ГОСУДАРСТВЕННОЕ ОРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ПИ

Факультет КНТ

Индивидуальное работа

Курс: Конструирование программного обеспечения

Выполнил

ст. гр. ПИ-18б

Моргунов А. Г.

Проверил

Чернышова А.В.

Донецк – 2021

СОДЕРЖАНИЕ

[1 Паттерн Абстрактная фабрика (Abstract factory) 3](#_Toc72363683)

[1.1 Краткое теоретическое описание паттерна 3](#_Toc72363684)

[1.2 Представить теоретические сведения по паттерну 3](#_Toc72363685)

[1.3 Целесообразность применения 4](#_Toc72363686)

[1.4 Диаграмма классов и диаграмма последовательностей спроектированного ранее программного продукта 5](#_Toc72363687)

[1.5 Диаграмма последовательностей с использованием новой техники 6](#_Toc72363688)

[1.6 Модифицированная диаграмма классов 7](#_Toc72363689)

[1.7 Вывод об эффективности использования паттерна 7](#_Toc72363690)

[2 Паттерн Посетитель (Visitor) 8](#_Toc72363691)

[2.1 Краткое описание 8](#_Toc72363692)

[2.2 Теоретические сведения о паттерне 8](#_Toc72363693)

# Паттерн Абстрактная фабрика (Abstract factory)

## 1.1 Краткое теоретическое описание паттерна

Паттерн Abstract Factory - предоставляет клиенту интерфейс (набор методов) для создания семейств взаимосвязанных или взаимозависимых объектов-продуктов, при этом скрывает от клиента информацию о конкретных классах создаваемых объектов-продуктов.

## 1.2 Представить теоретические сведения по паттерну

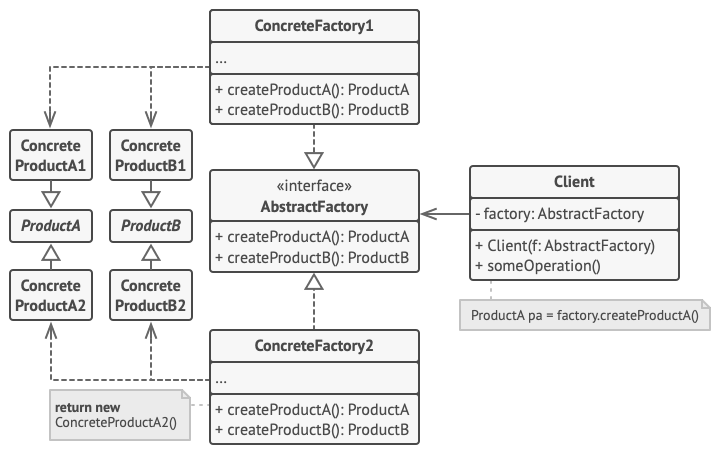


Рисунок 1.1 – Структура паттерна «Абстрактная фабрика»

1. **Абстрактные продукты** объявляют интерфейсы продуктов, которые связаны друг с другом по смыслу, но выполняют разные функции.
2. **Конкретные продукты** — большой набор классов, которые относятся к различным абстрактным продуктам (кресло/столик), но имеют одни и те же вариации (Викторианский/Модерн).
3. **Абстрактная фабрика** объявляет методы создания различных абстрактных продуктов (кресло/столик).
4. **Конкретные фабрики** относятся каждая к своей вариации продуктов (Викторианский/Модерн) и реализуют методы абстрактной фабрики, позволяя создавать все продукты определённой вариации.
5. Несмотря на то, что конкретные фабрики порождают конкретные продукты, сигнатуры их методов должны возвращать соответствующие абстрактные продукты. Это позволит клиентскому коду, использующему фабрику, не привязываться к конкретным классам продуктов. Клиент сможет работать с любыми вариациями продуктов через абстрактные интерфейсы.

## 1.3 Целесообразность применения

Выбранный проект – графический редактор.

При разработке графического редактора удобно использовать паттерн абстрактная фабрика для реализации смены вида. У нас есть вид, отображающий объекты, и вид, отображающий чертежи. При проектировании без паттерна «Абстрактная фабрика» приходится создавать методы для всех объектов для каждого вида, что затрудняет добавление новых фигур и видов, а также усложняет переключение между видами. Паттерн абстрактная фабрика решает эти проблемы, что позволяет облегчить модификацию и поддержку программного продукта.

## 1.4 Диаграмма классов и диаграмма последовательностей спроектированного ранее программного продукта

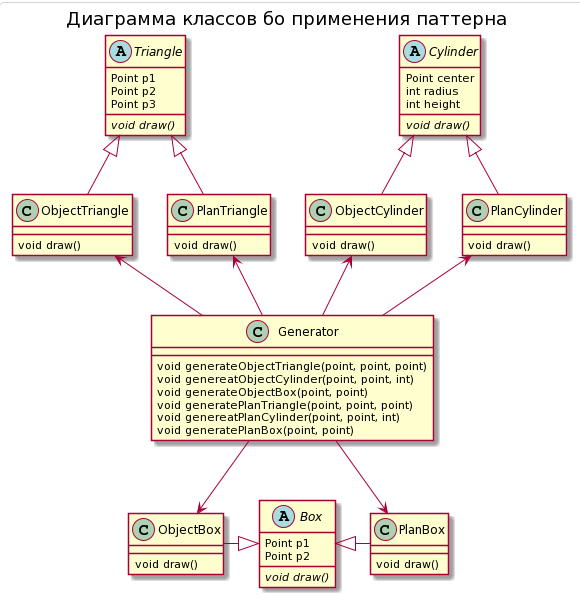


Рисунок 1.2 – Диаграмма классов до применения паттернов



Рисунок 1.3 – Диаграмма последовательности до применения паттерна

### 1.5 Диаграмма последовательностей с использованием новой техники



Рисунок 1.4 – Диаграмма последовательности после применения паттерна

### 1.6 Модифицированная диаграмма классов



Рисунок 1.5 – Диаграмма классов после применения паттерна

### 1.7 Вывод об эффективности использования паттерна

После применения паттерна «Абстрактная фабрика» к программному продукту сильно упростилась его поддержка и модификация. При изменении вида происходит изменение всего 1 поля класса (указателя на генератор) а методы, используемые при генерации, не меняются. Все изменения находятся в классе-наследнике Абстрактного генератора, который и определяет, как будут отрисовываться все фигуры.

# Паттерн Посетитель (Visitor)

### 2.1 Краткое описание

Посетитель — это поведенческий паттерн проектирования, который позволяет добавлять в программу новые операции, не изменяя классы объектов, над которыми эти операции могут выполняться.

### 2.2 Теоретические сведения о паттерне

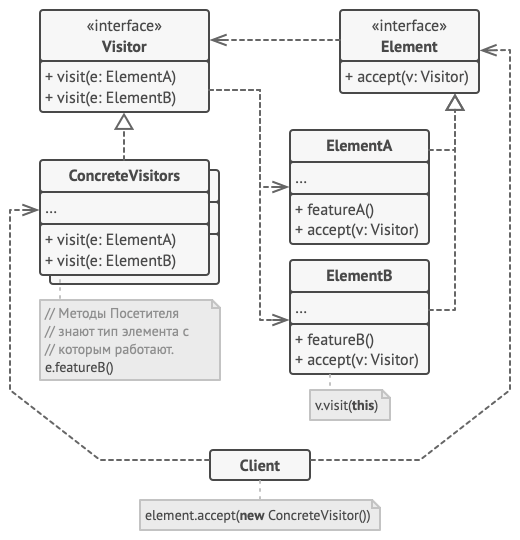


Рисунок 2.1 – Структура паттерна «Посетитель»

1. **Посетитель** описывает общий интерфейс для всех типов посетителей. Он объявляет набор методов, отличающихся типом входящего параметра, которые нужны для запуска операции для всех типов конкретных элементов. В языках, поддерживающих перегрузку методов, эти методы могут иметь одинаковые имена, но типы их параметров должны отличаться.
2. **Конкретные посетители** реализуют какое-то особенное поведение для всех типов элементов, которые можно подать через методы интерфейса посетителя.
3. **Элемент** описывает метод *принятия* посетителя. Этот метод должен иметь единственный параметр, объявленный с типом интерфейса посетителя.
4. **Конкретные элементы** реализуют методы *принятия* посетителя. Цель этого метода — вызвать тот метод посещения, который соответствует типу этого элемента. Так посетитель узнает, с каким именно элементом он работает.
5. **Клиентом** зачастую выступает коллекция или сложный составной объект, например, дерево **Компоновщика**. Зачастую клиент не привязан к конкретным классам элементов, работая с ними через общий интерфейс элементов.