**Введение**

Шаблоны проектирования — это испытанный способ решения проблемы в заданном контексте. Они скорее открыты, нежели придуманы, что очевидно и из применения слова «шаблон». Используя шаблон проектирования, вы получаете знания всех сообществ для безопасного решения этой проблемы. Если говорить максимально кратко, то шаблон состоит из распространенной проблемы и ее решения, которое уже можно считать своего рода стандартом. Мы хотим рассказать об одном из таких шаблонов под названием «Адаптер».

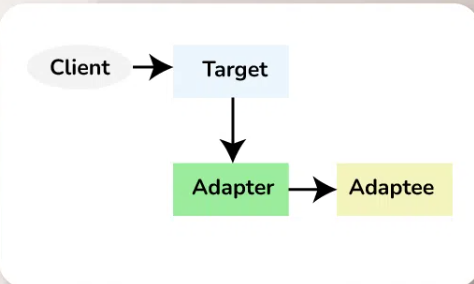
**План презентации:**  
Чтобы понять, что такое Адаптеры и для чего они используются, мы поговорим об:

1. Структуре адаптера
2. Различных реализациях этого шаблона
3. Преимуществах и недостатках
4. А также скажем о вариантах использования

Начнем с того, что же такое шаблон адаптер.   
Шаблон адаптера действует как соединитель между двумя несовместимыми интерфейсами, которые в противном случае не могут быть соединены напрямую. Основная цель этого шаблона — преобразовать существующий интерфейс в другой, который ожидает клиент. Название его говорящее, и вы не раз сталкивались с адаптерами в реальной жизни. Одним из самых распространенных адаптеров являются картридеры, которыми оснащены многие компьютеры и ноутбуки.

**Структура адаптера включает в себя следующие компоненты:**

1. Целевой интерфейс
2. Адаптируемый
3. Адаптер
4. Клиент



A diagram of a plug and two plugs

Description automatically generated with medium confidence

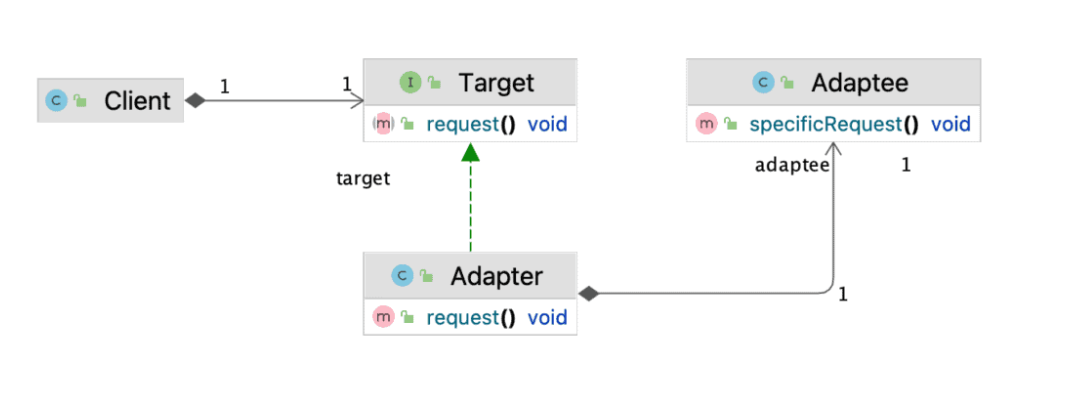
**Целевой интерфейс:** определяет интерфейс, ожидаемый клиентом. Он представляет собой набор операций, которые может использовать клиентский код. Это общий интерфейс, с которым взаимодействует клиентский код.

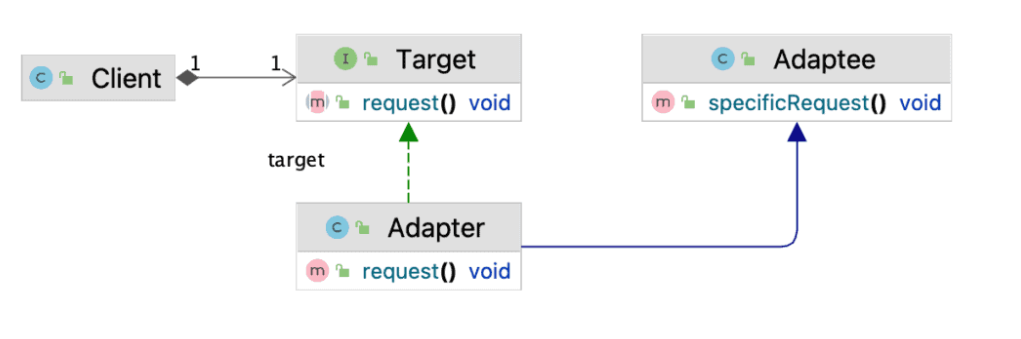
**Адаптируемый:** существующий класс или система с несовместимым интерфейсом, который необходимо интегрировать в новую систему. Это класс или система, которые клиентский код не может использовать напрямую из-за несоответствия интерфейсов.

**Адаптер:** класс, который реализует целевой интерфейс и внутренне использует экземпляр адаптируемого, чтобы сделать его совместимым с целевым интерфейсом. Он действует как мост, адаптируя интерфейс адаптируемого для соответствия целевому интерфейсу.

**Клиент:** код, который использует целевой интерфейс для взаимодействия с объектами. Он остается в неведении относительно конкретных деталей реализации адаптируемого и адаптера. Это код, который получает выгоду от интеграции адаптируемого в систему через адаптер.

**Различные реализации шаблона проектирования адаптера:**

1. **Адаптер объекта**: Эта композиция реализации используется для делегирования логики адаптеру. Это довольно простой способ добиться соответствия интерфейса: В этом случае адаптер содержит Adaptee и делегирует метод request() методу specificRequest() в Adaptee. 
2. **Адаптер класса**: Эта версия шаблона Adapter требует множественного наследования, что технически невозможно в Java, если мы не рассматриваем интерфейсы с методами по умолчанию. Основная идея заключается в создании Adapter путем расширения классов Target и Adapter. Однако мы можем реализовать это в Java, когда у нас есть Target как интерфейс, что проще сделать, поскольку Target — это та часть, которую мы контролируем. Выглядит очень похоже на Object Adapter, но теперь Adapter расширяет Adaptee, а не содержит его композиционно. Одним из преимуществ этого подхода является то, что Adapter может использоваться в обоих контекстах, как Target и как Adaptee. Технически мы создали двусторонний адаптер, что может быть очень удобно в определенных случаях (когда адаптированный объект нужно использовать в разных системах. Например, он может выступать в роли целевого класса в новых системах, работающих с целевыми классами, или в роли адаптируемого класса в других системах, работающих с а даптируемыми классами).



**Преимущества шаблона проектирования адаптера:**

1. Создавая адаптер, вы можете повторно использовать существующий код без необходимости его модификации. Это способствует повторному использованию кода и помогает поддерживать более чистую архитектуру.
2. Разделяя проблемы адаптации интерфейса, шаблон адаптера позволяет классам сосредоточиться на своих основных обязанностях, не имея дела с кодом адаптации, который засоряет их логику. Потому что вы можете просто переключить несколько адаптеров для поддержки разных интерфейсов, не изменяя базовую систему.
3. Отделяя вашу систему от конкретных реализаций, адаптеры упрощают замену или изменение частей без ущерба для функциональности других частей.

**Минусы шаблона проектирования адаптера:**

1. Внедрение адаптеров может добавить уровень сложности в вашу систему. Наличие нескольких адаптеров может усложнить навигацию и понимание кода.
2. Дополнительный уровень косвенности может привести к небольшому снижению производительности, особенно если адаптеру необходимо выполнять сложные преобразования.
3. При отсутствии должного управления использование адаптеров может привести к проблемам обслуживания. Отслеживание нескольких адаптеров для различных интерфейсов может стать обременительным.
4. Существует риск чрезмерного использования адаптеров для тривиальных изменений, что может привести к ненужной сложности. Крайне важно оценить, действительно ли требуется адаптер в конкретных обстоятельствах.
5. Адаптеры могут транслировать только два интерфейса; если вам нужно настроиться на несколько интерфейсов, вам может потребоваться много разных адаптеров, что может еще больше усложнить проект.

**Варианты использования:**

1. Для интеграции сторонних библиотек или устаревшего кода в систему.
2. Если у вас есть класс с несовместимым интерфейсом, но вы не хотите изменять исходный класс.
3. Если вы хотите разрешить системам взаимодействовать, имеющим разные интерфейсы.