МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук
Кафедра автоматизированных систем управления

Лабораторная работа №1

по дисциплине «Архитектура программных систем»

«Использование порождающих шаблонов проектирования»

Студент АС-21-1 _____ Станиславчук С. М.

(подпись, дата)

Руководитель

Доцент _____ Алексеев В. А.

(подпись, дата)

Липецк 2025

Содержание

2. Цель работы

Изучить шаблоны проектирования, относящиеся к классу порождающих, освоить применение шаблонов этого класса при разработке программных систем с применением объектно-ориентированных языков программирования.

Задание

Реализовать программу для решения задачи согласно варианту с использованием требуемого порождающего шаблона проектирования.

Вариант 2.3

Информационная система предназначена для «сборки» комплектации автомобиля. В зависимости от марки и модели комплектация может включать, например, следующие категории элементов: экстерьер, интерьер, комфорт, безопасность, мультимедиа и т.п. Реализовать «сборку» различных комплектаций для выбранной модели автомобиля и подготовку сравнительной таблицы опций в формате html. При выполнении задания использовать реальный каталог выбранного автопроизводителя.

3. Краткие теоретические сведения

3.1 Описание шаблона проектирования

Паттерн "Строитель" (Builder) — это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Он отделяет процесс конструирования объекта от его представления, что позволяет использовать один и тот же процесс конструирования для создания различных представлений объекта.

Структура паттерна:

1. **Продукт (Product)**: Это конечный объект, который будет создан. Обычно это сложный объект с множеством параметров.

2. **Строитель (Builder)**: Интерфейс, который определяет шаги для создания продукта.

3. **Конкретный строитель (Concrete Builder)**: Реализация интерфейса `Builder`. Определяет конкретные шаги для создания продукта.

4. **Директор (Director)**: Управляет процессом создания объекта. Он использует `Builder` для пошагового создания продукта.

5. **Клиент (Client)**: Создает объект `Director` и передает ему конкретный `Builder`.

3.2 Классовая диаграмма для шаблона проектирования в нотации UML.

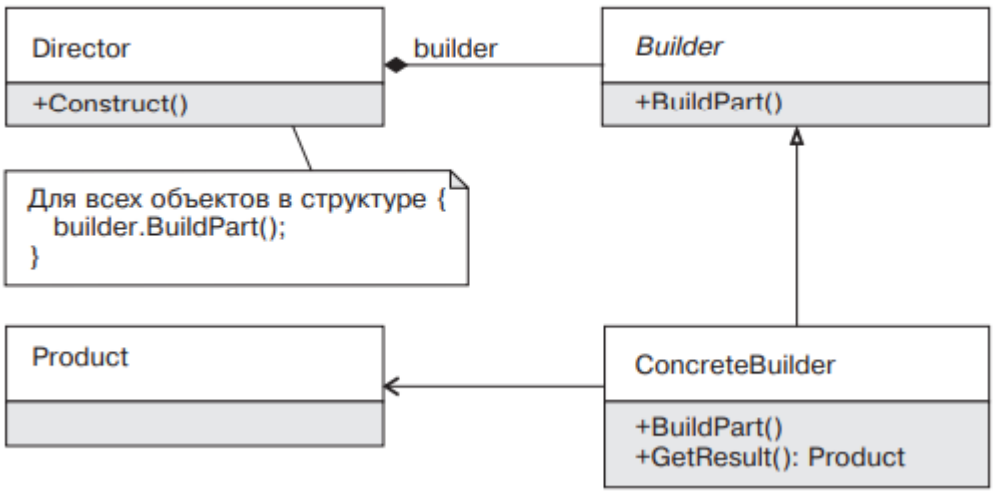

Рисунок 1 — UML схема паттерна строитель

4. Описание условий и хода работы
4.1 Используемые аппаратные и программные средства.

Аппаратные средства:

- Процессор: AMD Ryzen 7 4800h.

- Оперативная память: 16 ГБ DDR4 x3200.

- Жесткий диск: SAMSUNG SSD 512 ГБ.

Программные средства:

- Текстовый редактор: Vim.
- Компилятор, линкер: GCC.
- Система контроля версий: Git.

4.2 Интерпретация шаблона проектирования для решения задачи.

Класс `Car`. Представляет продукт, который нужно создать. Содержит поля для хранения информации о цене, экстерьере, интерьере, комфорте, безопасности и мультимедиа.

Абстрактный класс `CarBuilder`. Определяет интерфейс для пошагового создания объекта `Car`. Включает методы для добавления каждой категории опций (`buildPrice`, `buildExterior`, `buildInterior` и т.д.).

Конкретные строители. `BMW3SeriesLuxuryBuilder`: Реализует методы для создания комплектации BMW 3 320D Luxury. `BMW3SeriesSportBuilder`: Реализует методы для создания комплектации BMW 3 320D Sport.

Класс `CarDirector`. Управляет процессом сборки. Использует методы строителя для пошагового создания объекта `Car`.

4.3 Классовая диаграмма для задачи в нотации UML

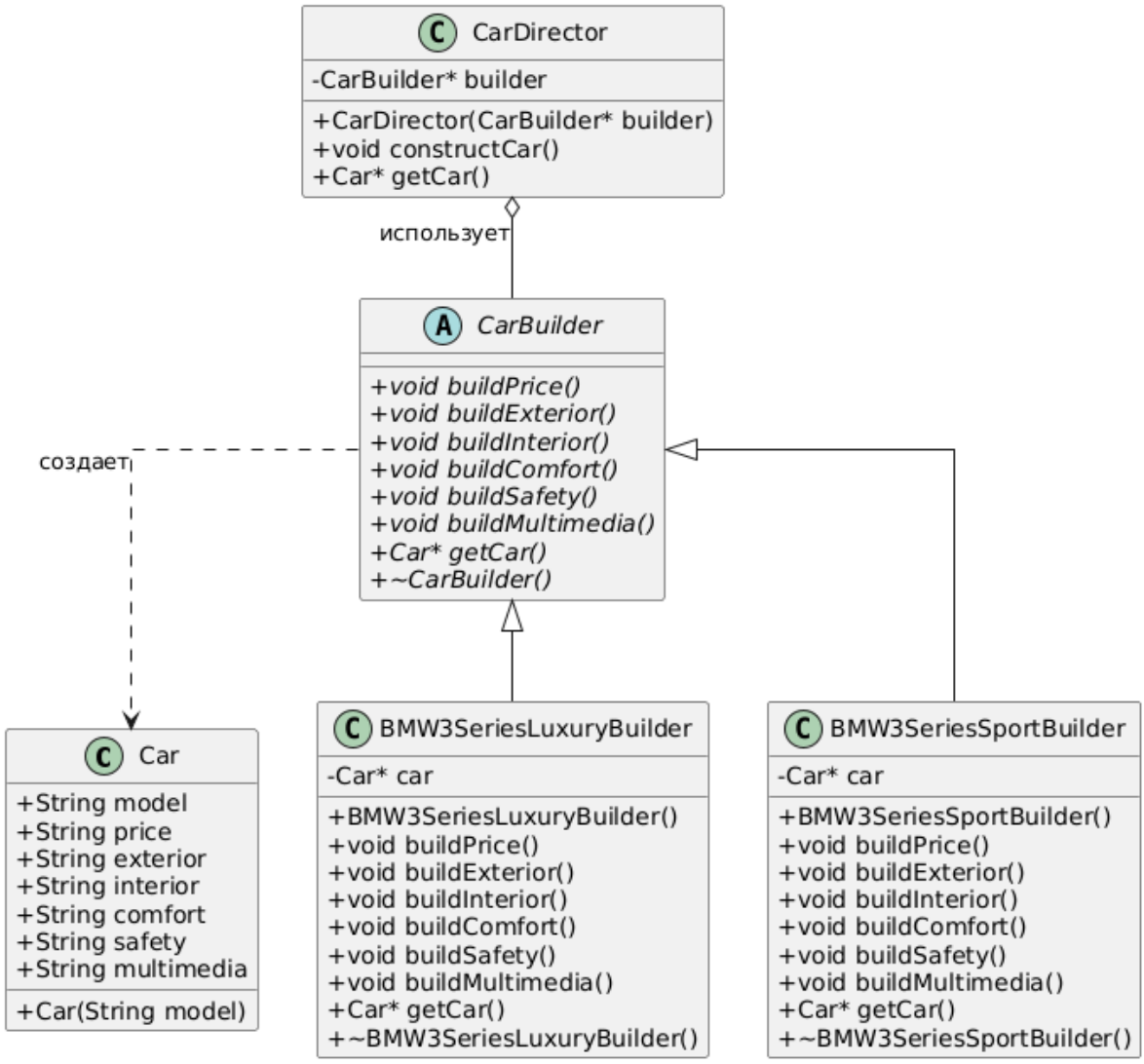

Рисунок 2 — Классовая диаграмма

5.1 Текст программы

Класс директора:

```
class CarDirector {
  private:
    CarBuilder* builder;

  public:
    CarDirector(CarBuilder* b) : builder(b) {}
```

```
    void constructCar() {
        builder->buildPrice();
        builder->buildExterior();
        builder->buildInterior();
        builder->buildComfort();
        builder->buildSafety();
        builder->buildMultimedia();
    }
};
```

Класс строителя:

```
#include "car.h"
class CarBuilder {
  public:
    virtual void buildPrice() = 0;
    virtual void buildExterior() = 0;
    virtual void buildInterior() = 0;
    virtual void buildComfort() = 0;
    virtual void buildSafety() = 0;
    virtual void buildMultimedia() = 0;
    virtual Car* getCar() = 0;
    virtual ~CarBuilder() {}
};
```

Класс комплектации автомобиля:

```
class Car {
  public:
    std::string price;
    std::string model;
    std::string exterior;
    std::string interior;
    std::string comfort;
    std::string safety;
    std::string multimedia;

    Car(const std::string& model);
};
```

Класс строителя комплектации «Luxury»:

```
class BMW3SeriesLuxuryBuilder : public CarBuilder {
  public:
    BMW3SeriesLuxuryBuilder();
    void buildPrice() override;
    void buildExterior() override;
    void buildInterior() override;
    void buildComfort() override;
    void buildSafety() override;
    void buildMultimedia() override;
    Car* getCar() override;
    ~BMW3SeriesLuxuryBuilder();

  private:
    Car* car;
};
```

Класс строителя комплектации «Sport»:

```
class BMW3SeriesSportBuilder : public CarBuilder {
  public:
    BMW3SeriesSportBuilder();
    void buildPrice() override;
    void buildExterior() override;
    void buildInterior() override;
    void buildComfort() override;
    void buildSafety() override;
    void buildMultimedia() override;
    Car* getCar() override;
    ~BMW3SeriesSportBuilder();

  private:
    Car* car;
};
```

6. Анализ результатов работы.

В результате выполнения программы создается html-файл, содержащий таблицу сравнения двух комплектаций автомобиля BMW 3 320d:

<html><head><meta charset="UTF-8"><title>Сравнение комплектаций</title></head><body><h1>Сравнение комплектаций автомобилей</h1><table border="1" cellspacing="0" cellpadding="5"><tr><th>Модель</th><th>Стоимость</th><th>Экстерьер</th><th>Интерьер</th><th>Комфорт</th><th>Безопасность</th><th>Мультимедиа</th></tr><tr><td>BMW 3 Series Sport</td><td>$47,361.41</td><td>M Sport Package, Adaptive LED headlights</td><td>Sport seats with Vernasca leather</td><td>Automatic climate control, Panoramic sunroof</td><td>Adaptive cruise control, Lane departure warning</td><td>iDrive 7.0, 10.25-inch display, Harman Kardon sound system</td></tr><tr><td>BMW 3 Series Luxury</td><td>$56,223.16</td><td>Luxury Exterior Package with Xenon headlights</td><td>Premium wood trim, Nappa leather seats</td><td>Multi-zone climate control, Ambient lighting</td><td>Blind spot detection, Parking sensors</td><td>iDrive 8.0, 12.3-inch display, Bowers & Wilkins sound system</td></tr></table></body></html>

7. Выводы о достоинствах и недостатках используемого шаблона проектирования.

Достоинства:

Возможность контролировать процесс создания сложного продукта
Возможность получения разных представлений некоторых данных
Позволяет использовать один и тот же код для создания различных продуктов
Изолирует сложный код сборки продукта от его основной бизнес-логики

Недостатки:
ConcreteBuilder и создаваемый им продукт жестко связаны между собой, поэтому при внесеннии изменений в класс продукта скорее всего придется соотвествующим образом изменять и класс ConcreteBuilder.
Усложняет код программы за счёт дополнительных классов.

Вывод: в ходе выполненной работы реализовал паттерн программирования «Строитель»