**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Инженерная школа информационных технологий и робототехники Отделение информационных технологий**

**Информатика и вычислительная техника**

Отчет по Лабораторной работе 4

Создание логики приложения

Выполнил

Студент группы 8В23 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Балашов А.Д.

Проверил

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Хамухин А. А.

Томск 2024

**Цель работы:**

Познакомится с разбиением приложения на слои. Освоить основные принципы объектно-ориентированного проектирования. Научиться применять шаблон Factory Method.

**Задание**

Для созданных классов предметной области в предыдущей лабораторной работе необходимо создать логику приложения в соответствии с вариантом задания (см. приложение). При выполнении лабораторной работы необходимо соблюдать следующие правила:

1. Необходимо создавать логику приложения в отдельных классах;
2. Классы логики приложения не должны зависеть от интерфейса пользователя, их необходимо выделить в отдельный пакет либо библиотеку классов;
3. Для использования логики приложения в интерфейсе пользователя необходимо вызывать методы классов, содержащих логику

**Процесс выполнения**

Создал отдельную DLL для реализации необходимых для этого задания методов. В соответствии с пунктом номер 1 в задании, создаю отдельный класс, в котором реализую основную функцию, которая считает стоимость маршрута. (рисунок 1)

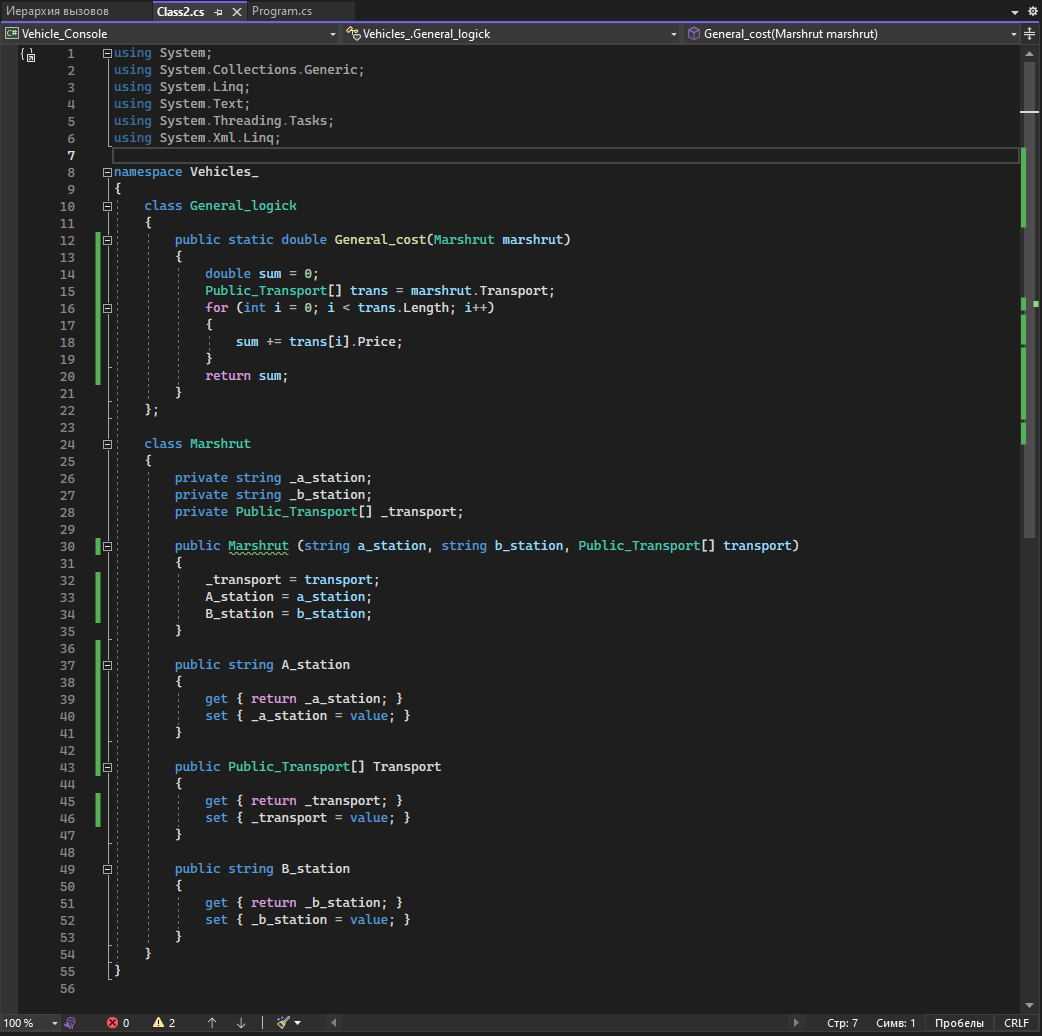
****

Рисунок 1 – Вспомогательная DLL

Создаю абстрактный тип данных, под названием «Marshrut», описывающий сущность понятия маршрут. Создаю файл консольного приложения. И добавляю ссылку на написанные ранее библиотеки. (рисунок 2)

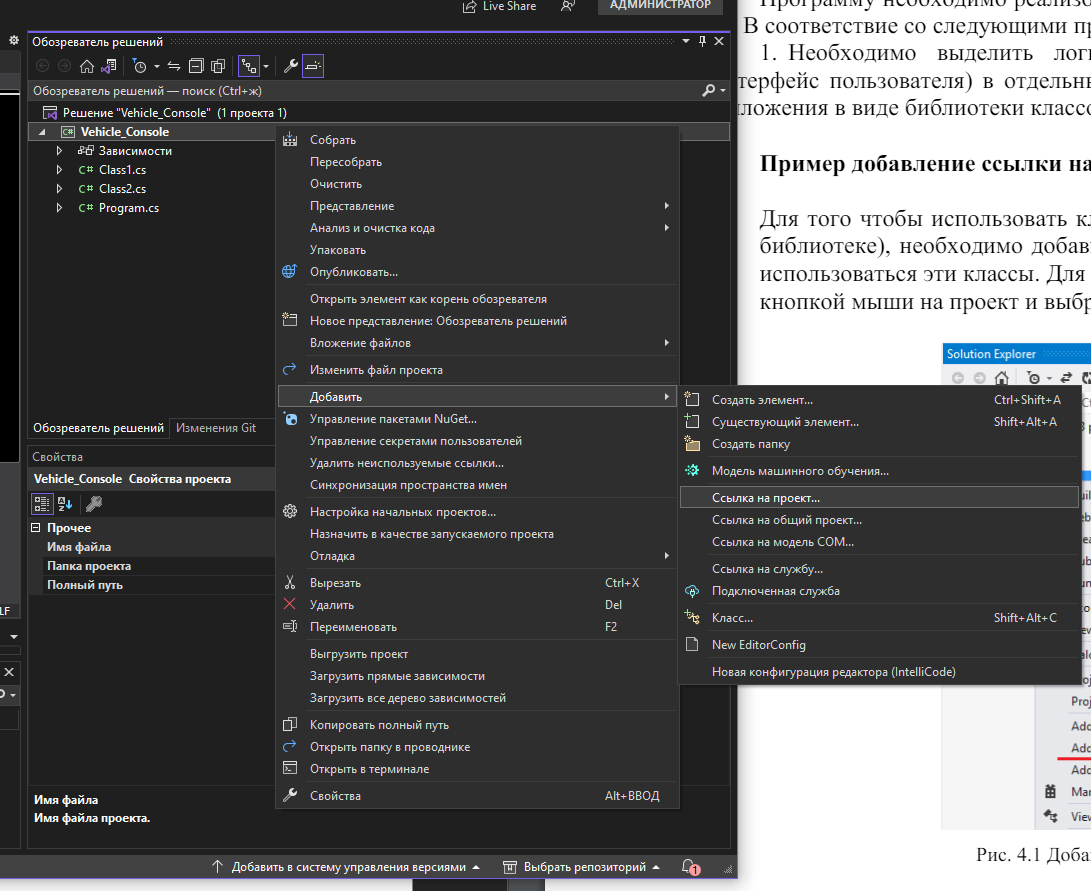
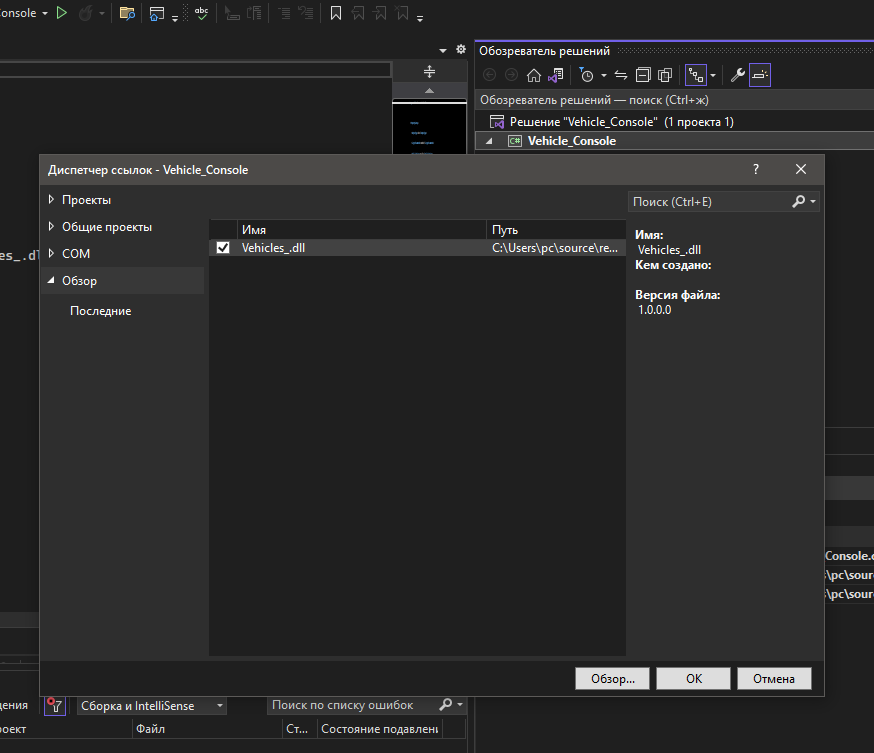


Рисунок 2 – Добавление ссылки на библиотеку

Затем связываю все воедино, создаю объекты описанных классов, вывожу нужную информацию в консоль. Реализовано это в отдельном от остальных методов классе, поэтому пункт 3 задания выполнен.

**Ответы на контрольные вопросы**

1. **Принципы разбиения приложения на модули** - это общепринятые подходы к организации кода при проектировании приложения. Основные принципы включают в себя:
   * **Принцип единственной ответственности (Single Responsibility Principle)**: Каждый модуль или класс должен быть ответственен только за одну часть функциональности программы.
   * **Принцип открытости/закрытости (Open/Closed Principle)**: Программные сущности должны быть открыты для расширения, но закрыты для модификации. Это означает, что изменения должны вноситься путем добавления нового кода, а не изменения существующего.
   * **Принцип подстановки Барбары Лисков (Liskov Substitution Principle)**: Объекты в программе должны быть заменяемыми экземплярами их подтипов без изменения корректности программы.
   * **Принцип инверсии зависимостей (Dependency Inversion Principle)**: Модули верхнего уровня не должны зависеть от модулей нижнего уровня. Оба должны зависеть от абстракций. Инверсия зависимостей также включает в себя предпочтение использования интерфейсов вместо конкретных классов при определении зависимостей.
2. **Интерфейс** - это абстрактный тип данных, представляющий собой набор методов, которые должны быть реализованы классом, который этот интерфейс имплементирует. Интерфейсы определяют только сигнатуры методов, но не их реализацию. Они используются для определения контрактов между различными частями программы.
3. **Отличие абстрактного класса от интерфейса**:
   * Абстрактный класс может содержать как абстрактные методы, так и обычные методы с реализацией, в то время как интерфейс содержит только сигнатуры методов без реализации.
   * Класс может наследовать только один абстрактный класс, но реализовать несколько интерфейсов.
   * Абстрактные классы могут содержать поля (переменные класса), тогда как интерфейсы могут содержать только константы.
4. **Два принципа проектирования**:
   * **Принцип единственной ответственности (Single Responsibility Principle)**: Как упомянуто ранее, каждый модуль или класс должен быть ответственен только за одну часть функциональности программы. Это помогает сделать код более понятным, тестируемым и поддерживаемым.
   * **Принцип открытости/закрытости (Open/Closed Principle)**: Также упомянутый ранее, принцип открытости/закрытости поддерживает возможность расширения функциональности программы без изменения уже существующего кода.
5. **Шаблон Factory Method** - это порождающий шаблон проектирования, который позволяет делегировать создание объектов подклассам. Фабричный метод определяет метод, который должен быть использован для создания объектов, но оставляет реализацию этого метода подклассам. Это позволяет легко изменять тип создаваемых объектов без изменения кода, который их использует.

**Вывод**

В ходе выполнения лабораторной работы был получен опыт практической и теоретической работы в составлении проектов типа Factory Method, написан соответствующий заданию проект.

**Список используемых источников:**

1. Фаулер М. Архитектура корпоративных приложений, Вильямс, 2006 г.
2. Блинов И.М., Романчик В.С. Java. Промышленное программирование: практ. пособие — Минск: УниверсалПресс, 2007. — 2007.
3. Блинов И.Н., Романчик В.С Java. Методы программирования, Четыре четверти, 2013
4. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектноориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2007. — 266 с.