**Facade**

Шаблонът за дизайн "Фасада" е използван главно в обектно ориентираното програмиране, като името му идва от фасадата на сграда. Фасадата е обект, който предоставя опростен интерфейс към много класове или библиотека. Той спомага за по-лесното и разбираемо използване на софтуера, а също така и при тестването му. Фасадата предотвратява зависимостта на допълнително добавено приложение към съществуващото, изолзващо библиотеката, като така го прави по-лесно за поддръжка и доработка. Този вид шаблон се използва често, когато софтуерът е много сложен и комплексен или труден за разбиране, поради многото взаимозависими класове, съдържащи многобройни обекти. Фасадата често крие детайли за имплементацията си, поради съображения за сигурност, а и за улесненние на клиента.

**Decorator**

Декораторът е структурен шаблон за дизайн, който се използва в обектно-ориентираното програмиране. Този шаблон може да бъде използван за разширяването на функционалността на определен клас по времето на изпълнение на програмата, като запазва интерфейса му. Той е много подходящ, когато се следва Принципът за еднолична отговорност (Single responsibility principle), според който един клас трябва да е отговорен за една единствена операция. Този вид шаблон може да се използва за добавяне на функционалност към определен обект динамично, без това да засяга другите инстанции на същия клас. Това се постига чрез създаването на нов клас Декоратор, който обхваща конкретния клас. Особеностите на декоратора (методи, свойства и други) обикновено се определят от интерфейса, който е еднакъв за декораторите и за декорираните обекти.

**Adapter**

Адаптер е структурен шаблон за дизайн, който се използва в обектно-ориентираното програмиране. Той позволява интерфейса на даден клас да бъде използван от друг интерфейс. Често се ползва при необходимост, за да могат съществуващи класове да работят с други, без да се променя техния изходен код. Адаптер шаблона е инструмент, който прави възможна работата на два несъвместими интерфейса. Интерфейсите могат да бъдат несъвместими, но вътрешните функционалности трябва да отговарят на приложението. Адаптерът позволява иначе несъвместими класове да работят заедно чрез превръщането на интерфейса на един от тях в интерфейс с вид, очакван от клиента. Адаптерът е полезен в ситуации, в които един вече съществуващ клас осигурява някои или всички необходими услуги, но не използва необходимия интерфейс.