Факультет компьютерных технологий

Лабораторная работа №6

«Разработка программ с использованием управляющих инструкций»

Вариант №10

Выполнил: студент гр. 981063 Ефименко Павел Викторович

Проверил: Воробей В.А.

Минск 2020

Задание №1

Условие:

Реализовать классы по заданию (в задачах второго уровня должно присутствовать не менее 4 классов). Наследование применяется в логически нужных местах. Создать объект класса Щенок, используя классы Животное, Собака. Методы: вывести на консоль имя, подать голос, прыгать, бегать, кусать.

internal class Dog : IAnimal

{

public string Name { get; set; }

public virtual void Bark()

{

Console.WriteLine("Вуф-вуф");

}

public virtual void Bite()

{

Console.WriteLine("Кусь");

}

public virtual void Jump()

{

Console.WriteLine("Прыг");

}

public virtual void Run()

{

Console.WriteLine("Побежал за зайцем");

}

public override string ToString()

{

return $"{Name}";

}

}

public interface IAnimal

{

string Name { get; set; }

void Bark();

void Jump();

void Run();

void Bite();

}

internal class Puppy : Dog

{

public Puppy()

{

}

public Puppy(string name)

{

Name = name;

}

public override void Bark()

{

System.Console.WriteLine("Гав-гав");

}

public override void Bite()

{

System.Console.WriteLine("Кусь за ногу");

}

public override void Jump()

{

System.Console.WriteLine("Прыгаю на двух лапах");

}

public override void Run()

{

System.Console.WriteLine("Бегу за кошкой");

}

}

class Program

{

static void Main(string[] args)

{

var dog = new Dog { Name = "Собака" };

Console.WriteLine(dog);

dog.Run();

dog.Jump();

dog.Bite();

dog.Bark();

var puppy = new Puppy { Name = "Щенок" };

Console.WriteLine($"\n{puppy}");

puppy.Run();

puppy.Jump();

puppy.Bite();

puppy.Bark();

var husky = new HuskyPuppy();

Console.WriteLine($"\n{husky}");

husky.Run();

husky.Jump();

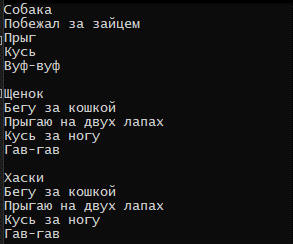
husky.Bite();

husky.Bark();

}

}

Результат:



Задание №2

Условие:

Таксопарк. Определить иерархию легковых автомобилей. Создать таксопарк. Подсчитать стоимость автопарка. Провести сортировку автомобилей парка по расходу топлива. Найти автомобиль в компании, соответствующий заданному диапазону параметров скорости.

public interface ITaxi

{

byte Speed { get; set; }

float Price { get; set; }

byte FuelVolume { get; set; }

}

internal class Economy : ITaxi

{

public Economy(byte speed, float price, byte fuelVolume)

{

Speed = speed;

Price = price;

FuelVolume = fuelVolume;

}

public byte Speed { get; set; }

public float Price { get; set; }

public byte FuelVolume { get; set; }

public override string ToString()

{

return $"Скорость:\t{Speed}\tЦена поездки:\t{Price}\tОбъем топлива:\t{FuelVolume}";

}

}

internal class Comfort : ITaxi

{

public Comfort(byte speed, float price, byte fuelVolume)

{

Speed = speed;

Price = price;

FuelVolume = fuelVolume;

}

public byte Speed { get; set; }

public float Price { get; set; }

public byte FuelVolume { get; set; }

public override string ToString()

{

return $"Скорость:\t{Speed}\tЦена поездки:\t{Price}\tОбъем топлива:\t{FuelVolume}";

}

}

internal class ComfortPlus : ITaxi

{

public ComfortPlus(byte speed, float price, byte fuelVolume)

{

Speed = speed;

Price = price;

FuelVolume = fuelVolume;

}

public byte Speed { get; set; }

public float Price { get; set; }

public byte FuelVolume { get; set; }

public override string ToString()

{

return $"Скорость:\t{Speed}\tЦена поездки:\t{Price}\tОбъем топлива:\t{FuelVolume}";

}

}

internal class Business : ITaxi

{

public Business(byte speed, float price, byte fuelVolume)

{

Speed = speed;

Price = price;

FuelVolume = fuelVolume;

}

public byte Speed { get; set; }

public float Price { get; set; }

public byte FuelVolume { get; set; }

public override string ToString()

{

return $"Скорость:\t{Speed}\tЦена поездки:\t{Price}\tОбъем топлива:\t{FuelVolume}";

}

}

class Program

{

static void Main(string[] args)

{

var economy = new Economy(75, 3.5f, 50);

var comfort = new Comfort(80, 4.5f, 70);

var comfortPlus = new ComfortPlus(100, 6.5f, 70);

var business = new Business(120, 7.5f, 80);

var taxiPark = new List<ITaxi> { economy, comfort, comfortPlus, business };

foreach (var taxi in taxiPark)

{

Console.WriteLine(taxi);

}

Console.WriteLine($"Сумма парка:\t{taxiPark.Sum(taxi => taxi.Price)}");

var filtredPark = taxiPark.OrderByDescending(taxi => taxi.FuelVolume);

Console.WriteLine($"Output\n{string.Join("\n", filtredPark)}\n");

var rangedPark = taxiPark.Where(taxi => taxi.Speed >= 80 && taxi.Speed <=110);

Console.WriteLine($"Output\n{string.Join("\n", rangedPark)}\n");

}

}

Результат:

