Факультет компьютерных технологий

Лабораторная работа №3

«Разработка программ с использованием управляющих инструкций»

Вариант №10

Выполнил: студент гр. 981063 Ефименко Павел Викторович

Проверил: Воробей В.А.

Минск 2020

Задание №1 Проектирование класса

Условие:

Дорожно-транспортная техника

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);

}

}

Задание №2 Создание класса

Условие:

Класс: равнобедренный треугольник

Поля: боковая сторона и угол при основании

Операции:

увеличение/уменьшение размера в заданное количество раз;

вычисление длины биссектрисы, принадлежащей любому углу;

вычисление длины высот, принадлежащей любой стороне;

определение значений сторон.

Код:

public interface IIsosceles

{

double Side { get; set; }

double AngleAtTheBase { get;set; }

void IncreaseSize(double value);

void DecreaseSize(double value);

void CalcLengthBisector();

void CalculatingLengthHeights();

void DefiningValuesSides();

}

public class IsoscelesTriangle : IIsosceles

{

public IsoscelesTriangle(double side, double angleAtTheBase)

{

Side = side;

AngleAtTheBase = angleAtTheBase;

}

public double Side { get ; set ; }

public double AngleAtTheBase { get ; set ; }

public void CalcLengthBisector()

{

var bisector = FindBisector(this.Side, this.AngleAtTheBase);

Console.WriteLine($"Длина биссектрисы равна:\t{bisector}, " +

$"при угле равному:\t{this.AngleAtTheBase} (градусов)");

}

public void CalculatingLengthHeights()

{

var basisSide = FindBasisSide(Side, AngleAtTheBase);

var length = Math.Sqrt(Math.Pow(Side, 2) - Math.Pow((basisSide / 2), 2));

Console.WriteLine($"Длина высоты при сторонах равным {Side}&{basisSide} - равна:\t{length}");

}

public void DecreaseSize(double value)

{

this.Side /= value;

}

public void DefiningValuesSides()

{

var basisSize = FindBasisSide(this.Side, this.AngleAtTheBase);

Console.WriteLine($"Две стороны равнобедренного треугольника равны:\t{this.Side}, основание:{basisSize}");

}

public void IncreaseSize(double value)

{

this.Side \*= value;

}

private double FindBisector(double side, double angle)

{

return side \* Math.Sin(angle);

}

private double FindBasisSide(double sideA, double angle)

{

return 2 \* sideA \* Math.Cos(angle);

}

}

Результат:

