using System;

namespace FurnitureStore

{

// Интерфейс для функциональности сборки мебели

public interface IAssemble//Любой класс, который реализует этот интерфейс, должен предоставить свои версии этих методов. **Интерфейс** — это конструкция в программировании, определяющая набор методов и свойств, которые должен реализовать класс, подписывающийся на этот интерфейс. В отличие от классов, интерфейс не содержит реализаций этих методов — только их сигнатуры (имя, параметры и возвращаемое значение). Класс, реализующий интерфейс, обязуется предоставить реализацию для всех методов и свойств, объявленных в интерфейсе.

{

void Assemble();

void ShowInfo();

}

public abstract class Product

{//абстрактный класс, представляющий базовый продукт в магазине. У него есть свойства Name (имя) и Price (цена). **Абстрактный класс** может содержать как реализованные методы, так и абстрактные (нереализованные). Это позволяет ему предоставлять некоторый базовый функционал, который может быть унаследован дочерними классами.

public string Name;

public decimal Price;

public Product(string name, decimal price)

{//Конструктор Product(string name, decimal price) инициализирует эти свойства.

Name = name;

Price = price;

}

public abstract void DisplayInfo();

public abstract void ShowInfo(); // Новый метод, совпадающий с интерфейсом

public override string ToString() => $"{GetType().Name} - {Name}, Цена: {Price:C}";

}

// Базовый класс Мебель

public abstract class Furniture : Product, IAssemble

{

public string Material;

public string Color;

public Furniture(string name, decimal price, string material, string color)

: base(name, price)

{

Material = material;

Color = color;

}

public virtual void Assemble()//можно переопределить в любых классах наследуемых

{

Console.WriteLine($"Сборка {Name} из {Material} цвета {Color}.");

}

public override void ShowInfo()

{

Console.WriteLine($"Мебель - {Name}, Материал: {Material}, Цвет: {Color}, Цена: {Price:C}");

}

}

// Класс Диван

public class Sofa : Furniture

{

public int SeatCount;

public Sofa(decimal price, string material, string color, int seatCount)

: base("Диван", price, material, color)

{

SeatCount = seatCount;

}

public override void DisplayInfo()

{

Console.WriteLine($"{Name}: {Material}, {Color}, Кол-во мест: {SeatCount}, Цена: {Price:C}");

}

public override void ShowInfo()//переопределение метода из Furniture

{

Console.WriteLine($"Диван - Кол-во мест: {SeatCount}, Материал: {Material}, Цвет: {Color}, Цена: {Price:C}");

}

}

// Класс Кровать

public sealed class Bed : Furniture

{

public string Size;

public Bed(decimal price, string material, string color, string size)

: base("Кровать", price, material, color)

{

Size = size;

}

public override void DisplayInfo()

{

Console.WriteLine($"{Name}: {Material}, {Color}, Размер: {Size}, Цена: {Price:C}");

}

public override void Assemble()

{

Console.WriteLine($"Сборка кровати размера {Size} цвета {Color}.");

}

public override void ShowInfo()

{

Console.WriteLine($"Кровать - Размер: {Size}, Материал: {Material}, Цвет: {Color}, Цена: {Price:C}");

}

}

// Класс Вешалка

public class Hanger : Product, IAssemble

{

public string Type;

public Hanger(decimal price, string type) : base("Вешалка", price)

{

Type = type;

}

public override void DisplayInfo()

{

Console.WriteLine($"{Name}: {Type}, Цена: {Price:C}");

}

public void Assemble()

{

Console.WriteLine($"Установка вешалки типа {Type}.");

}

public override void ShowInfo()

{

Console.WriteLine($"Вешалка - Тип: {Type}, Цена: {Price:C}");

}

}

// Класс Printer с полиморфным методом

public class Printer

{

public void IAmPrinting(Product someObj)

{

Console.WriteLine(someObj.ToString());

someObj.ShowInfo();

}

}

class Program

{

static void Main()

{

var sofa = new Sofa(499.99m, "Кожа", "Чёрный", 3);

var bed = new Bed(799.99m, "Дерево", "Белый", "King-size");

var hanger = new Hanger(15.99m, "Настенная");

Product[] products = { sofa, bed, hanger };

Printer printer = new Printer();

// Демонстрация вызова через ссылки на интерфейсы и абстрактные классы

Console.WriteLine("Демонстрация работы с объектами через интерфейсы и абстрактные классы:");

foreach (var product in products)

{

// Проверяем, поддерживает ли объект интерфейс IAssemble

if (product is IAssemble assemble)

{

assemble.Assemble();

}

product.DisplayInfo();

Console.WriteLine();

}

// Использование класса Printer для полиморфного вызова

Console.WriteLine("Использование класса Printer для вывода информации о товаре:");

foreach (var product in products)

{

printer.IAmPrinting(product);

Console.WriteLine();

}

}

}

}