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 2018 г.

Оглавление

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

## Диаграмма классов

## Описание программного кода классов

## Тестирование функциональности классов

## Пример использования библиотеки для построения интерфейса прикладного решения

## 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:

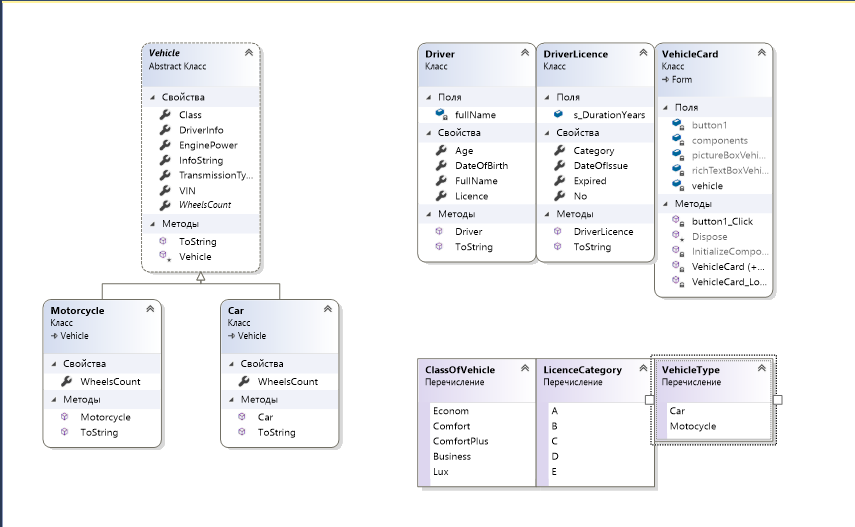
* Строковое представление объекта
* Корректность объекта
* Конструктор без параметров
* Конструктор с параметрами
* Вычисление максимальной массы грузы
* Вычисление максимальной длины груза

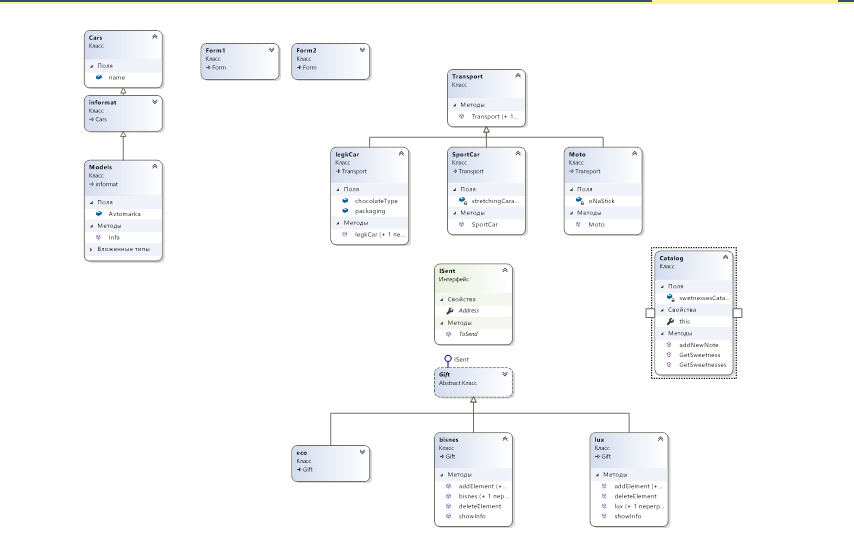
Для работы с этими объектами необходимо создать абстрактный базовый класс «Транспорт», в котором будет находиться информация о бренде, основные характеристики автомобиля, картинка, методы строкового представления и проверки корректности объектов. Также нужно создать и классы для хранения информации о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

## Диаграмма классов.





## Описание программного кода классов.

В абстрактном классе **Vehicle** созданы перегруженные конструкторы, вышеописанные поля, такие методы: ToString , VehicleCard, Motorcyle

Метод ToString возвращает значение.

Метод VehicleCard перегруженный.

Метод Motorcyle. конструктор класса мотоцикл, который берет значения из класса Vehicle.

Класс **Transport** хранит в себе типы автомобилей и коллекцию всех классов . Также в нем хранится значение равное количеству созданных подарков. В нем присутствуют такие методы: addGift, addNewNote, findElement, GetGift, GetGifts, GetSweetness, GetSweetnesses, removeGift, send, sortSelectGift.

Метод addGift добавляет в коллекцию новый каталог автомобиля.

Метод findElement ищет и выводит на экран элемент, соответствующий заданному диапазону.

Метод GetGft возвращает автомобиль из коллекции.

Метод GetGifts возвращает коллекцию классов.

Метод GetTransport возвращает название автомобиля из коллекции созданных.

Метод removeGift удаляет из коллекции подарок по переданному индексу.

Метод send вызывает у всех объектов интерфейса ISent метод toSend который выводит информацию об отправление в richTextBox.

Метод sortSelectGift вызывает метод.

Класс **Gift** хранит в себе значения адреса, коллекцию автомобилей, флаг показывающий дополнительный пакет, максимальную комплектацию и количество элементов в коллекции. Методы: deleteElement, getComposition, getMaxSize, getSize, showInfo.

Метод deleteElement удаляет элемент из коллекции.

Метод getComposition возвращает коллекцию автомобилей.

Метод getMaxSize возвращает максимальную комплектацию.

Метод getSize возвращает текущий.

Метод showInfo возвращает информацию об объекте.

## Тестирование функциональности классов.

Между классами Vehicle и Gift выстроено отношение - композиция. Так как коллекция с объектами Gift находится в классе Storage. Аналогичная ситуация и у классов Sweetness и Gift. Коллекция с объектамиTransport, в которой хранятся типы созданных сладостей, находится в классе Storage.

public class Vehicle

{

// созданная коллекция с типами

List<Transport>TransportsCatalog=new List<Sweetness>();

//коллекция

List<Gift> gifts = new List<Gift>();

}

Между классами Disk и Phonogram также выстроено отношение – композиция, так как коллекция с объектами Phonogram находится в классе Disk.

public abstract class Vihicle

{

private List<Vehicle > Vehicles;

Создание нового пользователя.

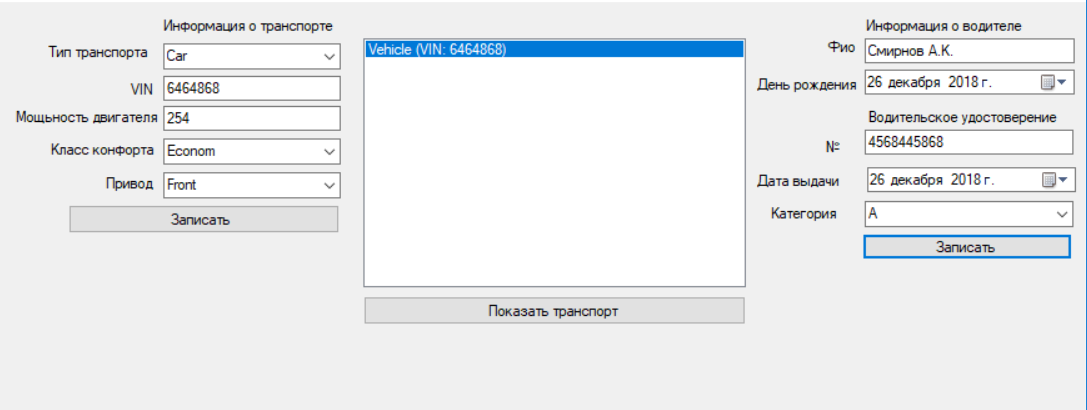


Рисунок 1. Тест метода класса.

Добавление пользователя в коллекцию каталога.

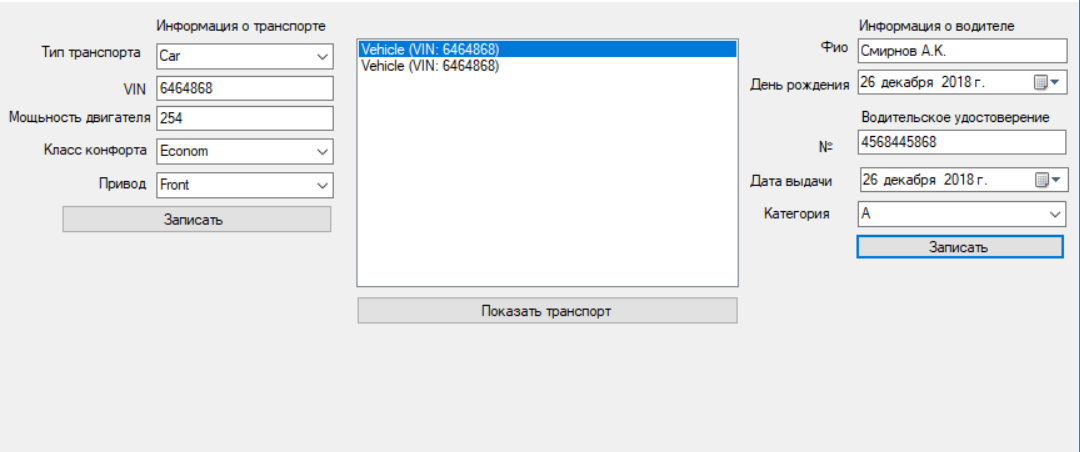


Рисунок 2. Тест метода каталога.

Вывод полной информации о пользователе.



Рисунок 3. Тест метода класса.

Вывод полной информации об автомобили.

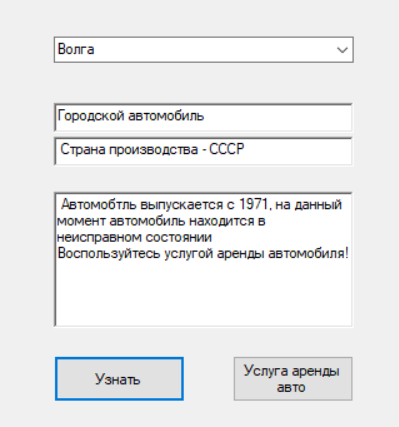


Рисунок 4. Тест метода, который возвращает несколько значений (модификатор out)

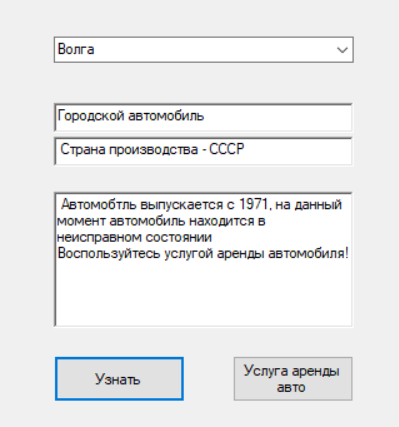


Рисунок 7. Тест метода класса.

Услуга аренды автомобиля. Создание и добавление в лист нового автомобиля.

1. Пример использования библиотеки для построения интерфейса прикладного решения.

Программа реализована в виде приложения Windows Forms.

При запуске перед пользователем отображается вкладка, где пользователь может создать новый тип нужного ему автомобиля.

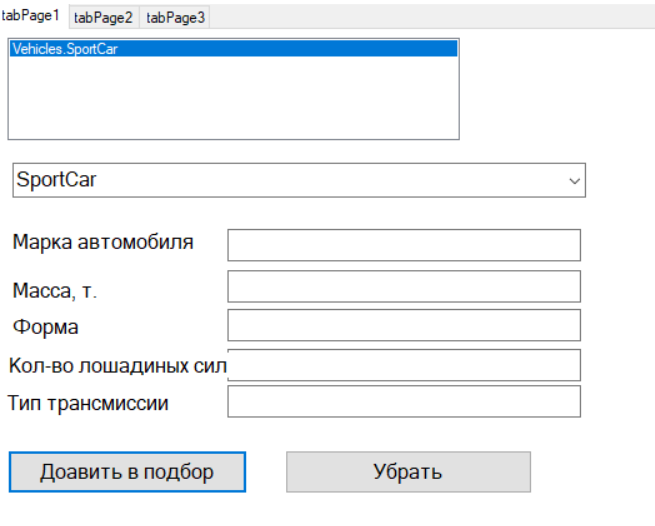
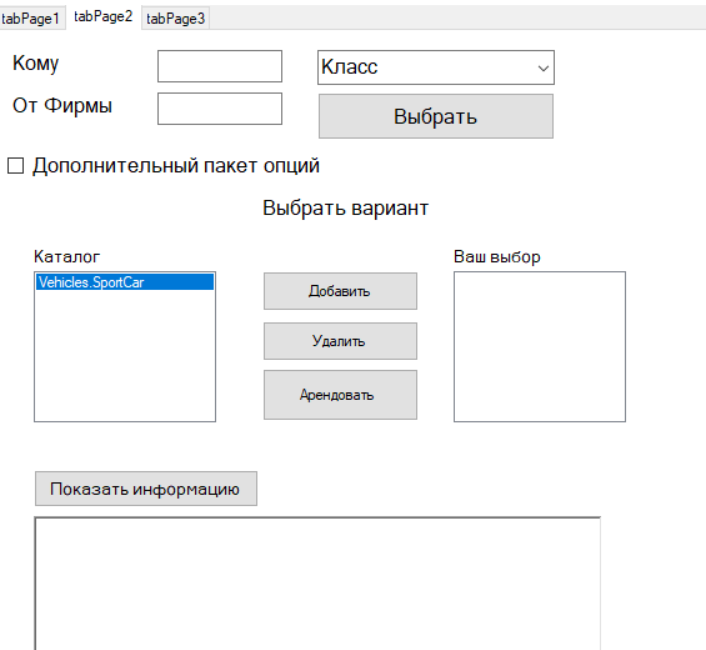
 

Рис8. Тест метода

Добавление в список выбранных пользователем авто

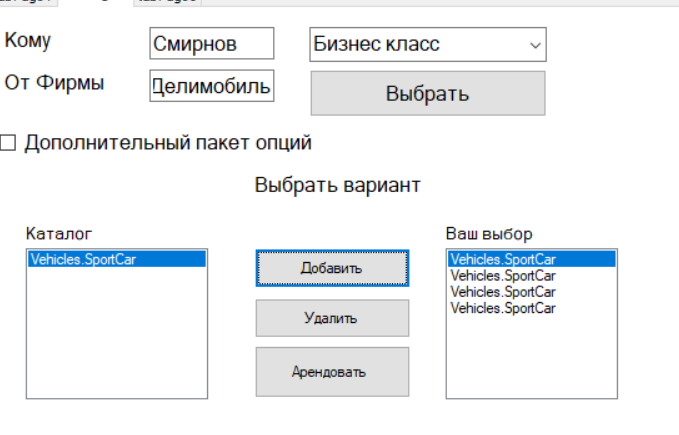
.

Рис9 Тест метода

Вывод информации в textbox.

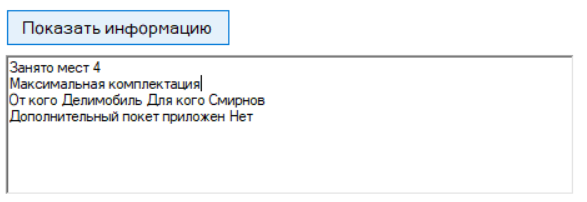


Рис 10 Тест метода

Удаление выбранного подарка из списка.

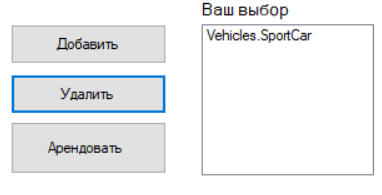
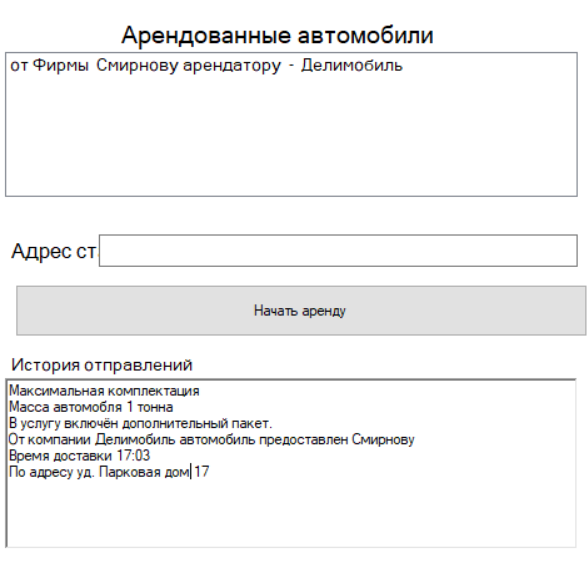
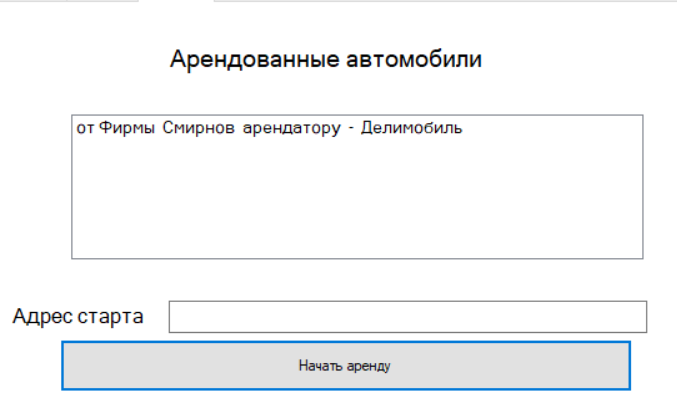


Рис 11 тест метода

Перейдя на вкладку «Арендовать», пользователь сможет указать адрес и отправить выбранный подарок по указанному адресу.



## Руководство пользователя.

В текстовое поле: Водительское удостоверение, Вин, необходимо вводить арабские цифры.

При аренде автомобиля вводе массы и кол-во лошадиных сил, когда вы хотите создать новый тип автомобиля, необходимо вводить арабскими цифрами.

Если вы хотите изменить что-то, который вы хотите добавить, то выберите необходимый тип из выпадающего списка.

Также если вы хотите изменить тип автомобиля, который вы хотите создать, то аналогично выберите необходимый тип из предложенного списка.

## Заключение

В процессе выполнения курсового проекта была разработана библиотека классов моделирующая объект подарок. Она позволяет создать систему способную создать и добавлять созданные сладости в подарок, работать с элементами этого подарка.

1. Библиографический список

[www.cyberforum.ru](http://www.cyberforum.ru)

https://docs.microsoft.com

https://professorweb.

## Приложения

**Приложение 1. Листинг программы**

public abstract class Vehicle

{

public abstract int WheelsCount { get; }//Кол-во rjk`c

public string VIN { get; set; }//Индивидуальный номер автомобиля

public Driver DriverInfo { get; set; }// Информация о водителе

public ClassOfVehicle Class { get; set; }//С класса об транспорте

public int EnginePower { get; set; }//Мощьность двигателя

public TransmissionType TransmissionType { get; set; }

public string InfoString { get { return "Vehicle (VIN: " + this.VIN + ")"; } }//Возращение информации о вине транспорта

protected Vehicle(string vin, ClassOfVehicle classOfVehicle, int enginePower, TransmissionType transmissionType)//Конструктор классса

{

VIN = vin;

Class = classOfVehicle;

EnginePower = enginePower;

TransmissionType = transmissionType;

}

public class Motorcycle : Vehicle

{//Класс мотоцикл который наследуется от класса Vehicle

public override int WheelsCount { get { return 2; } }//Возвращаем значение что у мотоцикла 2 колеса

public Motorcycle(string vin, ClassOfVehicle classOfVehicle, int enginePower, TransmissionType transmissionType) : base(vin, classOfVehicle, enginePower, transmissionType) { }//коструктор класса мотоцикл, котопый берет значения из класса Vehicle

public override string ToString()//Использование метода ToString

{

return string.Format("Мотоцикл ({0} Колеса) ", this.WheelsCount) + base.ToString();

}

}

}

public class Car : Vehicle

{//Класс автомобили кторый наследуется от класса Vehicle

public override int WheelsCount { get { return 4; } }//Возращаем значения, что у автомобиля 4 колеса

public Car(string vin, ClassOfVehicle classOfVehicle, int enginePower, TransmissionType transmissionType) : base(vin, classOfVehicle, enginePower, transmissionType) { }

//коструктор класса Сar, котопый берет значения из класса Vehicle

public override string ToString()

{

return string.Format("Автомобиль ({0} колеса) ", this.WheelsCount) + base.ToString();

}

}

}

public class Driver//Класс водитель Сдесь данные которые ввводятся о водителе

{

private string fullName;//ФиО

public string FullName

{

get { return fullName.ToUpper(); }

set

{

foreach (char ch in value)

{

if (char.IsDigit(ch))

{ this.fullName = "Ошибачное имя"; break; }

else this.fullName = value;

}

if (value.IndexOf('.') == value.Length - 3 && value.LastIndexOf('.') == value.Length - 1)

{ fullName = value; }

else

{

MessageBox.Show("Фамилия должна включать инициалы."); return;

}

}

}

public DateTime DateOfBirth { get; set; }//Дата рождения

public DriverLicence Licence { get; set; }//Дата получение прав

public int Age { get { return (DateTime.Now.Year - DateOfBirth.Year); } }

public Driver(string fullName, DateTime dateOfBirth, DriverLicence driverLicence)//Конструктор класса Водитель

{

FullName = fullName;

DateOfBirth = dateOfBirth;

Licence = driverLicence;

}

public override string ToString()//Возращения значения

{

return string.Format("ФИО {0}\n, Дата рождение: {1}\n, Номер прав № {2}\n", FullName, DateOfBirth, Licence.No);

}

}

}

public class DriverLicence//Класс водительские права

{

public static int s\_DurationYears = 10;//Стаж

public string No { get; set; }//Нет

public LicenceCategory Category { get; set; }//Категория прав

public DateTime DateOfIssue { get; set; }//Дата выпуска

public DateTime Expired { get { return DateOfIssue.AddYears(s\_DurationYears); } }//Итек срок

//ctor

public DriverLicence(string no, LicenceCategory category, DateTime dateOfIssue)//Конструктор класса Водительское удостовирениие

{

No = no;

Category = category;

DateOfIssue = dateOfIssue;

}

public override string ToString()//Возращаемое значение

{

return string.Format("Licence №{0}, {1}, {2}", this.No, this.Category, this.DateOfIssue);

}

}

}

public partial class VehicleCard : Form//Фотографи транспорта

{

private Vehicle vehicle;

private VehicleCard() { InitializeComponent(); }

public VehicleCard(Vehicle vehicle) : this()

{

this.vehicle = vehicle;

}

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

{

string imagePath = Application.StartupPath;//К фотке

if (vehicle is Car)

imagePath += "\\Images\\car.jpg";

else if (vehicle is Motorcycle)

imagePath += "\\Images\\moto.jpg";

pictureBoxVehiclePicture.Image = Image.FromFile(imagePath);

//richTextBoxVehicleInfo.DataBindings.Add("Text", vehicle, "InfoString");

richTextBoxVehicleInfo.Text = this.vehicle.ToString();

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 f = new Form1();

f.Show();

}

}

}

public partial class MainForm : Form

{

private BindingList<Vehicle> Vehicles { get; set; }

//ctor

public MainForm() { InitializeComponent(); }

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

{

Vehicles = new BindingList<Vehicle>();

//Vehicle part

comboBoxVehicleType.DataSource = Enum.GetValues(typeof(VehicleType));

comboBoxClassOfVehicle.DataSource = Enum.GetValues(typeof(ClassOfVehicle));

comboBoxTransmissionType.DataSource = Enum.GetValues(typeof(TransmissionType));

//Listbox

listBox1.DataSource = Vehicles;

listBox1.DisplayMember = "InfoString";

//Driver part

comboBoxLicenceType.DataSource = Enum.GetValues(typeof(LicenceCategory));

}

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

{

if (listBox1.SelectedItem == null)

return;

Vehicle vehicle = listBox1.SelectedItem as Vehicle;

VehicleCard vc = new VehicleCard(vehicle);

vc.ShowDialog();

}

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

{

string VIN = textBoxVIN.Text;//Вовод Vin в textbox значение стринг

int enginePower = int.Parse(textBoxEnginePower.Text);// Ввод мощьности транспортного средства значения int

ClassOfVehicle classOfVehicle = (ClassOfVehicle)Enum.Parse(typeof(ClassOfVehicle), comboBoxClassOfVehicle.SelectedValue.ToString());//Выыбор класса транспорта

TransmissionType transmissionType = (TransmissionType)Enum.Parse(typeof(TransmissionType), comboBoxTransmissionType.SelectedValue.ToString());//Выбор привода транспорта

Vehicle newVehicle = null;

VehicleType vehicleType = (VehicleType)Enum.Parse(typeof(VehicleType), comboBoxVehicleType.SelectedValue.ToString());

switch (vehicleType)// оператор выбора, который выбирает один из списка транспортов

{

case VehicleType.Car:

newVehicle = new Car(VIN, classOfVehicle, enginePower, transmissionType);

break;

case VehicleType.Motocycle:

newVehicle = new Motorcycle(VIN, classOfVehicle, enginePower, transmissionType);

break;

}

Vehicles.Add(newVehicle);//добавление

}

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

{

if (listBox1.SelectedItem == null)

return;

string licenceNo = textBoxLicenceNo.Text;//Ввод лицензии(номер прав)

DateTime issueDate = dateTimePicker1.Value;//Дата ввыдачи

LicenceCategory licenceCategory = (LicenceCategory)Enum.Parse(typeof(LicenceCategory), comboBoxLicenceType.SelectedValue.ToString());//выбо водительской категории

string fullName = textBoxName.Text;//Ввод ФОИ

DateTime birthday = dateTimePicker.Value;//Дата рождения

DriverLicence driverLicence = new DriverLicence(licenceNo, licenceCategory, issueDate);//Водительская лицензия

Driver driver = new Driver(fullName, birthday, driverLicence);

Vehicle vehicle = listBox1.SelectedItem as Vehicle;//Занос данных

vehicle.DriverInfo = driver;

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 f = new Form1();

f.Show();

}

}

public partial class Form1 : Form

{

private string imagePath;

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

Models m = new Models();

OldPlane op = new OldPlane();

if (comboBox1.Text == "Волга")

{

richTextBox1.Text = op.Infa();

}

Models m1 = new Models();

informat i = new informat();

i["type"] = "Городской автомобиль";

i["proizvodstvo"] = " Страна производства - СССР";

richTextBox1.Text = i["type"];

richTextBox2.Text = i["proizvodstvo"];

richTextBox3.Text = " Автомобтль выпускается с 1971, на данный момент автомобиль находится в неисправном состоянии\n" +

"Воспользуйтесь услугой аренды автомобиля!";

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 fa = new Form2();

fa.Show();

}

}

}

public class Transport

{

public Transport(string name, string weight, string form, string sugarContent)

{

}

public Transport(string name, string weight, string form, string sugarContent, string manufacturer)

{

}

}

public class legkCar : Transport

{

public string chocolateType;

public readonly string packaging = "Обертка с фольгой";

public legkCar(string name, string weight, string form, string sugarContent, string manufacturer, string chocolateType) : base(name, weight, form, sugarContent, manufacturer)

{

this.chocolateType = chocolateType;

}

public legkCar(string name, string weight, string form, string sugarContent, string chocolateType) : base(name, weight, form, sugarContent)

{

}

}

public class SportCar : Transport

{

bool stretchingCaramel;

public SportCar(string name, string weight, string form, string sugarContent, bool stretchingCaramel) : base(name, weight, form, sugarContent)

{

this.stretchingCaramel = stretchingCaramel;

}

}

ublic class Moto : Transport

{

bool oNaStick;

public Moto(string name, string weight, string form, string sugarContent, bool oNaStick) : base(name, weight, form, sugarContent)

{

this.oNaStick = oNaStick;

}

}

}

public interface ISent

{

string Address { get; set; }

void ToSend(DateTime dateTime, ref RichTextBox richTextBox);

}

public abstract class Gift : ISent

{

protected List<Transport> giftComposition;

protected int maxSize;

private string forWhom;

private string fromWhom;

private bool isPostcard;

protected int numElements;

private string address;

public string DisplayMember

{

get

{

string str = "от Фирмы " + forWhom + " арендатору - " + fromWhom;

return str;

}

}

public string ForWhom { get => forWhom; set => forWhom = value; }

public bool IsPostcard { get => isPostcard; set => isPostcard = value; }

public string FromWhom { get => fromWhom; set => fromWhom = value; }

public string Address { get => address; set => address = value; }

protected Gift()

{

giftComposition = new List<Transport>();

}

public Gift(int size)

{

giftComposition = new List<Transport>(size);

}

public Gift(List<Transport> giftComposition)

{

this.giftComposition = giftComposition;

}

public Gift(string forWhom, string fromWhom, bool isPostcard) : this()

{

this.forWhom = forWhom;

this.fromWhom = fromWhom;

this.isPostcard = isPostcard;

}

//абстрактные методы

//можно добавить в лист, передав объект

public abstract void addElement(Transport sweetness);

//можно добавить и по id

public abstract void addElement(int id, Catalog catalog);

//виртуальный метод в родительском классе

public virtual string showInfo()

{

string str = "\nМаксимальная комплектация" + maxSize +

"\nОт кого " + FromWhom + " Для кого " + ForWhom + "\nДополнительный покет приложен " + (IsPostcard ? "Да" : "Нет");

return str;

}

//удаление из списка по индексу

public abstract void deleteElement(int index);

public List<Transport> getComposition()

{

return giftComposition;

}

//реализованный метод интерфейса

public void ToSend(DateTime dateTime, ref RichTextBox richTextBox)

{

richTextBox.Text = "Автомобиль " + DisplayMember + " отправлен в " + dateTime.ToShortTimeString() + "\n по адрессу " + Address;

}

}

public class eco : Gift

{

public eco()

{

maxSize = 4;

}

public eco(string forWhom, string fromWhom, bool isPostcard) : base(forWhom, fromWhom, isPostcard)

{

maxSize = 4;

}

//перегруженный метод

//переопределенный метод

public override void addElement(Transport sweetness)

{

giftComposition.Add(sweetness);

numElements++;

}

//перегруженный метод

//переопределенный метод

public override void addElement(int id, Catalog catalog)

{

giftComposition.Add(catalog.GetSweetness(id));

numElements++;

}

public override void deleteElement(int index)

{

giftComposition.RemoveAt(index);

}

public override string showInfo()// переопределенный метод в дочернем классе

{

string str;

str = "Эконом класс\n Кол-во предпочитавший такой же класс " + numElements;

str += base.showInfo();

return str;

}

}

}

public class bisnes : Gift

{

public bisnes()

{

numElements = 8;

}

public bisnes(string forWhom, string fromWhom, bool isPostcard) : base(forWhom, fromWhom, isPostcard)

{

}

public override void addElement(Transport sweetness)

{

giftComposition.Add(sweetness);

numElements++;

}

public override void addElement(int id, Catalog catalog)

{

giftComposition.Add(catalog.GetSweetness(id));

numElements++;

}

public override void deleteElement(int index)

{

giftComposition.RemoveAt(index);

}

public override string showInfo()

{

string str;

str = "Средний подарок\nЗанято место " + numElements;

str += base.showInfo();

return str;

}

}

public class lux : Gift

{

public lux()

{

numElements = 15;

}

public lux(string forWhom, string fromWhom, bool isPostcard) : base(forWhom, fromWhom, isPostcard)

{

}

public override void addElement(Transport sweetness)

{

giftComposition.Add(sweetness);

numElements++;

}

public override void addElement(int id, Catalog catalog)

{

//использование индексатора

giftComposition.Add(catalog[id]);

numElements++;

}

public override void deleteElement(int index)

{

giftComposition.RemoveAt(index);

}

public override string showInfo()

{

string str;

str = "Большой подарок\nЗанято место " + numElements;

str += base.showInfo();

return str;

}

}

public class Catalog

{

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

//индексатор

public Transport this[int index]

{

get

{

return swetnessesCatalog[index];

}

}

public void addNewNote(Transport sweetness)

{

swetnessesCatalog.Add(sweetness);

}

public List<Transport> GetSweetnesses()

{

return swetnessesCatalog;

}

public Transport GetSweetness(int index)

{

return swetnessesCatalog[index];

}

}

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

Catalog catalog;

List<Gift> gifts;

Gift gift = null;

private void button1\_Click(object sender, EventArgs e)

{

//создаем объект класса Transport

Transport sweetness = null;

switch (comboBox1.SelectedIndex)

{

case 0:

sweetness = new legkCar(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, textBox5.Text);

break;

case 1:

sweetness = new SportCar(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, checkBox1.Checked);

break;

case 2:

sweetness = new Moto(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, checkBox2.Checked);

break;

}

//добавляем объект в каталог

catalog.addNewNote(sweetness);

updateForm();

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

textBox5.Clear();

}

private void updateForm()

{

listBox1.DataSource = null;

listBox1.DataSource = catalog.GetSweetnesses();

listBox1.DisplayMember = "DisplayMember";

listBox3.DataSource = null;

listBox3.DataSource = catalog.GetSweetnesses();

listBox3.DisplayMember = "DisplayMember";

listBox4.DataSource = null;

listBox4.DataSource = gifts;

listBox4.DisplayMember = "DisplayMember";

if (gift != null)

{

listBox2.DataSource = null;

listBox2.DataSource = gift.getComposition();

listBox2.DisplayMember = "DisplayMember";

}

else

{

listBox2.DataSource = null;

}

}

private void button3\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

switch (comboBox2.SelectedIndex)

{

case 0: gift = new eco(textBox6.Text, textBox7.Text, checkBox3.Checked); break;

case 1: gift = new bisnes(textBox6.Text, textBox7.Text, checkBox3.Checked); break;

case 2: gift = new lux(textBox6.Text, textBox7.Text, checkBox3.Checked); break;

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (gift != null)

{

if (listBox1.SelectedIndex >= 0)

{

int tmp = listBox1.SelectedIndex;

gift.addElement(tmp, catalog);

}

}

updateForm();

}

private void button5\_Click(object sender, EventArgs e)

{

if (listBox2.SelectedIndex >= 0)

{

int num = listBox2.SelectedIndex;

gift.deleteElement(num);

updateForm();

}

}

private void button6\_Click(object sender, EventArgs e)

{

if (gift != null) gifts.Add(gift);

gift = null;

textBox6.Clear();

textBox7.Clear();

updateForm();

}

private void button7\_Click(object sender, EventArgs e)

{

if (gift != null) richTextBox1.Text = gift.showInfo();

}

private void button8\_Click(object sender, EventArgs e)

{

if (textBox8.Text != "")

{

gifts[listBox4.SelectedIndex].Address = textBox8.Text;

gifts[listBox4.SelectedIndex].ToSend(DateTime.Now, ref richTextBox2);

gifts.RemoveAt(listBox4.SelectedIndex);

textBox8.Text = "";

}

updateForm();

}

private void Form2\_Load(object sender, EventArgs e)

{

catalog = new Catalog();

gifts = new List<Gift>();

}

}

}