МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №53

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | А. К. Малышев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЁТ ПО КУРСОВОЙ РАБОТЕ  «Разработка архитектуры классов для просмотра характеристики автомобилей»  по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5836 |  |  |  | Д. А. Храмченко |  |
|  |  |  | подпись, дата |  | инициалы, фамилия |  |

Санкт-Петербург 2019 г.

Оглавление

## Анализ поставленной задачи: функциональность разрабатываемой библиотеки и назначение ее использования

## Диаграмма классов

## Описание программного кода классов

## Тестирование функциональности классов

## Руководство пользователя

## Заключение

## Библиографический список

## Приложения

## Анализ поставленной задачи: функциональность разрабатываемой библиотеки и назначение ее использования.

Требуется разработать библиотеку классов для просмотра характеристик грузового и легкового транспорта. Для этого необходимо определить иерархию классов, которой может воспользоваться пользователь: транспорт, легковой автомобиль, грузовой автомобиль. Модель должна позволять пользователю создавать, просматривать и удалять легковые и грузовые автомобили.

Объекты класса «легковой автомобиль» должны иметь следующие поля:

* Бренд
* Страна происхождения бренда
* Дата создания бренда
* Цвет
* Цена
* Масса
* Год выпуска
* Картинка
* Тип кузова

Объекты класса «легковой автомобиль» должны иметь следующие методы:

* Строковое представление объекта
* Корректность объекта
* Конструктор без параметров
* Конструктор с параметрами

Объекты класса «грузовой автомобиль» должны иметь такие поля:

* Бренд
* Страна происхождения бренда
* Дата создания бренда
* Цвет
* Цена
* Масса
* Год выпуска
* Картинка
* Длина прицепа

Объекты класса «грузовой автомобиль» иметь методы:

* Строковое представление объекта
* Корректность объекта
* Конструктор без параметров
* Конструктор с параметрами
* Вычисление максимальной массы грузы
* Вычисление максимальной длины груза

Для работы с этими объектами необходимо создать абстрактный базовый класс «Транспорт», в котором будет находиться информация о бренде, основные характеристики автомобиля, картинка, методы строкового представления и проверки корректности объектов. Также нужно создать и классы для хранения информации о бренде и основных характеристик автомобиля.

В классах описать следующие элементы:

* Поля;
* Индексатор для доступа к свойству по имени свойства (для легковых);
* Конструкторы с параметрами и конструкторы по умолчанию;
* Свойства для установки и получения всех характеристик, информации об автомобиле и картинки;
* Методы строкового представления объектов, методы проверки корректности;
* Методы для вычисления максимальной массы и максимальной длины груза (для грузового);

## Диаграмма классов.

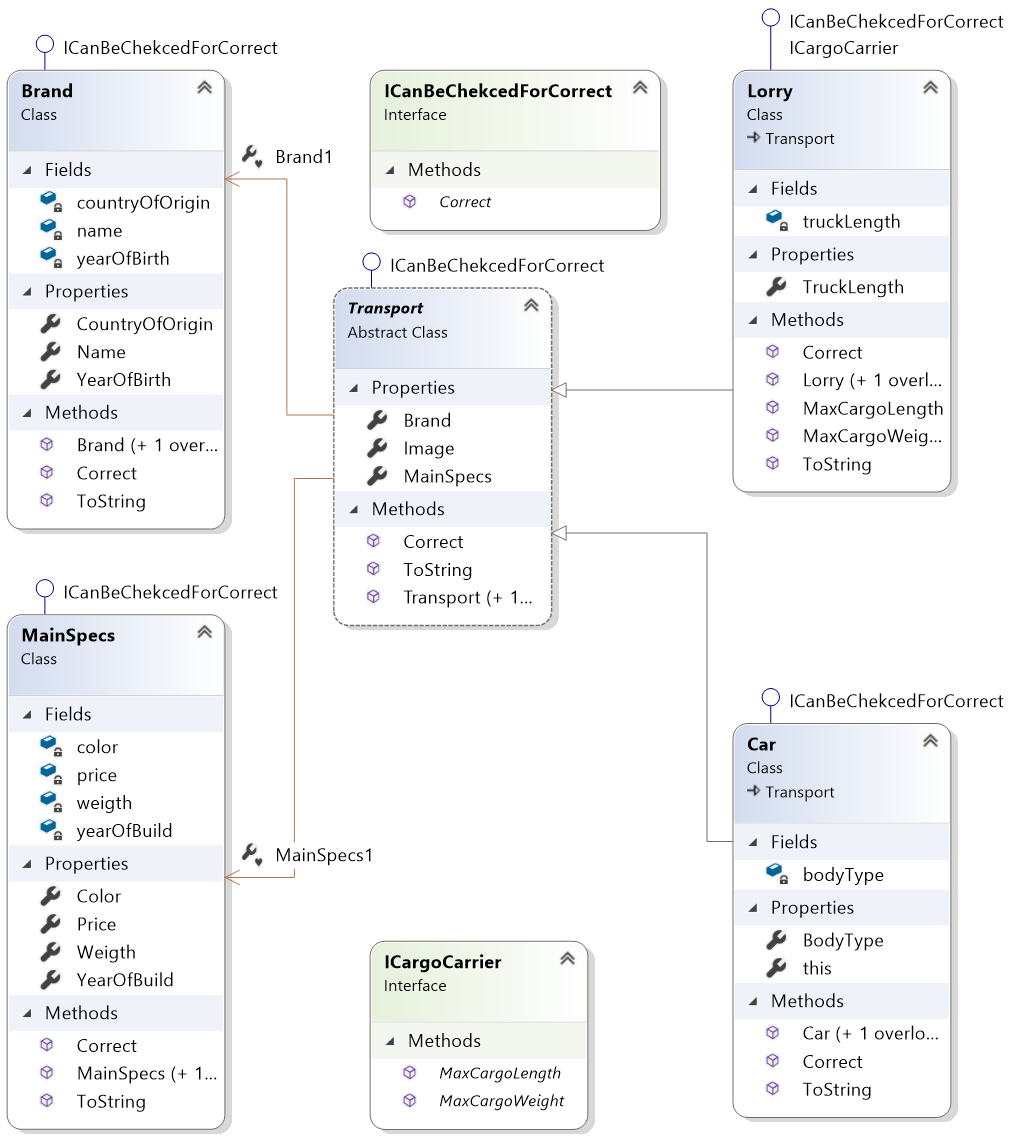
****

Рисунок 1 – диаграмма классов

## Описание программного кода классов.

Интерфейс ICanBeCheckedForCorrect создан для проверки объекта на корректность и содержит метод correct, который возвращает true или false.

Интерфейс ICargoCarrier создан для вычисления максимальной массы груза и максимальной длины грузы.

Методы

* MaxCargoLength – возвращает максимальную длину груза
* MaxCargoWeight – возвращает максимальную массу груза

Класс Brand хранит информацию о бренде автомобиля и реализует интерфейс ICanBeCheckedForCorrect.

Поля

* Имя
* Страна происхождения
* Год создания

Так же он содержит свойства для чтения и записи в/эти поля.

Методы

* Correct для проверки на корректность
* ToString возвращает строковое представление объекта

Конструкторы

* Конструктор без параметров
* Конструктор с параметрами, позволяющий инициализировать все поля объекта через свойства

Класс MainSpecs хранит основные характеристики автомобиля.

Поля

* Цвет
* Цена
* Масса
* Год выпуска

Так же он содержит свойства для чтения и записи в/эти поля.

Методы

* Correct для проверки на корректность
* ToString возвращает строковое представление объекта

Конструкторы

* Конструктор без параметров
* Конструктор с параметрами, позволяющий инициализировать все поля объекта через свойства

Абстрактный класс Transport реализует интерфейс ICanBeCheckedForCorrect.

Transport связан отношением ассоциации с классами Brand и MainSpecs.

Свойства

* Brand типа Brand хранит всю информацию о бренде
* MainSpecs типа MainSpecs хранит основные характеристики автомобиля
* Image типа Image хранит изображение автомобиля

Методы

* Correct для проверки на корректность
* ToString возвращает строковое представление объекта

Конструкторы

* Конструктор без параметров
* Конструктор с параметрами, позволяющий инициализировать все поля объекта через свойства

Класс Car создан для легковых автомобилей. Наследник класса Transport.  
Pеализует интерфейс ICanBeCheckedForCorrect.

Поля

* Тип кузова

Свойства

* BodyType для записи/ чтения из/в поле для типа кузова
* Индексатор доступа к свойствам объекта через их имя

Методы

* Correct для проверки на корректность
* ToString возвращает строковое представление объекта

Конструкторы

* Конструктор без параметров
* Конструктор с параметрами, позволяющий инициализировать все поля объекта через свойства

Класс Lorry создан для грузовых автомобилей. Наследник класса Transport.  
Pеализует интерфейс ICanBeCheckedForCorrect и ICargoCarrier.

Поля

* Длина прицепа

Свойства

* Для записи и чтения в/из поле для длины прицепа

Методы

* Correct для проверки на корректность
* ToString возвращает строковое представление объекта
* MaxCargoLength – возвращает максимальную длину груза
* MaxCargoWeight – возвращает максимальную массу груза

Конструкторы

* Конструктор без параметров
* Конструктор с параметрами, позволяющий инициализировать все поля объекта через свойства

## Тестирование функциональности классов.

Программа реализована в виде приложения Windows Forms.

При запуске перед пользователем отображается вкладка, где пользователь может создать новый автомобиль.

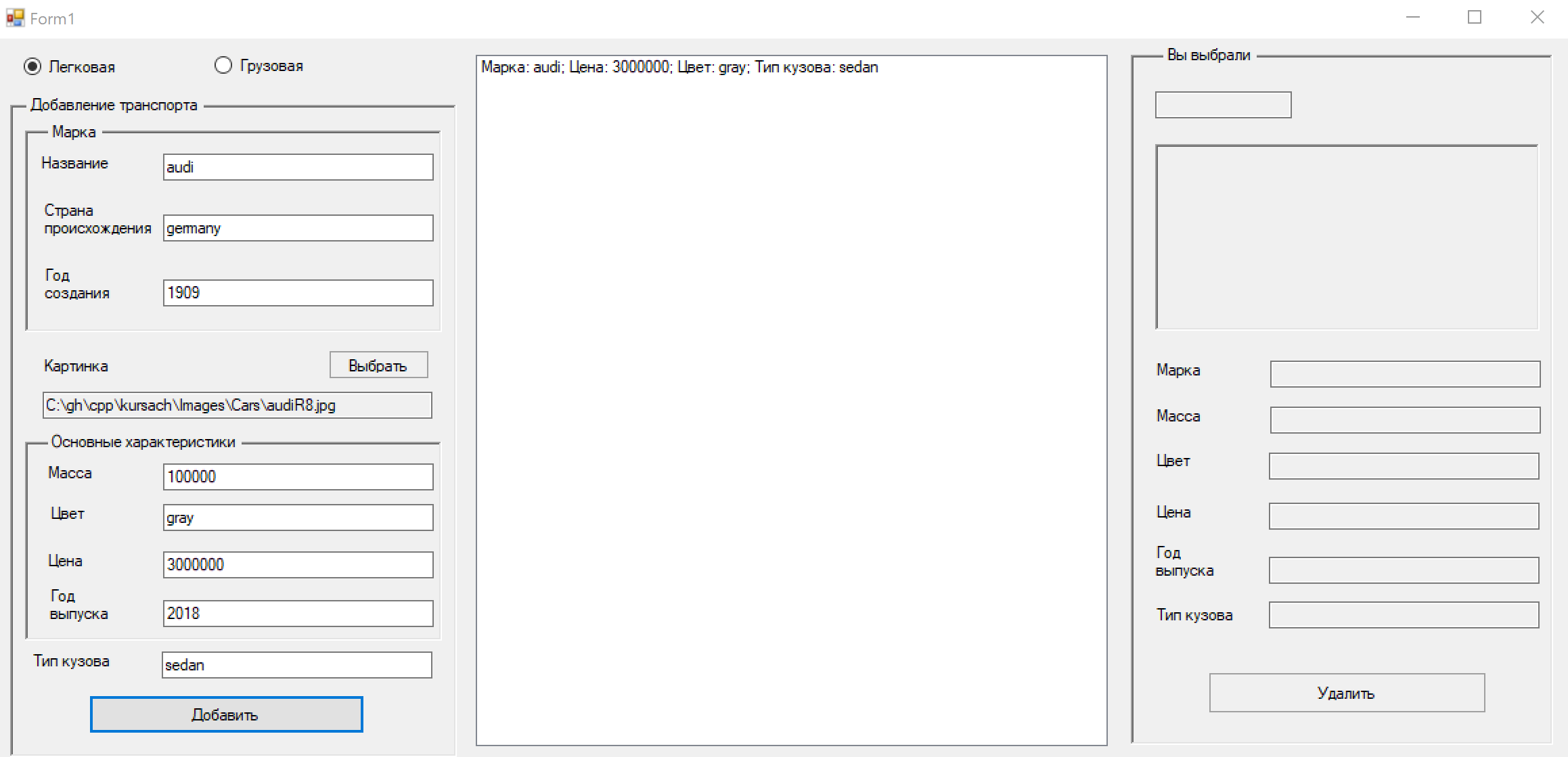


Рисунок 2 – создание и добавление легкового автомобиля

Рисунок 3 – просмотр выбранного автомобиля

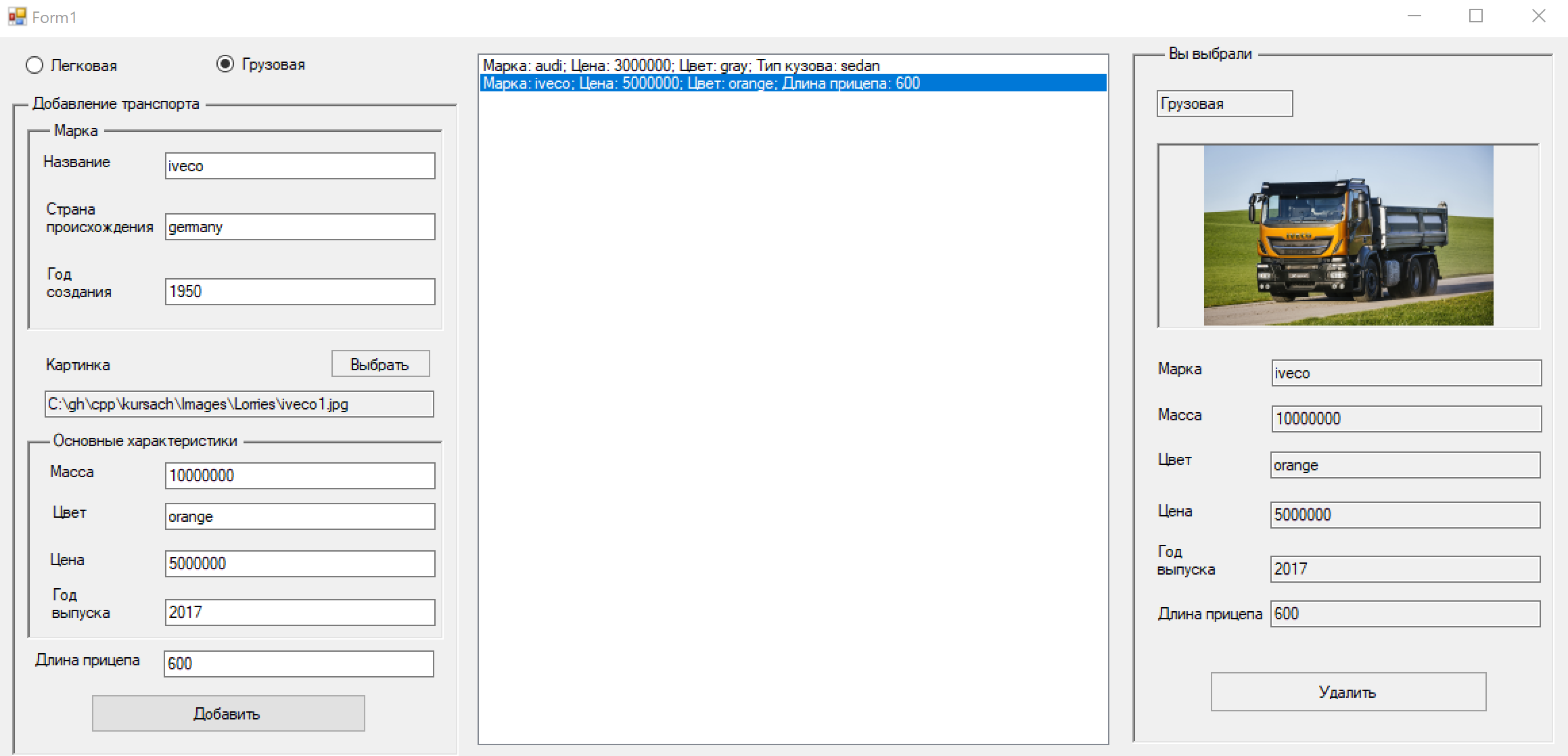
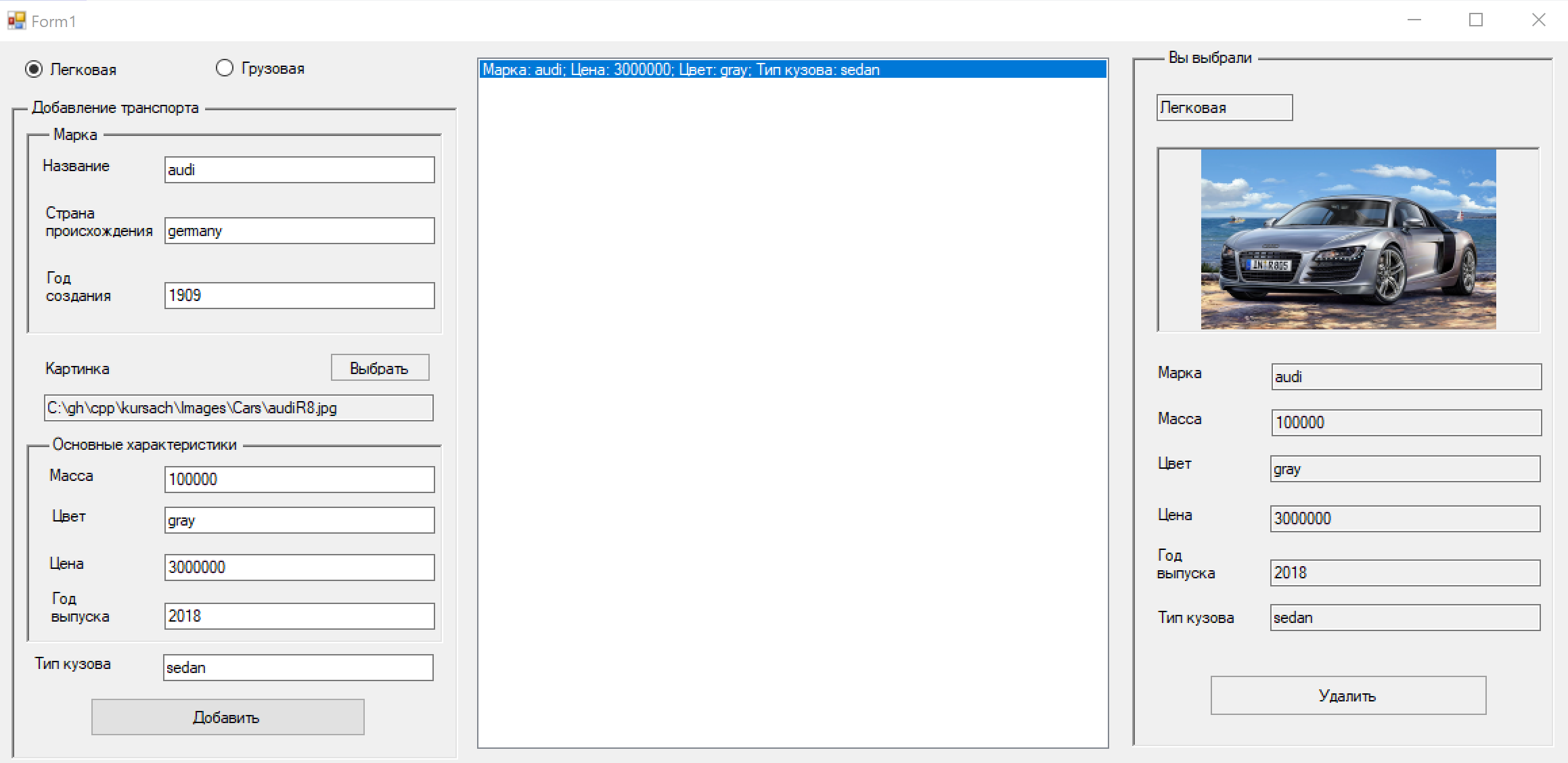


Рисунок 4 – создание, добавление и просмотр грузового автомобиля

Рисунок 5 – удаление выбранного автомобиля

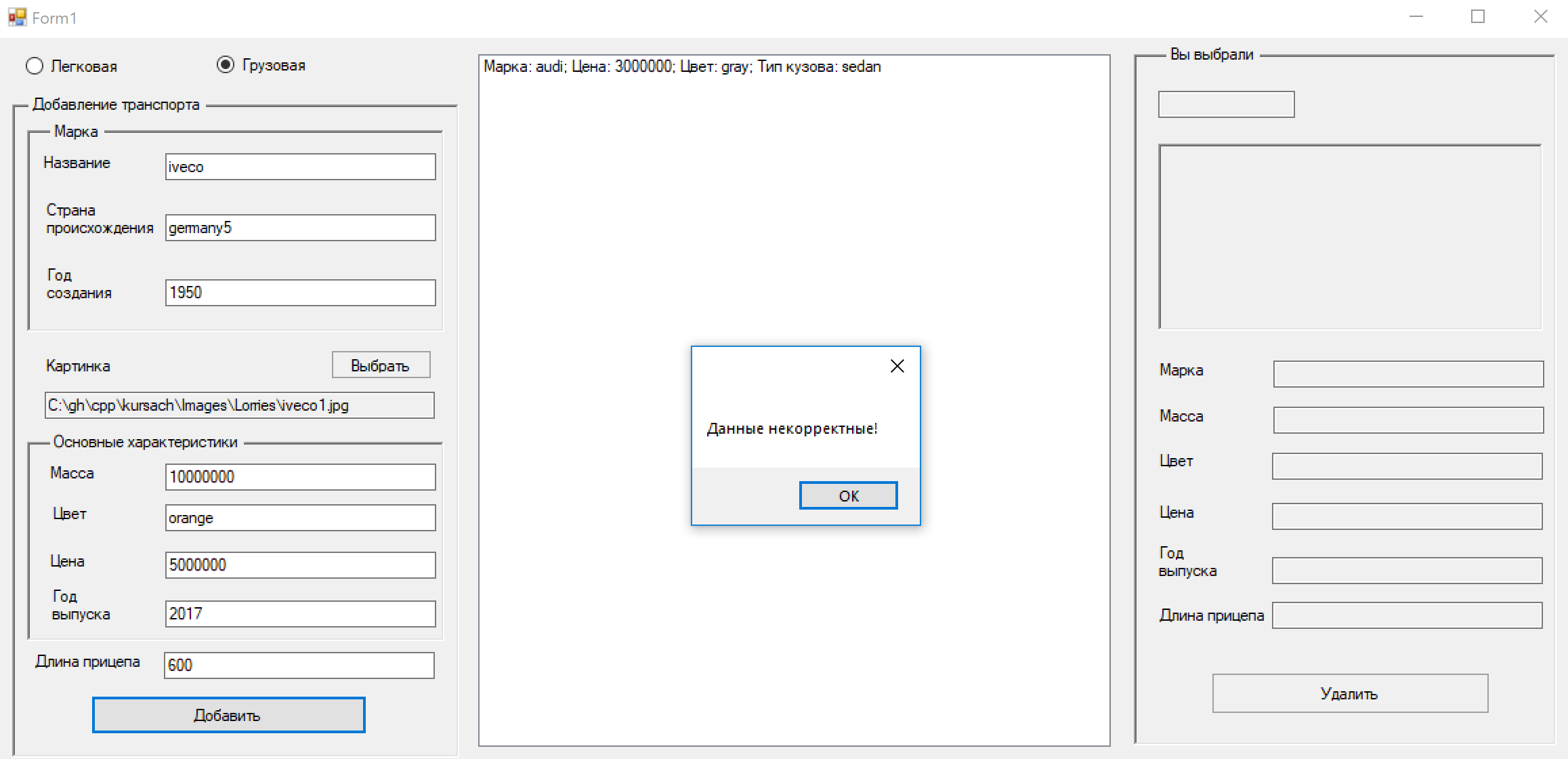
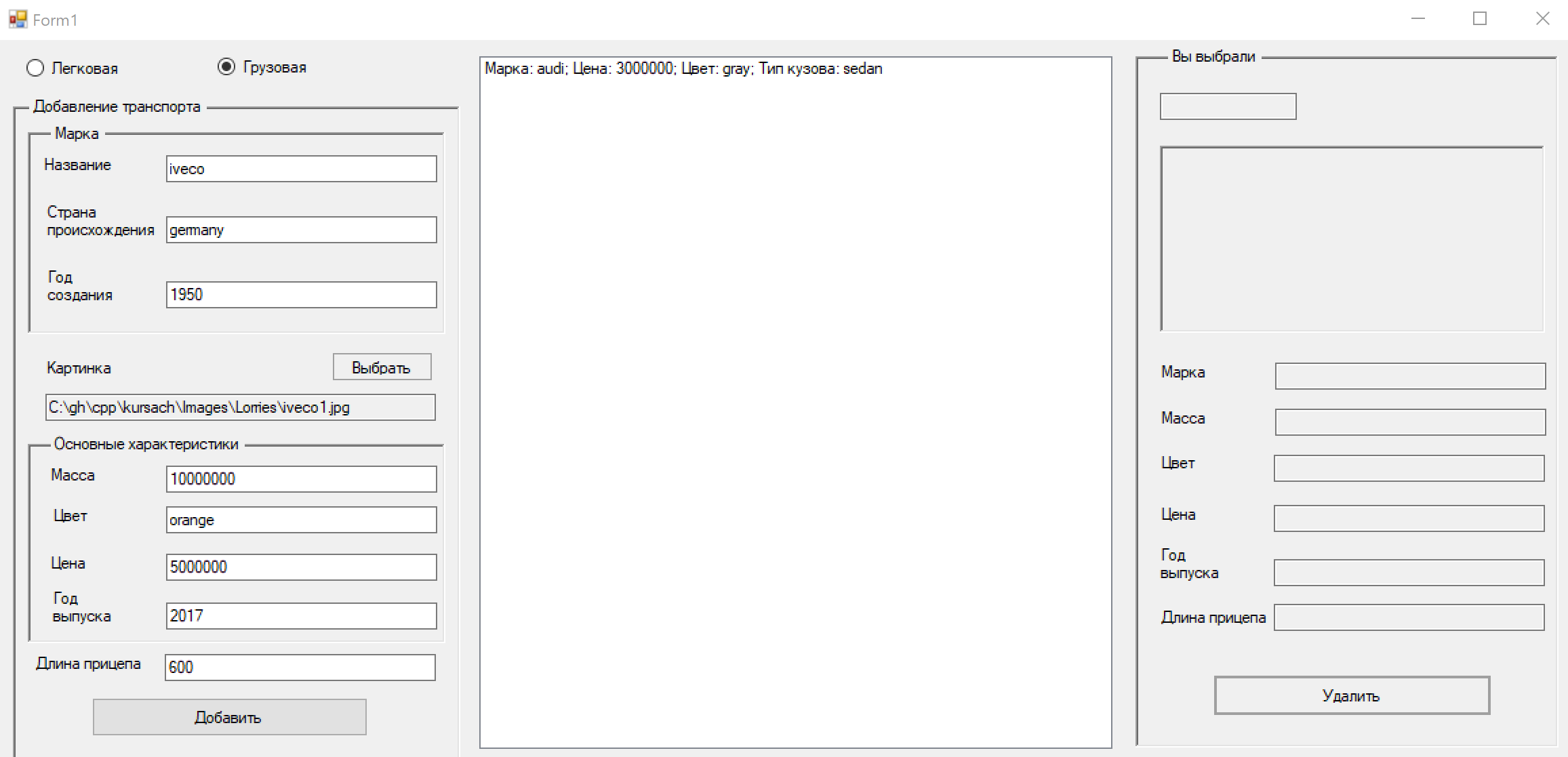


Рисунок 6 – предупреждение о некорректности данных

## Руководство пользователя.

С самого начала необходимо выбрать тип автомобиля, по умолчанию выбирается легковой автомобиль, но если вы хотите выбрать грузовой, то нужно нажать на точечку выбора рядом с названием “Грузовой”.

В текстовые поля: название, страна происхождения, цвет, тип кузова – необходимо вводить только буквы.

В текстовые поля: год создания, масса, цена, год выпуска, длина прицепа – необходимо вводить только цифры.

Если вы заполнили все текстовые поля и выбрали картинку автомобиля, то нужно нажать кнопку “добавить” для добавления автомобиля в список.

Если вы хотите посмотреть автомобиль, то следует выбрать нужный автомобиль из списка автомобилей и вся информация о нем отобразится справа в разделе “Вы выбрали”.

Если вы хотите удалить автомобиль из списка, то необходимо выбрать нужный автомобиль из списка и нажать на кнопку “удалить”, после чего вся информация о нем в разделе выбранного вами автомобиля очистится.

## Заключение

В ходе выполнения данного курсового проекта была разработана библиотека классов для просмотра характеристик грузового и легкового транспорта, которая позволяет пользователю создавать, просматривать и удалять легковые и грузовые автомобили.

1. Библиографический список

<http://www.cyberforum.ru/>

<https://docs.microsoft.com/ru-ru/>

<https://metanit.com/sharp/>

## Приложения

**Приложение 1. Листинг программы**

namespace superProject

{

// абстрактный класс транспорта, реализует интерфейс проверки корректности

abstract class Transport : ICanBeChekcedForCorrect

{

#region properties

// свойство для чтения и записи информации о марке

public Brand Brand

{

get;

set;

}

// свойство для чтения и записи информации об основных характеристиках

public MainSpecs MainSpecs

{

get;

set;

}

// свойство для чтения и записи картинки

public Image Image

{

get;

set;

}

#endregion properties

#region constructors

// конструктор без параметров

public Transport()

{

this.Brand = new Brand();

this.MainSpecs = new MainSpecs();

this.Image = null;

}

// перегруженный конструктор

public Transport(Brand brand, MainSpecs mainSpecs, Image image)

{

this.Brand = brand;

this.MainSpecs = mainSpecs;

this.Image = image;

}

#endregion constructors

#region other methods

// метод для текстового представления объекта

public virtual string ToString(string mode = "short")

{

return String.Format("Марка: {0}; Цена: {1}; Цвет: {2};",

this.Brand.Name, this.MainSpecs.Price, this.MainSpecs.Color);

}

// реализация интерфейса определения корректности объекта

public virtual bool Correct()

{

if (this.Brand.Correct() && this.MainSpecs.Correct() && this.Image != null)

{

return true;

}

return false;

}

#endregion other methods

}

}

namespace superProject

{

// класс легкового автомобиля, наследник транспорта, реализует интерфейс проверки корректности

class Car : Transport, ICanBeChekcedForCorrect

{

#region fields

private string bodyType = ""; // тип кузова

#endregion fields

#region properties

// свойство для чтения и записи из/в поле bodyType

public string BodyType

{

get

{

return this.bodyType;

}

set

{

foreach (var ch in value)

{

if (!Char.IsLetter(ch))

{

this.bodyType = "";

return;

}

}

bodyType = value;

}

}

// индексатор для получения строкового представляния свойств по их имененам

public string this[string propertyName]

{

get

{

if (propertyName == "Brand")

{

return this.Brand.ToString();

}

if (propertyName == "MainSpecs")

{

return this.MainSpecs.ToString();

}

if (propertyName == "Image")

{

return this.Image.ToString();

}

if (propertyName == "BodyType")

{

return this.BodyType;

}

return null;

}

}

#endregion properties

#region constructors

// конструктор без параметров. Вызывает конструктор без параметров базового класса Transport

public Car() : base()

{

this.BodyType = "";

}

// перегруженный конструктор. Вызывает перегруженный конструктор базового класса Transport

public Car(Brand brand, MainSpecs mainSpecs, Image image, string bodyType) : base(brand, mainSpecs, image)

{

this.BodyType = bodyType;

}

#endregion constructors

#region other methods

// реализация интерфейса определения корректности объекта

public override bool Correct()

{

if (base.Correct() && this.BodyType != "")

{

return true;

}

return false;

}

// переопределённый метод для текстового представления объекта

public override string ToString(string mode = "short")

{

if (mode == "short")

{

return String.Format("{0} Тип кузова: {1}", base.ToString(), this.BodyType);

}

return String.Format("Легковой Автомобиль\n" +

"Марка:\n{0}\n" +

"Основные характеристики:\n{1}\n" +

"Тип кузова: {2}\n" ,

this.Brand, this.MainSpecs, this.BodyType

);

}

#endregion other methods

}

}

namespace superProject

{

// класс грузовика, наследник транспорта, реализует интерфейс проверки корректности и интерфейс грузоперевозок

class Lorry : Transport, ICanBeChekcedForCorrect, ICargoCarrier

{

#region fields

private int truckLength; // длина прицепа

#endregion fields

#region properties

// свойство для чтения и записи из/в поле truckLength

public string TruckLength

{

get

{

return this.truckLength.ToString();

}

set

{

if (value == "")

{

this.truckLength = -1;

return;

}

foreach (var ch in value)

{

if (!Char.IsDigit(ch))

{

this.truckLength = -1;

return;

}

}

int truckLength = Convert.ToInt32(value);

if (truckLength > 0)

{

this.truckLength = truckLength;

return;

}

this.truckLength = -1;

}

}

#endregion properties

#region constructors

// конструктор без параметров. Вызывает конструктор без параметров базового класса Transport

public Lorry() : base()

{

this.TruckLength = "-1";

}

// перегруженный конструктор. Вызывает перегруженный конструктор базового класса Transport

public Lorry(Brand brand, MainSpecs mainSpecs, Image image, string trucklength = "-1") : base(brand, mainSpecs, image)

{

this.TruckLength = trucklength;

}

#endregion

#region other methhods

// реализация интерфейса определения корректности объекта

public override bool Correct()

{

if (base.Correct() && this.TruckLength != "-1")

{

return true;

}

return false;

}

// переопределённый метод для текстового представления объекта

public override string ToString(string mode = "short")

{

if (mode == "short")

{

return String.Format("{0} Длина прицепа: {1}", base.ToString(), this.TruckLength);

}

return String.Format("Грузовой автомобиль\n" +

"Марка:\n{0}\n" +

"Основные характеристики:\n{1}\n" +

"Длина прицепа: {2}\n",

this.Brand, this.MainSpecs, this.TruckLength

);

}

// реализация интерфейса грузоперевозок

public double MaxCargoWeight()

{

return 2 \* Convert.ToInt32(this.MainSpecs.Weigth);

}

// реализация интерфейса грузоперевозок

public double MaxCargoLength()

{

return 0.8 \* Convert.ToDouble(this.TruckLength);

}

#endregion

}

}

namespace superProject

{

// класс для информации о марке, реализует интерфейс проверки корректности

class Brand : ICanBeChekcedForCorrect

{

#region fields

private string name; // название марки

private string countryOfOrigin; // страна происхождения

private int yearOfBirth; // год создания

#endregion fields

#region properties

// свойство для чтения и записи из/в поле name

public string Name

{

get

{

return this.name;

}

set

{

foreach (var ch in value)

{

if (!Char.IsLetter(ch))

{

this.name = "";

return;

}

}

this.name = value;

}

}

// свойство для чтения и записи из/в поле countryOfOrigin

public string CountryOfOrigin

{

get

{

return this.countryOfOrigin;

}

set

{

foreach (var ch in value)

{

if (!Char.IsLetter(ch))

{

this.countryOfOrigin = "";

return;

}

}

this.countryOfOrigin = value;

}

}

// свойство для чтения и записи из/в поле yearOfBirth

public string YearOfBirth

{

get

{

return this.yearOfBirth.ToString();

}

set

{

if (value == "")

{

this.yearOfBirth = -1;

return;

}

foreach (var ch in value)

{

if (!Char.IsDigit(ch))

{

this.yearOfBirth = -1;

return;

}

}

int yearOfBirth = Convert.ToInt32(value);

if (yearOfBirth > 0)

{

this.yearOfBirth = yearOfBirth;

return;

}

this.yearOfBirth = -1;

}

}

#endregion properties

#region constructors

// конструктор без параметров

public Brand()

{

this.Name = "";

this.CountryOfOrigin = "";

this.YearOfBirth = "-1";

}

// перегруженный конструктор

public Brand(string name, string countryOfOrigin, string yearOfBirth)

{

this.Name = name;

this.CountryOfOrigin = countryOfOrigin;

this.YearOfBirth = yearOfBirth;

}

#endregion constructors

#region other methods

// переопределённый метод для текстового представления объекта

public override string ToString()

{

return String.Format("Название: {0}; Страна: {1}; Год создания: {2}",

this.Name, this.CountryOfOrigin, this.YearOfBirth);

}

// реализация интерфейса определения корректности объекта

public bool Correct()

{

if (this.Name != "" && this.CountryOfOrigin != "" && this.YearOfBirth != "-1")

{

return true;

}

return false;

}

#endregion

}

}

namespace superProject

{

// класс для информации об основных харатекристиках транспорта, реализует интерфейс проверки корректности

class MainSpecs : ICanBeChekcedForCorrect

{

#region fields

private int weigth; // вес

private string color; // цвет

private int price; // цена

private int yearOfBuild; // год выпуска

#endregion fields

#region properties

// свойство для чтения и записи из/в поле weight

public string Weigth

{

get

{

return this.weigth.ToString();

}

set

{

if (value == "")

{

this.weigth = -1;

return;

}

foreach (var ch in value)

{

if (!Char.IsDigit(ch))

{

this.weigth = -1;

return;

}

}

int weigth = Convert.ToInt32(value);

if (weigth > 0)

{

this.weigth = weigth;

return;

}

this.weigth = -1;

}

}

// свойство для чтения и записи из/в поле color

public string Color

{

get

{

return this.color;

}

set

{

foreach(var ch in value)

{

if (!Char.IsLetter(ch))

{

this.color = "";

return;

}

}

this.color = value;

}

}

// свойство для чтения и записи из/в поле price

public string Price

{

get

{

return this.price.ToString();

}

set

{

if (value == "")

{

this.price = -1;

return;

}

foreach (var ch in value)

{

if (!Char.IsDigit(ch))

{

this.price = -1;

return;

}

}

int price = Convert.ToInt32(value);

if (price > 0)

{

this.price = price;

return;

}

this.price = -1;

}

}

// свойство для чтения и записи из/в поле yearOfBuild

public string YearOfBuild

{

get

{

return this.yearOfBuild.ToString();

}

set

{

if (value == "")

{

this.yearOfBuild = -1;

return;

}

foreach (var ch in value)

{

if (!Char.IsDigit(ch))

{

this.yearOfBuild = -1;

return;

}

}

int yearOfBuild = Convert.ToInt32(value);

if (yearOfBuild > 0)

{

this.yearOfBuild = yearOfBuild;

return;

}

this.yearOfBuild = -1;

}

}

#endregion properties

#region constructors

// конструктор без параметров

public MainSpecs()

{

this.Weigth = "-1";

this.Color = "";

this.Price = "-1";

this.YearOfBuild = "-1";

}

// перегруженный конструктор

public MainSpecs(string weight, string color, string price, string yearOfBuild)

{

this.Weigth = weight;

this.Color = color;

this.Price = price;

this.YearOfBuild = yearOfBuild;

}

#endregion constructors

#region other methods

// переопределённый метод для текстового представления объекта

public override string ToString()

{

return String.Format("Масса: {0}; Цвет: {1}; Цена: {2}; Год выпуска: {3}",

this.Weigth, this.Color, this.Price, this.YearOfBuild);

}

// реализация интерфейса определения корректности объекта

public bool Correct()

{

if (this.Weigth != "-1" && this.Color != "" && this.Price != "-1" && this.YearOfBuild != "-1")

{

return true;

}

return false;

}

#endregion

}

}

namespace superProject

{

// интерфейс для реализации проверки корректности

interface ICanBeChekcedForCorrect

{

// метод, проверяющий корректность объекта

bool Correct();

}

}

namespace superProject

{

// интерфейс для перевозящего грузы транспорта

interface ICargoCarrier

{

// метод, вычисляющий максимальный вес груза

double MaxCargoWeight();

// метод, вычисляющий максимальную длину груза

double MaxCargoLength();

}

}

namespace superProject

{

public partial class Form1 : Form

{

// конструктор формочки

public Form1()

{

InitializeComponent();

}

// список для хранения объектов типа Transoprt

List<Transport> transportList = new List<Transport>();

// метод, который вызывается при запускается формочки

private void Form1\_Load(object sender, EventArgs e)

{

// при запуске поумолчанию выбран легковой автомобиль

radioButtonCar.Checked = true;

labelAdditionalInput.Text = "Тип кузова";

}

// выбор легкового

private void radioButtonCar\_CheckedChanged(object sender, EventArgs e)

{

if (radioButtonCar.Checked)

{

labelAdditionalInput.Text = "Тип кузова";

}

}

// выбор грузового

private void radioButtonLorry\_CheckedChanged(object sender, EventArgs e)

{

if (radioButtonLorry.Checked)

{

labelAdditionalInput.Text = "Длина прицепа";

}

}

// добавление картинки

private void buttonImageInput\_Click(object sender, EventArgs e)

{

if (openFileDialogAddImage.ShowDialog() != DialogResult.Cancel)

{

textBoxImagePath.Text = openFileDialogAddImage.FileName;

}

}

// добавление транспорта

private void buttonAddTransport\_Click(object sender, EventArgs e)

{

// проверяем, что картинка выбрана

if (textBoxImagePath.Text == "")

{

MessageBox.Show("Данные некорректные!");

return;

}

// читаем из текстбоксов информацию о марке и записываем в объект типа Brand

Brand brand = new Brand(textBoxBrandNameInput.Text, textBoxBrandCountryInput.Text, textBoxBrandYearInput.Text);

// читаем из текстбоксов информацию об основных характеристиках и записываем в объект типа MainSpecs

MainSpecs mainSpecs = new MainSpecs(textBoxWeightInput.Text, textBoxColorInput.Text, textBoxPriceInput.Text, textBoxYearOfBuildInput.Text);

// читаем из текстбокса путь к картинке и записываем в объект типа Image

Image image = Image.FromFile(textBoxImagePath.Text);

// если выбрана легковая

if (radioButtonCar.Checked)

{

// создаём объект типа Car

Car car = new Car(brand, mainSpecs, image, textBoxAdditionalInput.Text);

// проверка на корректность введённых данных

if (!car.Correct())

{

MessageBox.Show("Данные некорректные!");

return;

}

// добавляем краткое текстовое представление объекта car в listBoxOfTransport

listBoxOfTransport.Items.Add(car.ToString());

// добавляем объект car в transportList

transportList.Add(car);

}

// если выбрана грузовая

if (radioButtonLorry.Checked)

{

// создаём объект типа Lorry

Lorry lorry = new Lorry(brand, mainSpecs, image, textBoxAdditionalInput.Text);

// проверка на корректность введённых данных

if (!lorry.Correct())

{

MessageBox.Show("Данные некорректные!");

return;

}

// добавляем краткое текстовое представление объекта car в listBoxOfTransport

listBoxOfTransport.Items.Add(lorry.ToString());

// добавляем объект car в transportList

transportList.Add(lorry);

}

}

// вывод выбранного объекта

private void listBoxOfTransport\_SelectedIndexChanged(object sender, EventArgs e)

{

// узнаём номер объекта

int i = listBoxOfTransport.SelectedIndex;

if (i == -1)

{

return;

}

// если он типа Car

if (transportList[i].GetType().Name == "Car")

{

textBoxTransportTypeOut.Text = "Легковая";

labelAdditionalOut.Text = "Тип кузова";

// преобразовали элемент коллекции transportList к типу Car(коллекция из типа Transport)

textBoxAdditionalOut.Text = ((Car)(transportList[i])).BodyType;

}

// если он типа Lorry

if (transportList[i].GetType().Name == "Lorry")

{

textBoxTransportTypeOut.Text = "Грузовая";

labelAdditionalOut.Text = "Длина прицепа";

// преобразовали элемент коллекции transportList к типу Lorry(коллекция из типа Transport)

textBoxAdditionalOut.Text = ((Lorry)(transportList[i])).TruckLength;

}

// пихаем картинку в бокс

pictureBoxTransportImage.Image = transportList[i].Image;

// подгоняем размеры картинки

pictureBoxTransportImage.SizeMode = PictureBoxSizeMode.Zoom;

// заполняем остальные поля вывода

textBoxBrandOut.Text = transportList[i].Brand.Name;

textBoxWeightOutput.Text = transportList[i].MainSpecs.Weigth;

textBoxColorOut.Text = transportList[i].MainSpecs.Color;

textBoxPriceOut.Text = transportList[i].MainSpecs.Price;

textBoxYearOfBuildOut.Text = transportList[i].MainSpecs.YearOfBuild;

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

// проверяем, есть ли, что удалять

if (transportList.Count == 0)

{

MessageBox.Show("Список пуст!");

return;

}

int i = listBoxOfTransport.SelectedIndex;

// проверяем выбрано ли, что удалять

if (i == -1)

{

MessageBox.Show("Транспорт не выбран!");

return;

}

// собственно, удаляем

listBoxOfTransport.Items.RemoveAt(i);

transportList.RemoveAt(i);

// очищаем элементы вывода

textBoxTransportTypeOut.Clear();

pictureBoxTransportImage.Image = null;

textBoxBrandOut.Clear();

textBoxWeightOutput.Clear();

textBoxColorOut.Clear();

textBoxPriceOut.Clear();

textBoxYearOfBuildOut.Clear();

textBoxAdditionalOut.Clear();

}

}

}