Факультет компьютерных технологий

Лабораторная работа №4

«Разработка программ с использованием управляющих инструкций»

Вариант №10

Выполнил: студент гр. 981063 Ефименко Павел Викторович

Проверил: Воробей В.А.

Минск 2020

Задание №1 Разработка методов и использование их в

программах

Условие:

Необходимо реализовать классы из задания 1 лабораторной работы №3, и в каждом классе реализовать:

не менее 4 полей (к каждому полю соответствуют нужные свойства);

не менее 2 конструкторов (без параметров и со всеми параметрами);

не менее 4 методов (с возвращающим значением и без возвращающего значения)

Код:

public class Bus : IRoadTransportEquipment, IEquatable<Bus>

{

public Bus()

{

}

public Bus(string name, string color, string type, int power)

{

Name = name;

Color = color;

Type = type;

Power = power;

}

public string Name { get; set; }

public string Color { get; set; }

public string Type { get; set; }

public int Power { get; set; }

public override bool Equals(object obj)

{

return Equals(obj as Bus);

}

public bool Equals(Bus other)

{

return other != null &&

Name == other.Name &&

Color == other.Color &&

Type == other.Type &&

Power == other.Power;

}

public override int GetHashCode()

{

return HashCode.Combine(Name, Color, Type, Power);

}

public override string ToString()

{

return $"{Name}\t{Color}\t{Type}\t{Power}";

}

public static bool operator ==(Bus left, Bus right)

{

return EqualityComparer<Bus>.Default.Equals(left, right);

}

public static bool operator !=(Bus left, Bus right)

{

return !(left == right);

}

}

public class Car : IRoadTransportEquipment, IEquatable<Car>

{

public Car()

{

}

public Car(string name, string color, string type, int power)

{

Name = name;

Color = color;

Type = type;

Power = power;

}

public string Name { get; set; }

public string Color { get; set; }

public string Type { get; set; }

public int Power { get; set; }

public override bool Equals(object obj)

{

return Equals(obj as Car);

}

public bool Equals(Car other)

{

return other != null &&

Name == other.Name &&

Color == other.Color &&

Type == other.Type &&

Power == other.Power;

}

public override int GetHashCode()

{

return HashCode.Combine(Name, Color, Type, Power);

}

public override string ToString()

{

return $"{Name}\t{Color}\t{Type}\t{Power}";

}

public static bool operator ==(Car left, Car right)

{

return EqualityComparer<Car>.Default.Equals(left, right);

}

public static bool operator !=(Car left, Car right)

{

return !(left == right);

}

}

public interface IRoadTransportEquipment

{

public string Name { get; set; }

public string Color { get; set; }

public string Type { get; set; }

public int Power { get; set; }

}

public class Tractor : IRoadTransportEquipment, IEquatable<Tractor>

{

public Tractor()

{

}

public Tractor(string name, string color, string type, int power)

{

Name = name;

Color = color;

Type = type;

Power = power;

}

public string Name { get; set; }

public string Color { get; set; }

public string Type { get; set; }

public int Power { get; set; }

public override bool Equals(object obj)

{

return Equals(obj as Tractor);

}

public bool Equals(Tractor other)

{

return other != null &&

Name == other.Name &&

Color == other.Color &&

Type == other.Type &&

Power == other.Power;

}

public override int GetHashCode()

{

return HashCode.Combine(Name, Color, Type, Power);

}

public override string ToString()

{

return $"{Name}\t{Color}\t{Type}\t{Power}";

}

public static bool operator ==(Tractor left, Tractor right)

{

return EqualityComparer<Tractor>.Default.Equals(left, right);

}

public static bool operator !=(Tractor left, Tractor right)

{

return !(left == right);

}

}

Результат тестирования:

