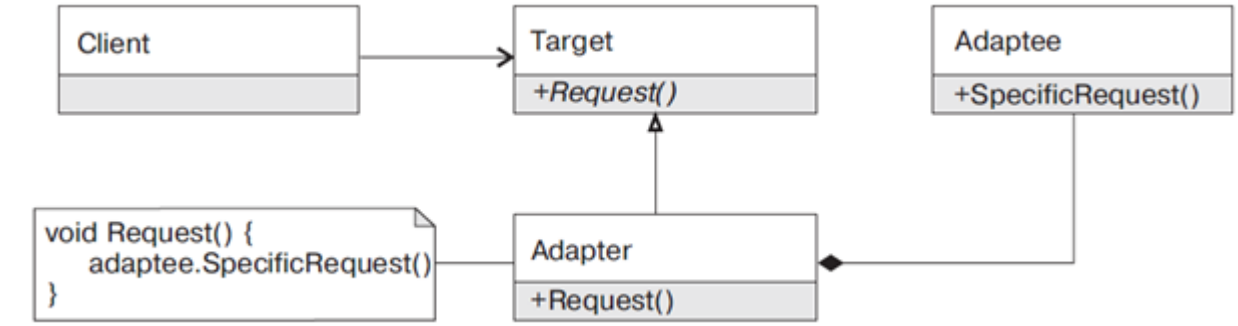
Шаблон проектирования "Адаптер"[1][2]

Адаптер (Adapter) – это структурный паттерн, который позволяет адаптировать интерфейс класса в соответствии с требованиями системы. То есть, это своеобразная прослойка между классами, приводящая интерфейс одного класса к используемому в другом.

Диаграмма



- **Target** — целевой интерфейс, к которому нужно преобразовать интерфейс существующих классов;
- **Adaptee** — существующий класс, чей интерфейс нужно преобразовать;
- **Adapter** — класс-адаптер, который преобразует интерфейс адаптируемого класса к целевому;
- **Client** — клиенты нового интерфейса, которые работают с адаптированными классами полиморфным образом.

Пример

In [1]:

```
class Target:
    """
    Целевой класс объявляет интерфейс, с которым может работать клиентский код.
    """
    def request(self) -> str:
        return "Target: The default target's behavior."


class Adaptee:
```

[1] https://refactoring.guru/ru/design-patterns/adapter/python/example
[2] https://shwanoff.ru/adapter/

```
    """
    Адаптируемый класс содержит некоторое полезное поведение, но его интерфейс
    несовместим с существующим клиентским кодом. Адаптируемый класс нуждается в
    некоторой доработке, прежде чем клиентский код сможет его использовать.
    """

    def specific_request(self) -> str:
        return ".eetpadA eht fo roivaheb laicepS"


class Adapter(Target, Adaptee):
    """
    Адаптер делает интерфейс Адаптируемого класса совместимым с целевым
    интерфейсом благодаря множественному наследованию.
    """
    def request(self) -> str:
        return f"Adapter: (TRANSLATED) {self.specific_request()[::-1]}"


def client_code(target: "Target") -> None:
    """
    Клиентский код поддерживает все классы, использующие интерфейс Target.
    """
    print(target.request(), end="")


if __name__ == "__main__":
    print("Client: I can work just fine with the Target objects:")
    target = Target()
    client_code(target)
    print("\n")
```

```
    adaptee = Adaptee()
    print("Client: The Adaptee class has a weird interface. "
          "See, I don't understand it:")
    print(f"Adaptee: {adaptee.specific_request()}", end="\n\n")

    print("Client: But I can work with it via the Adapter:")
    adapter = Adapter()
    client_code(adapter)
```

```
Client: I can work just fine with the Target objects:
Target: The default target's behavior.

Client: The Adaptee class has a weird interface. See, I don't understand it:
Adaptee: .eetpadA eht fo roivaheb laicepS

Client: But I can work with it via the Adapter:
Adapter: (TRANSLATED) Special behavior of the Adaptee.
```