Отчет по лабораторной работе № 4

“Исследование механизма множественного наследования”

по дисциплине

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Выполнил студент гр. ИВТ/б-22о

Горбенко К.Н.

Проверил:

Тимофеев И.С.

1. **Цель работы:** исследование основных средств описания класса, наследующего свойства нескольких базовых классов.
2. **Задание на работу**:   
    Описать интерфейс ввода-вывода. Описать иерархию классов, заданную по варианту. Для каждого класса описать конструкторы и деструктор (по необходимости), функции ввода и вывода значений полей. В каждом классе должны присутствовать минимум одно уникальное поле и один уникальный метод. Проиллюстрировать корректную работу механизма множественного наследования - для этого создать объекты базовых классов и заполнить их поля данными, вывести на печать. Создать объект класса-наследника, его поля заполнить значениями соответствующих полей базовых классов. Вывести на печать данные полученного объекта.   
    Условие для **варианта № 8**:

Базовые классы: Машина (марка, мощность, скорость), Транспорт (тоннаж, тип). Класс-наследник: Грузовик.

1. **Код программы на языке С#** Интерфейс Vehicle

|  |
| --- |
| public interface Vehicle  {  int NumberOFWheels { get; }   string Type { get; }   double Weight { get; }  } |

Класс Program:

|  |
| --- |
| class Program  {  static void Main()  {  var truck = new Truck();  var car = new Car();   WriteLine("Car status:\n" + car.Status());  WriteLine();  WriteLine("Truck status:\n" + truck.Status());   ReadLine();  }  } |

Класс Car:

|  |
| --- |
| public class Car  {  public string Model { get; private set; }   public int Power { get; private set; }   public int MaxSpeed { get; private set; }   public Car()  {  Model = "Mercedes";  Power = 500;  MaxSpeed = 300;  }   public Car(string model, int power, int maxSpeed)  {  if (string.IsNullOrWhiteSpace(model)) throw new ArgumentNullException(nameof(model));  if (power <= 0) throw new ArgumentException("Power can not be zero or less");  if (maxSpeed <= 0) throw new ArgumentException("Speed can not be zero or less");   Model = model;  Power = power;  MaxSpeed = maxSpeed;  }   public string Status()  {  return $"Model: {Model}; " +  $"Power: {Power}; " +  $"Max speed {MaxSpeed};";  }  } |

Класс Truck:

|  |
| --- |
| public class Truck : Car, Vehicle  {  //Properties are not inherited from interface,  //should be implemented manually   public int NumberOFWheels { get; private set; }   public string Type { get; private set; }   public double Weight { get; private set; }   public Truck() : base()  {  NumberOFWheels = 4;  Type = "tipper";  Weight = 1000;  }   public Truck(int numberOfWheels, string type, double weight, string model, int power, int maxSpeed)   : base(model, power, maxSpeed)  {  if (string.IsNullOrWhiteSpace(type)) throw new ArgumentNullException(nameof(type));  if (numberOfWheels % 2 != 0) throw new ArgumentException("Number of wheels could not be even");  if (weight <= 0) throw new ArgumentException("Weight could not be zero or less");   NumberOFWheels = numberOfWheels;  Type = type;  Weight = weight;  }   public new string Status()  {  return $"Number of wheels: {NumberOFWheels}; " +  $"Type: {Type}; " +  $"Weight: {Weight}; " +  $"Model: {Model}; " +  $"Power: {Power}; " +  $"Max speed: {MaxSpeed}";  }  } |

1. **Описание логики программы:** Так как в языке C# множественное наследование не реализовано, в лабораторной работе вместо второго базового класса был использован интерфейс. Чтобы реализовать поля, объявленные в интерфейсе, эти поля были объявлены в классе-наследнике.   
    Конструкторы класса-наследника обращаются к базовым конструкторам для инициализации полей, унаследованных от базового класса.
2. **Описание программы:** В программе создаются объекты базового класса и класса-наследника. Для каждого объекта вызывается метод Status(), который должен быть реализован в классах в соответствии с их полями. Метод Status() базового класса должен возвращать значения трех его полей, объявленных явно: Model, Power, MaxSpeed. Метод класса-наследника должен выводить значения полей, унаследованных от базового класса, а также значения полей, реализованных в соответствии с интерфейсом.
3. **Вывод:** в ходе лабораторной работы был изучен механизм множественного наследования. Учитывая особенности языка С#, все классы-наследники кроме первого были заменены на интерфейсы. В результате класс-наследник содержал как элементы, унаследованные от базового класса, так и элементы, реализованные в соответствии с интерфейсами.