



Федеральное агентство железнодорожного транспорта
Сибирский государственный университет путей сообщения
Кафедра «Системный анализ и управление проектами»

Фабричный метод

Выполнила студентка
гр.БПИ-411
Шестакова А.А
Проверила Доцент
Красникова К.В.

Новосибирск 2017

Определение

- 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.

Задачи

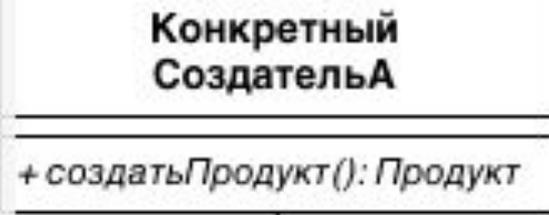
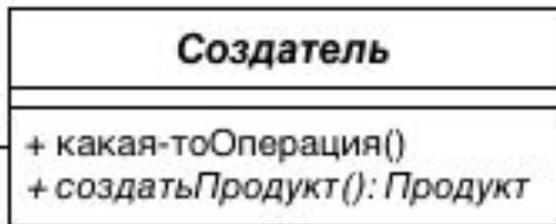
- Система должна оставаться расширяемой путем добавления объектов новых типов. Непосредственное использование выражения `new` является нежелательным, так как в этом случае код создания объектов с указанием конкретных типов может получиться разбросанным по всему приложению. Тогда такие операции как добавление в систему объектов новых типов или замена объектов одного типа на другой будут затруднительными. Паттерн `Factory Method` позволяет системе оставаться независимой как от самого процесса порождения объектов, так и от их типов.
- Заранее известно, когда нужно создавать объект, но неизвестен его тип.

Описание паттерна Factory Method

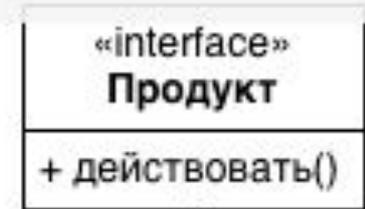
- Механизм полиморфизм (определение единого интерфейса);
- Создание объектов конкретных типов в специальном классе-фабрике:
 - ❖ Обобщенный конструктор;
 - ❖ Классический вариант фабричного метода.

Структура

```
Продукт p = создатьПродукт();  
p.действовать();
```



```
return new КонкретныйПродуктА();
```



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 нее можно легко вводить новые классы, объекты которых система должна создавать.
- Когда создание новых объектов необходимо делегировать из базового класса классам наследникам

Достоинства

- позволяет сделать код создания объектов более универсальным, не привязываясь к конкретным классам (ConcreteProduct), а оперируя лишь общим интерфейсом (Product);
- позволяет установить связь между параллельными иерархиями классов.

Недостатки

- необходимость создавать наследника Creator для каждого нового типа продукта (ConcreteProduct).

Формальное определение паттерна на языке C#

```
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 }
```

Пример

```
1 class Program
2 {
3     static void Main(string[] args)
4     {
5         Developer dev = new PanelDeveloper("ООО КирпичСтрой");
6         House house2 = dev.Create();
7
8         dev = new WoodDeveloper("Частный застройщик");
9         House house = dev.Create();
10
11         Console.ReadLine();
12     }
13 }
14 // абстрактный класс строительной компании
15 abstract class Developer
16 {
17     public string Name { get; set; }
18
19     public Developer (string n)
20     {
```

```
21         Name = n;
22     }
23     // фабричный метод
24     abstract public House Create();
25 }
26 // строит панельные дома
27 class PanelDeveloper : Developer
28 {
29     public PanelDeveloper(string n) : base(n)
30     { }
31
32     public override House Create()
33     {
34         return new PanelHouse();
35     }
36 }
37 // строит деревянные дома
38 class WoodDeveloper : Developer
39 {
40     public WoodDeveloper(string n) : base(n)
```

```
41     { }
42
43     public override House Create()
44     {
45         return new WoodHouse();
46     }
47 }
48
49 abstract class House
50 { }
51
52 class PanelHouse : House
53 {
54     public PanelHouse()
55     {
56         Console.WriteLine("Панельный дом построен");
57     }
58 }
59 class WoodHouse : House
60 {
61     public WoodHouse()
62     {
63         Console.WriteLine("Деревянный дом построен");
64     }
65 }
```