//This real-world code demonstates the Builder pattern in which different vehicles are assembled in a step-by-step fashion.

//The Shop uses VehicleBuilders to construct a variety of Vehicles in a series of sequential steps.

using System;

using System.Collections.Generic;

namespace BuilderExRealWorld

{

/// <summary>

/// MainApp startup class for Real-World

/// Builder Design Pattern.

/// </summary>

public class MainApp

{

/// <summary>

/// Entry point into console application.

/// </summary>

public static void Main()

{

VehicleBuilder builder;

// Create shop with vehicle builders

Shop shop = new Shop();

// Construct and display vehicles

builder = new ScooterBuilder();

shop.Construct(builder);

builder.Vehicle.Show();

builder = new CarBuilder();

shop.Construct(builder);

builder.Vehicle.Show();

builder = new MotorCycleBuilder();

shop.Construct(builder);

builder.Vehicle.Show();

// Wait for user

Console.ReadKey();

}

}

/// <summary>

/// The 'Director' class

/// </summary>

class Shop

{

// Builder uses a complex series of steps

public void Construct(VehicleBuilder vehicleBuilder)

{

vehicleBuilder.BuildFrame();

vehicleBuilder.BuildEngine();

vehicleBuilder.BuildWheels();

vehicleBuilder.BuildDoors();

}

}

/// <summary>

/// The 'Builder' abstract class

/// </summary>

abstract class VehicleBuilder

{

protected Vehicle vehicle;

// Gets vehicle instance

public Vehicle Vehicle

{

get { return vehicle; }

}

// Abstract build methods

public abstract void BuildFrame();

public abstract void BuildEngine();

public abstract void BuildWheels();

public abstract void BuildDoors();

}

/// <summary>

/// The 'ConcreteBuilder1' class

/// </summary>

class MotorCycleBuilder : VehicleBuilder

{

public MotorCycleBuilder()

{

vehicle = new Vehicle("MotorCycle");

}

public override void BuildFrame()

{

vehicle["frame"] = "MotorCycle Frame";

}

public override void BuildEngine()

{

vehicle["engine"] = "500 cc";

}

public override void BuildWheels()

{

vehicle["wheels"] = "2";

}

public override void BuildDoors()

{

vehicle["doors"] = "0";

}

}

/// <summary>

/// The 'ConcreteBuilder2' class

/// </summary>

class CarBuilder : VehicleBuilder

{

public CarBuilder()

{

vehicle = new Vehicle("Car");

}

public override void BuildFrame()

{

vehicle["frame"] = "Car Frame";

}

public override void BuildEngine()

{

vehicle["engine"] = "2500 cc";

}

public override void BuildWheels()

{

vehicle["wheels"] = "4";

}

public override void BuildDoors()

{

vehicle["doors"] = "4";

}

}

/// <summary>

/// The 'ConcreteBuilder3' class

/// </summary>

class ScooterBuilder : VehicleBuilder

{

public ScooterBuilder()

{

vehicle = new Vehicle("Scooter");

}

public override void BuildFrame()

{

vehicle["frame"] = "Scooter Frame";

}

public override void BuildEngine()

{

vehicle["engine"] = "50 cc";

}

public override void BuildWheels()

{

vehicle["wheels"] = "2";

}

public override void BuildDoors()

{

vehicle["doors"] = "0";

}

}

/// <summary>

/// The 'Product' class

/// </summary>

class Vehicle

{

private string \_vehicleType;

private Dictionary<string, string> \_parts =

new Dictionary<string, string>();

// Constructor

public Vehicle(string vehicleType)

{

this.\_vehicleType = vehicleType;

}

// Indexer

public string this[string key]

{

get { return \_parts[key]; }

set { \_parts[key] = value; }

}

public void Show()

{

Console.WriteLine("\n---------------------------");

Console.WriteLine("Vehicle Type: {0}", \_vehicleType);

Console.WriteLine(" Frame : {0}", \_parts["frame"]);

Console.WriteLine(" Engine : {0}", \_parts["engine"]);

Console.WriteLine(" #Wheels: {0}", \_parts["wheels"]);

Console.WriteLine(" #Doors : {0}", \_parts["doors"]);

}

}

}