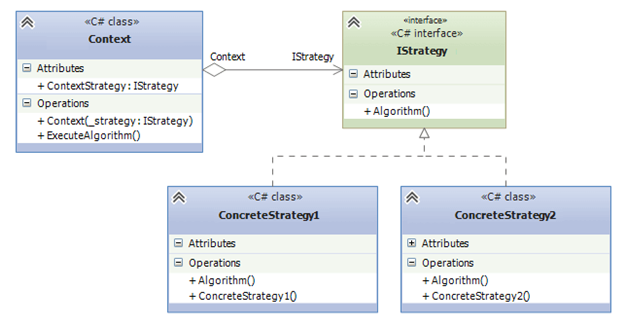
Паттерн Стратегия (Strategy) представляет шаблон проектирования, который определяет набор алгоритмов, инкапсулирует каждый из них и обеспечивает их взаимозаменяемость. В зависимости от ситуации мы можем легко заменить один используемый алгоритм другим. При этом замена алгоритма происходит независимо от объекта, который использует данный алгоритм.

Когда использовать стратегию?

* Когда есть несколько родственных классов, которые отличаются поведением. Можно задать один основной класс, а разные варианты поведения вынести в отдельные классы и при необходимости их применять
* Когда необходимо обеспечить выбор из нескольких вариантов алгоритмов, которые можно легко менять в зависимости от условий
* Когда необходимо менять поведение объектов на стадии выполнения программы
* Когда класс, применяющий определенную функциональность, ничего не должен знать о ее реализации

Формально паттерн Стратегия можно выразить следующей схемой UML:



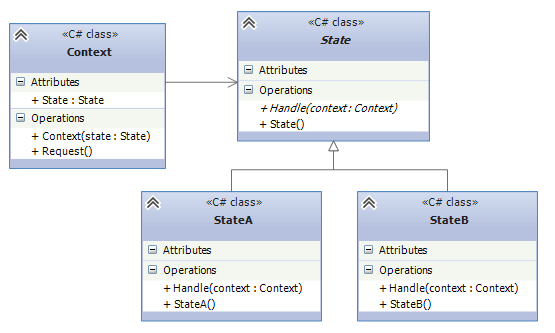
Состояние (State) - шаблон проектирования, который позволяет объекту изменять свое поведение в зависимости от внутреннего состояния.

Когда применяется данный паттерн?

Когда поведение объекта должно зависеть от его состояния и может изменяться динамически во время выполнения

Когда в коде методов объекта используются многочисленные условные конструкции, выбор которых зависит от текущего состояния объекта

UML-диаграмма данного шаблона проектирования предлагает следующую систему:



Паттерн Хранитель (Memento) позволяет выносить внутреннее состояние объекта за его пределы для последующего возможного восстановления объекта без нарушения принципа инкапсуляции.

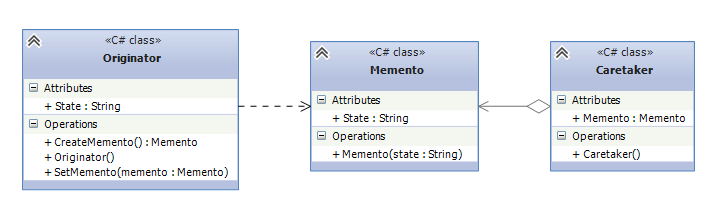
Когда использовать Memento?

Когда нужно сохранить состояние объекта для возможного последующего восстановления

Когда сохранение состояния должно проходить без нарушения принципа инкапсуляции

То есть ключевыми понятиями для данного паттерна являются сохранение внутреннего состояния и инкапсуляция, и важно соблюсти баланс между ними. Ведь, как правило, если мы не нарушаем инкапсуляцию, то состояние объекта хранится в объекте в приватных переменных. И не всегда для доступа к этим переменным есть методы или свойства с сеттерами и геттерами. Например, в игре происходит управление героем, все состояние которого заключено в нем самом - оружие героя, показатель жизней, силы, какие-то другие показатели. И нередко может возникнуть ситуация, сохранить все эти показатели во вне, чтобы в будущем можно было откатиться к предыдущему уровню и начать игру заново. В этом случае как раз и может помочь паттерн Хранитель.

С помощью диаграмм структуру паттерна можно изобразить следующим образом:



апрапро