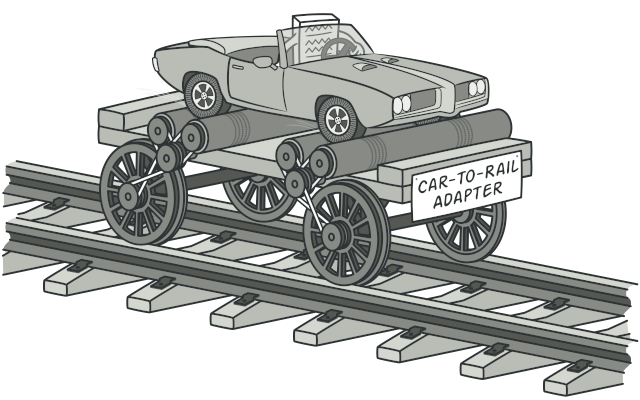
Практическая работа №5: адаптер

Дуботолкова Натали

**Также известен как:**Wrapper, Обёртка, Adapter

**Суть паттерна**

**Адаптер** — это *структурный* паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

**Пример из жизни**

Переходник из старых розеток в новые

Переводчик с одного языка на другой

**Применение на практике**

* **Когда вы хотите использовать сторонний класс, но его интерфейс не соответствует остальному коду приложения.**  Адаптер позволяет создать объект-прокладку, который будет превращать вызовы приложения в формат, понятный стороннему классу.
* **Когда вам нужно использовать несколько существующих подклассов, но в них не хватает какой-то общей функциональности, причём расширить суперкласс вы не можете.**  Вы могли бы создать ещё один уровень подклассов и добавить в них недостающую функциональность. Но при этом придётся дублировать один и тот же код в обеих ветках подклассов. Более элегантным решением было бы поместить недостающую функциональность в адаптер и приспособить его для работы с суперклассом. Такой адаптер сможет работать со всеми подклассами иерархии. Это решение будет сильно напоминать паттерн [**Декоратор**](https://refactoring.guru/ru/design-patterns/decorator).

**Сравнение с другими паттернами**

* [**Мост**](https://refactoring.guru/ru/design-patterns/bridge) проектируют сразу при разработке проекта, чтобы развивать большие части приложения отдельно друг от друга. [**Адаптер**](https://refactoring.guru/ru/design-patterns/adapter) применяется после, чтобы заставить несовместимые классы работать вместе.
* [**Адаптер**](https://refactoring.guru/ru/design-patterns/adapter) меняет интерфейс существующего объекта. [**Декоратор**](https://refactoring.guru/ru/design-patterns/decorator) улучшает другой объект без изменения его интерфейса.
* [**Адаптер**](https://refactoring.guru/ru/design-patterns/adapter) предоставляет классу альтернативный интерфейс. [**Декоратор**](https://refactoring.guru/ru/design-patterns/decorator) предоставляет расширенный интерфейс. [**Заместитель**](https://refactoring.guru/ru/design-patterns/proxy) предоставляет тот же интерфейс.
* [**Фасад**](https://refactoring.guru/ru/design-patterns/facade) задаёт новый интерфейс, тогда как [**Адаптер**](https://refactoring.guru/ru/design-patterns/adapter) повторно использует старый. *Адаптер* оборачивает только один класс, а *Фасад* оборачивает целую подсистему. Кроме того, *Адаптер* позволяет двум существующим интерфейсам работать сообща вместо того, чтобы задать полностью новый.
* [**Мост**](https://refactoring.guru/ru/design-patterns/bridge), [**Стратегия**](https://refactoring.guru/ru/design-patterns/strategy) и [**Состояние**](https://refactoring.guru/ru/design-patterns/state) (а также слегка и [**Адаптер**](https://refactoring.guru/ru/design-patterns/adapter)) имеют схожие структуры классов — все они построены на принципе «композиции», то есть делегирования работы другим объектам.

**Мой пример**

У нас имеется COM-порт, но в данный момент он нам не нужен, зато требуется работать с USB-накопителем. Поэтому, с помощью драйвера компьютер может включить в COM-порт USB-накопитель.

При реализации выбирала между двумя паттернами: декоратором и адаптером. Эти шаблоны схожи тем, что создают некоторую "обертку" вокруг класса. Однако различаются они тем, какой интерфейс предоставляют и что делают с функциональностью оригинального класса.

|  |  |  |
| --- | --- | --- |
| Шаблон | Что делает с интерфейсом | Что делает с функционалом |
| Декоратор | Не изменяет/расширяет | Расширяет |
| Адаптер | Изменяет | Не изменяет |

В моём примере у нас с функциональностью проблем нет. СОМ-порт делает всё, что нам нужно.

Список используемой литературы

Лекция 5 на dl.spbstu.ru

<https://refactoring.guru/ru/design-patterns/adapter>

<https://refactoring.guru/ru/design-patterns/decorator>

<https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B5%D1%80_(%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)>

<https://ru.stackoverflow.com/questions/693654/%D0%92-%D1%87%D1%91%D0%BC-%D0%BE%D1%82%D0%BB%D0%B8%D1%87%D0%B8%D1%8F-%D0%B8-%D1%81%D1%85%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%B0-%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B2-adapter-decorator-wrapper-%D0%B8-proxy>