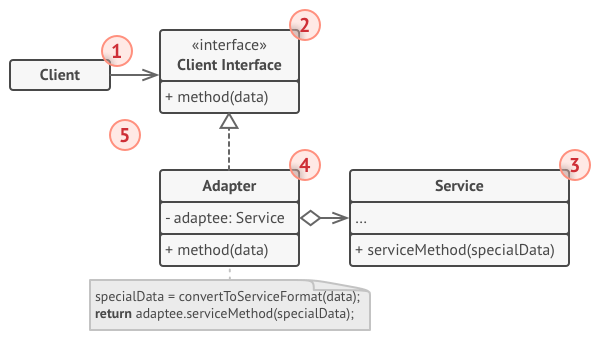
**АДАПТЕР**: преобразует интерфейс класса к другому интерфейсу, на который рассчитан клиент. Позволяет объектам с несовместимыми интерфейсами работать вместе. Работая с адаптером через интерфейс, клиент не привязывается к конкретному классу адаптера - благодаря этому, можно добавлять в программу новые виды адаптеров, независимо от клиентского кода.

ПРИМЕНЕНИЕ:

- когда хотите использовать сторонний класс, но его интерфейс не соответствует остальному коду приложения

- когда нужно использовать несколько существующих подклассов, но в них не хватает какой-то общей функциональности, причём расширить суперкласс вы не можете

РЕАЛИЗАЦИЯ:

**1. КЛИЕНТ**: класс, который содержит существующую бизнес-логику программы. Например, работает с методами "утки quack() и fly())"

**2. (Целевой) ИНТЕРФЕЙС КЛИЕНТА**: описывает протокол, через который клиент может работать с другими классами. Т.е. то, к чему адаптируется. Например, интерфейсные методы "утки (quack() и fly())"

**3. (Адаптируемый неизвестный) СЕРВИС**: какой-то полезный класс, обычно сторонний. Клиент не может использовать этот класс напрямую, т.к. сервис имеет непонятный ему интерфейс. Например, "дикая индейка (с методами gobble() и fly()) "

**4. АДАПТЕР**: класс, который может одновременно работать и с клиентом, и с сервисом. Реализует клиентский интерфейс и содержит ссылку на объект сервиса, при этом ссылок может быть даже не одна. Получает вызовы от клиента через методы клиентского интерфейса, а затем переводит их в вызовы методов обёрнутого объекта в правильном формате. Например, "адаптерИндейки (приводит методы Индейки к методам Утки)"

**5.** Работая с адаптером через интерфейс, клиент не привязывается к конкретному классу адаптера. Благодаря этому, вы можете добавлять в программу новые виды адаптеров, независимо от клиентского кода. Это может пригодиться, если интерфейс сервиса вдруг изменится, например, после выхода новой версии сторонней библиотеки

ПЛЮСЫ:

- отделяет и скрывает от клиента подробности преобразования различных интерфейсов

МИНУСЫ:

- усложняет код программы из-за введения дополнительных классов