Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «СПП»

за 5-й семестр

Выполнил:

студент 2 курса

группы ПО-3 (1)

Афанасьев В.В.

Проверил:

Крощенко А.А.

Брест, 2020

**Цель работы:** приобрести базовые навыки создания и использования классов в программах на языке программирования C#.

**Вариант:** 2

**Задание 1:**

**Реализовать простой класс.**Требования к выполнению: **•** Реализовать пользовательский класс по варианту. **•** Создать другой класс с методом main, в котором будут находится примеры использованияпользовательского класса.Для каждого класса: **•** Создать поля классов **•** Создать методы классов **•** Добавьте необходимые get и set методы (по необходимости) **•** Укажите соответствующие модификаторы видимости **•** Добавьте конструкторы **•** Переопределить методы toString() и equals()

2) *Равносторонний треугольник, заданный длинами сторон* – Предусмотреть возможностьопределения площади и периметра, а так же логический метод, определяющий существует илитакой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

**Задание 2:**

***Разработать автоматизированную систему на основе некоторой структуры дан- ных,  
манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных.***  
Требования к выполнению:  
*•* Задание посвящено написанию классов, решающих определенную задачу автоматизации;  
*•* Данные для программы загружаются из файла (формат произволен). Файл создать и написать  
вручную.

2) *Автоматизированная система проката автомобилей*Составить программу, которая хранит и обрабатывает информацию о прокате автомобилей. Окаждом автомобиле (Car) содержится следующая информация:• id;• Марка;• Модель;• Год выпуска;• Цвет;• Цена;• Регистрационный номер;• Номер машины.• ФИО лица, взявшего на прокат (при наличии);• Номер паспорта лица-арендатора (при наличии).Программа должна обеспечить вывод списков:• автомобилей;• автомобилей заданной марки;• автомобилей заданной модели, которые эксплуатируются больше n лет;• автомобилей заданного года выпуска, цена которых больше указанной;• автомобилей, взятых на прокат;• автомобилей, взятых на прокат с выводом личной информации об арендаторах.

**Код программы:**

**1)**

using System;

using System.ComponentModel.Design;

using System.Threading;

namespace spp\_lab3

{

class Program

{

static void Main(string[] args)

{

try

{

Equilateral obj1 = new Equilateral();

Equilateral obj2 = new Equilateral(4, 4, 4);

if (obj1.Equals(obj2))

{

Console.WriteLine("Equils");

}

else Console.WriteLine("Not equils");

Console.WriteLine(obj1.ToString());

}

catch(Exception Ex)

{

Console.WriteLine(Ex.Message);

}

}

}

class Equilateral

{

private double Side { get; set; }

private bool Checking(double \_side1, double \_side2, double \_side3)

{

if (\_side1 == \_side2 && \_side2 == \_side3)

{

return true;

}

else return false;

}

public Equilateral()

{

Side = 1;

}

public Equilateral(double \_side1, double \_side2, double \_side3)

{

if (Checking(\_side1, \_side2, \_side3))

{

Side = \_side1;

}

else throw new Exception("Wrong values!");

}

public double Area()

{

return Math.Pow(Side, 2) \* Math.Sqrt(3)/4;

}

public double Perimeter()

{

return Side \* 3;

}

public override bool Equals(object obj)

{

Equilateral other = obj as Equilateral;

return Side == other.Side;

}

public override string ToString()

{

return new string("Side is " + Side);

}

}

}

**2)**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Xml;

using System.Xml.Serialization;

namespace task2

{

class Program

{

static void Main(string[] args)

{

CarDB db = new CarDB();

// writing for testing

//Car obj1 = new Car()

//{

// Id = 1,

// Brand = "Lada",

// Model = "154",

// Year = 1980,

// Color = "green",

// Price = 300,

// CarNum = 6544,

// RegNum = 45475

//};

//db.Add(obj1);

//Car obj2 = new Car()

//{

// Id = 2,

// Brand = "Mercedes",

// Model = "154",

// Year = 1995,

// Color = "gray",

// Price = 2500,

// CarNum = 6344,

// RegNum = 454375

//};

//db.Add(obj2);

//Car obj3 = new Car()

//{

// Id = 3,

// Brand = "Renault",

// Model = "567",

// Year = 2005,

// Color = "green",

// Price = 5500,

// CarNum = 6544,

// RegNum = 45475

//};

//db.Add(obj3);

//Car obj4 = new Car()

//{

// Id = 4,

// Brand = "Lada5",

// Model = "154",

// Year = 1985,

// Color = "blue",

// Price = 350,

// CarNum = 6544,

// RegNum = 45475,

// FIO\_p = "Astapov Peter",

// PassNum\_p = 865849

//};

//db.Add(obj4);

//db.Writer();

Console.WriteLine("\nModel with old");

foreach (var item in db.GetModelWithOld("154", 5))

{

item.Show();

}

Console.WriteLine("\nYear with price");

foreach (var item in db.GetYearWithPrice(1985, 300))

{

item.Show();

}

Console.WriteLine("\nHire with info");

foreach (var item in db.GetCarsHireWithInfo())

{

item.Show();

}

}

}

[Serializable]

public class Car

{

public int Id { get; set; }

public string Brand { get; set; }

public string Model { get; set; }

public int Year { get; set; }

public string Color { get; set; }

public int Price { get; set; }

public int CarNum { get; set; }

public int RegNum { get; set; }

public string FIO\_p { get; set; }

public int PassNum\_p { get; set; }

public void Show()

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("ID: " + Id);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("Brand: " + Brand);

Console.WriteLine("Model: " + Model);

Console.WriteLine("Year: " + Year);

Console.WriteLine("Color: " + Color);

Console.WriteLine("Price: " + Price);

Console.WriteLine("Car number: " + CarNum);

Console.WriteLine("Registration number: " + RegNum);

Console.WriteLine("FIO: " + FIO\_p);

Console.WriteLine("Password number: " + PassNum\_p);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

}

class CarDB

{

private List<Car> cars = new List<Car>();

public void Add(Car \_car)

{

cars.Add(\_car);

}

public CarDB()

{

if(File.Exists("cars.xml"))

{

Reader();

}

}

public void Reader()

{

XmlSerializer formatter = new XmlSerializer(typeof(Car[]));

using (FileStream fs = new FileStream("cars.xml", FileMode.OpenOrCreate))

{

Car[] newcars = (Car[])formatter.Deserialize(fs);

cars = newcars.ToList();

}

}

public void Writer()

{

XmlSerializer formatter = new XmlSerializer(typeof(Car[]));

using (FileStream fs = new FileStream("cars.xml", FileMode.OpenOrCreate))

{

formatter.Serialize(fs, cars.ToArray());

}

}

public List<Car> GetCars()

{

return cars;

}

public List<Car> GetBrand(string \_Brand)

{

return cars.Where(i => i.Brand == \_Brand).ToList();

}

public List<Car> GetModelWithOld(string \_Model, int n)

{

int CurrentYear = 2020;

return cars.Where(i => (i.Model == \_Model) && ((CurrentYear - i.Year) > n)).ToList();

}

public List<Car> GetYearWithPrice(int \_Year, int \_Price)

{

return cars.Where(i => i.Year == \_Year && i.Price > \_Price).ToList();

}

public List<Car> GetCarsHire()

{

return cars.Where(i => i.FIO\_p != null).ToList();

}

public List<Car> GetCarsHireWithInfo()

{

var info = cars.Where(i => i.FIO\_p != null).ToList();

foreach (var item in info)

{

Console.WriteLine("ID: "+item.Id);

Console.WriteLine("FIO: " + item.FIO\_p);

Console.WriteLine("Password number: " + item.PassNum\_p);

}

return info;

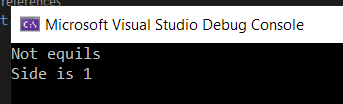
}

}

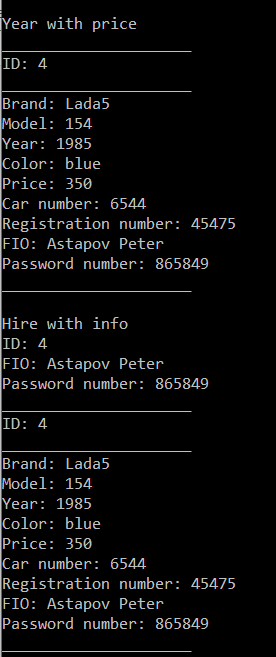
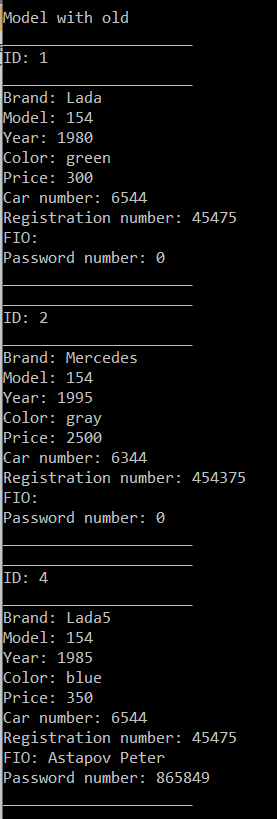
}

**Результаты работы:**

**1)**

****

**2)**

****

**Выводы:** в ходе выполнения лабораторной работы были получены базовые навыки создания и использования классов в программах на языке программирования C#.