# Адаптер шаблон

**Цел –** Преобразува даден интерфейс на един клас във друг, който се изисква и очаква от клиентите (другите софтуерни модули на дадено приложение). Осигурява съвместимост между класовете в случаите, когато това е невъзможност поради несъвместимост между интерфейсите им.

**Мотивация –** Понякога когато работим по дизайна на едно приложение стигаме до момент, къде е необходимо използването на даден модул (клас), който вече имаме разработен по друг проект. Проблемът възниква, когато този вече съществуващ модул е частично или напълно не съвместим с интефейса използван в текущото приложение. В такива случаи едно от решенията е да пренапишем изцяло функционалността, което като цяло е безмислено и чиста загуба на време (още повече ако става въпрос за голям модул, които освен че е вече написан, добре тестван и доказано работещ). Друго решение е да вземем съществуващият модул и да го модифицираме за нуждите на новото, но и това не е най-доброто, защото отново трябва да се изтестват всички функционалности на готовият модул, но най-вече този вариант отпада, ако не разполагаме със сорс кода му. Най-удачният вариант на решение да използваме междиннен клас, който да капсулира готовият модул и да го представи чрез интерфейс, който е напълно съвместим с останалата част от новото приложение.

**Приложимост –** Необходимо е да се използва съществуващ клас, но интерфейса му е несъвместим със това от което имаме нужда.

**Имплементиране –** Модула (класът), който е необходимо да направим достъпен като функционалност към текущата система трябва да се инстанциира като private от адаптера вътрешно за него (например локална променлива, която да реферира към него), за да не е достъпен за останалата част от кода, докато клиента (разработвания софтуер) трябва да се наследи от самият адаптер следвайки ООП принципите.

**Известни употреби –** Може да се използва например ако имаме готова библиотека с клас съдържащ методи за текстообработка, но се налага да я използваме за графичното моделиране на текст в софтуер за чертане. В случая имаме нужда от капсулиране на функционалността на съществуващите тексобработващи методи, за употребата им в графично приложение за обработка на текст. Посредствум този шаблон ще имаме възможност да донадградим и липсващата функционалност във съществуващата библиотека, но изисквана от интерфейса на графичната среда.

**Следствия -** Може да се използва както за класове, така и за обекти. Той прави възможно използването на съществуващ код след неговото създаване – пост фактум.

**Свързани шаблони –** Мост, Декоратор, Пълномощно

**Структура –** виж графичния файл в директорията.