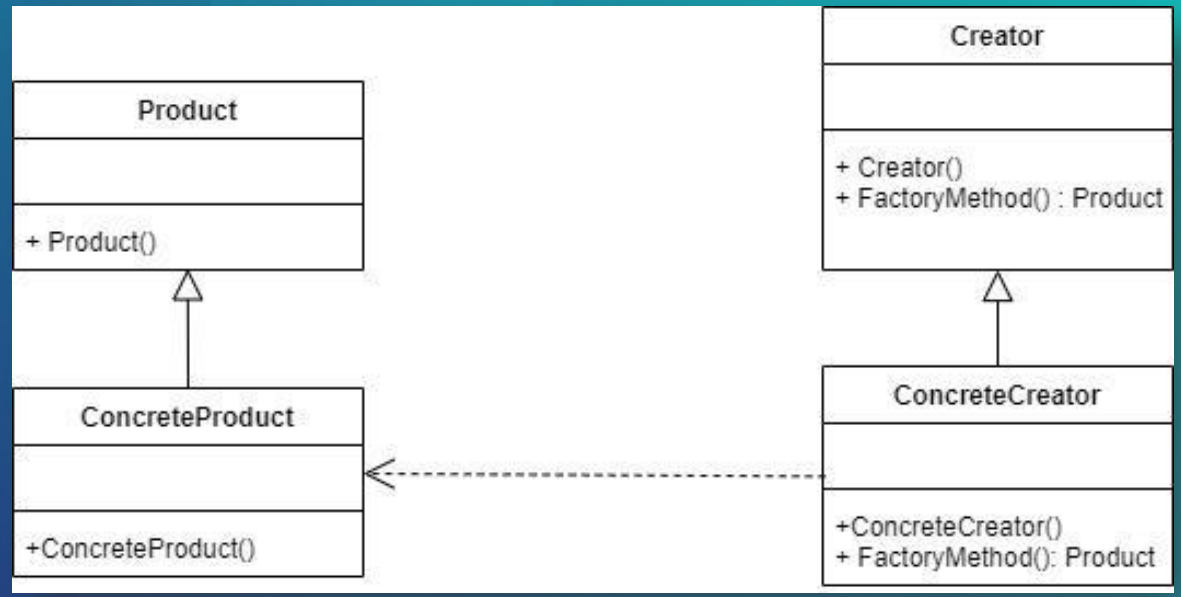
Общее описание

Фабричный метод (Factory Method) - это паттерн, который определяет интерфейс для создания объектов некоторого класса, но непосредственное решение о том, объект какого класса создавать происходит в подклассах.

В момент создания наследники могут определить, какой класс создавать.

UML Диаграмма

- Product — определяет интерфейс объектов, создаваемых абстрактным методом
- ConcreteProduct — реализует интерфейс Product
- Creator — объявляет фабричный метод, возвращающий объект типа Product. Может вызывать фабричный метод для создания объекта типа Product.
- ConcreteCreator — переопределяет фабричный метод таким образом, чтобы он создавал и возвращал объект класса ConcreteProduct.



Реализация паттерна C#

- Абстрактный класс Product определяет интерфейс класса, объекты которого надо создавать
- Конкретные классы **ConcreteProductA** и **ConcreteProductB** представляют реализацию класса Product.
- 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().

```
1  abstract class Product
2  {}
3
4  class ConcreteProductA : Product
5  {}
6
7  class ConcreteProductB : Product
8  {}
9
10 abstract class Creator
11 {
12     public abstract Product FactoryMethod();
13 }
14
15 class ConcreteCreatorA : Creator
16 {
17     public override Product FactoryMethod() { return new ConcreteProductA(); }
18 }
19
20 class ConcreteCreatorB : Creator
21 {
22     public override Product FactoryMethod() { return new ConcreteProductB(); }
23 }
```

Реализация паттерна C#

Для каждого конкретного класса продукта определяется свой конкретный класс создателя.

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

Последствия применения паттерна

Применение паттерна позволяет использовать в коде программы не конкретные классы, а манипулировать абстрактными объектами на более высоком уровне.

Достоинства и недостатки

- Паттерн позволяет сделать код создания объектов более универсальным, не привязываясь к конкретным классам (ConcreteProduct), а оперируя лишь общим интерфейсом (Product), а также позволяет установить связь между параллельными иерархиями классов.
- Недостатком же является необходимость создавать наследника Creator для каждого нового типа продукта (ConcreteProduct).

Slide title

1. Click to add Title
2. Click to add Title
3. Click to add Title
4. Click to add Title
5. Click to add Title