**Отчёт по лабораторной работе №4: Паттерн "Фиктивная служба"**

**Описание проблемы предметной области:**

В повседневной практике многие компании, предоставляющие услуги доставки, нуждаются в программных решениях, позволяющих рассчитывать стоимость доставки товаров в зависимости от их характеристик и расстояния. Однако при разработке и тестировании подобных систем возникает проблема: невозможность протестировать логику интерфейса без наличия доступа к реальному серверу или API, или же ответ с сервера занимает слишком много времени. Это замедляет процесс разработки, поскольку усложняет тестирование и увеличивает сложность отладки.

**Решение:**

Для решения этой проблемы в проекте применён архитектурный паттерн 'Фиктивная служба' (Mock Object). Суть паттерна заключается в создании поддельной реализации интерфейса сервиса доставки, которая имитирует реальное поведение. Это позволяет производить тестирование пользовательского интерфейса без подключения к серверу, а также ускоряет разработку за счёт быстрого получения предсказуемого результата.

Код с паттерном:  
using System;

namespace DeliveryServiceExample

{

// Интерфейс службы доставки

public interface IDeliveryService

{

DeliveryResult Calculate(DeliveryParams parameters);

}

// Параметры доставки

public class DeliveryParams

{

public double Weight { get; set; }

public double Length { get; set; }

public double Width { get; set; }

public double Height { get; set; }

public double Distance { get; set; }

}

// Результат доставки

public class DeliveryResult

{

public double Price { get; set; }

public int EstimatedDays { get; set; }

}

// Реальная служба доставки (конкретная реализация)

public class RealDeliveryService : IDeliveryService

{

public DeliveryResult Calculate(DeliveryParams parameters)

{

double volume = parameters.Length \* parameters.Width \* parameters.Height;

double price = parameters.Weight \* 10 + volume \* 0.05 + parameters.Distance \* 0.1;

int days = (int)Math.Ceiling(parameters.Distance / 500.0);

return new DeliveryResult

{

Price = Math.Round(price, 2),

EstimatedDays = days

};

}

}

// Фиктивная служба доставки (для тестов)

public class MockDeliveryService : IDeliveryService

{

public DeliveryResult Calculate(DeliveryParams parameters)

{

return new DeliveryResult

{

Price = 999,

EstimatedDays = 3

};

}

}

// Клиентский код

class Program

{

static void Main(string[] args)

{

// Переключение между реалной и фиктивной службой

Console.WriteLine("Использовать Mock-сервис? (yes/no): ");

string input = Console.ReadLine();

IDeliveryService service = input == "yes"

? new MockDeliveryService()

: new RealDeliveryService();

// Ввод данных

var deliveryParams = new DeliveryParams

{

Weight = 5,

Length = 40,

Width = 30,

Height = 20,

Distance = 1000

};

// Расчет и вывод результата

var result = service.Calculate(deliveryParams);

Console.WriteLine($"Стоимость доставки: {result.Price} руб.");

Console.WriteLine($"Примерный срок: {result.EstimatedDays} дней.");

}

}

}

Использование паттерна "Фиктивная служба" (Mock Object) позволяет отделить бизнес-логику от конкретных реализаций внешних сервисов, благодаря чему упрощается тестирование, улучшается модульность и обеспечивается гибкость архитектуры. Такой подход позволяет легко подменять реальные службы на фиктивные в тестовой среде, не влияя на остальной код, и тем самым повышает надёжность системы, ускоряет разработку.

**Диаграмма классов**

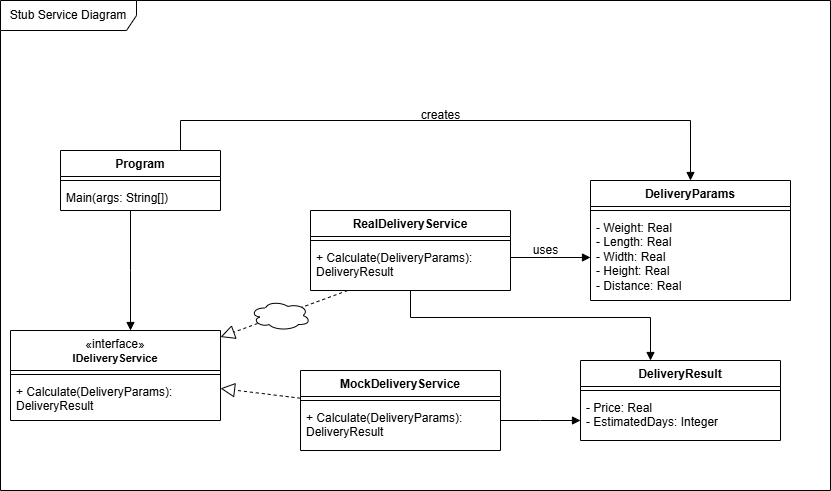


Рисунок 1 – Паттерн “Фиктивная служба” в архитектуре приложения.

**Вывод**

Применение паттерна 'Фиктивная служба' позволило разделить бизнес-логику и реализацию пользовательского интерфейса, а также обеспечило удобство тестирования без зависимости от сетевого окружения. Внедрение интерфейса и mock-класса значительно упростило отладку и сделало возможным быстрое переключение между реальным и тестовым режимом работы.