|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Демина Дарья Сергеевна  **ПОВЕДЕНЧЕСКИЙ ПАТТЕРН «СТРАТЕГИЯ»**  Лабораторная работа  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»   |  |  | | --- | --- | |  | Руководитель  Доцент кафедры информационных технологий в бизнесе.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  А.В. Кычкин | |

Пермь, 2018 год

# Назначение и цель применения паттерна Strategy

Стратегия – поведенческий паттерн, которые создает семейство схожих алгоритмов и помещает каждый из них в отдельный класс. Классы взаимозаменяемы, замену можно осуществлять во время выполнения программы. Причем замена алгоритма происходит независимо от объекта, который использует алгоритм.

Преимущества:

* Быстрая смена алгоритмов
* Уход от наследования к делегированию
* Изолирует код и данные алгоритмов от основных классов

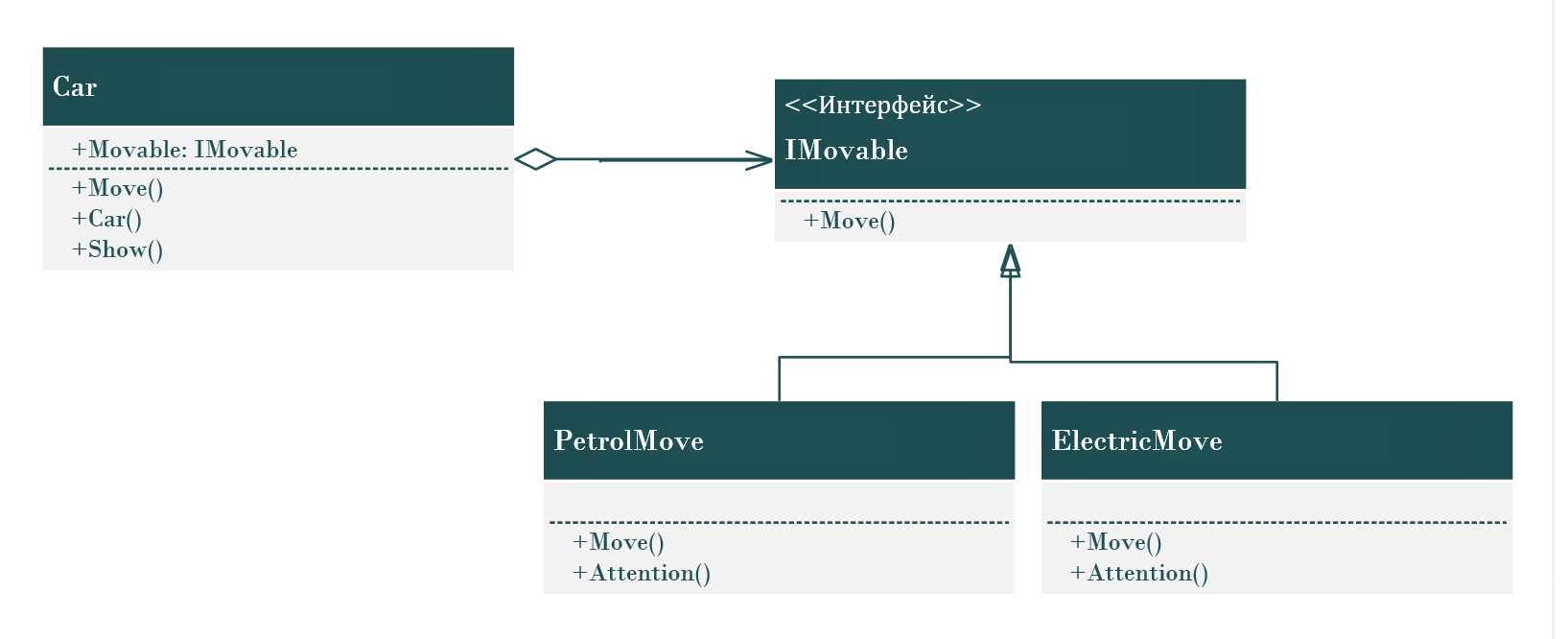
Недостатки:

* Усложняет код за счет дополнительных классов.
* Чтобы выбрать подходящую стратегию, клиент должен в них разбираться

Цели применения:

1. Вынести разные варианты поведения в отдельные классы и при необходимости их применять (когда несколько родственных классов отличаются поведением)
2. Изменение поведения объектов на стадии выполнения программы
3. Класс, применяющий определенную функциональность, ничего не знает о ее реализации

# UML-диаграмма



Car – хранит ссылку на объект IMovable, с его помощью решает поставленную задачу

IMovable – Определяет интерфейс для всех вариантов алгоритмов

PetrolMove, ElectricMove – классы, которые реализуют интерфейс IMovable

# Области применения

Архитектура Microsoft WDF основана на этом паттерне. У каждого объекта «драйвер» и «устройство» есть неизменяемая часть, вшитая в систему, в которой регистрируется изменяемая часть (стратегия), написанная в конкретной реализации. Изменяемая часть может быть и вовсе пустой, что даст ничего не делающий драйвер, но при этом способный участвовать в PnP и управлении питанием.

Библиотека ATL содержит в себе набор классов threading model, которые являются стратегиями (различными реализациями Lock/Unlock, которые потом используются основными классами системы). При этом в этих стратегиях используется статический полиморфизм через параметр шаблона, а не динамический полиморфизм через виртуальные методы.

# Особенности паттерна

Особенность паттерна заключается в том, что можно создавать неограниченное количество вариантов поведения, при этом не загромождая отдельную функцию.

# Пример реализации

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace strategyPattern

{

class Program

{

static void Main(string[] args)

{

Car auto = new Car(4, "Volvo", new PetrolMove());

auto.show();

auto.Move();

auto.Movable = new ElectricMove();

auto.show();

auto.Move();

Console.ReadLine();

}

}

interface IMovable

{

void Move();

}

class PetrolMove : IMovable

{

public void Move()

{

Console.WriteLine("Автомобиль работает на бензиновом топливе");

}

}

class ElectricMove : IMovable

{

public void Move()

{

Console.WriteLine("Автомобиль работает на электричестве");

}

}

class Car

{

protected int passengers; // кол-во пассажиров

protected string model; // модель автомобиля

public Car(int num, string model, IMovable mov)

{

this.passengers = num;

this.model = model;

Movable = mov;

}

public IMovable Movable { private get; set; }

public void Move()

{

Movable.Move();

}

public void show()

{

Console.WriteLine("Движение осуествляется на автомобиле {0}, в котором находится {1} человек.", model, passengers);

}

}

}