**Министерство высшего образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение Высшего образования

«**Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

Колледж информационных технологий и предпринимательства (КИТП)

**Кафедра физики и прикладной математики (ФиПМ)**

Задание 13

Тема: «**Абстрактные классы**»

Вариант 4

Выполнил студент группы ПКсп-120

Кирсанов К.А.

Принял(а) Кабанова М.Ю.

Владимир 2022

**Постановка задачи:**

Требуется реализовать два варианта приложений для каждой задачи: консольное приложение и приложение с графическим интерфейсом на языке программирования C#.

1) Создать абстрактный класс Trans с методами позволяющим вывести на экран информацию о транспортном средстве, а также определить грузоподъемность транспортного средства.

2) Создать производные классы: Легковая\_машина (марка, номер, скорость, грузоподъемность), Мотоцикл (марка, номер, скорость, грузоподъемность, наличие коляски, при этом если коляска отсутствует, то грузоподъемность равна 0), Грузовик (марка, номер, скорость, грузоподъемность, наличие прицепа, при этом если есть прицеп, то грузоподъемность увеличивается в два раза) со своими методами вывода информации на экран, и определения грузоподъемности.

3) Создать базу (массив) из n машин, вывести полную информацию из базы на экран, а также организовать поиск машин, удовлетворяющих требованиям грузоподъемности.

**Практическая часть:**

**Листинг консольного приложения**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practicum13

{

//абстрактный класс

abstract class Trans

{

public abstract int LoadCapacity { get; }

public abstract void OutInfo();

public abstract void GetLoadCapacity();

}

//реализующие классы

class Car : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Car(string brand, int number, int speed, int loadCapacity)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

}

public override void OutInfo()

{

Console.WriteLine("\nИнформация о машине: ");

Console.WriteLine($"Марка: {brand}");

Console.WriteLine($"Номер: {number}");

Console.WriteLine($"Скорость: {speed}");

Console.WriteLine($"Грузоподъемность: {loadCapacity}");

}

public override void GetLoadCapacity()

{

Console.WriteLine($"Грузоподъемность: {loadCapacity}");

}

}

class Motorcycle : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

bool isСarriage;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Motorcycle(string brand, int number, int speed, int loadCapacity, bool isCarriage)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

this.isСarriage = isCarriage;

}

public override void OutInfo()

{

Console.WriteLine("\nИнформация о мотоцикле: ");

Console.WriteLine($"Марка: {brand}");

Console.WriteLine($"Номер: {number}");

Console.WriteLine($"Скорость: {speed}");

Console.WriteLine($"Грузоподъемность: {loadCapacity}");

string carriage = (isСarriage) ? "Да" : "Нет";

Console.WriteLine($"Есть коляска: {carriage}");

}

public override void GetLoadCapacity()

{

if (isСarriage)

{

Console.WriteLine($"Так как есть коляска, то грузоподъемность мотоцикла {loadCapacity}");

}

else

{

loadCapacity = 0;

Console.WriteLine($"Так как коляски нет, то грузоподъемность мотоцикла {loadCapacity}");

}

}

}

class Truck : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

bool isTrailer;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Truck(string brand, int number, int speed, int loadCapacity, bool isTrailer)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

this.isTrailer = isTrailer;

}

public override void OutInfo()

{

Console.WriteLine("\nИнформация о грузовике: ");

Console.WriteLine($"Марка: {brand}");

Console.WriteLine($"Номер: {number}");

Console.WriteLine($"Скорость: {speed}");

Console.WriteLine($"Грузоподъемность: {loadCapacity}");

string trailer = (isTrailer) ? "Да" : "Нет";

Console.WriteLine($"Есть прицеп: {trailer}");

}

public override void GetLoadCapacity()

{

if (isTrailer)

{

loadCapacity \*= 2;

Console.WriteLine($"Так как есть прицеп, то грузоподъемность грузовика: {loadCapacity}");

}

else

{

Console.WriteLine($"Так как прицепа нет, то грузоподъемность грузовика: {loadCapacity}");

}

}

}

internal class Program

{

static void Main(string[] args)

{

List <Trans> transport = new List<Trans>();

int amount;

while (true)

{

try

{

Console.Write("Введите количество транспортных средств: ");

amount = int.Parse(Console.ReadLine());

if (amount < 1) throw new Exception("Значение не может равняться нулю или меньше!");

break;

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите целочисленное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

for (int i = 0; i < amount; i++)

{

int choose;

while (true)

{

try

{

Console.WriteLine("\nКакое т/с нужно добавить?\n1 - Легковая машина, 2 - Мотоцикл, 3 - Грузовик");

Console.Write("Введите нужное значение: ");

choose = int.Parse(Console.ReadLine());

if (choose < 1 || choose > 3) throw new Exception("Введите цифру в диапазоне от 1 до 3!");

break;

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите целочисленное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

string brand;

int number = 0, speed = 0, loadCapacity = 0;

switch (choose)

{

case 1:

Console.Write("Введите марку т/с: ");

brand = Console.ReadLine();

try

{

Console.Write("Введите номер т/с: ");

number = int.Parse(Console.ReadLine());

Console.Write("Введите скорость т/с: ");

speed = int.Parse(Console.ReadLine());

Console.Write("Введите грузоподъемность т/с: ");

loadCapacity = int.Parse(Console.ReadLine());

if (number < 0 || speed < 0 || loadCapacity < 0) throw new Exception("Значение не может быть отрицательным!");

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите численное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Car car = new Car(brand, number, speed, loadCapacity);

transport.Add(car);

break;

case 2:

Console.Write("Введите марку т/с: ");

brand = Console.ReadLine();

bool isCarriage = true;

try

{

Console.Write("Введите номер т/с: ");

number = int.Parse(Console.ReadLine());

Console.Write("Введите скорость т/с: ");

speed = int.Parse(Console.ReadLine());

Console.Write("Имеется ли у т/с мотоколяска? (Да/Нет): ");

string answ = Console.ReadLine();

switch (answ)

{

case "Да":

isCarriage = true;

try

{

Console.Write("Введите грузоподъемность: ");

loadCapacity = int.Parse(Console.ReadLine());

if (loadCapacity < 0) throw new Exception("Грузоподъемность не может быть отрицательной!");

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Нужно ввести численное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

break;

case "Нет":

isCarriage = false;

break;

}

if (number < 0 || speed < 0 || loadCapacity < 0) throw new Exception("Значение не может быть отрицательным!");

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите численное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Motorcycle bike = new Motorcycle(brand, number, speed, loadCapacity, isCarriage);

bike.GetLoadCapacity();

transport.Add(bike);

break;

case 3:

Console.Write("Введите марку т/с: ");

brand = Console.ReadLine();

bool isTrailer = true;

try

{

Console.Write("Введите номер т/с: ");

number = int.Parse(Console.ReadLine());

Console.Write("Введите скорость т/с: ");

speed = int.Parse(Console.ReadLine());

Console.Write("Введите грузоподъемность т/с: ");

loadCapacity = int.Parse(Console.ReadLine());

if (number < 0 || speed < 0 || loadCapacity < 0) throw new Exception("Значение не может быть отрицательным!");

Console.Write("Имеется ли у грузовика прицеп? (Да/Нет): ");

string str = Console.ReadLine();

switch (str)

{

case "Да":

isTrailer = true;

break;

case "Нет":

isTrailer = false;

break;

}

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите численное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Truck truck = new Truck(brand, number, speed, loadCapacity, isTrailer);

truck.GetLoadCapacity();

transport.Add(truck);

break;

}

}

Console.WriteLine("\nИнформация из базы данных по всем внесенным т/с\n");

foreach (var e in transport)

{

e.OutInfo();

}

int loadCap;

while (true)

{

try

{

Console.Write("\nВведите требуемое значение грузоподъемности: ");

loadCap = int.Parse(Console.ReadLine());

if (loadCap < 0) throw new Exception("Значение грузоподъемности не может быть отрицательным!");

break;

}

catch (FormatException)

{

Console.WriteLine("Ошибка! Введите численное значение");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

int count = 0;

Console.WriteLine("Т/с, с подходящей грузоподъемностью:\n");

foreach (var t in transport)

{

if (t.LoadCapacity >= loadCap)

{

t.OutInfo();

count++;

}

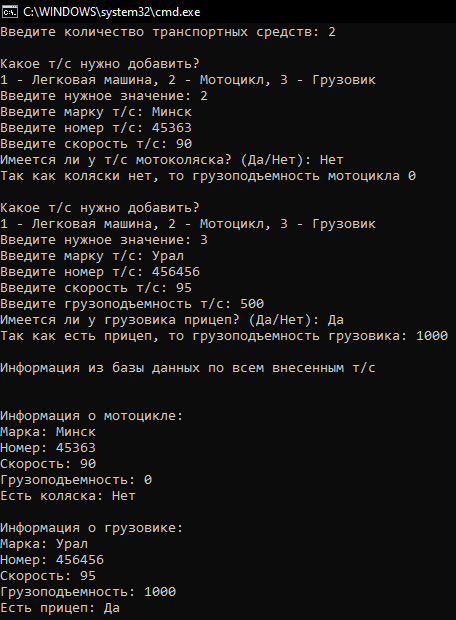
}

if (count == 0) Console.WriteLine("Подходящих т/с не обнаружено");

}

}

}

.

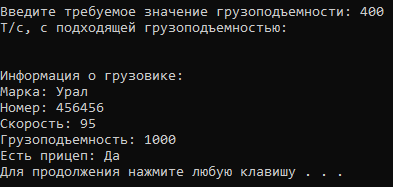


Рисунок 1 и 2 – Результат выполнения первого задания

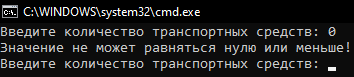


Рисунок 3 – Исключение при вводе нуля или отрицательного числа

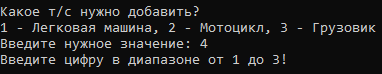


Рисунок 4 – Исключение при вводе значения, отличного от 1-3

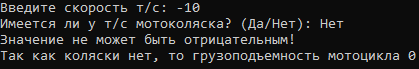


Рисунок 5 – Исключения при вводе отрицательного значения скорости



Рисунок 6 – Исключения при вводе отрицательного значения грузоподъемности

**Листинг графического приложения**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace Practicum13\_Task1\_WF

{

public partial class Form1 : Form

{

List<Trans> transport = new List<Trans>();

int amount;

int count;

string brand;

int number = 0, speed = 0, loadCapacity = 0;

bool isCarriage = true;

bool isTrailer = true;

public Form1()

{

InitializeComponent();

}

private void CountIsFull()

{

if (count == amount)

{

MessageBox.Show("Необходимое кол-во т/с внесено в базу данных", "Успех!");

button2.Enabled = false;

button3.Enabled = false;

button4.Enabled = false;

groupBox3.Enabled = false;

groupBox4.Enabled = false;

groupBox5.Enabled = false;

richTextBox1.Clear();

foreach (var e in transport)

{

richTextBox1.Text += e.OutInfo();

}

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

groupBox4.Enabled = false;

groupBox5.Enabled = false;

button2.Enabled = true;

button3.Enabled = false;

button4.Enabled = false;

numericUpDown4.Enabled = true;

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

groupBox4.Enabled = true;

groupBox5.Enabled = false;

button2.Enabled = false;

button3.Enabled = true;

button4.Enabled = false;

numericUpDown4.Enabled = false;

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

groupBox4.Enabled = false;

groupBox5.Enabled = true;

button2.Enabled = false;

button3.Enabled = false;

button4.Enabled = true;

numericUpDown4.Enabled = true;

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

isCarriage = true;

numericUpDown4.Enabled = true;

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

isCarriage = false;

numericUpDown4.Enabled = false;

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

isTrailer = true;

}

private void radioButton7\_CheckedChanged(object sender, EventArgs e)

{

isTrailer = false;

}

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

{

brand = textBox1.Text;

number = (int)numericUpDown2.Value;

speed = (int)numericUpDown3.Value;

loadCapacity = (int)numericUpDown4.Value;

Motorcycle bike = new Motorcycle(brand, number, speed, loadCapacity, isCarriage);

transport.Add(bike);

count++;

textBox1.Clear();

radioButton4.Checked = false;

radioButton5.Checked = false;

numericUpDown2.Value = 0;

numericUpDown3.Value = 0;

numericUpDown4.Value = 0;

numericUpDown4.Enabled = true;

radioButton1.Checked = false;

radioButton2.Checked = false;

radioButton3.Checked = false;

CountIsFull();

}

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

{

brand = textBox1.Text;

number = (int)numericUpDown2.Value;

speed = (int)numericUpDown3.Value;

loadCapacity = (int)numericUpDown4.Value;

Truck truck = new Truck(brand, number, speed, loadCapacity, isTrailer);

MessageBox.Show(truck.GetLoadCapacity());

transport.Add(truck);

count++;

textBox1.Clear();

radioButton6.Checked = false;

radioButton7.Checked = false;

numericUpDown2.Value = 0;

numericUpDown3.Value = 0;

numericUpDown4.Value = 0;

radioButton1.Checked = false;

radioButton2.Checked = false;

radioButton3.Checked = false;

CountIsFull();

}

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

{

int loadCap = (int)numericUpDown5.Value;

richTextBox2.Clear();

int count = 0;

foreach (var t in transport)

{

if (t.LoadCapacity >= loadCap)

{

richTextBox2.Text += t.OutInfo();

count++;

}

}

if (count == 0) MessageBox.Show("Подходящих т/с не найдено", "Ошибка!");

}

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

{

try

{

amount = (int)numericUpDown1.Value;

if (amount <= 0) throw new Exception("Количество т/с не может быть 0-м или отрицательным");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!");

return;

}

button2.Enabled = true;

button3.Enabled = true;

button4.Enabled = true;

groupBox3.Enabled = true;

groupBox4.Enabled = true;

groupBox5.Enabled = true;

count = 0;

}

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

{

brand = textBox1.Text;

number = (int)numericUpDown2.Value;

speed = (int)numericUpDown3.Value;

loadCapacity = (int)numericUpDown4.Value;

Car car = new Car(brand, number, speed, loadCapacity);

transport.Add(car);

count++;

textBox1.Clear();

numericUpDown2.Value = 0;

numericUpDown3.Value = 0;

numericUpDown4.Value = 0;

radioButton1.Checked = false;

radioButton2.Checked = false;

radioButton3.Checked = false;

CountIsFull();

}

}

//абстрактный класс

abstract class Trans

{

public abstract int LoadCapacity { get; }

public abstract string OutInfo();

public abstract string GetLoadCapacity();

}

//реализующие классы

class Car : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Car(string brand, int number, int speed, int loadCapacity)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

}

public override string OutInfo()

{

return $"\n------------------------\n" +

$"Информация о машине:\n" +

$"Марка: {brand}\n" +

$"Номер: {number}\n" +

$"Скорость: {speed}\n" +

$"Грузоподъемность: {loadCapacity}";

}

public override string GetLoadCapacity()

{

return $"Грузоподъемность: {loadCapacity}";

}

}

class Motorcycle : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

bool isСarriage;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Motorcycle(string brand, int number, int speed, int loadCapacity, bool isCarriage)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

this.isСarriage = isCarriage;

}

public override string OutInfo()

{

string carriage = (isСarriage) ? "Да" : "Нет"; ;

return $"\n------------------------\n" +

$"Информация о мотоцикле:\n" +

$"Марка: {brand}\n" +

$"Номер: {number}\n" +

$"Скорость: {speed}\n" +

$"Грузоподъемность: {loadCapacity}\n" +

$"Есть коляска: {carriage}";

}

public override string GetLoadCapacity()

{

if (isСarriage)

{

return $"Так как есть коляска, то грузоподъемность мотоцикла {loadCapacity}";

}

else

{

loadCapacity = 0;

return $"Так как коляски нет, то грузоподъемность мотоцикла {loadCapacity}";

}

}

}

class Truck : Trans

{

string brand;

int number;

int speed;

int loadCapacity;

bool isTrailer;

public override int LoadCapacity

{

get { return loadCapacity; }

}

public Truck(string brand, int number, int speed, int loadCapacity, bool isTrailer)

{

this.brand = brand;

this.number = number;

this.speed = speed;

this.loadCapacity = loadCapacity;

this.isTrailer = isTrailer;

}

public override string OutInfo()

{

string trailer = (isTrailer) ? "Да" : "Нет";

return $"\n------------------------\n" +

$"Информация о грузовике:\n" +

$"Марка: {brand}\n" +

$"Номер: {number}\n" +

$"Скорость: {speed}\n" +

$"Грузоподъемность: {loadCapacity}\n" +

$"Есть прицеп: {trailer}";

}

public override string GetLoadCapacity()

{

if (isTrailer)

{

loadCapacity \*= 2;

return $"Так как есть прицеп, то грузоподъемность грузовика: {loadCapacity}";

}

else

{

return $"Так как прицепа нет, то грузоподъемность грузовика: {loadCapacity}";

}

}

}

}

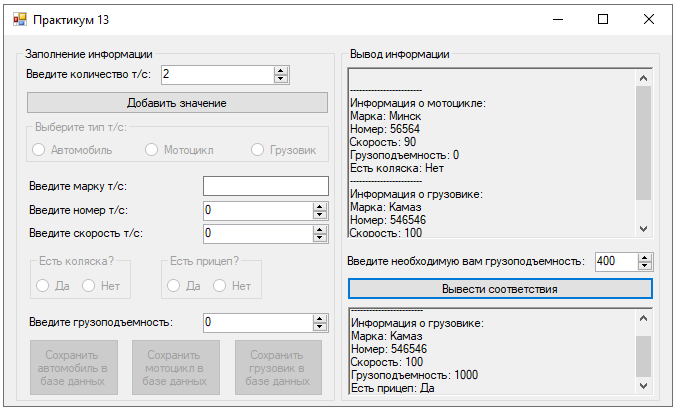


Рисунок 7 – Графический интерфейс задания 1

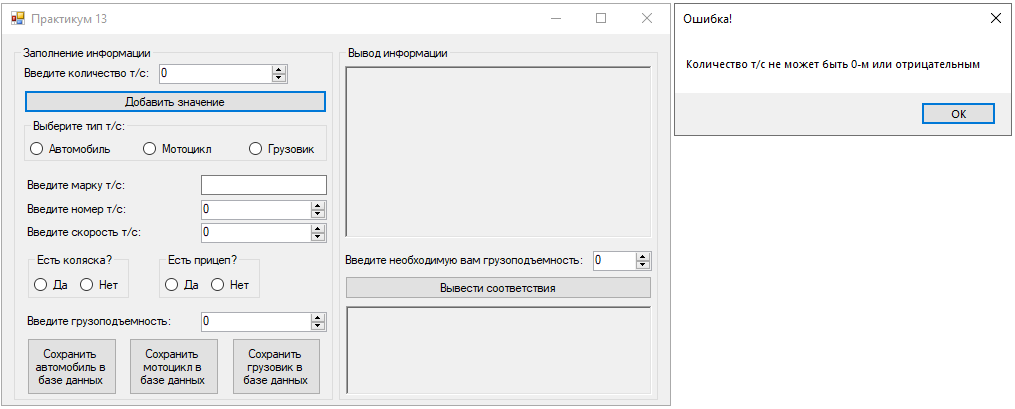


Рисунок 8 – Сообщение при вводе 0-го или отрицательного количества т/с

**Ссылка на github:**

Все работы будут храниться в следующих репозиториях:

1. репозиторий со всеми консольными реализациями заданий

https://github.com/sotonoche/Practicum13

1. репозиторий со всеми реализациями заданий с графическим интерфейсом

https://github.com/sotonoche/Practicum13