**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Информационных технологий |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и технологии |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Лабораторный практикум № 6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **по дисциплине** |  | Программирование на IOS | | |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | **Уткин Иван Сергеевич** |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | **группа ДБИ-301рки** |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев Иван Валерьевич |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2024 г.**

**Цель лабораторного практикума:** Ознакомиться с понятием Swift шаблонов проектирования. Порождающих паттернов. Структурных паттернов. Поведенческих паттернов. На базе сетевого слоя и базы данных, создать фабрику моделей. Привести примеры кода.

**Шаблоны проектирования в Swift**

Я ознакомился с сайтом https://refactoringguru.cn/ru/design-patterns/swift содержимое которого гласит, что:

Шаблоны проектирования (design patterns) представляют собой обобщенные решения распространенных проблем, возникающих при проектировании программного обеспечения. В языке Swift, как и в других языках программирования, можно использовать разнообразные паттерны для повышения качества и гибкости кода. Основные виды шаблонов проектирования делятся на три категории: порождающие (creational), структурные (structural) и поведенческие (behavioral) паттерны.

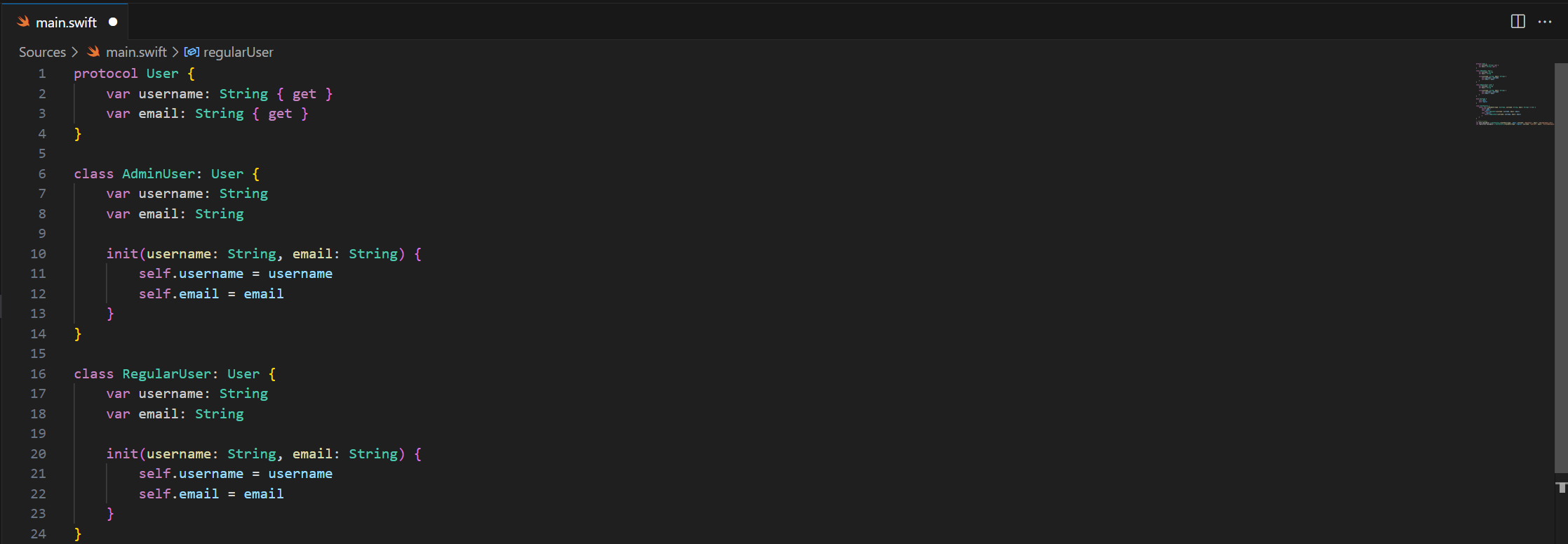


Рисунок 1 Главное меню сайта-источника информации

**Порождающие паттерны**

Порождающие паттерны касаются создания объектов. Они помогают создавать экземпляры классов или структур, обеспечивая гибкость и использование различных подходов к созданию объектов. Рассмотрим несколько основных порождающих паттернов:

1. Фабрика (Factory Method): позволяет создавать объекты без указания конкретного класса создаваемого объекта.





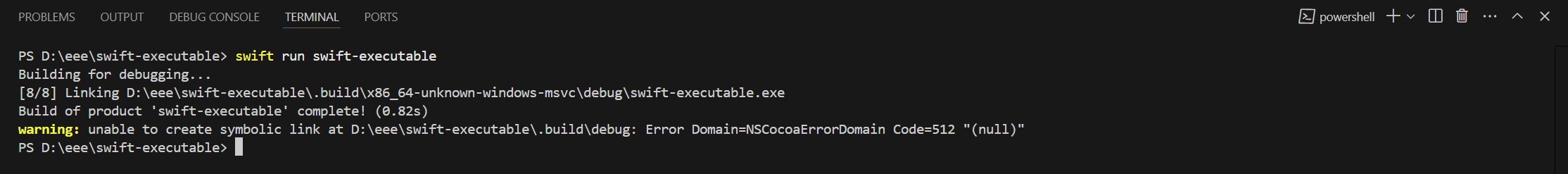


Рисунок 2 Пример кода фабрики для создания моделей данных

1. Паттерн Singleton гарантирует, что у класса есть только один экземпляр, и предоставляет глобальную точку доступа к нему. Этот паттерн часто используется для управления общими ресурсами, такими как соединения с базой данных или конфигурационные настройки.

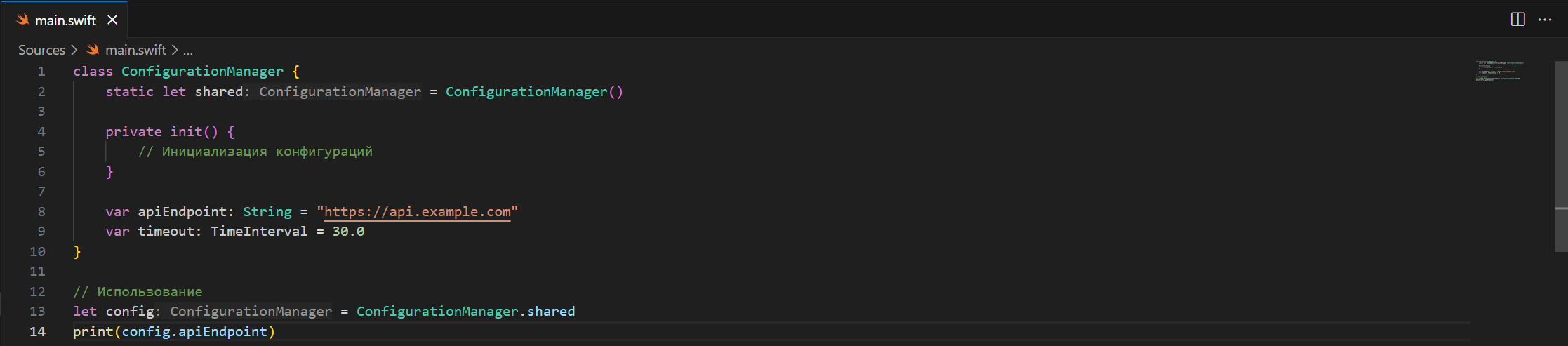


Рисунок 3 Пример реализации Singleton

1. Паттерн Builder используется для конструкций сложных объектов пошагово. Это позволяет создавать разные представления объекта с одинаковым процессом построения.

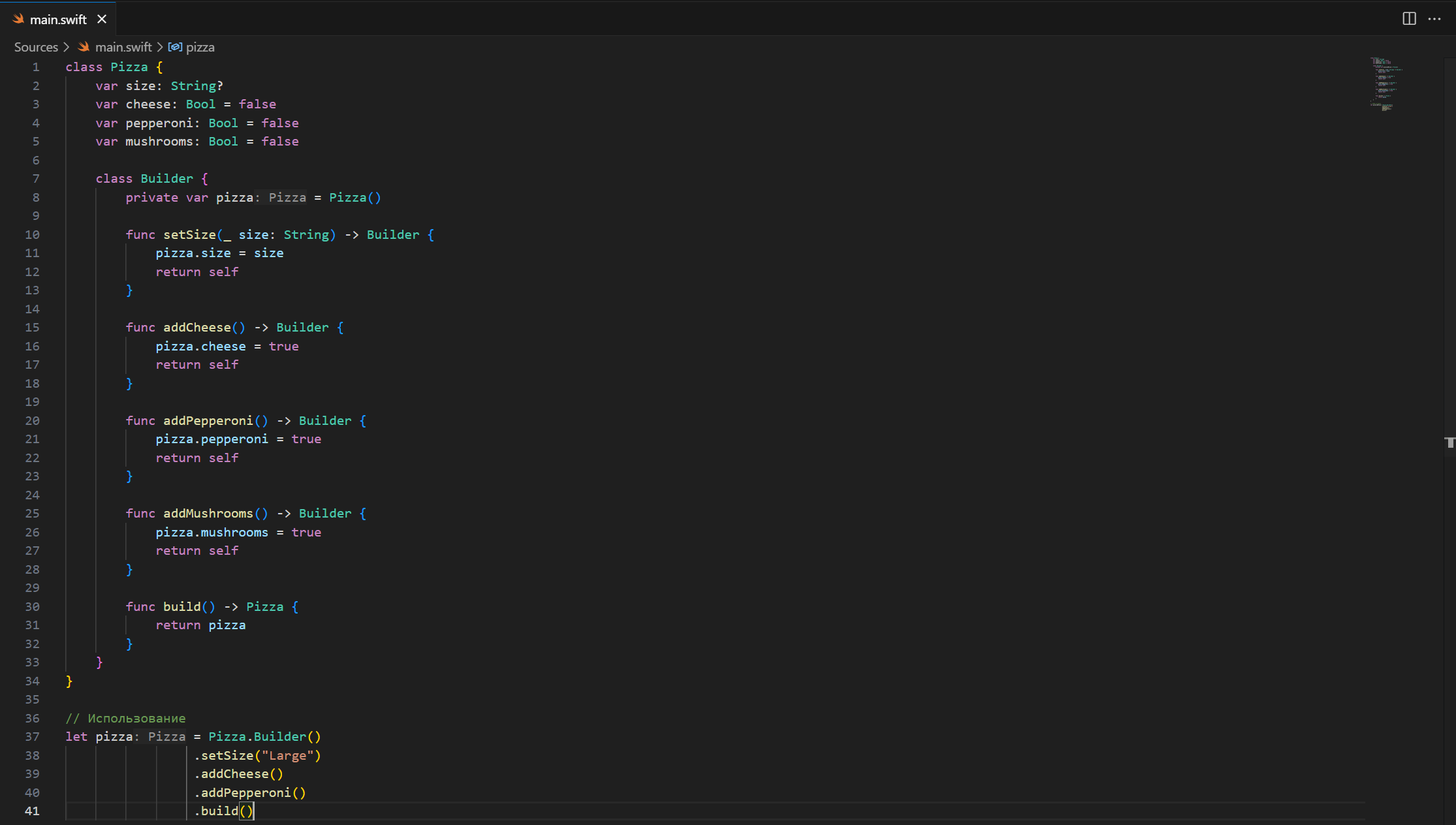


Рисунок 4 Пример реализации Builder

1. Паттерн Prototype позволяет создавать новые объекты, копируя существующие, что удобно, когда создание нового экземпляра требует затратных операций или сложных настроек.

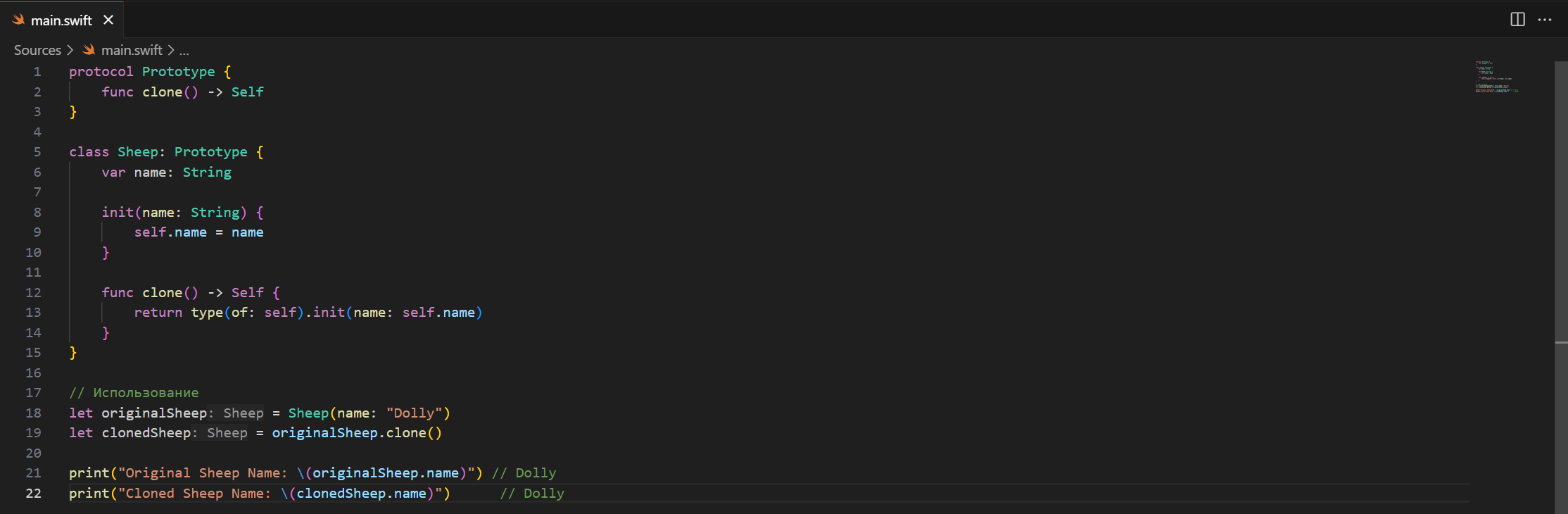
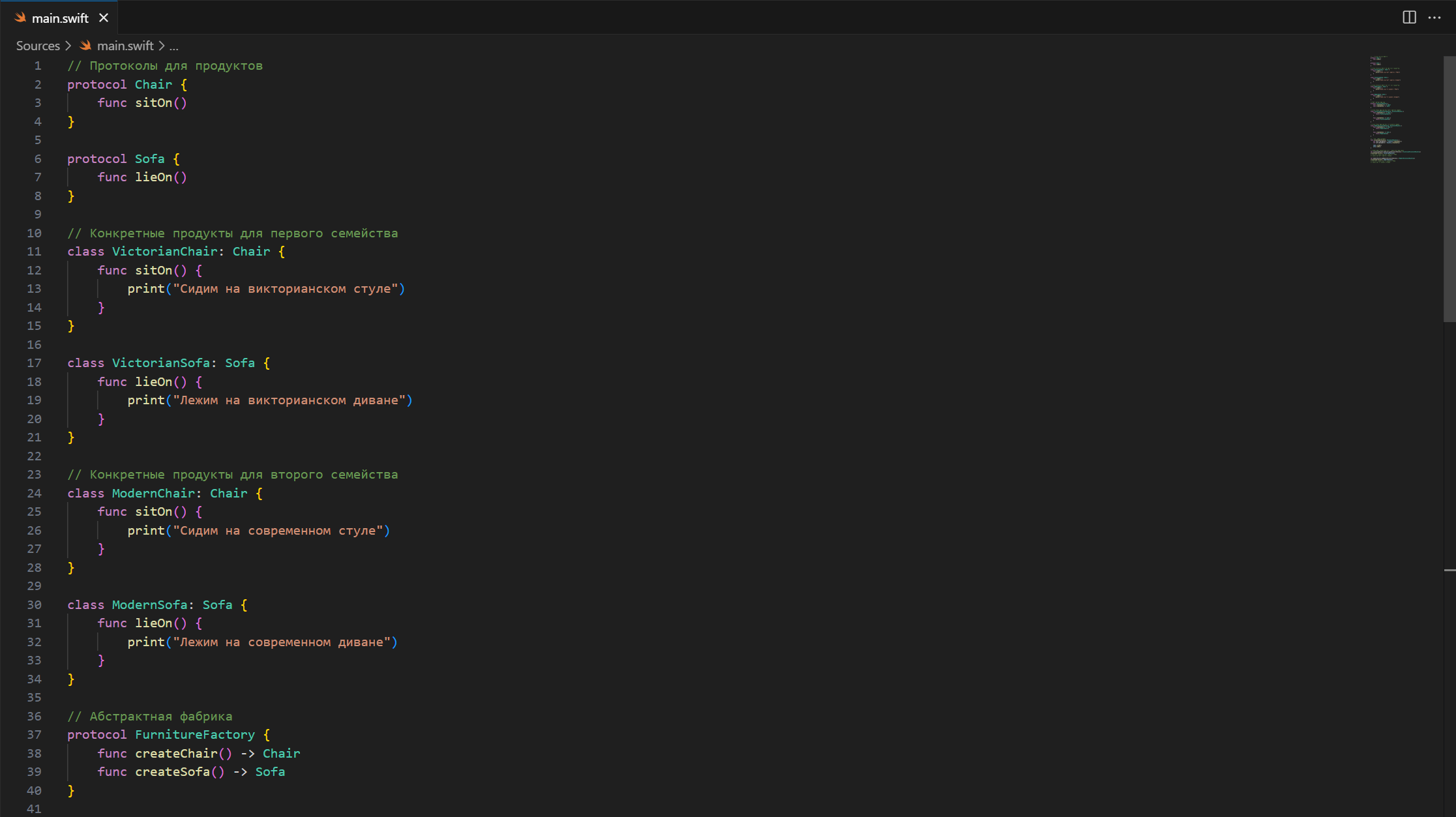


Рисунок 5 Пример реализации Prototype

1. Паттерн Abstract Factory предоставляет интерфейс для создания семейств связанных или зависимых объектов, не указывая их конкретные классы. Этот паттерн полезен, когда система должна быть независимой от способа создания, композиции и представления своих объектов.





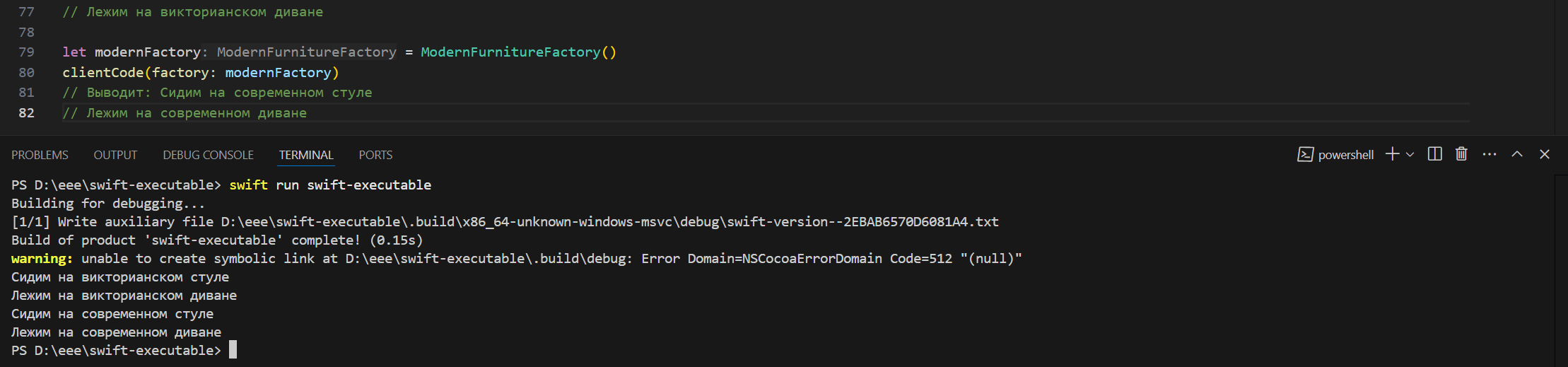
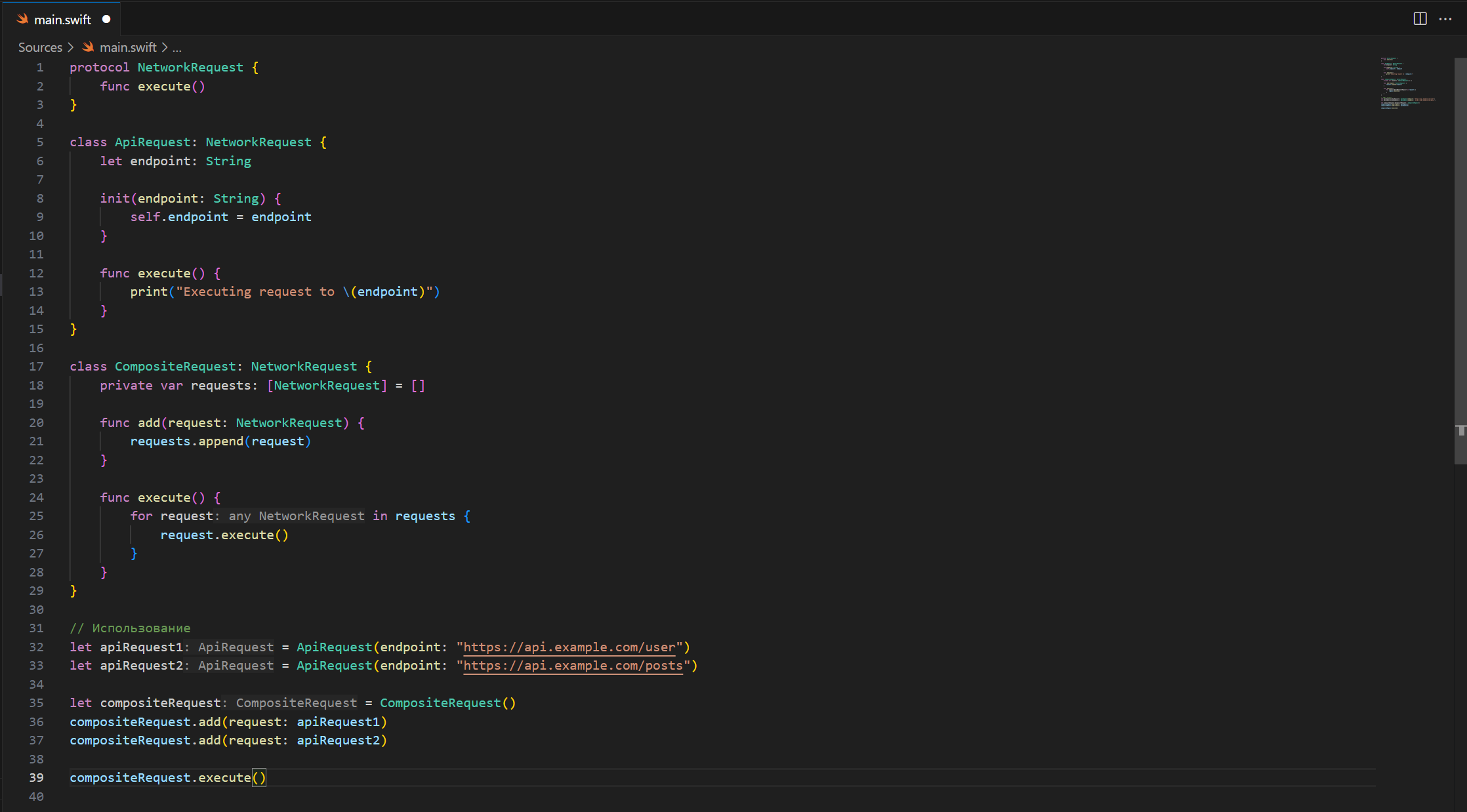


Рисунок 6 Пример реализации Abstract Factory

**Структурные паттерны**

Структурные паттерны описывают, как классы и объекты могут объединяться для формирования больших структур. Они помогают обеспечить гибкость в проектировании. Основные структурные паттерны включают:

1. Композит (Composite): позволяет объединять объекты в древовидные структуры для представления иерархий.



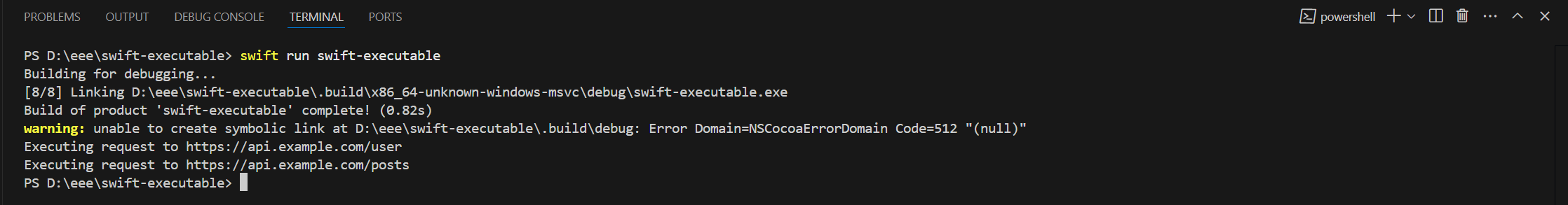
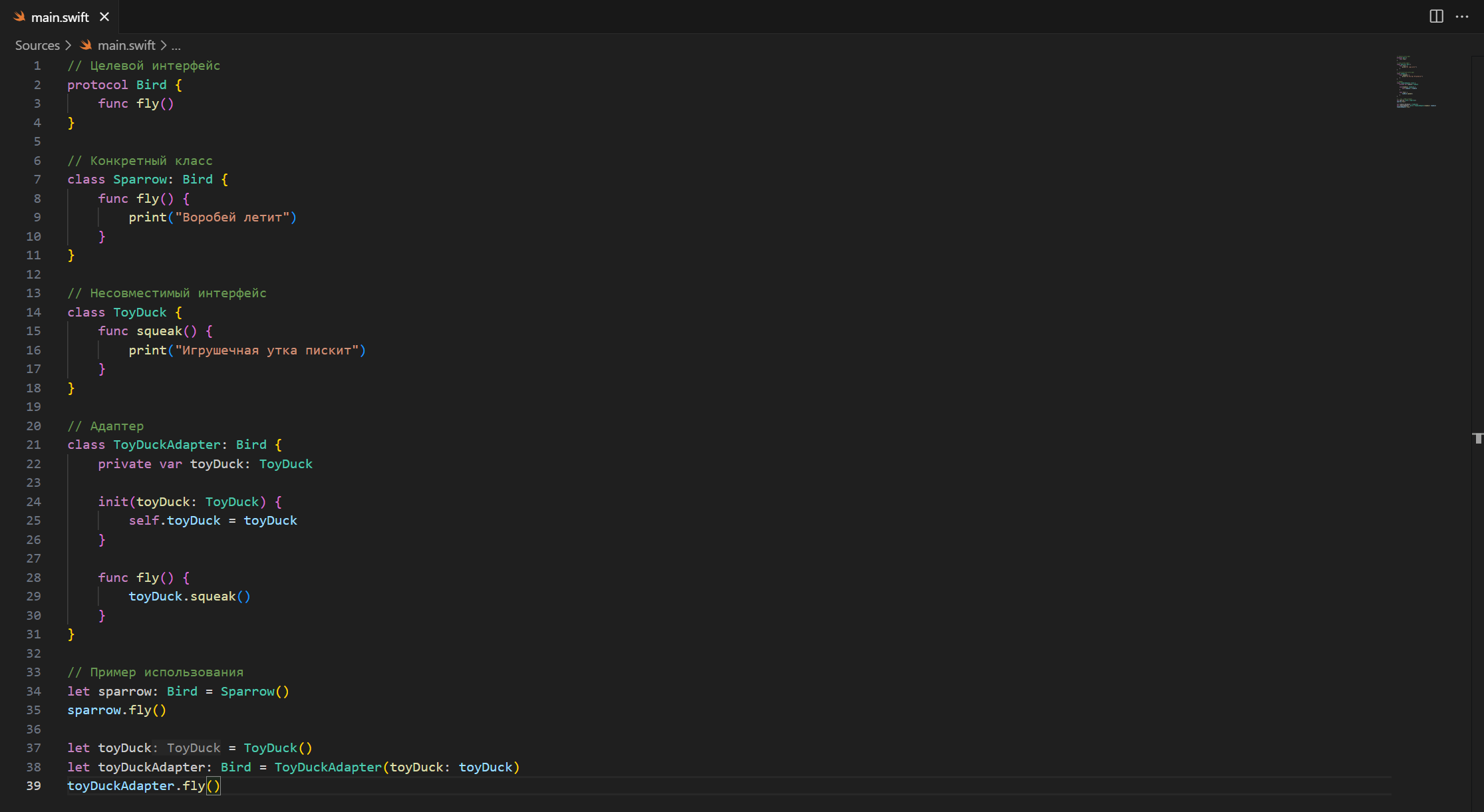


Рисунок 7 Пример композита для создания сетевых запросов

1. Паттерн Adapter позволяет объектам с несовместимыми интерфейсами работать вместе. Это достигается путем создания обертки, которая преобразует интерфейс одного класса в интерфейс, ожидаемый другим классом.



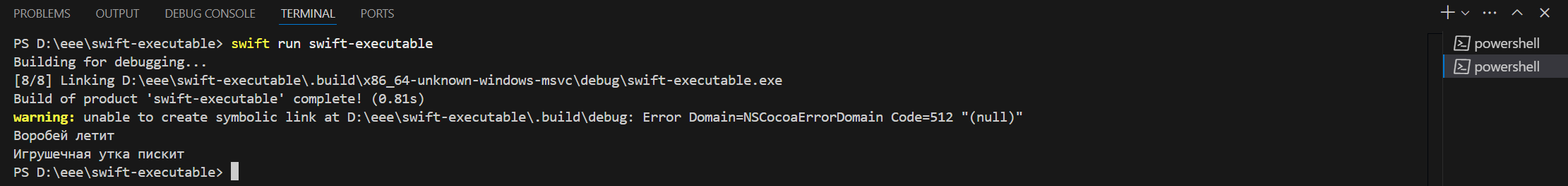
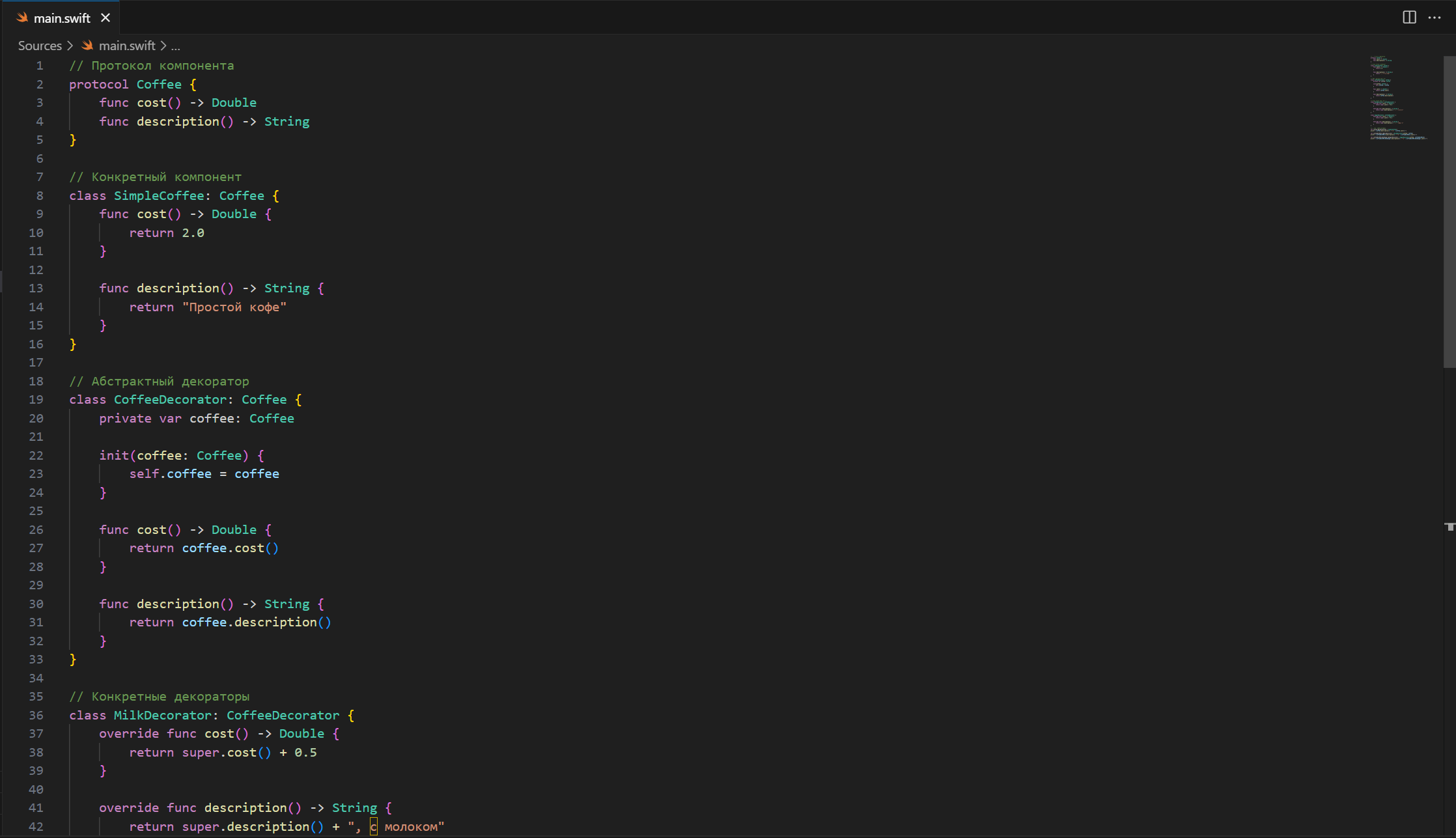


Рисунок 8 Пример реализации Adapter

1. Паттерн "Декоратор" (Decorator) позволяет динамически добавлять дополнительные обязанности объекту, оборачивая его в класс-декоратор.



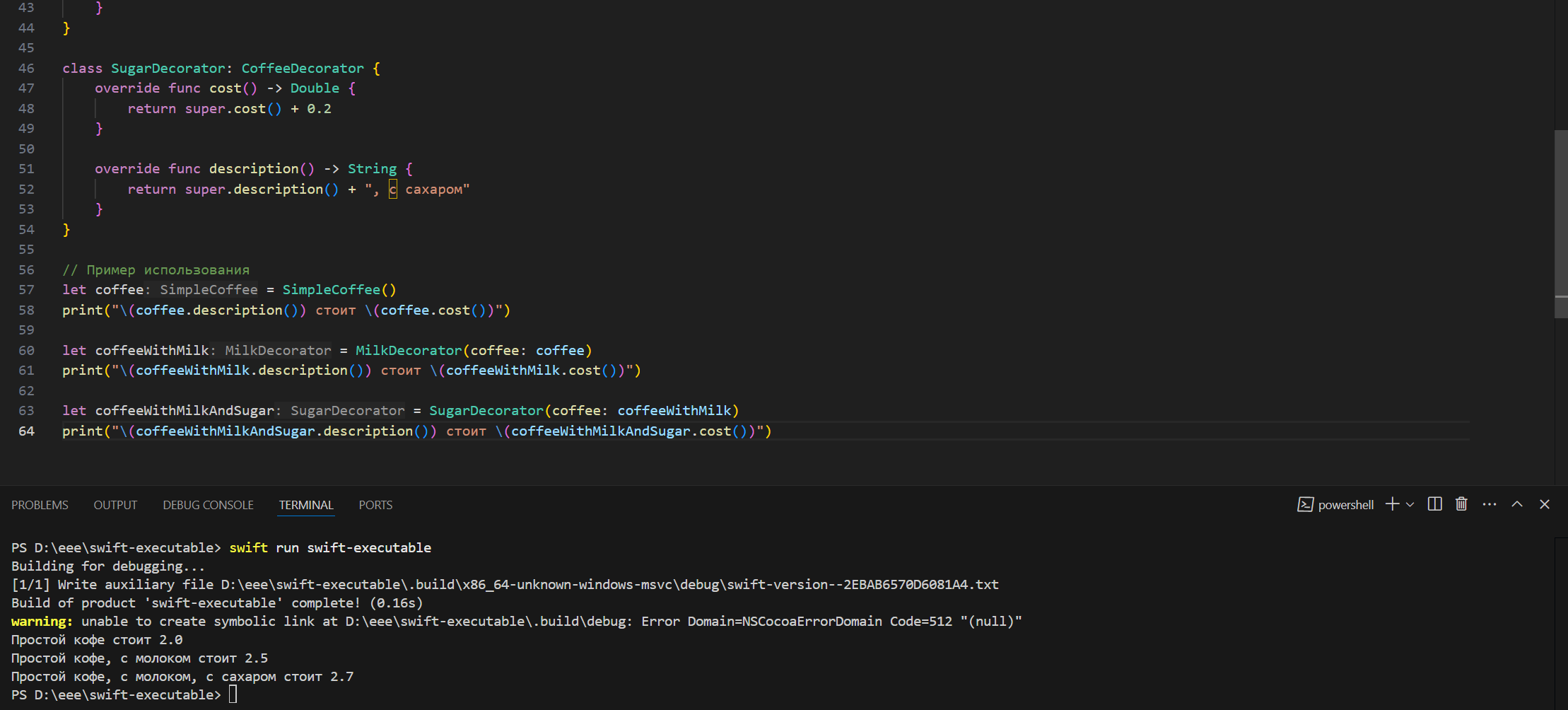
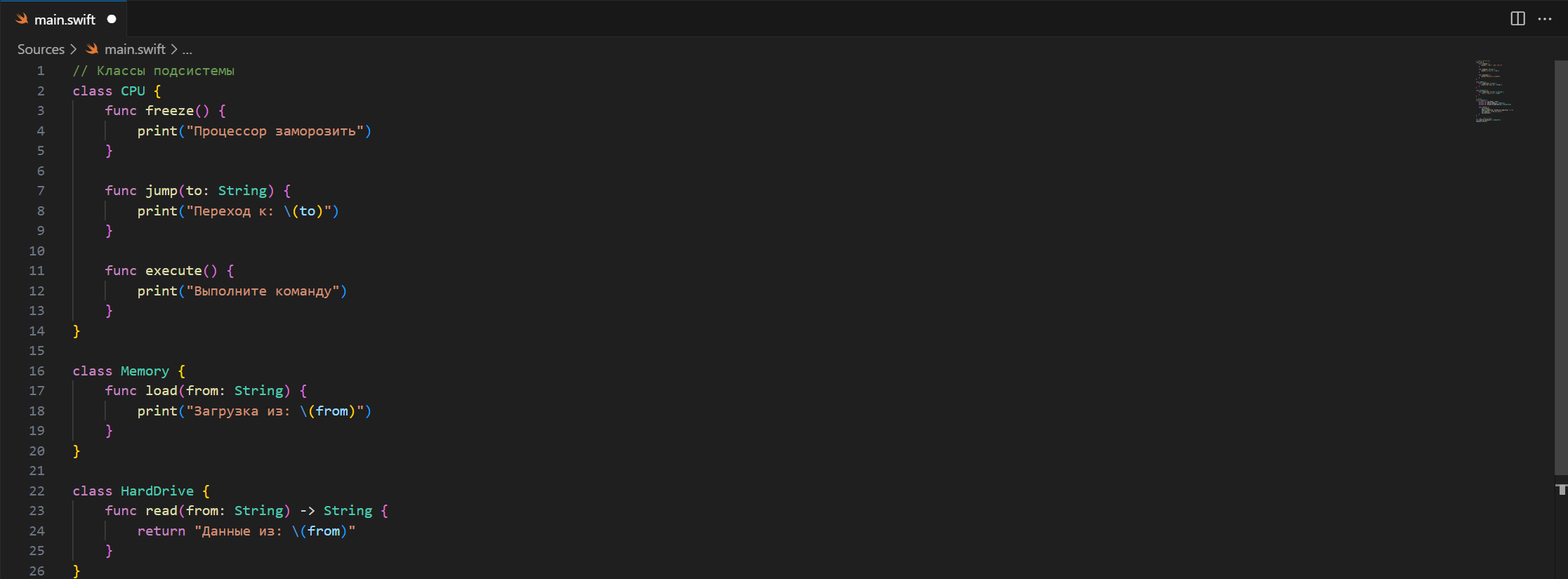


Рисунок 9 Пример реализации Decorator

1. Паттерн "Фасад" (Facade) предоставляет упрощённый интерфейс к более сложной системе.



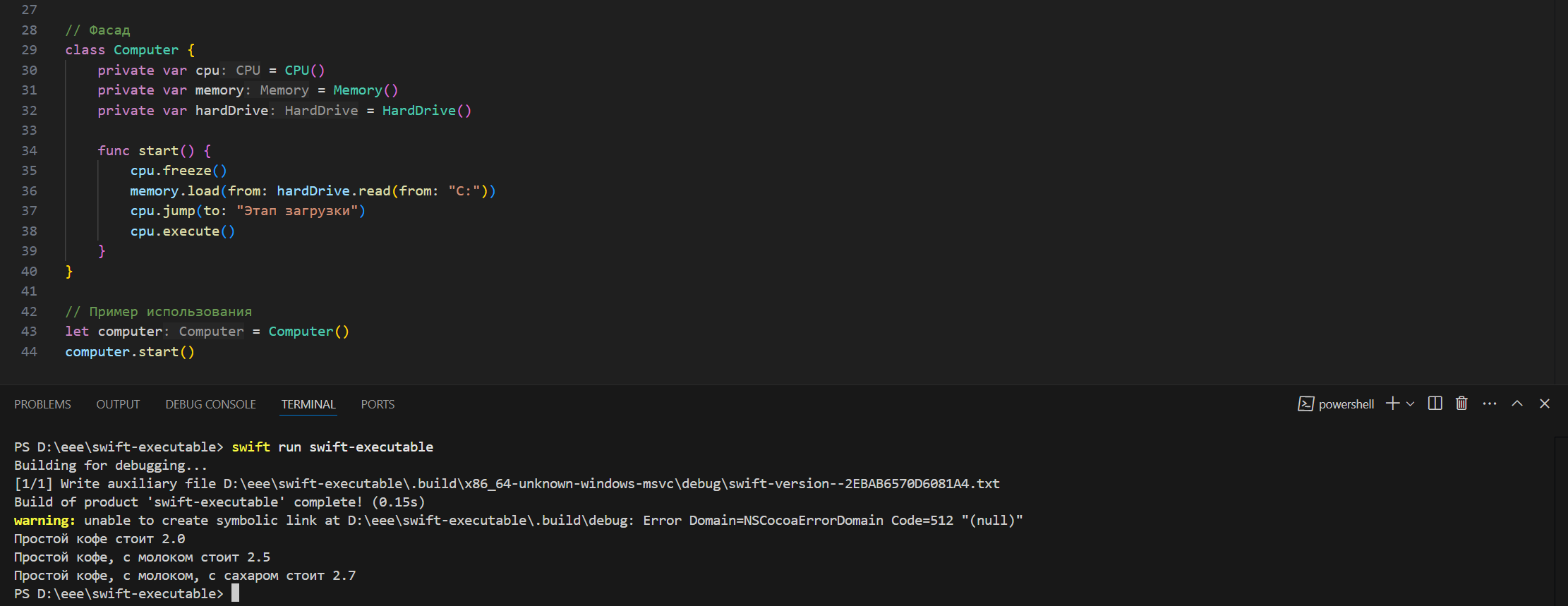
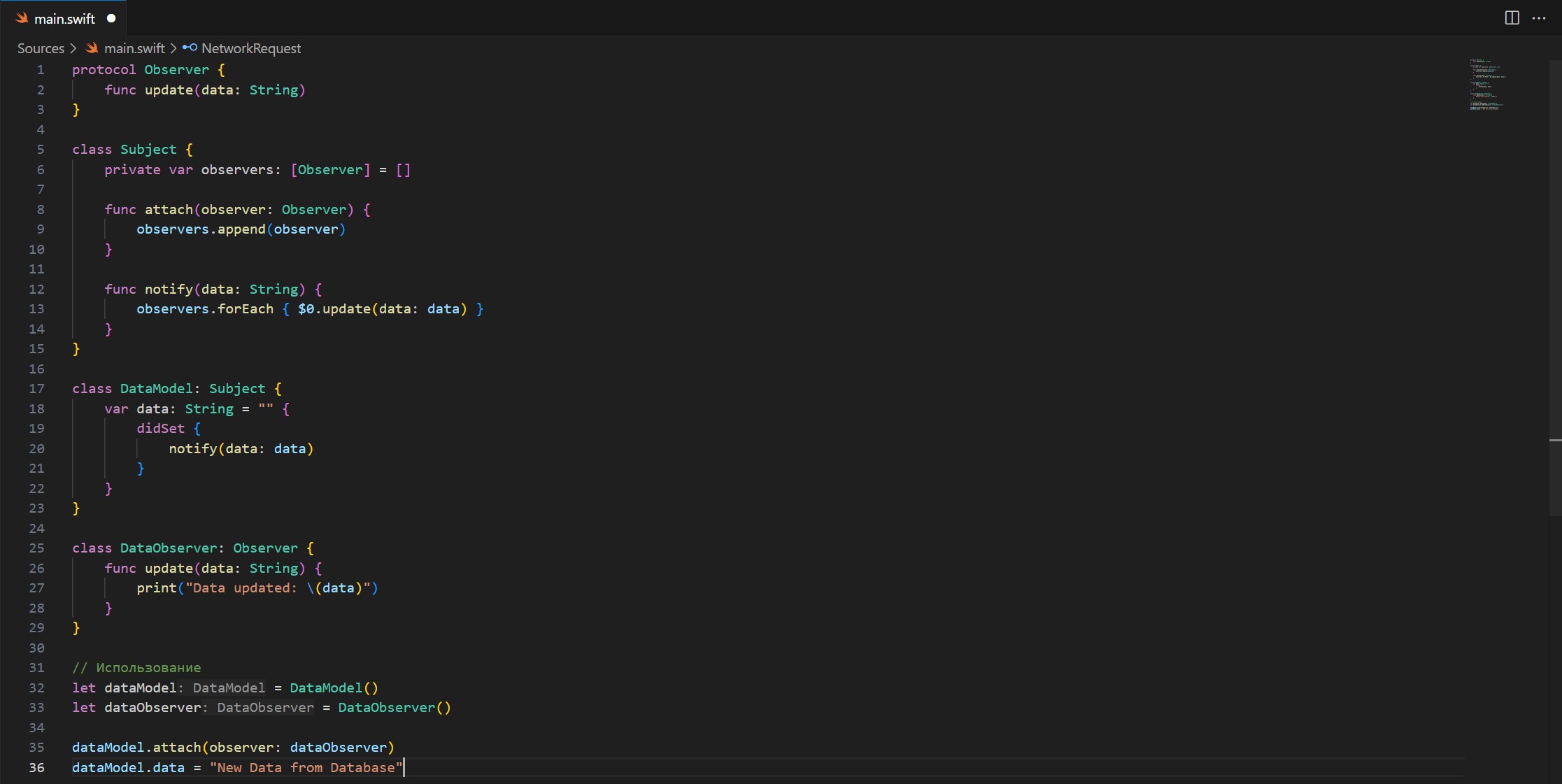


Рисунок 10 Пример реализации Facade

**Поведенческие паттерны**

Поведенческие паттерны относятся к взаимодействию между объектами и обеспечивают различные способы передачи сообщений или распределения ответственности. Основные поведенческие паттерны включают:

1. Наблюдатель (Observer): позволяет объектам подписываться на события других объектов.

**

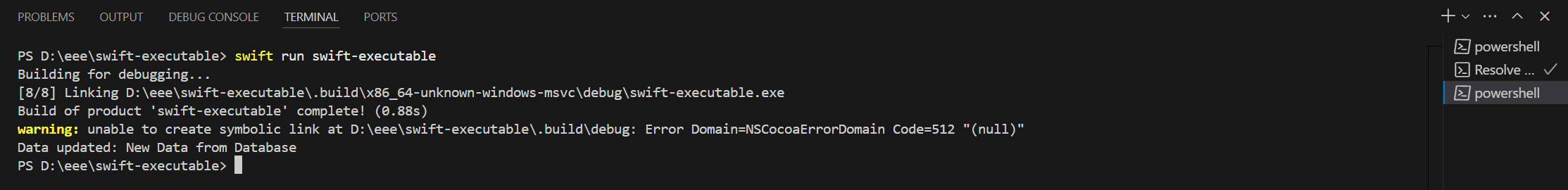
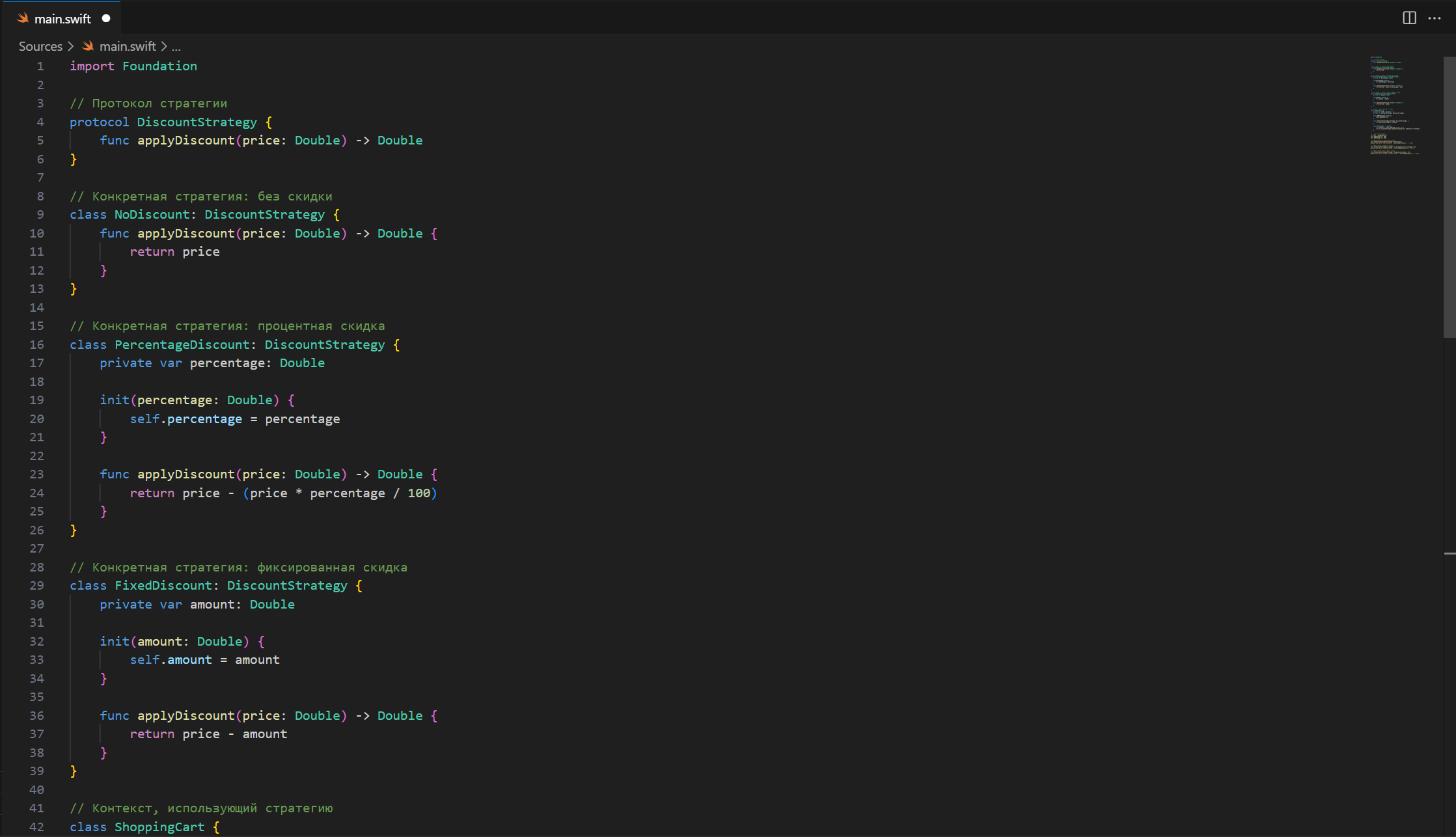
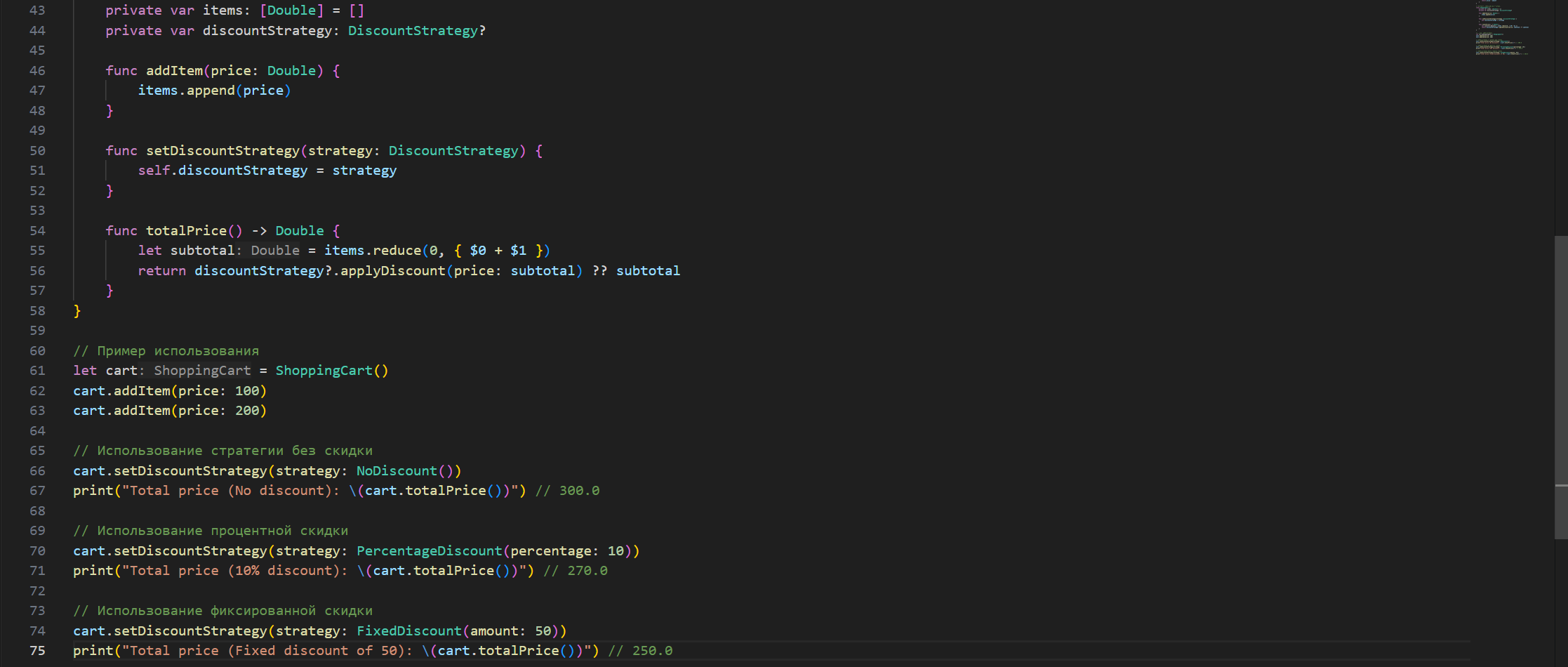
**

Рисунок 11 Пример реализации паттерна Наблюдателя в контексте обновления состояния модели

1. Паттерн Стратегия (Strategy) позволяет определять семейство алгоритмов, инкапсулировать каждый из них и делать их взаимозаменяемыми.





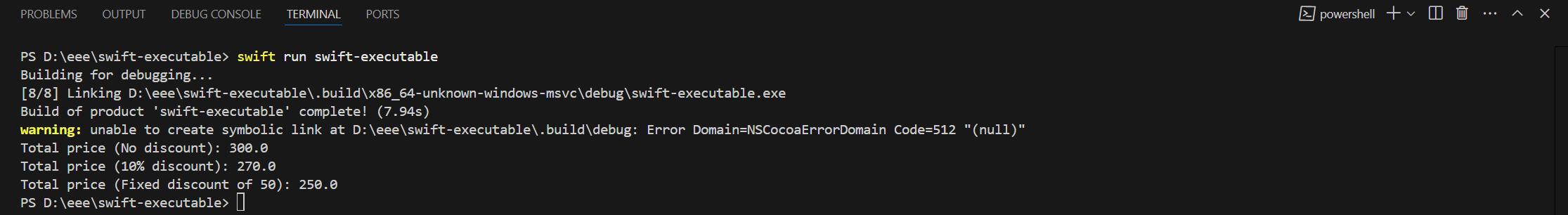
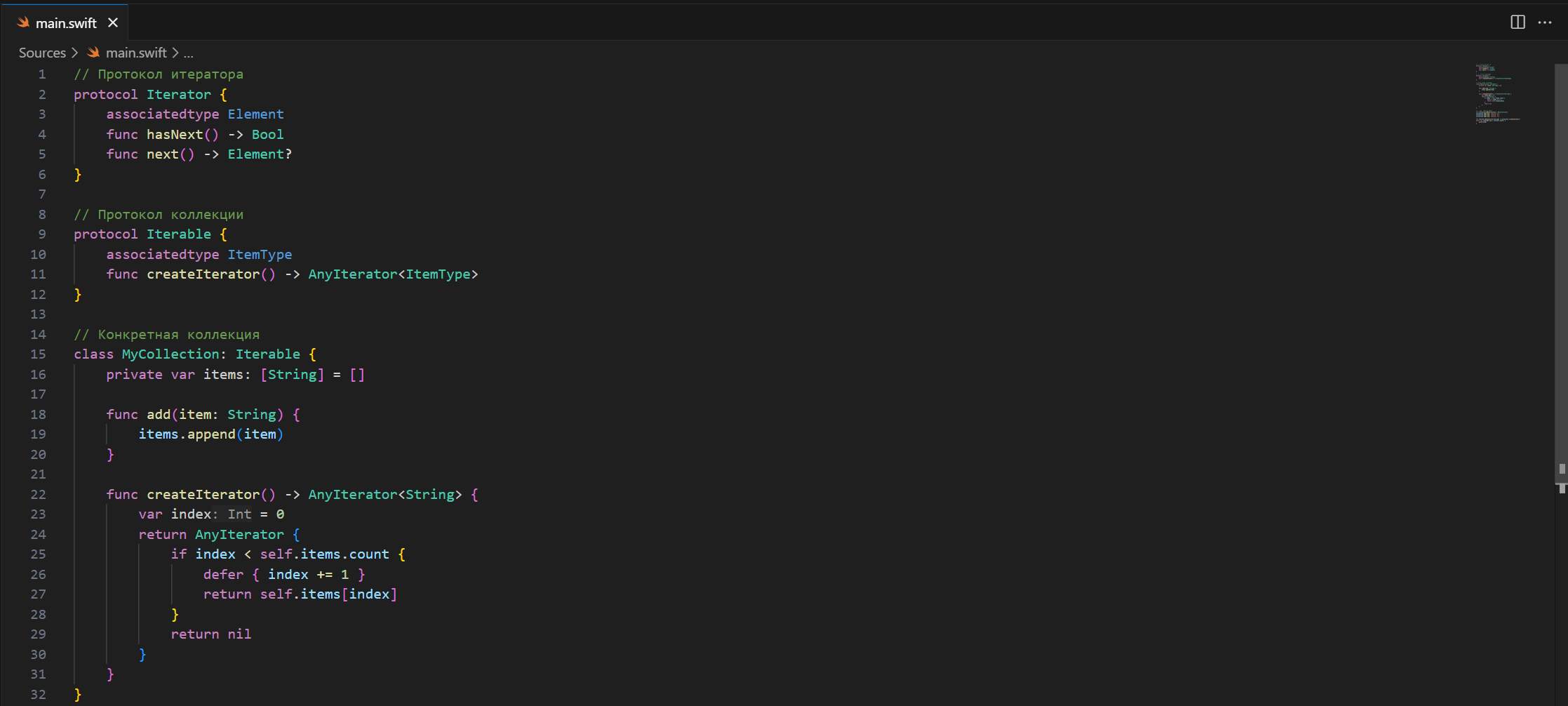


Рисунок 12 Пример реализации Strategy

1. Паттерн "Итератор" (Iterator) предоставляет способ последовательного доступа к элементам составного объекта, не раскрывая его внутреннее представление.



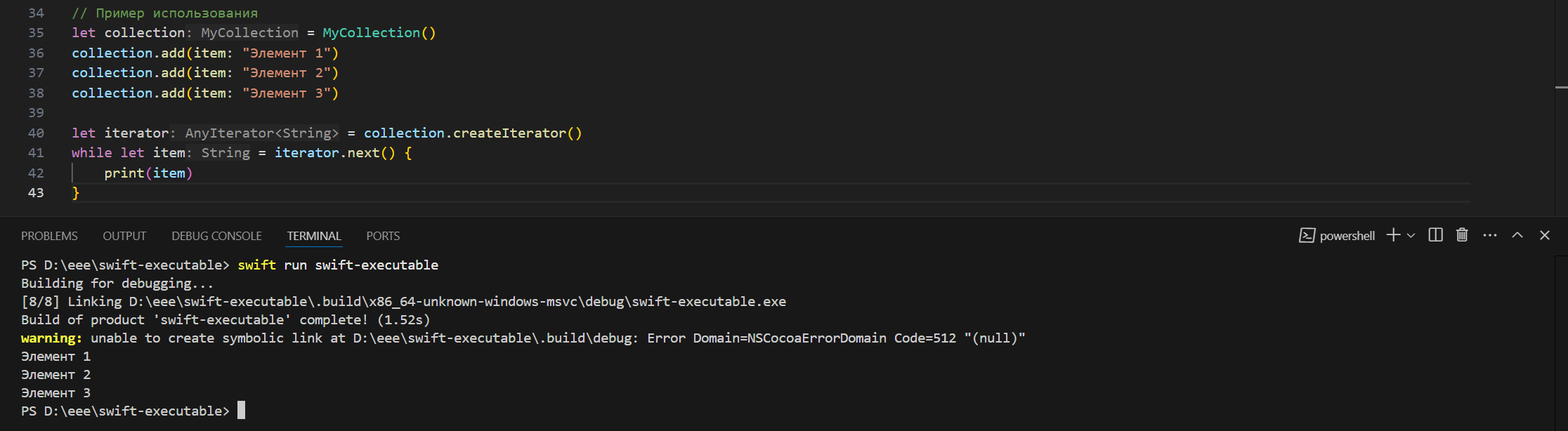
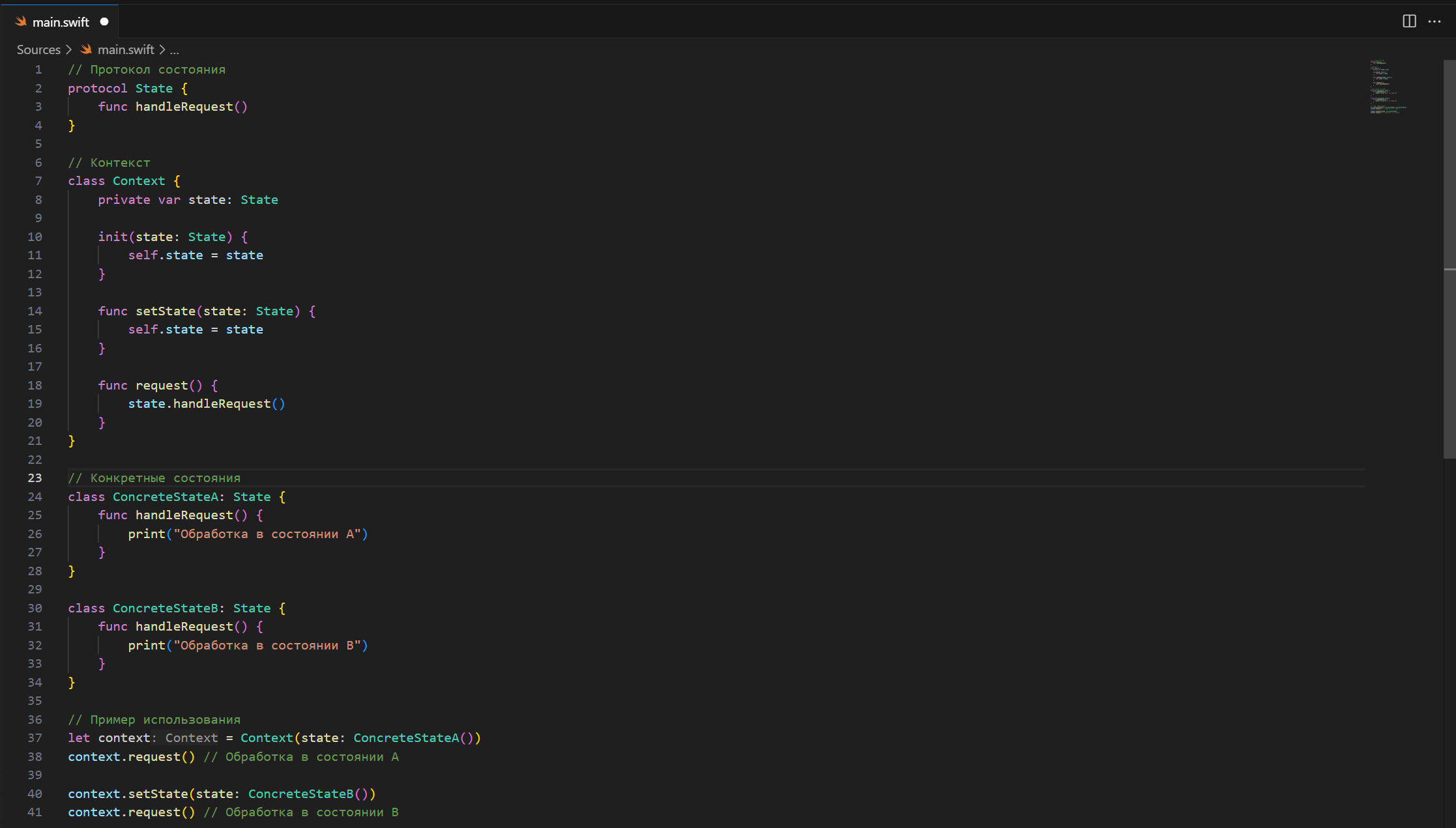


Рисунок 13 Пример реализации iterator

1. Паттерн "Состояние" (State) позволяет объекту изменять свое поведение в зависимости от своего состояния. Объект будет казаться изменившим свой класс.



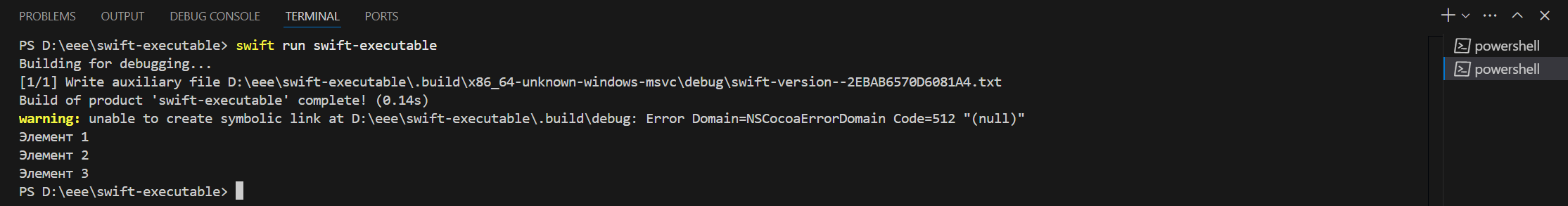
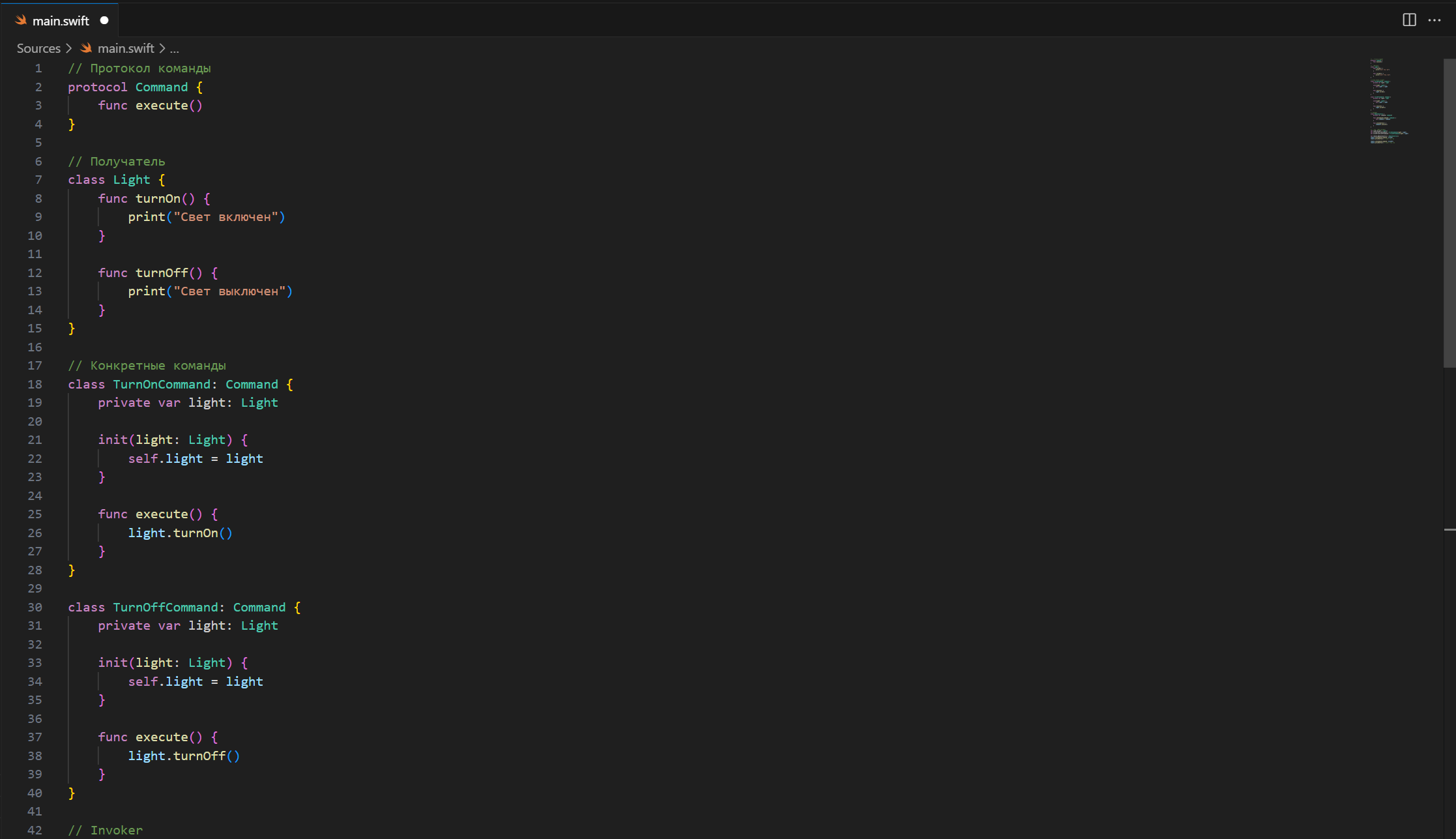


Рисунок 14 Пример реализации State

1. Паттерн "Команда" (Command) превращает запросы в объекты, позволяя параметризовать методы с помощью очередей, логирования и поддержания отмены выполнения операций.



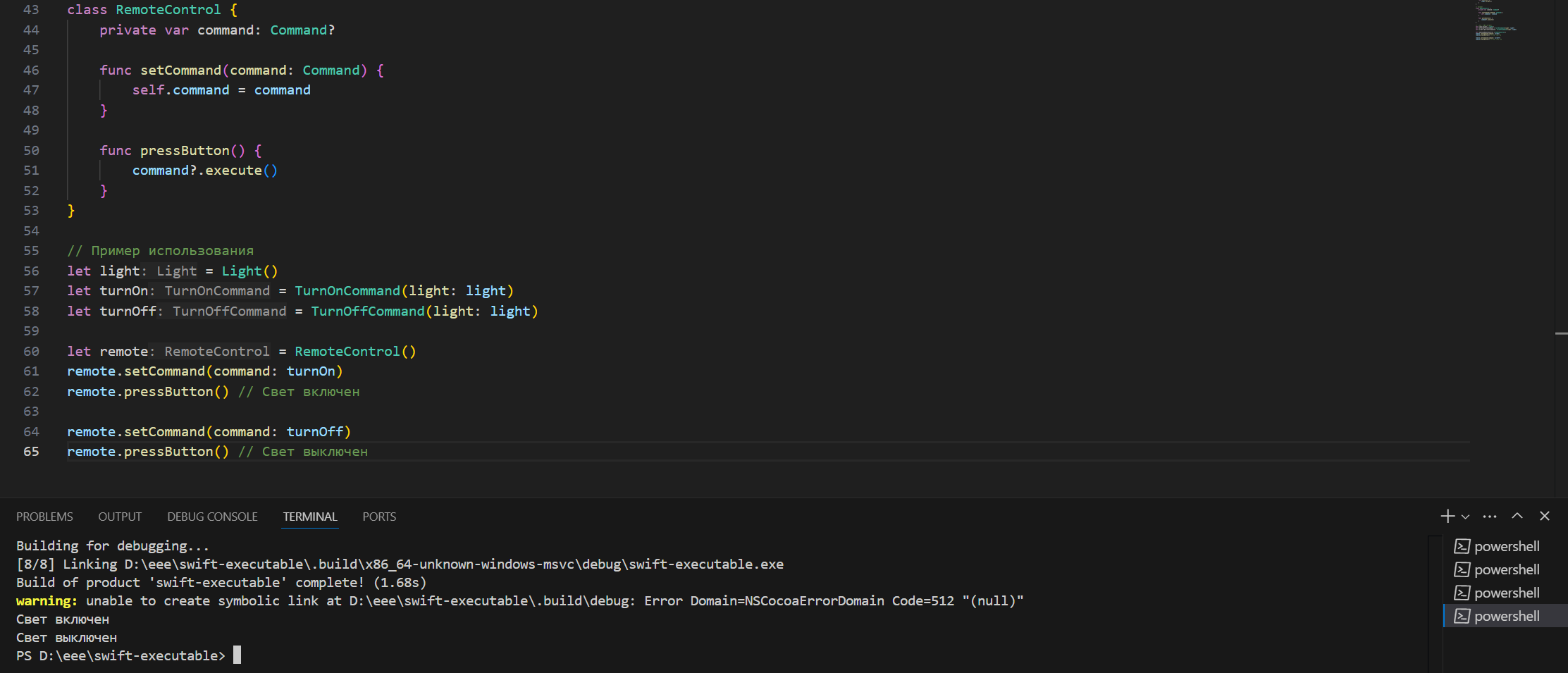


Рисунок 15 Пример реализации Command

**Вывод**

Я ознакомился с понятием Swift шаблонов проектирования, классами порождающих паттернов, структурных паттернов и поведенческих паттернов. На базе сетевого слоя и базы данных, создал фабрику моделей. Привёл примеры кода некоторых паттернов.