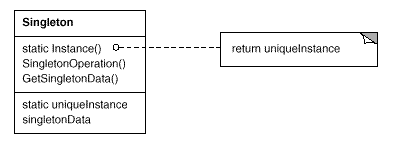
# Strategy Design Pattern

В софтуерното инженерство Strategy е дизайн модел, който променя поведението си по време на изпълнението на програмата. Целта е предоставяне на опростено API и по-лесна работа. Дефинира група от алгоритми, които са взаимно заменяеми и независими от клиента. Поведението не трябва да е наследено, а капсулирано чрез интерфейси. По този начин се гарантира независимостта на алгоритмите един от друг. В примера подаваме числа на обекта context. В зависимост от това с каква операция са зададени, той взима решение, кой алгоритъм за обработка да се изпълни.



# Singleton Design Pattern

Singleton е един от creational шаблоните. Идеята му е да дава глобален достъп до обект и ограничи възможността му за инстанциране. Полезен е, когато искаме да координираме действия в дадена система или сверим даден state. Има предимство пред глобални променливи, защото не замърсява неймспейса и в зависимост от имплементацията позволява lazy initialization и възможност за използване в multithreaded приложения.



# Facade Pattern

Facade е един от structural шаблоните. Идеята му е да енкапсулира сложна подсистема от различни обекти извършващи различни действия, в един общ обект. Така се намалява използването на различни обекти за определени действия, а се използва само един който знае кой клас да се извика и координира с всеки за всякко определено действие. Така също се отделя подсистемата от потенциално многото клиенти, които биха я ползвали. Но от друга страна, ако фасадата е единствената точка за достъп към подсистемата, така се намаляват възможностите и гъвкавостта от която потребителите, които я ползват, биха имали нужда.

