ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| старший преподаватель | | С.Ю.Гуков |
|---|---|---|
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

# ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

## Паттерны проектирования

по курсу: Технологии программирования

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ гр. № 4321 | | Г.В. Буренков |
|---|---|---|
| | подпись, дата | инициалы, фамилия |

Санкт-Петербург 2025

**СОДЕРЖАНИЕ**

## 1 Цель работы

Целью данной лабораторной работы является изучение и применение паттернов проектирования в процессе разработки программного обеспечения. В рамках работы необходимо ознакомиться с назначением и принципами работы различных паттернов, а также научиться применять их на практике в контексте реального программного проекта.

Основное внимание уделяется проектированию архитектуры программных решений с использованием двух паттернов различных категорий, что позволяет улучшить масштабируемость, гибкость и поддержку кода. В результате выполнения работы студент приобретает практические навыки использования паттернов проектирования, анализа архитектурных решений и создания UML-диаграмм, что способствует развитию компетенций в области программирования и проектирования программных систем.

## 2 Задание

Разработать программное обеспечение, в котором используются паттерны «Абстрактная фабрика» (Abstract Factory) и «Адаптер» (Adapter) в едином контексте. В рамках работы необходимо спроектировать систему, где «Абстрактная фабрика» будет отвечать за создание групп связанных объектов без указания их конкретных классов, а «Адаптер» обеспечит совместимость интерфейсов между различными компонентами системы.

Проект должен быть реализован с соблюдением принципов объектно-ориентированного программирования и SOLID. В качестве наглядной демонстрации работы необходимо разработать пользовательский интерфейс, который позволит взаимодействовать с созданной системой. Также требуется подготовить UML-диаграмму, отражающую архитектуру решения, и оформить отчет с описанием процесса разработки, исходного кода и анализа примененных паттернов.

## 3 Ход работы и назначение используемых технологий

В ходе выполнения работы была разработана программная система, реализующая паттерны «Абстрактная фабрика» (Abstract Factory) и «Адаптер» (Adapter). В качестве предметной области выбрана модель производства автомобилей, где «Абстрактная фабрика» отвечает за создание различных типов автомобилей и их комплектующих, а «Адаптер» обеспечивает совместимость между различными интерфейсами деталей и систем управления банковским счетом пользователей.

Проект разработан с применением принципов объектно-ориентированного программирования, включая инкапсуляцию, наследование и полиморфизм. Использование SOLID-принципов позволило создать гибкую и расширяемую архитектуру. UML-диаграмма классов отражает основные связи между компонентами системы. Для реализации использован TypeScript, что обеспечивает строгую типизацию и упрощает поддержку кода.

Взаимодействие с пользователем реализовано в виде консольного интерфейса, демонстрирующего процесс создания трех видов доступа пользователя: Customer, Editor, Admin. Разные уровни пользователей дают различные права для управления. Для реализации системы пользователей был создан адаптер платежных систем на выбор (Webmoney, F2Wallet). Для связи был реализован файл view.ts. Его главная задача соединение взаимодействия системы и пользователя в одно целое. В файлах users.ts, interfaces.ts, factory.ts код разбит на модули для улучшение ориентирования и читаемости кода. На рисунке 1 изображена схема UML приложения.

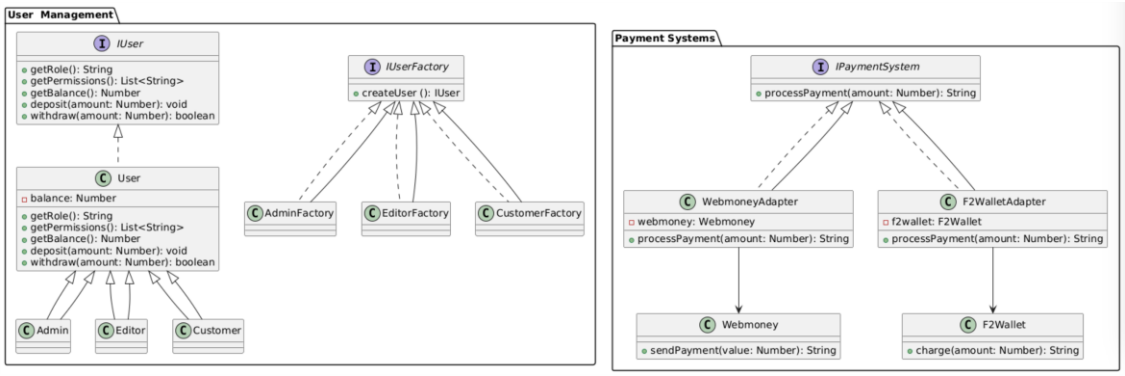

Рисунок 1 – Результат работы приложения

## 4 Аргументация использования паттернов

Паттерн фабрика помогает создавать объекты разных типов, не указывая конкретный класс каждого объекта. В данном проекте он используется для управления пользователями, позволяя легко создавать администраторов, редакторов и клиентов. Благодаря этому подходу логика создания пользователей сосредоточена в одном месте, что делает код более понятным. Если в будущем потребуется добавить новый тип пользователя, это можно сделать, не изменяя уже существующий код. Кроме того, такой способ удобен для тестирования, так как фабрики можно заменять на тестовые версии.

Паттерн адаптер позволяет сделать так, чтобы системы с разными интерфейсами могли работать вместе. В данном проекте он используется для интеграции нескольких платежных систем, которые изначально имеют разные способы обработки платежей. Адаптеры приводят их к общему формату, что делает код более удобным и гибким. Если понадобится добавить новую платежную систему, достаточно создать для нее адаптер, не изменяя остальную программу. Такой подход облегчает поддержку и тестирование кода, так как он не зависит от конкретных платежных сервисов.

## 5 Исходный код

Полную структуру кода можно увидеть в репозитории github по ссылке: https://github.com/skv0r/user-factory

## 6 Результат работы приложения

Программа успешно выполняет взаимодействие с пользователем через CLI с помощью выбранных паттернов. На рисунке 2 изображен результат.

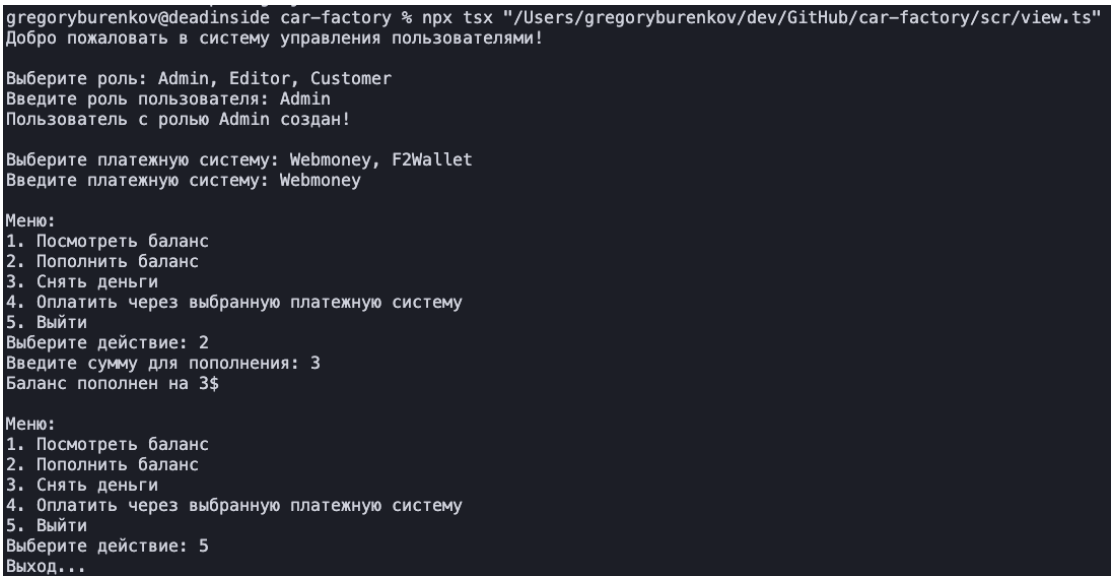


Рисунок 2– Результат работы приложения

## 7 Вывод

В ходе выполнения работы была разработана программная система, использующая паттерны проектирования фабрика и адаптер. В результате удалось создать гибкую и расширяемую архитектуру, которая позволяет управлять процессом перевода денег, создавая различные модели банковских систем, а также адаптировать права для использования.

Использование фабричного паттерна упростило процесс создания систем, сделав код более структурированным и удобным для дальнейшего расширения. Применение адаптера позволило интегрировать детали с разными интерфейсами, обеспечивая их совместимость без внесения изменений в основной код. Благодаря этому система получилась более модульной, гибкой и удобной для тестирования.

Работа продемонстрировала, как использование паттернов проектирования помогает решать задачи масштабируемости, повторного использования кода и поддержки в разработке программных систем. Полученные результаты и разработанные подходы могут быть применены в более сложных проектах, требующих продуманной архитектуры и эффективного управления зависимостями между компонентами.