

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.02

Выполнила: Зыкова Софья Александровна

Группа: ПР-21

Преподаватель: Мирошниченко Г.В.

2025

Содержание

[1. **Задание №3 приложение «Учет качества машин»** 3](#_qnuy7idubhzs)

[1.1](#_y8u24gz4p0y3) Описание задачи 3

[1.2](#_67g23rmf2iv) Структура проекта 3

[1.3 Описание разработанных функций 3](#_yzzrow6oydfb)

[1.4 Алгоритм решения 3](#_fygxzbbatt33)

[1.5 Используемые библиотеки 3](#_ixhetek99xb5)

[1.6 Тестовые случаи 3](#_uwy98cj82ux4)

[1.7 Используемые инструменты 3](#_s9f8fz2djv28)

[1.8 Описание пользовательского интерфейса 3](#_vlw01m52jfps)

[1.9 Приложение (pr screen экранов) 3](#_wp62y88opdbt)

# 1. **Задание №3 приложение «Учет качества машин»**

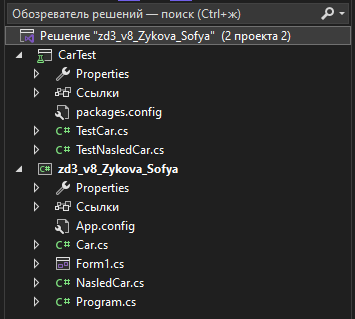
## **Описание задачи**

Требуется разработать программу для расчета показателя Qp автомобиля.

Требования к задаче:

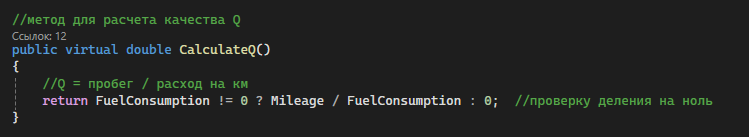
* Пользователь должен ввести:
* Год выпуска автомобиля (целое число, например, 2015)
* Пробег (в км, число с плавающей точкой, например, 150000,5)
* Расход топлива на км (в литрах, число с плавающей точкой, например, 0,08)
* Программа должна вычислить:
* Q = пробег / расход на км
* Qp = Q \* 1,15 \* год выпуска
* Показать пользователю:
* Промежуточное значение Q
* Итоговое значение Qp
* Доп. требования:
* Добавление машин в список
* Удаление машин из списка
* Поиск машины

## **Структура проекта**



* CarTest - Библиотека классов, хранит в себе два класса, выполняющих модульные тесты.
* zd3\_v8\_ZykovaSofya - Приложение Windows Forms, приложение, в которое вносим данные о машинах и рассчитываем учет качества машин.

## **1.3 Описание разработанных функций**



#### CalculateQ() (базовый метод)

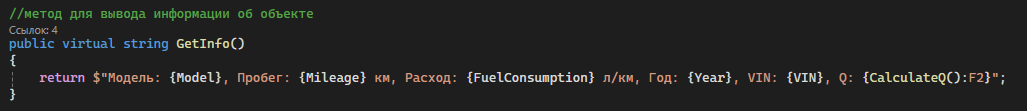
Назначение:  
Вычисляет базовый показатель качества автомобиля Q по формуле:  
Q = пробег / расход на км

Входные данные:

* Использует свойства класса:
  + Mileage (пробег, км)
  + FuelConsumption (расход топлива, л/км)

Выходные данные:

* Возвращает double – значение показателя Q
* Если расход равен 0, возвращает 0 (защита от деления на ноль)



#### GetInfo() (базовый метод)

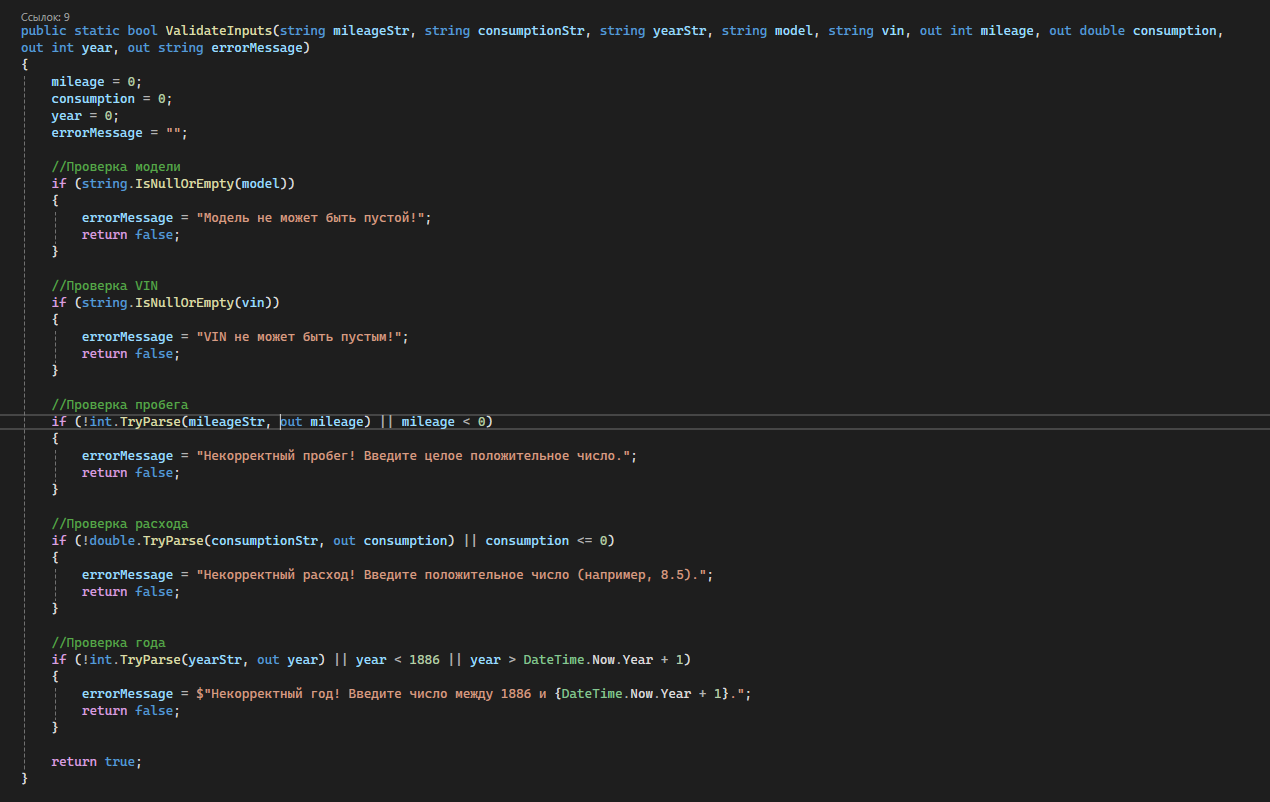
Назначение:  
Формирует строку с информацией об автомобиле.

Входные данные:

* Использует свойства класса:
  + Model (модель авто)
  + Mileage (пробег)
  + FuelConsumption (расход)
  + Year (год выпуска)
  + VIN (идентификационный номер)
  + Результат CalculateQ()

Выходные данные:

* Возвращает string в формате:  
  "Модель: {Model}, Пробег: {Mileage} км, Расход: {FuelConsumption} л/км, Год: {Year}, VIN: {VIN}, Q: {CalculateQ():F2}"



#### ValidateInputs() (статический метод)

Назначение:  
Проверяет корректность введенных пользователем данных.

Входные данные:

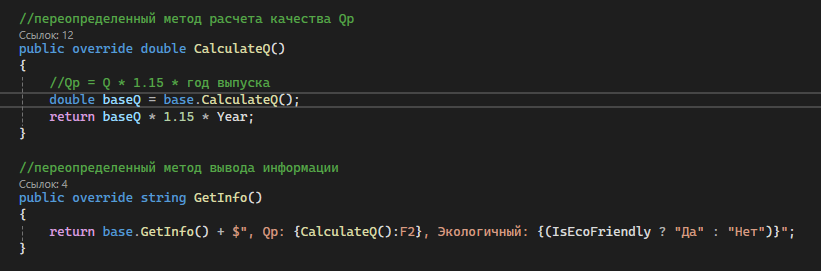
* Параметры:
  + mileageStr (пробег, строка)
  + consumptionStr (расход, строка)
  + yearStr (год выпуска, строка)
  + model (название модели)
  + vin (VIN-номер)
* Выходные параметры (out):
  + mileage (пробег, число)
  + consumption (расход, число)
  + year (год выпуска, число)
  + errorMessage (сообщение об ошибке)

Выходные данные:

* Возвращает bool:
  + true – если все данные корректны
  + false – если есть ошибки (с записью в errorMessage)

Проверки:

1. Модель и VIN не должны быть пустыми.
2. Пробег – целое положительное число.
3. Расход – положительное число (дробное).
4. Год выпуска – от 1886 (первый автомобиль) до текущего года + 1.



#### Переопределенные методы CalculateQ() и GetInfo() (в наследнике NasledCar)

Назначение:  
Расширяют функциональность базового класса:

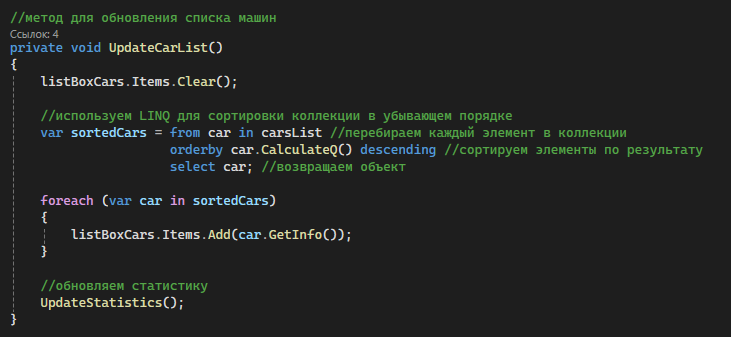
* CalculateQ() вычисляет Qp = Q × 1.15 × год выпуска
* GetInfo() добавляет информацию о Qp и экологичности.

Входные данные:

* Используют свойства базового класса + IsEcoFriendly (флаг экологичности).

Выходные данные:

* CalculateQ(): double (Qp)
* GetInfo(): string (информация + Qp и экологичность).



#### UpdateCarList()

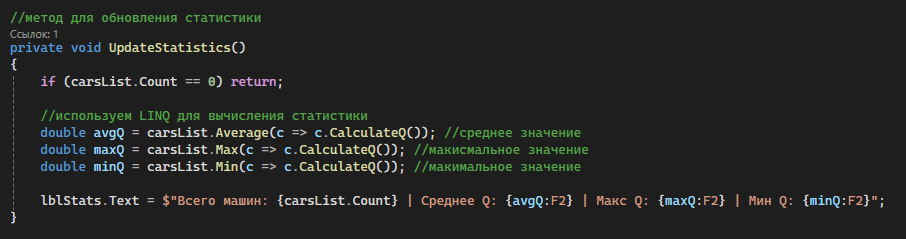
Назначение:  
Обновляет список автомобилей в интерфейсе, сортируя по убыванию Q.

Входные данные:

* Коллекция carsList (список автомобилей).

Логика:

1. Очищает listBoxCars.
2. Сортирует авто по Q (по убыванию, через LINQ).
3. Добавляет записи в listBoxCars через GetInfo().
4. Вызывает UpdateStatistics().



#### UpdateStatistics()

Назначение:  
Вычисляет и отображает статистику по коллекции автомобилей.

Входные данные:

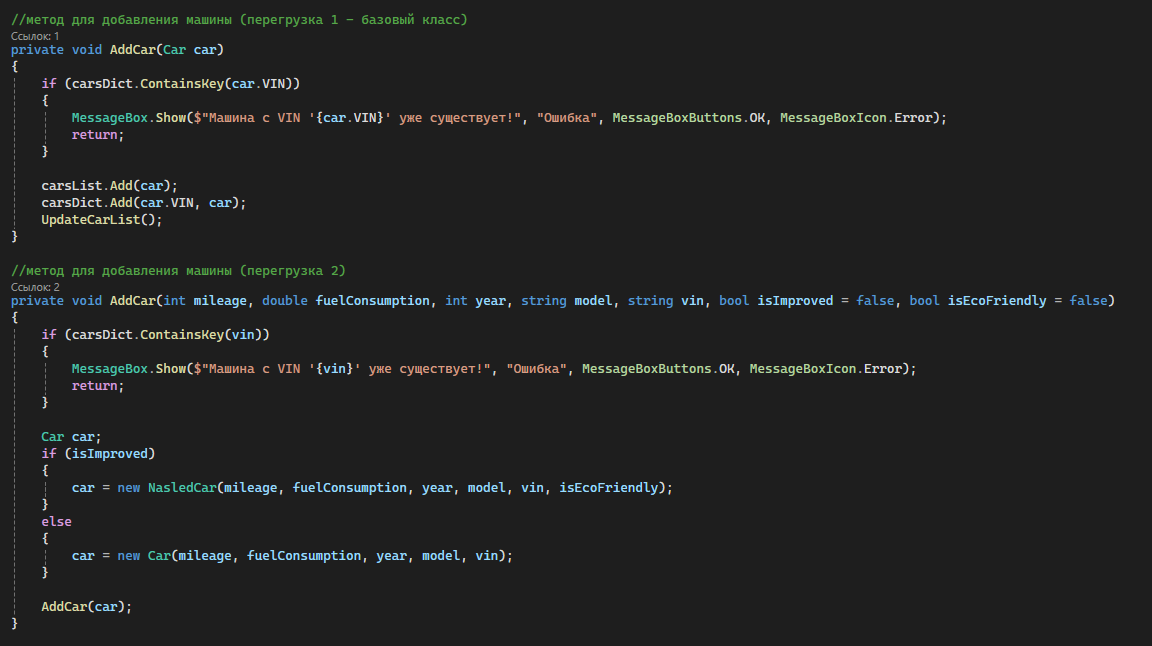
* Коллекция carsList.

Выходные данные:

* Обновляет lblStats, выводя:
  + Количество машин.
  + Среднее, максимальное и минимальное значение Q.

Используемые LINQ-методы:

* Average() – среднее значение.
* Max() / Min() – максимум и минимум.

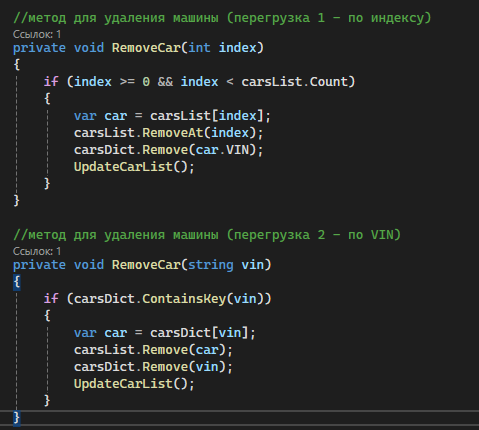


#### Перегруженные методы AddCar()

Назначение:  
Добавляют автомобиль в коллекцию.

Варианты:

1. Принимает объект Car:
   * Проверяет уникальность VIN.
   * Добавляет в carsList и carsDict.
2. Принимает параметры:
   * Создает объект Car или NasledCar (если isImproved = true).
   * Проверяет VIN и вызывает первый вариант.



#### Перегруженные методы RemoveCar()

Назначение:  
Удаляют автомобиль из коллекции.

Варианты:

1. По индексу:
   * Удаляет из carsList и carsDict.
2. По VIN:
   * Удаляет по ключу в словаре carsDict.

## 1.4 **Алгоритм решения**

## 1.5 **Используемые библиотеки**

* System.Collections.Generic (работа с коллекциями)
* System.Linq (сортировка и статистика)
* System.Windows.Forms (интерфейс)

## 1.6 **Тестовые случаи**

#### 1. Тест: Добавление обычного автомобиля

Назначение: Проверить корректность добавления автомобиля в коллекцию.  
Входные данные:

* Пробег: 100000
* Расход: 0.1 (л/км)
* Год выпуска: 2015
* Модель: Toyota Camry
* VIN: ABC123456789  
  Ожидаемый результат:
* Автомобиль добавлен в carsList и carsDict.
* Обновлен listBoxCars с новой записью.

#### 2. Тест: Добавление автомобиля с дублирующимся VIN

Назначение: Проверить обработку попытки добавления автомобиля с существующим VIN.  
Входные данные:

* VIN: ABC123456789 (уже существует)  
  Ожидаемый результат:
* Вывод сообщения об ошибке:

*Машина с VIN 'ABC123456789' уже существует!*

#### 3. Тест: Валидация некорректных данных

Назначение: Проверить обработку невалидных входных данных.  
Входные данные:

* Пробег: -50000 (отрицательное число)
* Расход: 0 (нулевой расход)
* Год выпуска: 3000 (некорректный год)  
  Ожидаемый результат:
* Сообщения об ошибках:
  + Для пробега:

*Некорректный пробег! Введите целое положительное число.*

* + Для расхода:

*Некорректный расход! Введите положительное число.*

* + Для года:

*Некорректный год! Введите число между 1886 и 2024.*

#### 4. Тест: Удаление автомобиля по индексу

Назначение: Проверить удаление автомобиля из коллекции по выбранному индексу в listBoxCars.  
Предварительные условия:

* В коллекции есть автомобиль с индексом 0.  
  Действие:
* Вызов RemoveCar(0).  
  Ожидаемый результат:
* Автомобиль удален из carsList и carsDict.
* listBoxCars обновлен (пуст).
* Статистика отображает:

*Всего машин: 0*

#### 5. Тест: Удаление автомобиля по VIN

Назначение: Проверить удаление автомобиля по VIN.  
Входные данные:

* VIN: XYZ987654321 (существует в коллекции).  
  Ожидаемый результат:
* Автомобиль удален из коллекций.
* Обновлен интерфейс.

#### 6. Тест: Поиск автомобиля по модели

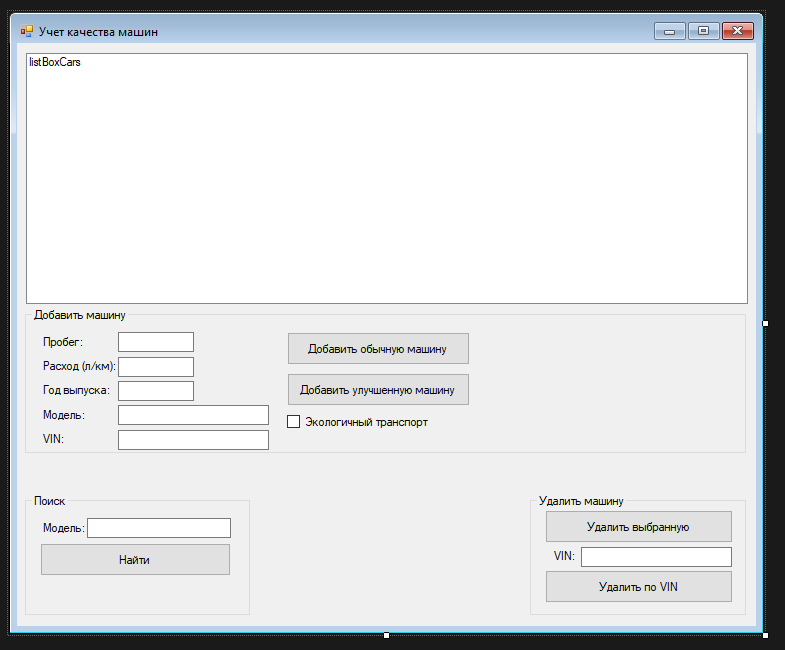
Назначение: Проверить фильтрацию списка по названию модели.  
Предварительные условия:

* В коллекции есть автомобили:
  + Toyota Camry
  + Honda Accord  
    Входные данные:
* Поисковый запрос: camry (без учета регистра).  
  Ожидаемый результат:
* В listBoxCars отображается только Toyota Camry.

## 1.7 **Используемые инструменты**

Для разработки использовался язык программирования С#. Среда разработки Visual Studio. Фреймворк: WinForms (C#, Windows)

## 1.8 **Описание пользовательского интерфейса**



#### 1. Блок "Добавить машину"

Поля ввода:

* Пробег: Числовое поле (например, 150000 км).
* Расход (л/км): Числовое поле (например, 0.08 л/км).
* Год выпуска: Числовое поле (например, 2015).
* Модель: Текстовое поле (например, Toyota Camry).
* VIN: Текстовое поле (уникальный идентификатор, например, XTA210997654321).

Кнопки:

* Добавить обычную машину – добавляет базовый объект Car.
* Добавить улучшенную машину – добавляет объект NasledCar (наследник Car).

Чекбокс:

* Экологичный транспорт – если отмечен, в листе будет показано как “Экологичный: Да”

Описание работы:

1. Пользователь заполняет все поля.
2. Выбирает тип машины (обычная/улучшенная).
3. Нажимает кнопку Добавить → данные валидируются:
   * Проверка на пустые поля.
   * Проверка формата (пробег > 0, расход > 0, год от 1886 до текущего).
   * Проверка уникальности VIN.
4. Если данные корректны – машина добавляется в listBoxCars и коллекции (carsList, carsDict).

#### 2. Блок "Поиск"

Поле ввода:

* Модель: Текстовое поле для поиска (например, Toyota).

Кнопка:

* Найти – фильтрует listBoxCars, оставляя только машины, в названии модели которых есть введенная подстрока (без учета регистра).

Описание работы:

1. Пользователь вводит текст в поле Модель.
2. Нажимает Найти → в listBoxCars отображаются только совпадения (например, Toyota Camry, Toyota RAV4).
3. Если поле пустое – выводится полный список.

#### 3. Блок "Удалить машину"

Варианты удаления:

1. Удалить выбранную:
   * Удаляет машину, выделенную в listBoxCars (по индексу).
   * Если ничего не выбрано – выводится сообщение:

*Выберите машину для удаления!*

1. Удалить по VIN:
   * Поле VIN: Текстовое поле для ввода идентификатора.
   * Кнопка Удалить по VIN – удаляет машину с указанным VIN.
   * Если VIN не существует – выводится сообщение:

*Машина с таким VIN не найдена!*

Описание работы:

* При удалении обновляются коллекции (carsList, carsDict) и интерфейс (listBoxCars, статистика).

#### 4. Список машин (listBoxCars)

Отображает:

* Список всех добавленных машин в формате:

Модель: …, Пробег: … км, Расход: … л/км, Год: …, VIN: …, Q: …

* Для улучшенных машин добавляется пометка:

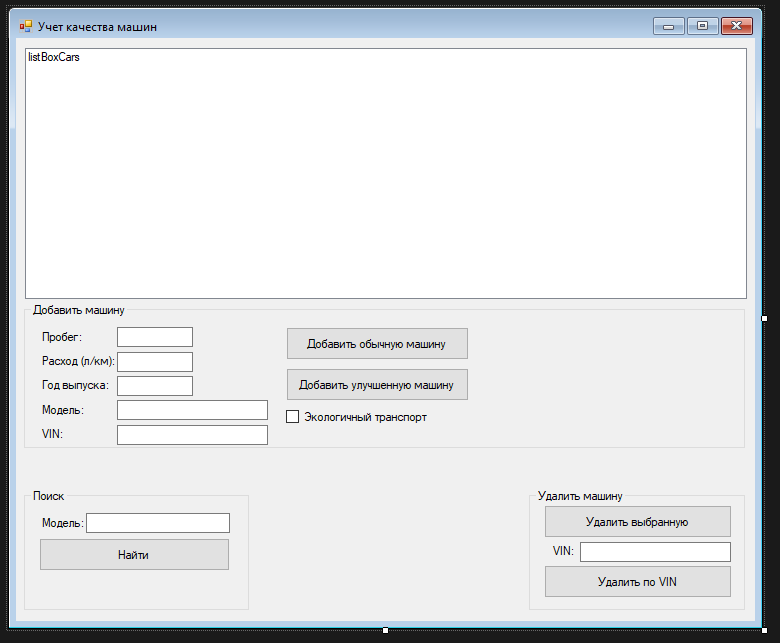
*Экологичный: Да/Нет*

#### 5. Статистика

Отображается под списком машин (не показана в схеме, но подразумевается):

*Всего машин: {Count} | Среднее Q: {avgQ:F2} | Макс Q: {maxQ:F2} | Мин Q: {minQ:F2}*

## 1.**9 Приложение (pr screen экранов)**

****