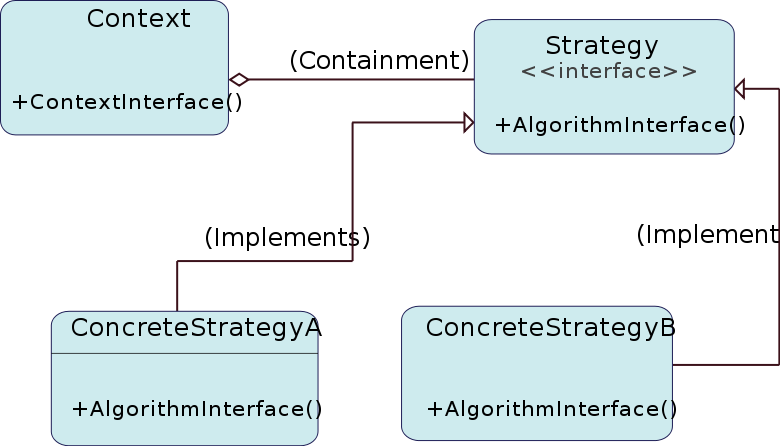
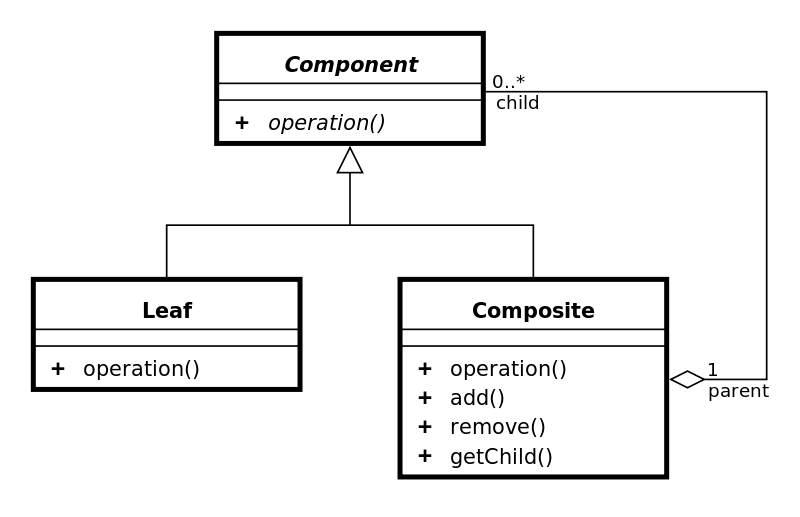
# Strategy Desing Pattern

В софтуерното инженерство Strategy е дизайн модел, който променя поведението си рънтайм. Целта е предоставяне на опростено API и по-лесна работа. Дефинира група от алгоритми, които са взаимно заменими 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 кое е по-голямо или са равни той взима решение, кой алгоритъм за обработка да се изпълни.



# Composite Design Pattern

Composite е структурен патърн. При прилагането му колекция от обекти се третира също като един обект. Целта му е да групира обектите по йерархичен начин инстанциите на класове. Така третираме по един начин индивидуални обекти и колекции. Ако искаме примерно да променим даден обект, всички под него също се променят. Примерът от домашното създава структура подобна на дърво. Когато извикаме Print() от корена всички негови под-елементи се принтират. След това правим същото само за един клон със съответният резултат – всички дъщерни елементи са принтирани.



# Singleton Design Pattern

Singleton е един от creational патърните. 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.

