avito.tech

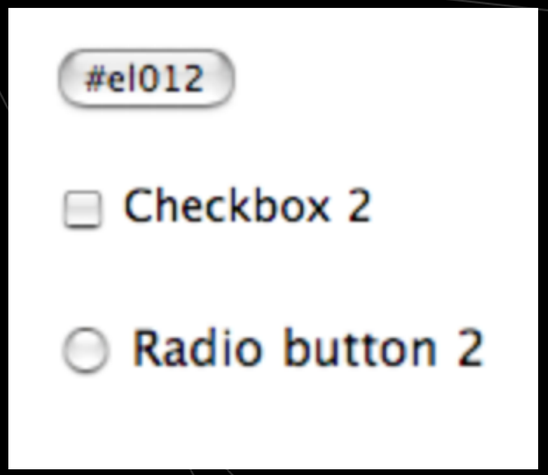
АБСТРАКТНАЯ ФАБРИКА

Афанасьев Юра

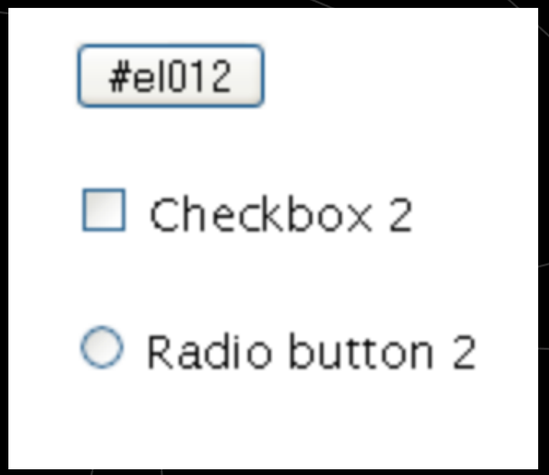
Предоставляет интерфейс
для создания семейств
взаимосвязанных
или взаимозависимых объектов,
не специфицируя
их конкретных классов

Linux

MacOS



Windows

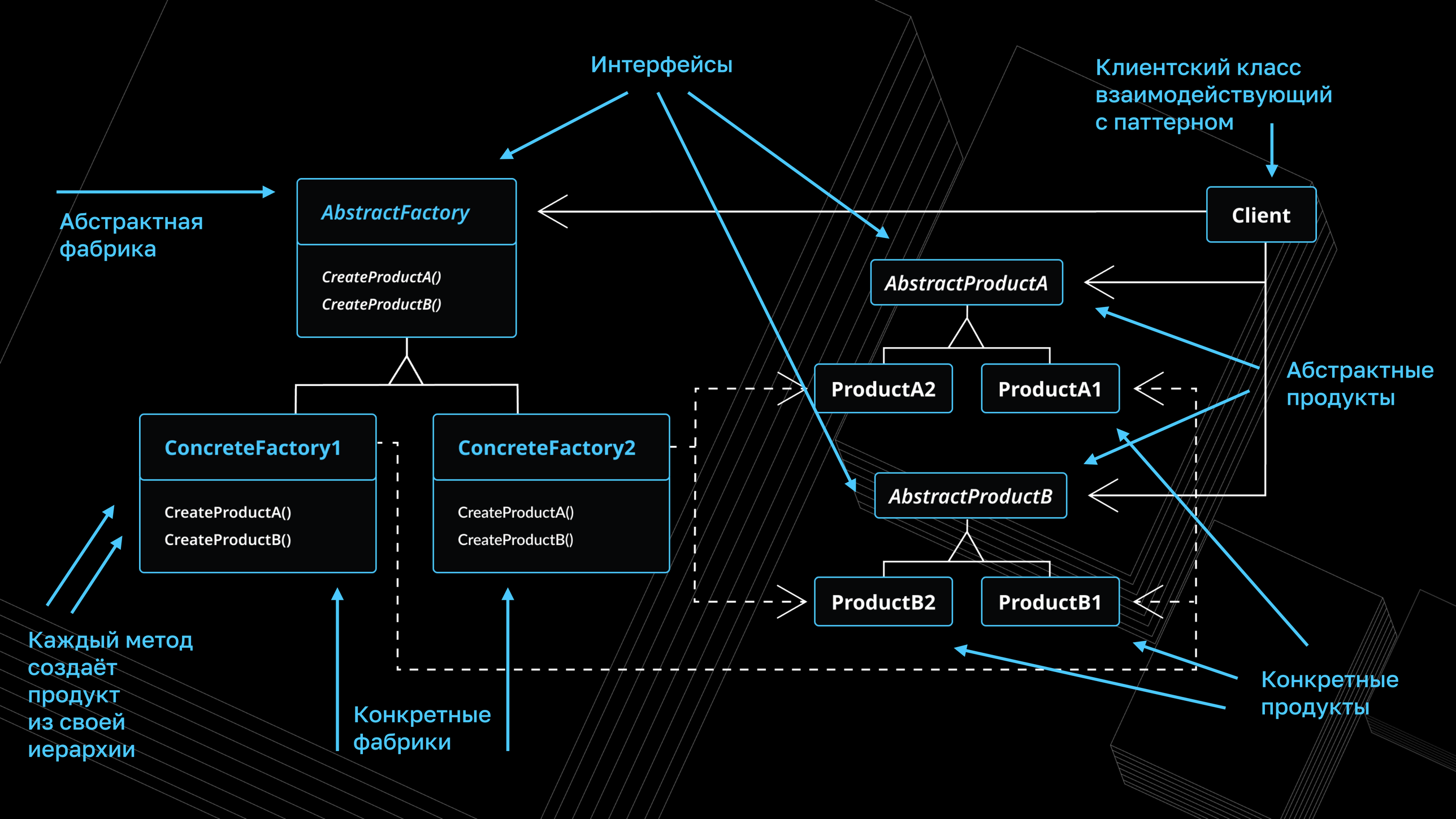


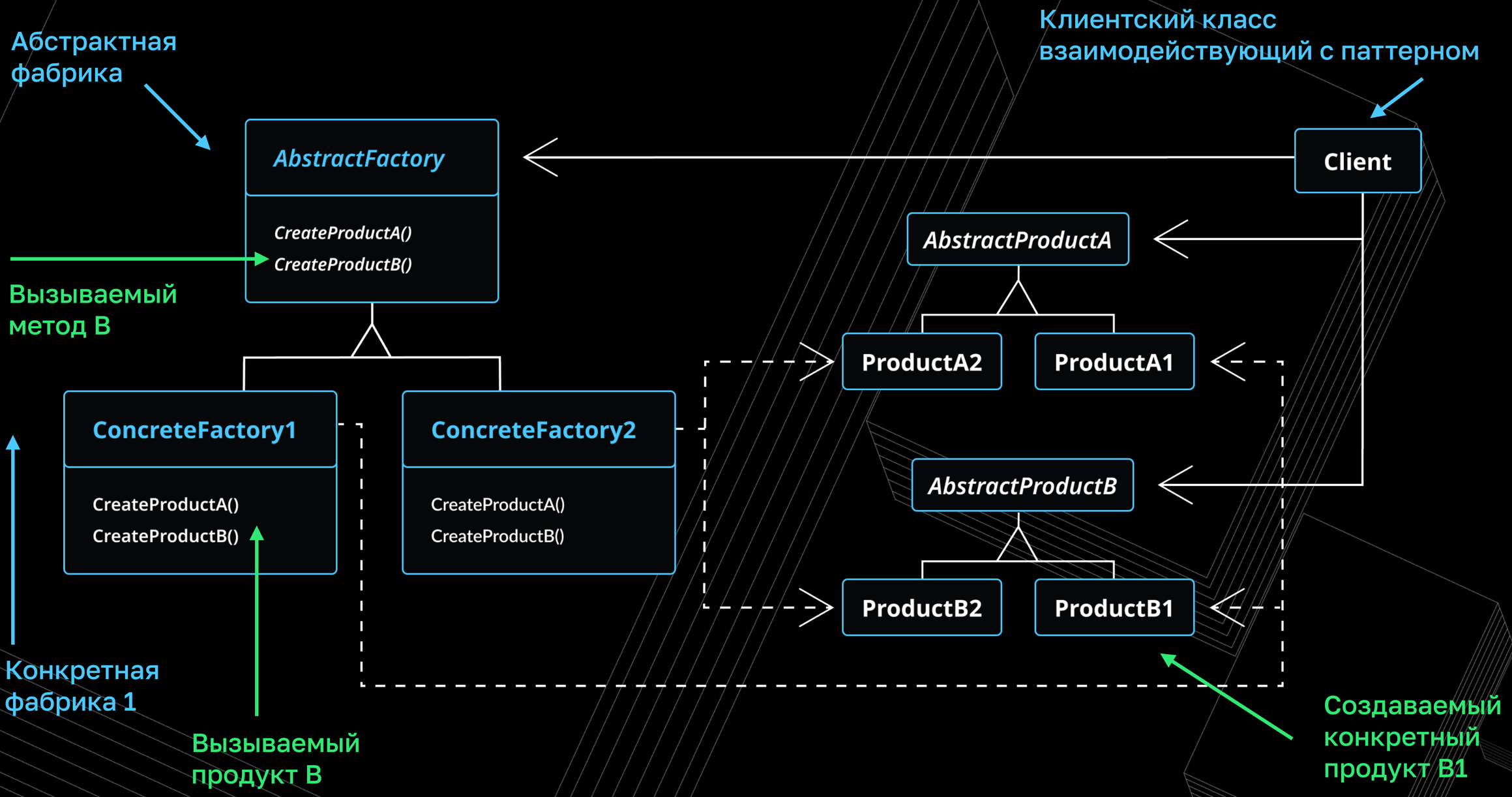
Механика

Предоставляет взаимозаменяемые семейства (группы), целью которых является инкапсуляция логики создания взаимосвязанных объектов

Решаемая задача

- Инкапсулирует логику создания объектов
- Объекты семейства объединены одной задачей
- В одном сеансе приложение работает с одним семейством
- Клиенту известны только интерфейсы возвращаемых классов


Интерфейсы
Клиентский класс взаимодействующий с паттерном
Абстрактная фабрика
AbstractFactory
CreateProductA()
CreateProductB()
Client
AbstractProductA
ProductA2
ProductA1
Абстрактные продукты
ConcreteFactory1
CreateProductA()
CreateProductB()
ConcreteFactory2
CreateProductA()
CreateProductB()
AbstractProductB
ProductB2
ProductB1
Каждый метод создаёт продукт из своей иерархии
Конкретные фабрики
Конкретные продукты


Абстрактная фабрика
Клиентский класс взаимодействующий с паттерном
AbstractFactory
CreateProductA()
CreateProductB()
Client
Вызываемый метод B
AbstractProductA
ProductA2
ProductA1
ConcreteFactory1
CreateProductA()
CreateProductB()
ConcreteFactory2
CreateProductA()
CreateProductB()
AbstractProductB
ProductB2
ProductB1
Конкретная фабрика 1
Вызываемый продукт B
Создаваемый конкретный продукт B1

Абстрактная фабрика

```
interface AbstractFactory
{
    public function createProductA(): AbstractProductA;

    public function createProductB(): AbstractProductB;

    // public function createProductC(): AbstractProductC;
}
```

Описывают продукты

Продуктов может быть много

Конкретные фабрики

Конкретные продукты

Конкретных фабрик может быть много

Абстрактные продукты

```
class ConcreteFactory1 implements AbstractFactory
{
    public function createProductA(): AbstractProductA
    {
        return new ProductA1();
    }

    public function createProductB(): AbstractProductB
    {
        return new ProductB1();
    }
}

class ConcreteFactory2 implements AbstractFactory
{
    public function createProductA(): AbstractProductA
    {
        return new ProductA2();
    }

    public function createProductB(): AbstractProductB
    {
        return new ProductB2();
    }
}

// class ConcreteFactory3 implements AbstractFactory {…}
```

Абстрактные продукты (задаёт методы продуктам)

```php
interface AbstractProductA
{
    public function someMethod(): void;

    // public function anotherMethod(): void;
}


interface AbstractProductB
{
    public function superMethod(): void;

    // public function anotherMethod();
}
```

Методов у продуктов может быть много

Конкретные продукты A*

```php
class ProductA1 implements AbstractProductA
{
    public function someMethod(): void
    {
        echo 'реализация A1';
    }
}

class ProductA2 implements AbstractProductA
{
    public function someMethod(): void
    {
        echo 'реализация A2';
    }
}
```

Абстрактный продукт A

Передаём
Конкретную Фабрику

В зависимости
от фабрики
будем получать
разные
продукты

```php
testAbstractFactory(new ConcreteFactory1());
// testAbstractFactory(new ConcreteFactory2());


function testAbstractFactory(AbstractFactory $factory)
{
    $createProductA = $factory->createProductA();
    $createProductA->someMethod();

    $createProductB = $factory->createProductB();
    $createProductB->superMethod();
}
```

Создание
Конкретных
Продуктов

Вызов
методов
фабрики

Плюсы

- Клиент не привязан к конкретным классам
- Изолируют конкретные продукты (классы)
- Удовлетворяет принципу открытости/закрытости из SOLID и слабой связанности (Low Coupling)
- Легко изменить один набор классов на другой, не производя рефакторинг
- Гарантирует сочетаемость продуктов

Минусы

- Реализация паттерна состоит из большого числа классов
- Добавление нового вида продукта может быть затруднительным

Особенности

- Для каждого семейства в приложении используется лишь одна конкретная фабрика
- Конкретная фабрика выбирается и создаётся в проекте один раз

Создание семейства наборов взаимозаменяемых классов

Совет

- Грамоздкий паттерн
- Для простых случаев используйте Фабричный метод

avito.tech