1. Все паттерны проектирования обеспечивают возможности изменения некоторой части системы без воздействия на остальные ее части.
2. Выделите аспекты проектирования, которые могут изменяться и отделите их от тех, которые всегда остаются постоянными. Инкапсулируйте эти изменяющиеся составляющие кода так, чтобы в любой момент их можно было изменить или расширить без воздействия на постоянные составляющие.
3. Программируйте на уровне интерфейса, а не на уровне реализации.
4. Отдавайте предпочтение композиции перед наследованием. Поведение лучше не наследовать, а представлять правильно выбранным объектом (содержать этот самый объект).

**Паттерн «Стратегия» определяет семейство алгоритмов, инкапсулирует каждый из них и обеспечивает их взаимозаменяемость. Он позволяет модифицировать алгоритмы независимо от их использования на стороне клиента.**

*А прикольные у меня утки получились! Въехал в идею стратегии: я своему "уткоконструктору" (duck.h) подсовываю указатели на интерфейсы, а инициализирую эти указатели реальными классами, которые эти интерфейсы реализуют. И как бы эти классы не менялись, пользователь "уткоконструктора" ничего не заметит. Зато теперь у пользователя появилась возможность создавать много разных уток, используя одни и те же интерфейсы.*

**Патерн «Наблюдатель» определяет отношения «один ко многим» таким образом, что при изменении состояния одного объекта (субъект) происходит автоматическое оповещение/обновление всех остальных объектов в системе (подписчики/наблюдатели).**

1. Стремитесь к слабой связанности объектов внутри системы. Это дает гибкость и возможность хорошо адаптироваться к изменениям.