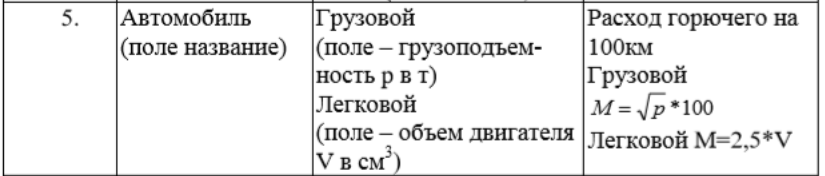
**10 «МЕХАНИЗМ НАСЛЕДОВАНИЯ. ПОЛИМОРФИЗМ».**

Задание 1. Составить программу с одним родительским классом и двумя потомками. Потомки должны содержать виртуальные функции. Создать виртуальную функцию выдачи результатов расчета методов на экран монитора с указанием названий и полей, и их значений соответствующего объекта. Составить тестирующую программу с выдачей протокола на экран монитора. При этом создать объекты базового и производных типов, используя полиморфный контейнер - массив ссылок базового класса на объекты базового и производных классов (количество объектов&gt; =5).



Листинг программы:

using System;

class Car

{

protected string brand;

protected double speed;

public Car(string brand, double speed)

{

this.brand = brand;

this.speed = speed;

}

public virtual double FuelConsumption()

{

return 0;

}

public virtual void DisplayResults()

{

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

Console.WriteLine("Speed: " + speed + " km/h");

}

}

class Truck : Car

{

private double cargoCapacity;

public Truck(string brand, double speed, double cargoCapacity) : base(brand, speed)

{

this.cargoCapacity = cargoCapacity;

}

public override double FuelConsumption()

{

return Math.Sqrt(cargoCapacity) \* 100;

}

public override void DisplayResults()

{

base.DisplayResults();

Console.WriteLine("Cargo Capacity: " + cargoCapacity + " tons");

Console.WriteLine("Fuel Consumption: " + FuelConsumption() + " liters/100 km");

}

}

class PassengerCar : Car

{

private double engineVolume;

public PassengerCar(string brand, double speed, double engineVolume) : base(brand, speed)

{

this.engineVolume = engineVolume;

}

public override double FuelConsumption()

{

return 2.5 \* engineVolume;

}

public override void DisplayResults()

{

base.DisplayResults();

Console.WriteLine("Engine Volume: " + engineVolume + " cubic cm");

Console.WriteLine("Fuel Consumption: " + FuelConsumption() + " liters/100 km");

}

}

class Program

{

static void Main(string[] args)

{

Car[] cars = new Car[5];

cars[0] = new Truck("Volvo", 80, 15);

cars[1] = new PassengerCar("Toyota", 100, 2000);

cars[2] = new Truck("MAN", 70, 12);

cars[3] = new PassengerCar("Ford", 90, 1800);

cars[4] = new PassengerCar("BMW", 120, 2500);

foreach (Car car in cars)

{

Console.WriteLine("\nCar Details:");

car.DisplayResults();

}

}

}Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Car Details:  Brand: Volvo  Speed: 80 km/h  Cargo Capacity: 15 tons  Fuel Consumption: 387,2983346207417 liters/100 km  Car Details:  Brand: Toyota  Speed: 100 km/h  Engine Volume: 2000 cubic cm  Fuel Consumption: 5000 liters/100 km  Car Details:  Brand: MAN  Speed: 70 km/h  Cargo Capacity: 12 tons  Fuel Consumption: 346,41016151377545 liters/100 km  Car Details:  Brand: Ford  Speed: 90 km/h  Engine Volume: 1800 cubic cm  Fuel Consumption: 4500 liters/100 km  Car Details:  Brand: BMW  Speed: 120 km/h  Engine Volume: 2500 cubic cm  Fuel Consumption: 6250 liters/100 km |

Анализ результатов:

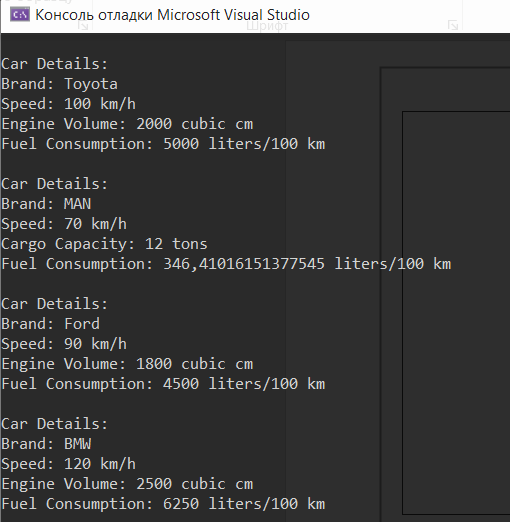


Рисунок 1.1 – Результат работы программы

Задание 2. Составить программу с абстрактным родительским классом и двумя объектами - потомками. Для этого модифицировать задание 2. Составить тестирующую программу с выдачей протокола на экран монитора. В ней нужно реализовать циклический вывод параметров объектов, используя полиморфный контейнер - массив объектов базового класса (количество объектов >=5). Организовать вычисление суммарного расхода горючего.

Листинг программы:

using System;

abstract class Car

{

protected string brand;

protected double speed;

public Car(string brand, double speed)

{

this.brand = brand;

this.speed = speed;

}

public abstract double FuelConsumption();

public virtual void DisplayParameters()

{

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

Console.WriteLine("Speed: " + speed + " km/h");

}

}

class Truck : Car

{

private double cargoCapacity;

public Truck(string brand, double speed, double cargoCapacity) : base(brand, speed)

{

this.cargoCapacity = cargoCapacity;

}

public override double FuelConsumption()

{

return Math.Sqrt(cargoCapacity) \* 100;

}

public override void DisplayParameters()

{

base.DisplayParameters();

Console.WriteLine("Cargo Capacity: " + cargoCapacity + " tons");

}

}

class PassengerCar : Car

{

private double engineVolume;

public PassengerCar(string brand, double speed, double engineVolume) : base(brand, speed)

{

this.engineVolume = engineVolume;

}

public override double FuelConsumption()

{

return 2.5 \* engineVolume;

}

public override void DisplayParameters()

{

base.DisplayParameters();

Console.WriteLine("Engine Volume: " + engineVolume + " cubic cm");

}

}

class Program

{

static void Main(string[] args)

{

Car[] cars = new Car[5];

cars[0] = new Truck("Volvo", 80, 15);

cars[1] = new PassengerCar("Toyota", 100, 2000);

cars[2] = new Truck("MAN", 70, 12);

cars[3] = new PassengerCar("Ford", 90, 1800);

cars[4] = new PassengerCar("BMW", 120, 2500);

double totalFuelConsumption = 0;

foreach (Car car in cars)

{

Console.WriteLine("\nCar Details:");

car.DisplayParameters();

double fuelConsumption = car.FuelConsumption();

Console.WriteLine("Fuel Consumption: " + fuelConsumption + " liters/100 km");

totalFuelConsumption += fuelConsumption;

}

Console.WriteLine("\nTotal Fuel Consumption for all cars: " + totalFuelConsumption + " liters/100 km");

}

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Car Details:  Brand: Volvo  Speed: 80 km/h  Cargo Capacity: 15 tons  Fuel Consumption: 387,2983346207417 liters/100 km  Car Details:  Brand: Toyota  Speed: 100 km/h  Engine Volume: 2000 cubic cm  Fuel Consumption: 5000 liters/100 km  Car Details:  Brand: MAN  Speed: 70 km/h  Cargo Capacity: 12 tons  Fuel Consumption: 346,41016151377545 liters/100 km  Car Details:  Brand: Ford  Speed: 90 km/h  Engine Volume: 1800 cubic cm  Fuel Consumption: 4500 liters/100 km  Car Details:  Brand: BMW  Speed: 120 km/h  Engine Volume: 2500 cubic cm  Fuel Consumption: 6250 liters/100 km  Total Fuel Consumption for all cars: 16483,708496134517 liters/100 km |

Анализ результатов:

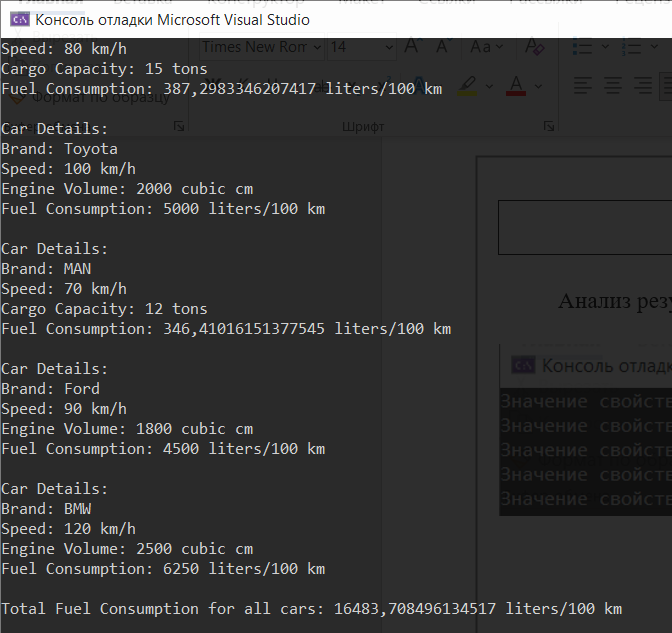


Рисунок 1.2 – Результат работы программы