

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языка»

Отчет по домашней
«Dependency Injection. C#»

Выполнил:
студент группы РТ5-31Б:
Сахарова О.П

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2025 г.

Описание задания

Реализовать на предметной области Dependency Injection (внедрение зависимостей). На языке C#. Выбранная предметная область транспорт

Текст программы

```
using System;
using System.Collections.Generic;
// Абстрактный класс для двигателя
public abstract class Engine
{
    public abstract string Start();
    public abstract string Stop();
    public abstract int GetPower();
}
// Конкретные реализации двигателей
public class Benz_Engine : Engine
{
    private readonly int _power;
    public Benz_Engine(int power)
    {
        _power = power;
    }

    public override string Start()
    {
        return "Бензиновый двигатель запущен";
    }
    public override string Stop()
    {
        return "Бензиновый двигатель остановлен";
    }
    public override int GetPower()
    {
        return _power;
    }
}
public class Diesel_Engine : Engine
{
    private readonly int _power;
    public Diesel_Engine(int power)
    {
        _power = power;
    }
    public override string Start()
    {
        return "Дизельный двигатель запущен";
    }
    public override string Stop()
    {
        return "Дизельный двигатель остановлен";
    }
}
```

```

    }
    public override int GetPower()
    {
        return _power;
    }
}
public class Electric_Motor : Engine
{
    private readonly int _power;
    public Electric_Motor(int power)
    {
        _power = power;
    }
    public override string Start()
    {
        return "Электромотор активирован";
    }
    public override string Stop()
    {
        return "Электромотор деактивирован";
    }
    public override int GetPower()
    {
        return _power;
    }
}
// Абстрактный класс для транспорта
public abstract class Transport
{
    protected readonly Engine _engine;
    protected readonly string _model;
    protected Transport(Engine engine, string model)
    {
        _engine = engine;
        _model = model;
    }
    public abstract string Move();
    public abstract string StopMoving();
    public virtual string GetInfo()
    {
        return $"Модель: {_model}, Мощность: {_engine.GetPower()} л.с.";
    }
}
public class Car : Transport
{
    private readonly int _doorsCount;
    public Car(Engine engine, string model, int doorsCount)
        : base(engine, model)
    {
        _doorsCount = doorsCount;
    }
    public override string Move()

```

```

    {
        return $"Автомобиль '{_model}' начал движение. {_engine.Start()}";
    }
    public override string StopMoving()
    {
        return $"Автомобиль '{_model}' остановился. {_engine.Stop()}";
    }
    public override string GetInfo()
    {
        return base.GetInfo() + $", Количество дверей: {_doorsCount}";
    }
}
public class Truck : Transport
{
    private readonly double _cargoCapacity;
    public Truck(Engine engine, string model, double cargoCapacity)
        : base(engine, model)
    {
        _cargoCapacity = cargoCapacity;
    }
    public override string Move()
    {
        return $"Грузовик '{_model}' начал движение. {_engine.Start()}";
    }
    public override string StopMoving()
    {
        return $"Грузовик '{_model}' остановился. {_engine.Stop()}";
    }
    public override string GetInfo()
    {
        return base.GetInfo() + $", Грузоподъемность: {_cargoCapacity} тонн";
    }
}
public class Bus : Transport
{
    private readonly int _passengerCapacity;
    public Bus(Engine engine, string model, int passengerCapacity)
        : base(engine, model)
    {
        _passengerCapacity = passengerCapacity;
    }
    public override string Move()
    {
        return $"Автобус '{_model}' начал движение. {_engine.Start()}";
    }
    public override string StopMoving()
    {
        return $"Автобус '{_model}' остановился. {_engine.Stop()}";
    }
    public override string GetInfo()
    {
        return base.GetInfo() + $", Вместимость: {_passengerCapacity} пассажиров";
    }
}

```

```

    }
}
public class DIContainer
{
    private readonly Dictionary<Type, Func<object>> _registrations = new();
    public void Register<T>(Func<T> factory)
    {
        _registrations[typeof(T)] = () => factory();
    }
    public T Resolve<T>()
    {
        if (_registrations.TryGetValue(typeof(T), out var factory))
        {
            return (T)factory();
        }
        throw new InvalidOperationException($"Тип {typeof(T)} не зарегистрирован");
    }
}

public class Transport_moving
{
    private readonly Transport _transport;
    // Внедрение зависимости через конструктор
    public Transport_moving(Transport transport)
    {
        _transport = transport;
    }

    public void OperateTransport()
    {
        Console.WriteLine(_transport.GetInfo());
        Console.WriteLine(_transport.Move());
        Console.WriteLine("Транспорт в пути...");
        Console.WriteLine(_transport.StopMoving());
        Console.WriteLine();
    }
}

class Program
{
    static void Main()
    {
        // Создание контейнера
        var container = new DIContainer();
        container.Register(() => new Benz_Engine(150));
        container.Register(() => new Diesel_Engine(300));
        container.Register(() => new Electric_Motor(200));

        // Пример 1: Автомобиль с бензиновым двигателем
        var carEngine = container.Resolve<Benz_Engine>();
        var car = new Car(carEngine, "Toyota Camry", 4);
        var carService = new Transport_moving(car);
        carService.OperateTransport();
    }
}

```

```

// Пример 2: Грузовик с дизельным двигателем
var truckEngine = container.Resolve<Diesel_Engine>();
var truck = new Truck(truckEngine, "Volvo FH16", 20.5);
var truckService = new Transport_moving(truck);
truckService.OperateTransport();

//Внедрение вручную (без контейнера)
var hybridEngine = new Benz_Engine(120);
var hybridCar = new Car(hybridEngine, "Toyota Prius", 5);
// Внедрение через свойство
var manualService = new Transport_moving(hybridCar);
manualService.OperateTransport();
}
}

```

Результаты

```

Модель: Toyota Camry, Мощность: 150 л.с., Количество дверей: 4
Автомобиль 'Toyota Camry' начал движение. Бензиновый двигатель запущен
Транспорт в пути...
Автомобиль 'Toyota Camry' остановился. Бензиновый двигатель остановлен

Модель: Volvo FH16, Мощность: 300 л.с., Грузоподъемность: 20,5 тонн
Грузовик 'Volvo FH16' начал движение. Дизельный двигатель запущен
Транспорт в пути...
Грузовик 'Volvo FH16' остановился. Дизельный двигатель остановлен

Модель: Toyota Prius, Мощность: 120 л.с., Количество дверей: 5
Автомобиль 'Toyota Prius' начал движение. Бензиновый двигатель запущен
Транспорт в пути...
Автомобиль 'Toyota Prius' остановился. Бензиновый двигатель остановлен

```