МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления



**ОТЧЁТ**

**по практической работе №5**

**по дисциплине:** Программирование.

Вариант: №1

Выполнили работу: Проверил работу:

Студенты гр. АП-127, АВТФ Эстрайх Игорь Викторович

Мавлонхонов Жобирхон, «\_\_» \_\_\_\_\_\_\_\_ 20\_\_г.

[Карпова](https://dispace.edu.nstu.ru/personal/index/109451) Яна, \_\_\_\_\_\_\_\_\_\_\_

Квашнёв Денис (подпись)

«13» октября 2022г

Новосибирск 2022г.

**Постановка задачи**

Использовать json-сериализацию для программы из лабораторной №1.

**Листинг программы**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Collections.Specialized;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Numerics;  using System.Drawing;  using System.Runtime.ConstrainedExecution;  using System.Runtime.Intrinsics;  using static lab3sem3.Class1;  using System.Timers;  using System.Text.Json;  using System.IO;  namespace lab3sem3  {  public class Class1  {  public enum ExceptionsFor  {  AutoNumber,  AutoWeight,  AutoVolume,  AutoFuel,  HorseAge,  HorseWeight,  CarriageNumber,  CarriageWeight,  }  static void Outputing(ExceptionsFor x)  {  if (x == ExceptionsFor.AutoNumber) { Console.WriteLine("Введите номер автомобиля:"); }  if (x == ExceptionsFor.AutoWeight) { Console.WriteLine("Введите массу автомобиля ( от 400 до 10000 кг):"); }  if (x == ExceptionsFor.AutoVolume) { Console.WriteLine("Введите объём бака (от 20 до 150 л): "); }  if (x == ExceptionsFor.AutoFuel) { Console.WriteLine("Введите тип топлива ( 1-бензин, 2-газ, 3-дизельное топливо):"); }  if (x == ExceptionsFor.HorseAge) { Console.WriteLine("Введите возраст лошади( от 20 до 62:)"); }  if (x == ExceptionsFor.HorseWeight) { Console.WriteLine("Введите массу лошади( от 300 до 1000 кг ):"); }  if (x == ExceptionsFor.CarriageNumber) { Console.WriteLine("Введите номер повозки: "); }  if (x == ExceptionsFor.CarriageWeight) { Console.WriteLine("Введите массу повозки( от 100 до 1000 кг ): "); }  }  static double Exceptions(double a, ExceptionsFor x)  {  if (x == ExceptionsFor.AutoNumber) { if (a < 0) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.AutoWeight) { if (a < 400 || a > 10000) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.AutoVolume) { if (a < 20 || a > 150) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.AutoFuel) { if (a < 1 || a > 3) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.HorseAge) { if (a < 20 || a > 62) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.HorseWeight) { if (a < 50 || a > 1600) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.CarriageNumber) { if (a < 0) throw new ArgumentOutOfRangeException(); else return a; }  if (x == ExceptionsFor.CarriageWeight) { if (a < 100 || a > 1000) throw new ArgumentOutOfRangeException(); else return a; }  return 0;  }  static double OutputExceptions(ExceptionsFor x)  {  double a;  for (; ; )  {  Outputing(x);  a = Convert.ToDouble(Console.ReadLine());  try  {  a = Exceptions(a, x);  break;  }  catch (ArgumentOutOfRangeException)  {  Console.WriteLine("Возникло исключение ArgumentOutOfRangeException. Попробуйте снова.");  continue;  }  }  return a;  }  public class Automobile  {  protected int numberauto; // номер участника  public double weight; // масса автомобиля  public double volume; // объём бака  public int fuel; // 1 - бензин , 2 - газ, 3 - дизель  protected double discar; // кол-во км, которое сможет проехать машина  public int NumberAuto  {  get { return numberauto; }  set  {  ExceptionsFor x = ExceptionsFor.AutoNumber;  numberauto = (int)Exceptions(value, x);  }  }  public double Weight  {  get { return weight; }  set  {  ExceptionsFor x = ExceptionsFor.AutoWeight;  weight = Exceptions(value, x);  }  }  public int Fuel  {  get { return fuel; }  set  {  ExceptionsFor x = ExceptionsFor.AutoFuel;  fuel = (int)Exceptions(value, x);  }  }  public double Volume  {  get { return volume; }  set  {  ExceptionsFor x = ExceptionsFor.AutoVolume;  volume = Exceptions(value, x);  }  }  public double Discar  {  get { return discar; }  set  {  double y = 0.0; // расход топлива  switch (fuel)  {  case 1: y = 8.0 + (3.0 \* weight / 1000.0); break;  case 2: y = 8.0 + (4.0 \* weight / 1000.0); break;  case 3: y = 4.0 + (3.0 \* weight / 1000.0); break;  }  discar = volume \* 100.0 / y;  }  }  }  public class Horse  {  public int age;//возраст  public double weightHorse;// вес коня  static protected double dishorse; // коэффициент уставания лошади  static public int horseCount = 0;  public int HorseCount  {  get { return horseCount; }  set  {  horseCount = value;  }  }  public int Age  {  get { return age; }  set  {  ExceptionsFor x = ExceptionsFor.HorseAge;  age = (int)Exceptions(value, x);  }  }  public double WeightHorse  {  get { return weightHorse; }  set  {  ExceptionsFor x = ExceptionsFor.HorseWeight;  weightHorse = Exceptions(value, x);  }  }  public double Dishorse  {  set  {  dishorse = horsedistance();  }  get { return dishorse; }  }  public double horsedistance()  {  double y1 = 0.0, y2 = 0.0;  y1 = -(age \* age) + 40 \* age + 100; // преобразование возраста в коэффициент  if (400 < weightHorse && weightHorse <= 1000)  {  y2 = -0.0007 \* weightHorse \* weightHorse + 0.56 \* weightHorse + 188;  }  if (30 <= weightHorse && weightHorse < 300)  {  y2 = weightHorse;  }  if (300 <= weightHorse && weightHorse <= 400)  {  y2 = 300.0;  }  double dishorse = y1 + y2;  return dishorse;  }  };  public class Carriage : Horse  {  private int numbercarriage; // номер повозки  protected double distance; // максимальное расстояние , которое может пройти повозка  static private double weightCarriage; // масса повозки  public int NumberCarriage  {  get { return numbercarriage; }  set  {  ExceptionsFor x = ExceptionsFor.CarriageNumber;  numbercarriage = (int)Exceptions(value, x);  }  }  public double WeightCarriage  {  get { return weightCarriage; }  set  {  ExceptionsFor x = ExceptionsFor.CarriageWeight;  weightCarriage = Exceptions(value, x);  }  }  public double Distance  {  set { distance = disdis(); }  get { return distance; }  }  public double disdis()  {  double distance = (dishorse / 3) / (weightCarriage \* 0.3 / (3 \* Math.Pow(horseCount, 0.5)));  return distance;  }  public double CarriageDistance(double m, double w1, int k)  {  double dis = (m / 3) / (w1 \* 0.3 / (3 \* Math.Pow(k, 0.5)));  return dis;  }  }  public class Competitions  {  public List<Automobile> VectorAuto = new List<Automobile>();  public List<Carriage> VectorCarriage = new List<Carriage>();  public int AutoCount = 0;  public int CarriageCount = 0;  public void AddCarriage() //adding carriage  {  int k = 1;  Carriage b = new Carriage(); // создание объекта для дальнейшего добавления его в вектор VectorCarriage  double w2 = 0;  double value = 0;  b.WeightCarriage = value;  List<Carriage> VectorHorse = new List<Carriage>(); // вектор для хранения объектов типа Horse  List<double> VectorMimK = new List<double>(); // вектор для хранения коэффициентов уставания  double d = 0.0;  int HorseCount = 0;  List<double> VectorMinK = new List<double>();// вектор для хранения коэффициентов уставания  do  {  if (k == 1)  {  Carriage l = new Carriage(); // создание объекта типа Horse для дальнейшего довавление в вектор VectorHorse  l.Age = (int)value;  l.WeightHorse = value;  w2 = l.WeightHorse;  d = l.horsedistance();  VectorMinK.Add(d);  HorseCount++;  }  if (k == 2)  {  int num;  for (; ; )  {  Console.WriteLine("Введите порядковый номер коня, которого нужно удалить: ");  num = Convert.ToInt32(Console.ReadLine());  if (num <= 0 || num > HorseCount) { Console.WriteLine("Такого коня нет! Введите номер коня заново."); continue; }  else break;  }  VectorHorse.RemoveAt(num);  HorseCount--;  }  Console.WriteLine("1 - Добавить ещё лошадь; 2 - Удалить коня по порядковому номеру ; любая другая цифра - выйти");  k = Convert.ToInt32(Console.ReadLine());  } while (k == 1 || k == 2);  VectorMinK.Sort();  double dis = b.CarriageDistance(VectorMinK[0], w2, HorseCount); // вызов метода получения минимального коэффициента уставания  b.NumberCarriage = ++CarriageCount;  b.Distance = dis;  VectorCarriage.Add(b);  }  public void AddAuto() //adding automobile  {  Automobile a = new Automobile();  double value = 0;  a.Weight = value;  a.Volume = value;  a.Fuel = (int)value;  a.NumberAuto = ++AutoCount;  a.Discar = value;  VectorAuto.Add(a);  }  public int GetKolAuto() { return AutoCount; } //getting  public int GetKolCarriage() { return CarriageCount; } //getting quantity of carriages  public void DeleteAuto(int index) { VectorAuto.RemoveAt(index); AutoCount--; } //removing automobile  public void DeleteCarriage(int index) { VectorCarriage.RemoveAt(index); CarriageCount--; } //removing carriage  public void contest() //conducting the competiton  {  for (int i = 0; i < GetKolAuto() - 1; i++)  {// сортировка автомобилей методом пузырька от максимального расстояния до минимпльного  for (int j = GetKolAuto() - 1; j > i; j--)  {  if (VectorAuto[j - 1].Discar < VectorAuto[j].Discar) // если текущий элемент больше предыдущего  {  Automobile temp = VectorAuto[j - 1]; // меняем их местами  VectorAuto[j - 1] = VectorAuto[j];  VectorAuto[j] = temp;  }  }  }  for (int i = 0; i < GetKolCarriage() - 1; i++)  {// сортировка автомобилей методом пузырька от максимального расстояния до минимпльного  for (int j = GetKolCarriage() - 1; j > i; j--)  {  if (VectorCarriage[j - 1].Distance < VectorCarriage[j].Distance) // если текущий элемент больше предыдущего  {  Carriage temp2 = VectorCarriage[j - 1]; // меняем их местами  VectorCarriage[j - 1] = VectorCarriage[j];  VectorCarriage[j] = temp2;  }  }  }  }  }  public static void Main(string[] args)  {  Carriage b = new Carriage();  b.NumberCarriage = 3;  b.WeightCarriage = 200;  b.Age = 39;  b.WeightHorse = 100;  b.HorseCount = 1;  b.Dishorse = 0;  b.Distance = 0;  var options = new JsonSerializerOptions  {  WriteIndented = true  };  string jsonString = JsonSerializer.Serialize<Carriage>(b, options);  Console.WriteLine("json.Serialize:");  File.WriteAllText("carriage.json", jsonString);  Console.WriteLine(File.ReadAllText("carriage.json"));  Carriage? restoredCarriage = JsonSerializer.Deserialize<Carriage>(jsonString);  Console.WriteLine("\njson.Deserialize:");  Console.WriteLine($"NumberCarriage: {restoredCarriage?.NumberCarriage}");  Console.WriteLine($"WeightCarriage:{restoredCarriage?.WeightCarriage}");  Console.WriteLine($"Distance:{restoredCarriage?.Distance}");  Console.WriteLine($"Age:{restoredCarriage?.Age}");  Console.WriteLine($"WeightHorse:{restoredCarriage?.WeightHorse}");  Console.WriteLine($"Dishorse:{restoredCarriage?.Dishorse}");  Console.WriteLine($"HorseCount: {restoredCarriage?.HorseCount}");  }  }  } |

**Результат работы программы**

