„Factory Method“

„Factory Method“ е дизайн патърн, който се използва за решаване на проблеми със създаването на обекти в дадено приложение.

Този патърн се използва с цел инстанцирането на различни обекти, като се дефинира интерфейс, който съдържа метод за създаването на обекти. Конкретната имплементация на метода, с който се създават конкретните обекти се прави в наследниците на интерфейса. Когато този метод бъде извикан, той връща нов обект по предварително зададени параметри, които конкретната фабрика знае.

„Factory Method“ е приложим, когато даден клас не знае как се създават различни обекти, а наследниците му могат да имплементират създаването на възможните обекти.

Чрез този патърн, създаването на обекти може да се изнесе в едно място от кода, а от друга страна се вдига абстракцията като се добавя създава нова фабрика създаваща нов обект.



Фиг.1 „Factory Method“

“Decorator”

Идеята на Декоратор патърна е, че може динамично да добавя нови функционалности върху обектите. Използването му се налга, когато е необходимо да се добави нова функционалност, върху съществуващ обект. По-лошия вариант това да се направи е чрез създаване на нов обект, който наследява този, който искаме да разширим, като новия съдържа новата функционалност. Това създава голям проблем при необходимост от много и различни функционалност, тъй като би означавало нов клас за всяка от тях, което води до т.нар. “class explosion”, което означава експоненциално нарастване на броя на класовете. Това води до по-сложна поддръжка и утежняване на кода като цяло.

Чрез Декоратор, може да се създаде клас, който “обгръща” разширяемия обект, като наследява абстрактния клас/интерфейса на конкретните класове и в същото време приема като параметър инстанция на конкретните класове. С други думи, самия Декоратор е един клас, който съдържа разширената функционалност, която може да се приложи за всички конкретни наследници на абстрактния клас/интерфейса.



Фиг.2 “Decorator Pattern”

“Strategy”

Дизайн патърн „Стратегия“ се прилага тогава, когато трябва върху едни и същи видове данни да се прилагат различни алгоритми за тяхната обработка. При този патърн имаме различни стратегии/алгоритми, които са наследници на базов клас/интерфейс общ за всички стратегии, който знае какви ще са обработваните данни. Употребата на този дизайн патърн е подходяща, когато има различни варианти за обработка на данните, но не е подходящо всеки от алгоритмите да присъства във всеки обект и на база някакво условие да се приложи определен алгоритъм.

Стратегия се използва, като се дефинира семейство от различни, но взаимнозаменяеми алгоритми, всеки един от тях енкапсулиран като отделен обект. Стратегията позволява превключването от един към друг алгоритъм да става независимо от клиентския код който ги използва, като може да използва и данни, които не са видими за клиента.



Фиг.3 “Strategy”