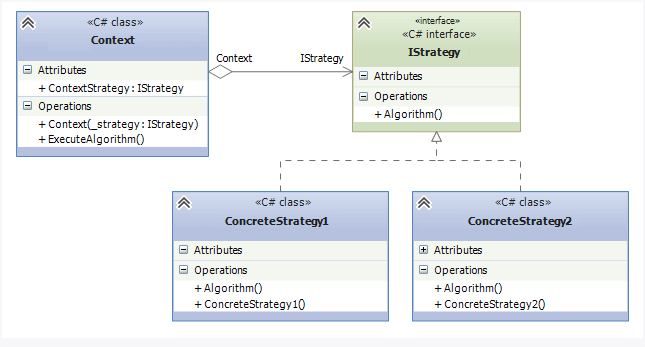
Паршаков Геннадий ПИ-15-1

Паттерн «Стратегия»

# Назначение и цель применения паттерна

**Стратегия** — это поведенческий паттерн проектирования, который определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс. После чего, алгоритмы можно взаимозаменять прямо во время исполнения программы.

# UML-диаграмма



* Интерфейс IStrategy, который определяет метод Algorithm(). Это общий интерфейс для всех реализующих его алгоритмов. Вместо интерфейса здесь также можно было бы использовать абстрактный класс.
* Классы ConcreteStrategy1 и ConcreteStrategy, которые реализуют интерфейс IStrategy, предоставляя свою версию метода Algorithm(). Подобных классов-реализаций может быть множество.
* Класс Context хранит ссылку на объект IStrategy и связан с интерфейсом IStrategy отношением агрегации.

# Области применения

Отделение процедуры выбора алгоритма от его реализации. Это позволяет сделать выбор на основании контекста.

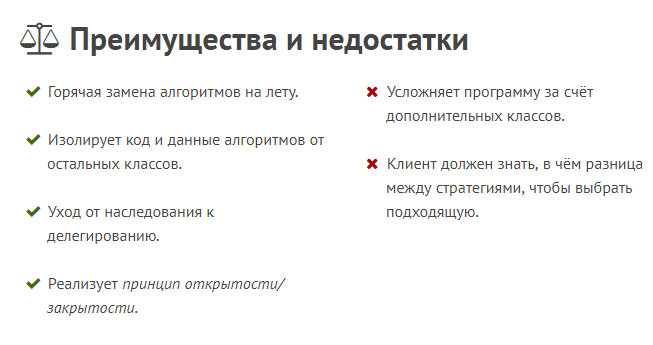
Например, программа – навигатор. Она прокладывает дорогу. Но человек может ехать на машине, идти пешком или пользоваться общественном транспортом. В результате мы получаем разные алгоритма прокладывания маршрута.

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



# Особенности паттерна



# Примеры реализации

public interface IStrategy

{

    void Algorithm();

}

public class ConcreteStrategy1 : IStrategy

{

    public void Algorithm()

    {}

}

public class ConcreteStrategy2 : IStrategy

{

    public void Algorithm()

    {}

}

public class Context

{

    public IStrategy ContextStrategy { get; set; }

    public Context(IStrategy \_strategy)

    {

        ContextStrategy = \_strategy;

    }

    public void ExecuteAlgorithm()

    {

        ContextStrategy.Algorithm();

    }

}

# Список источников

* 1. <https://refactoring.guru/ru/design-patterns/strategy>
  2. <https://metanit.com/sharp/patterns/3.1.php>
  3. <https://ru.wikipedia.org/wiki/Стратегия_(шаблон_проектирования>)