Шаблони в проектирането при създаване с ООП

**1.Структурни шаблони за проектиране с ООП:**

**-** Адаптер (Adapter, GOF 139) - може да се използва за

преобразуване на един интерфейс на клас до друг, с

цел по-лесно програмиране.

**-** Мост (Bridge, GOF 151) - разделя абстракция на даден обект от неговата имплементация, така че да можем да ги варираме независимо.

**-** Композит (Composite, GOF 163) - описва как да се изгради клас-йерархия на обекти, всеки от които може да бъде прост или съставен обект. Съставният обекти ви позволяват да композирате примитивни и композитни обекти в произволно сложни структури.

**-** Декоратор (Decorator, GoF 175) - описва как динамично да добавяме отговорности към обекти; композира обектите рекурсивно, за да позволи отворен брой добавени допълнителни отговорности. Например, декоратор обект, съдържащ компонент от потребителския интерфейс, може да добави декорация като граница или сянка на компонента, или пък функционалност като превъртане и мащабиране.

**-** Фасада (Facade, GoF 185) - показва как един обект може да представлява цялата подсистема. Фасадата е представител за набор от обекти. Фасадата изпълнява своите отговорности чрез изпращане на съобщения до обектите, които тя представлява.

**-**Миниобект (Flyweight, GoF 195) - определя структура за споделяне на обекти. Обекти се споделят наймалко по две причини: ефективност и съгласуваност. Flyweight се фокусира върху споделяне с цел ефективно използване на паметта. При използване на много обекти, трябва да се обърне особено внимание на цената на всеки обект. Пестим чрез споделяне на обекти, вместо чрез репликирането им.

**-** Пълномощник (Proxy, GoF 207) - действа като контейнер за друг обект, като местен представител за обекта в друго адресно пространство. Може да представлява тежък обект, който се зареждат при поискване, или да защитава достъп до чувствителен обект.

**2.Поведенчески шаблони за проектиране с ООП:**

Поведенчески шаблони за проектиране са свързани с алгоритми и възлагането на отговорности между обектите. ν Поведенческите модели описват не само образци с обекти или класове, но и модели на комуникация между тях. ν Тези модели характеризират комплексен контрол на изпълнение на процеса. Те обаче изместват фокуса си от потока на управление, за да ви позволят да се концентрирате само върху начина, по който обектите са свързани помежду си.

*Поведенчески шаблони, използващи наследяване, за да разпространят поведение между класовете:*

-Шаблонен метод - Template Method (GoF325) - отнася се до абстрактна дефиниция на алгоритъм, като определя алгоритъма стъпка по стъпка, където всяка стъпка се позовава на абстрактна операция или примитивна операция. Подкласът реализира алгоритъма чрез определяне на абстрактните операции.

-Интерпретатор - Interpreter (GoF243) - представя граматика като йерархия от класове и изпълнява интерпретатор като операция върху екземпляри на тези класове.

-Спомен - Memento (GoF381) - без да нарушава капсулирането, улавя и екстернализира вътрешното състояние на даден обект, така че обектът да може да бъде възстановен до това състояние по-късно.

*Други поведенчески шаблони използват композиции от обекти. Някои описват как група от партньорски обекти си сътрудничат за изпълнение на задачата, ако всеки един обект не може да извърши работата сам. Важен въпрос тук е какво партньорските обекти (рeers) знаят един за друг. Те може да поддържат изрични референции един към друг, но това ще увеличи тяхната свързаност*.

-Посредник - Mediator (GoF273) - избягва този с въвеждането на обект-посредник между равнопоставени; осигурява опосредственост, необходима за по-слабо свързване.

-Верига от отговорности - Chain of Responsibility (GoF223) - осигурява още по-слаба степен на свързаност. Позволява да се изпращат заявки към обект имплицитно чрез верига от кандидатобекти за обработка на заявката. Всеки кандидат може да изпълни заявката в зависимост от условията в реално време. Броят на кандидатите е отворен и можем да изберем кои кандидати участват във веригата по време на изпълнение.

-Наблюдател - Observer (GoF293) - определя и поддържа зависимост между обектите, напр. в Модел/Изглед/Контролер.

*Други поведенчески модели обекти са свързани с капсулиране на поведение в даден обект и делегиране на заявки към него:*

-Стратегия - Strategy (GoF315) - капсулира алгоритъм в даден обект. Стратегията прави алгоритъма лесен за въвеждане и промяна в обекта, който го използва;

-Команда - Command (GoF233) - капсулира заявка в обект, така че тя да може да се предава като параметър, съхранява в списъка с историята, или да се манипулира по друг начин;

-Състояние - State (GoF305) - капсулира състоянията на даден обект, така че обектът може да промени поведението си, когато неговия обект със състоянията се промени;

-Посетител - Visitor (GoF331) - капсулира поведение, което иначе би било разпределено по класове;

-Итератор - Iterator (GoF257) - абстрахира начина за достъп и преминаване (траверс) по обектите в агрегат.